EMPIRICAL VALIDATION OF CROSS-PROJECT METHODOLOGIES FOR SOFTWARE QUALITY PREDICTIVE MODELING

A Thesis Submitted
In Partial Fulfillment of the Requirements
for the Degree of

DOCTOR OF PHILOSOPHY

By

SHWETA MEENA
(2k18/Ph.D./CO/06)

Under the Supervision of
Prof. Ruchika Malhotra
Head of Department
Department of Software Engineering

Department of Software Engineering
DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Shahbad Daulatpur, Main Bawana Road, Delhi 110042. India

July, 2024

DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)

Shahbad Daulatpur, Main Bawana Road, Delhi-42

CANDIDATE'S DECLARATION

I **Shweta Meena** hereby certify that the work which is being presented in the thesis entitled **"Empirical Validation of Cross-Project Methodologies for Software Quality Predictive Modeling"** in partial fulfillment of the requirements for the award of the Degree of Doctor of Philosophy, submitted in the Department of **Software Engineering**, Delhi Technological University is an authenticate record of my own work carried out during the period from **2018** to **2024** under the supervision of **Prof. Ruchika Malhotra.**

The matter presented in the thesis has not been submitted by me for the award of any other degree of this or any other Institute.

Candidate's Signature

DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)

Shahbad Daulatpur, Main Bawana Road, Delhi-42

CERTIFICATE BY THE SUPERVISOR

Certified that **Ms. Shweta Meena (Roll no. 2k18/Ph.D./CO/06)** has carried out their search work presented in this thesis entitled **"Empirical Validation of Cross-Project Methodologies for Software Quality Predictive Modeling"** for the award of **Doctor of Philosophy** from Department of Software Engineering, Delhi Technological University, Delhi, under my supervision. The thesis embodies results of original work, and studies are carried out by the student herself and the contents of the thesis do not form the basis for the award of any other degree to the candidate or to anybody else from this or any other University/Institution.

Signature

PROF. RUCHIKA MALHOTRA

HoD and DRC Chairperson

Department of Software Engineering

Delhi Technological University, Delhi 110042

Date:

Acknowledgment

With profound gratitude, I extend heartfelt appreciation to my supervisor, **Prof. Ruchika Malhotra, Head of Department and DRC Chairperson** whose unwavering mentorship and encouragement have been instrumental in the successful completion of this research. The opportunity to undertake this journey under her guidance has been a blessing, unlocking my potential for Ph.D. I am immensely thankful for her priceless intellectual support, invaluable guidance, and sustained motivation throughout this endeavor.

I extend my gratitude to all **faculty members** of the department for their support.

I express deep appreciation to my father, **Mr. N. R. Meena**, for his steadfast encouragement and moral support. His inspiration for hard work has been pivotal in my achievements. I extend warm love and gratitude to my parents for their unconditional support and driving force in my life.

Special appreciation goes to my loving mother, **Mrs. Prem Devi**, whose presence brought immense happiness and joy, providing strength during challenging stages of my research.

Shweta Meena

ABSTRACT

Nowadays, the size of software's is increasing every day. The challenge faced by software developers and experts is to develop large and detailed software projects within a reasonable amount of time and with a reasonable amount of resources by satisfying the customer's requirements. These days, the focus of organizations is on the creation of quality software within a specified budget and time. However, the company focuses on delivering quality software considering the quality attributes satisfied by the software from the developer and customer perspective. Thus, developers keep a check on the software quality throughout all the phases of development. Software quality needs to be ensured from the beginning of requirement gathering to the delivery of software to the end user. However, the software requires continuous updates with technological advancements or any kind of updates requested in customer requirements after deployment at the customer site. However, after the deployment of software at the customer site, it also ensures software quality in terms of maintainability, user-friendliness, and reliability. The quality of software is important from every perspective, considering its internal and external qualities. Thus, considering internal quality attributes, software developers focus on aspects related to the software's structure, design, and implementation, such as maintainability, flexibility, scalability, and reusability. Thus, considering external quality attributes, software developers focus on aspects related to the software's behavior and performance as perceived by users, such as functionality, reliability, performance, and usability. Moreover, the software quality is also affected by the amount of data used to develop software. In the initial stages, developers design a prototype considering conventional software. Due to this, the performance of developed software is not much better than conventional

software.

One of the ways to improve quality is to develop predictive models using real-world data. The prediction models are developed using software's internal properties, such as cohesion, coupling, and inheritance. The prediction models can be developed using Machine Learning (ML) techniques by dividing the data into one part as training data and another part as testing/ validation. It has been seen that researchers tend to validate the ML models from the dataset used for development, which resulted in producing biased and non-generalizable results. Keeping this in view, it is important to develop models using training data from one project and validate the same models using validation data derived from another project. This concept is often known as Cross-Project Validation (CPV). Furthermore, the objective of CPV is to develop prediction models that produce generalized and unbiased results. Thus, this work focuses on developing and validating models using CPV techniques.

This research is focused on various techniques to improve the performance of models developed using CPV methodology. Due to the limited availability of data, the idea of cross-project arises. The cross-project prediction model is further categorized into two categories based on types of projects and types of features in train_data and test_data. The work of this research is to explore Intra-Project validation and Inter-Project validation. The feasibility of CPV with Intra-Project validation is analyzed in this research. The research is focused on using existing models for predictions on future data. The analysis conducted in this study considered ML techniques. Also, the primary objective of conducting an experiment in this study is to enhance the quality of software's considering four important aspects defect, change, maintenance, and effort. The study mainly focused on defect and change prediction models.

Another focus of this research is to explore and validate the applicability of Transfer Learning (TL). TL is one of the CPV techniques. Using TL concept, the predictive model is to be designed and will be used to validate other project data. The main concept is that one project is used as training data, which will be used for creating

a prediction model, and that model is used to validate other project data. Lastly, this work focuses on inter-version project validation, where different versions of the same project are used for training and testing data. Thus, a prediction model will be created for the inter-version project validation. Furthermore, the impact of feature selection techniques is also analyzed for cross-project models. A comparison was performed among ranking-based and greedy search algorithms to design efficient cross-project models for unseen datasets. The impact of TL models with cross-project models is also analyzed. The thesis also evaluates the CPV models for enhancing the maintainability and reusability of projects. We believe that this study's research outcomes would help improve the quality of software prediction models based on the concept of cross-project methodologies by developing software projects in a reasonable time and cost. This research's findings would benefit academicians, researchers, and industry experts in developing efficient software with better quality.

List of Publications

Papers Accepted/Published in International Journals

1. Ruchika Malhotra and Shweta Meena, "Defect prediction model using transfer learning", *Soft Computing*, vol. 26, no. 10, pp. 4713-4726, 2022 (Impact Factor: 4.1). (https://doi.org/10.1007/s00500-022-06846-x)

2. Ruchika Malhotra and Shweta Meena, "Empirical validation of machine learning techniques for heterogeneous cross-project change prediction and within-project change prediction", *Journal of Computational Science*, vol. 76, pp. 102230, 2024 (Impact Factor: 3.3). (https://doi.org/10.1016/j.jocs.2024.102230)

3. Ruchika Malhotra and Shweta Meena, "A Systematic Review of Transfer Learning in Software Engineering", *Multimedia Tools and Applications*, 2024 (Impact Factor: 3.6), Accepted.

4. Ruchika Malhotra and Shweta Meena, "Empirical validation of feature selection techniques for cross-project defect prediction", *International Journal of System Assurance Engineering and Management*, pp. 1-13, 2023 (Impact Factor: 2.0). (https://doi.org/10.1007/s13198-023-02051-7)

Papers Accepted/Published in International Conferences

5. Ruchika Malhotra and Shweta Meena, "Empirical validation of cross-version

and 10-fold cross-validation for defect prediction, *2021 Second International Conference on Electronics and Sustainable Communication Systems*, pp. 431-438, 2021.

6. Ruchika Malhotra and Shweta Meena, "Grid Search-Optimized Artificial Neural Network for Heterogeneous Cross-Project Defect Prediction", *In International Conference on Data Analytics and Management*, pp. 447-458, Singapore, 2023.

7. Ruchika Malhotra and Shweta Meena, "Empirical Validation of Software Defect Prediction Models using Grid Search Grey Wolf Optimization", *3rd International Conference on Computing and Communication Networks (ICCCNet-2023)*, United Kingdom, 2023.

Papers Communicated in International Journals

8. Ruchika Malhotra and Shweta Meena, "An improved cross-project defect prediction model developed using optimized neural network", *Arabian Journal for Science and Engineering*.

Contents

List of Tables

List of Figures

List of Symbols and Abbreviations

ACO	**Ant Colony Optimization**
ADB	**ADaBoost**
ADCNN	**Adversarial Discriminative Convolutional Neural Network**
AI	**Artificial Intelligence**
AMC	**Average Method Complexity**
ANN	**Artificial Neural Network**
ANOVA	**Analysis of Variance**
API	**Application Programming Interface**
ARTL	**Adaptation Regularization TL**
AST	**Abstract Syntax Tree**
AUC	**Area Under Curve**
AUCEC	**Area Under the Cost-Effectiveness Curve**
B	**Halstead Effort**
BBN	**Bayesian Belief Network**
BDA	**Balanced Distribution Adaptation**
BETL	**Bagging Based TL**
BiNN	**Bidirectional Neural Network**
BL	**Bayesian Learning**
BN	**Bayesian Network**
BNG	**Bagging**
BRC	**Branch Count**
Ca	**Afferent Coupling**
CAM	**Cohesion Among Methods of Class**

CBM	**Coupling Between Method**
CBO	**Coupling Between Objects**
CCA	**Canonical Correlation Analysis**
CCA+	**Canonical Correlation Analysis Extended**
CCB	**Change Control Board**
CD	**Critical Difference**
Ce	**Efferent Coupling**
CLL	**Conditional log-likelihood**
CNN	**Convolutional Neural Network**
CPCP	**Cross-Project Change Prediction**
CPDP	**Cross-Project Defect Prediction**
CPV	**Cross-Project Validation**
CSO	**Cat Swarm Optimization**
CVDP	**Cross-Version Defect Prediction**
CWCARM	**Correlation Coefficient Weighted Class Association Rule Mining**
D	**Halstead Difficulty**
DAM	**Data Access Metrics**
DBT	**Discriminability Based TL**
DE-SDP	**Dual Ensemble Software Defect Prediction**
DIT	**Depth of Inheritance Tree**
DL	**Deep Learning**
DoF	**Degree of Freedom**
DP	**Defect Prediction**
DT	**Decision Tree**
E_i	**Expected Count**
EL	**Ensemble Learning**
EOCM	**Entropy of Change Metric**
ev(g)	**Essential Complexity**
EVS	**Evolutionary Search**
FN	**False Negative**

FP	**False Positive**
FPR	**False Positive Rate**
FS	**Feature Selection**
FSR	**Feature Space Remapping**
GA	**Genetic Algorithm**
GAFSR	**GA for Feature-Space Remapping**
GBM	**Graph-Based Methods**
GCMF	**Graph co-regularized Collective Matrix tri-Factorization**
GFK	**Geodesic Flow Kernel**
GrFSR	**Greedy Search for Feature-Space Remapping**
GrS	**Greedy Search**
GTL	**Graph Co-Regularization TL**
GWO	**Grey Wolf Optimization**
HetCPCP	**Heterogeneous Cross-Project Change Prediction**
HetCPDP	**Heterogeneous Cross-Project Defect Prediction**
HoCP	**Homogeneous Cross Project**
HSD	**Honest Significant Difference**
HSO	**Harmony Search Optimization**
HV	**Hybrid Voting**
IC	**Inheritance Coupling**
IDE	**Integrated Development Environment**
IdTL	**Inductive Transfer Learning**
IG	**Information Gain**
IMM	**Information Measure Metric**
InPDP	**Intra Project Defect Prediction**
iv(g)	**Design Complexity**
JIT	**Just-In-Time**
KCCA+	**Kernel Canonical Correlation Analysis Extended**
KNN	**K-Nearest Neighbor**
KNNDD	**K-Nearest Neighbor Data Description**

KS test	**Kolmogorov-Smirnov test**
LB	**LogiBoost**
LCOM	**Lack of COhesion in Methods**
LCOM3	**Lack of Cohesion**
Le	**Halstead Length**
LIME	**Local Interpretable Model-Agnostic Explanations**
LDA	**Linear Discriminant Analysis**
LiR	**Linear Regression**
LOC	**Line of Code**
LR	**Logistic Regression**
LSTM	**Long Short Term Memory**
MAE	**Mean Absolute Error**
MBRE	**Mean Balanced Relative Error**
MCC	**Matthews Correlation Coefficient**
MER	**Magnitude of Error Relative to the Estimate**
MFA	**Measure of Functional Abstraction**
ML	**Machine Learning**
MLP	**Multi Layer Perceptron**
MO	**Multi-Objective**
MOA	**Measure of Aggregation**
MR	**Manifold Regularization**
MRE	**Magnitude of Relative Error**
MSVM	**Multiclass Support Vector Machine**
MTL	**Multitask Learning**
NB	**Naive Bayes**
NEDT	**Non-Ensemble Decision Trees**
NIV	**Instance Variable**
NN	**Neural Network**
NNS	**Nearest Neighbor Selection**
NOA	**Number of Attributes**

NOC	**Number of Children**
NOM	**Number of Methods**
Nopn	**Num Operands**
Nopt	**Num Operators**
NPrA	**Number of Private Attributes**
NPrM	**Number of Private Methods**
NPuA	**Number of Public Attributes**
NPuM	**Number of Public Methods**
O_i	**Observed Count**
OO	**Object-Oriented**
OOM	**Object-Oriented Metrics**
PSO	**Particle Swarm Optimization**
QDA	**Quadratic Discriminant Analysis**
QMOOD	**Quality Model for Object-Oriented Design**
RF	**Random Forest**
RFC	**Response for a Class**
RFE	**Recursive Feature Elimination**
RMSE	**Root Mean Square Error**
RNN	**Recurrent Neural Network**
RQs	**Research Questions**
SA	**Standardized Accuracy**
SCL	**Structural Corresponding Learning**
SCM	**Source Code Metric**
SCP	**Software Change Prediction**
SDP	**Software Defect Prediction**
SGD	**Stochastic Gradient Descent**
SHP	**SHapley Additive exPlanations**
SLOC	**Lines of Source Code**
SMO	**Sequential Minimal Optimization**
SMOTE	**Synthetic Minority Over-sampling TEchnique**

SoR	**Softmax Regression**
SpMO	**Spider Monkey Optimization**
SQPM	**Software Quality Predictive Modeling**
STL	**Standard Template Library**
SVDDD	**Support Vector Domain Data Description**
SVM	**Support Vector Machine**
T	**Halstead Program Time**
TANN	**Transfer Artificial Neural Network**
TCA	**Transfer Component Analysis**
TCNN	**Transfer Convolutional Neural Network**
TdTL	**Transductive Transfer Learning**
TGWO	**Transfer Grey Wolf Optimization**
TJM	**Transfer Joint Matching**
TKNN	**Transfer K-Nearest Neighbor**
TL	**Transfer Learning**
TN	**True Negative**
TNB	**Transfer Naive Bayes**
TP	**True Positive**
TPR	**True Positive Rate**
TRF	**Transfer Random Forest**
TRNN	**Transfer Recurrent Neural Network**
TSVM	**Transductive Support Vector Machine**
UAR	**Unweighted Average Recall**
UnTL	**Unsupervised Transfer Learning**
V	**Halstead Volume**
v(g)	**Cyclomatic Complexity**
V_d	**Food source**
V_s	**Ant Colony**
WEKA	**Waikato Environment for Knowledge Analysis**
WMC	**Weighted Method per Class**

WNBC	**Weighted NB classifier**
WPCP	**Within-Project Change Prediction**
WPDP	**Within-Project Defect Prediction**

Chapter 1

Introduction

1.1 Introduction

Software quality is a primary objective in software engineering. Functional and non-functional quality attributes are crucial for high-quality software. Most resources 60-70% go towards evolving software, including code, data, and documentation[1]. Customer satisfaction is key when delivering good-quality software [2]. To develop quality software, we must consider existing software's. Developers change and update existing software for detection and removal of defects to create good-quality software in the future. Software quality ensures future updates and respects technological advancements. Predictive modeling uses historical context and cross-project methodologies. Integrating predictive modeling aims to develop an ML model using past project data for better software in the future. The team defines software quality and attributes and then demonstrates predictive modeling. We discuss cross-project methodologies such as inter-version, cross-project,

cross-company, and TL. Due to non-availability of a sufficient amount of training data, cross-project methodologies emerged. The same dataset can be used for training and testing, but it leads to overfitting and biased results for a specified dataset. Thus, this issue was resolved using cross-project methodology. Utilization of different dataset for training and testing removes this problem for researchers.

The integration of TL provides effective results in the software engineering domain. The knowledge of existing models must be utilized in order to develop more efficient and reliable software. The following section examines software quality prediction and transfer learning applications. The predictive modeling considering the historical contexts is employed with cross-project methodologies. The main aim of integrating predictive modeling is to develop an ML model to utilize the existing project data to design better software in the future; it focuses on past practices followed.

The focus of this thesis is the development of efficient prediction models using cross-project and TL. The basic concepts involved in the thesis and the motivation of research work are discussed in this chapter. Firstly, the basic terminology of software quality and software quality attributes is defined, with examples such as functionality, usability, performance, security, reliability and maintainability (Section 1.1). Further, Software Quality Predictive Modeling (SQPM) is demonstrated using a hypothetical software project alongwith basic steps for development of SQPM (Section 1.3). Next, the cross-project methodologies, such as inter-version, cross-project, cross-company, TL and types of TL summarized with real-world example (Section 1.4). Section 1.5 discussed the literature work with respect to cross-project methodologies, defect prediction, change prediction, and TL. Section 1.6 presents the Vision of thesis. Section 1.7 discussed the objective of thesis. Section 1.8 presents overview of the work done to improvise the quality of

software using cross-project and TL. Section 1.9 presents the organization of thesis in the subsequent chapters.

1.2 Introduction to Software Quality

This section discusses the significance of software quality alongwith software quality attributes in detail.

1.2.1 Software Quality

Software quality is defined as the extent to which the software satisfies the customer requirements or end user needs and expectations [2].

- Conformance to requirements
- Fitness for the purpose
- Level of satisfaction

Software quality includes various aspects, including functionality, reliability, usability, efficiency, maintainability, and security. Thus, developing high-quality software requires rigorous testing, adherence to coding standards, and continuous improvement of processes throughout the software development life cycle. However, a constant focus on software quality ensures that the final software is robust, reliable, and capable of delivering a positive user experience. Moreover, the software quality is beyond functionality and focus on developing more robust, user-friendly, and secure software by utilizing existing software.

1.2.2 Software Quality Attributes

Software quality attributes are termed as software quality characteristics or quality factors. It represents the various dimensions or facets of software quality. The significance of these attributes is to assess and analyze the overall quality of software. The standard quality attributes used are as follows:

- *Reliability*: This attribute refers to the software's ability to perform its intended functions correctly and consistently under specific conditions for a defined period. Reliability indicates the software's stability and its resistance to failures or errors.
- *Functionality*: The functionality attribute focused on the functioning of the software. It states that the software has to do what it is designed for. However, software must perform its intended functionality correctly. Functional quality ensures that the software delivers on the promises outlined in its requirements and specifications.
- *Usability*: How easy is the software to learn and use? Does it guide users intuitively or leave them scratching their heads? Usability encompasses factors like navigation, clarity of instructions, and the overall ease of achieving goals within the software.
- *Security*: In today's digitally-connected world, software security is crucial. Quality software protects user data and resists cyberattacks and unauthorized access. Security measures like encryption, strong authentication, and regular vulnerability assessments are vital to maintain user trust.
- *Performance*: Performance analyzes software's capability by analyzing how well it performs in terms of various aspects such as speed, responsiveness, throughput, and

resource utilization. It encompasses response time, execution speed, and scalability under varying workloads.

- *Maintainability*: The ease with which the software can be modified, updated, repaired, or extended is termed as maintainability. It encompasses attributes such as modularity, code clarity, documentation quality, and adherence to coding standards.

These are some key software quality attributes, and many additional attributes depend on specific project requirements or industry standards. However, evaluating and optimizing software quality considering these attributes are essential for developing and delivering high-quality software that satisfies user requirements.

1.3 Software Quality Predictive Modeling

This section discusses predictive modeling, the validation process, the steps for SQPM.

1.3.1 What is Predictive Modeling?

Predictive modeling is a procedure for predictive analytics to create a model for predicting future behavior [3]. The predictive model consists of predictors, which are variable factors that affect the output variables. It collects historical data for designing predictive rules to apply to future data is essential for predictive modeling. SQPM means predicting software quality during the early phase of software development. SQPM helps in efficient utilization of available resources in a reasonable amount of time. SQPM is a kind of data mining that uses historical data for model development.

Development of prediction models uses a specified set of input variables with values to predict value of output variables/predictors. Predictive modeling is also considered an essential part of predictive analytics, which is part of data analytics. However, in software engineering, predictive modeling indicates the development of predictive models for estimating software quality in terms of defect, change, maintenance, and effort. Thus, prediction models use software metrics as input variables. Software metrics signifies Object-Oriented (OO) software characteristics and Halstead metrics such as inheritance, reusability, coupling, cohesion metrics, size of a program, vocabulary of a program, length of a program, volume of a program. Moreover, the output of the prediction model can be the presence or absence of defect, bug count, presence or absence of change, effort estimation, and maintainability prediction.

1.3.2 Steps in SQPM

The steps [3] for creating a predictive model are as follows:

- Collection of factual data.
- Choosing appropriate learning techniques for model development.
- Outlier analysis and handling of missing values are included in data cleaning.
- Relevant attribute selection.
- Model development using training dataset.
- Validation of the model for its performance determination.
- Use developed model for predictions on future dataset.

1.4 Validation Methodologies

The discussion encompasses the validation methodologies, the settings and types of TL.

1.4.1 Cross-Validation

It is a technique in ML to analyze the model's performance on future data. Cross-validation divides the dataset into subsets, using one subset as the validation set and the remaining folds as the training set. However, there will be multiple folds of the training set, so that each iteration uses a different fold as the training set. Thus, the average of all folds produces a more robust model. Cross-validation aims to avoid overfitting. The issue of overfitting arises when a model is trained too well on training data but needs to perform better on unseen or future data. However, various cross-validation techniques exist, such as *k*-fold cross-validation, leave-one-out cross-validation, and hold-out validation. Cross-validation focused on developing a model that provides unbiased and generalized results. The discussion encompasses the different types of cross-validation techniques as follows:

1. **Hold-out cross-validation:** The dataset is divides into two parts: 50% as a training dataset, and 50% as a testing purposes. Although it is a simple model. However, it is not advantageous for models because it may be possible that the remaining 50% of the dataset contains significant information, which may lead to high bias.

2. **Leave-one-out cross–validation:** In Leave-one-out cross-validation, training is performed on the complete dataset but leaves one data point for testing and further iterates for every data point. However, this method is helpful as *n*-1 samples are used for training, and left-out data samples are used for testing, that reduces bias in

the model. However, if the remaining data point is an outlier, then it would lead to higher variation.

3. ***K*-Fold cross–validation:** The dataset is divided into K subsets. Training is conducted on all subsets except for one, the K-1 subset, which is utilized to evaluate the model. This process is repeated K times, with different subsets being used for training each time. The value of K, which represents the number of subsets, is a crucial parameter. A higher value of K, such as 10, can provide a more accurate estimate of the model's performance, but it also increases the computational cost.

1.4.2 Inter-Version Validation

The inter-version validation is one of the cross-project methodologies in software engineering. It validates or assesses a software system's performance, functionality, or quality across different versions or releases. Inter-version validation aims to ensure that changes introduced in subsequent versions of the software doesn't introduce new errors, defects, or unexpected behaviors compared to the existing version of that software. However, it ensures the software's functionality remains consistent and reliable across different versions. Inter-version validation is not just a process, it's a shield. It plays a crucial role in software development and maintenance by ensuring the software's reliability, stability, and consistency across different releases.

1.4.3 Cross-Project/ Company Validation

The development of a prediction model, such as the cross-project model, requires a sufficient amount of training and testing data. In cases where data is not readily available,

data from other projects is used to develop the cross-project model. The cross-project model is then categorized into different types, such as cross-company, inter-project, and TL, based on the characteristics and behavior of the other projects.

Cross-project is employed in the non-availability of sufficient training dataset. It considers domain of feature set for both the projects utilize in the development of prediction model. In cross-company validation methodology, different company projects are used for training and testing and have similar data distribution. Let's discuss it with an example: we are considering ordering modules from Flipkart and Amazon. Here, we can train our model on the Flipkart module and test it on the Amazon module. The concept of cross-company is now involved here as both modules belong to different companies, i.e., E-commerce companies. However, a more advanced prediction model has been developed to determine which modules were utilized by these companies with feature set.

1.4.3.1 Transfer Learning

In the domain of ML, TL emerged as a powerful technique that empowers models to leverage knowledge acquired from one task (source task) and adapt it to a related but distinct task (target task). This approach proves particularly effective when faced with limited labeled data for the target task. The central concept of TL is to use the knowledge that has been learned from a task (source) having a lot of labeled training data available and transfer that knowledge to a new task (target) where we don't have a sufficient amount of data presented in Fig. 1.1.

TL can be utilized in various ways based on source and target data such as Inductive TL, Transductive TL, Unsupervised TL. Inductive TL can be used for multi-task learning and self-taught learning. Transductive TL can be used for domain adaptation, sample

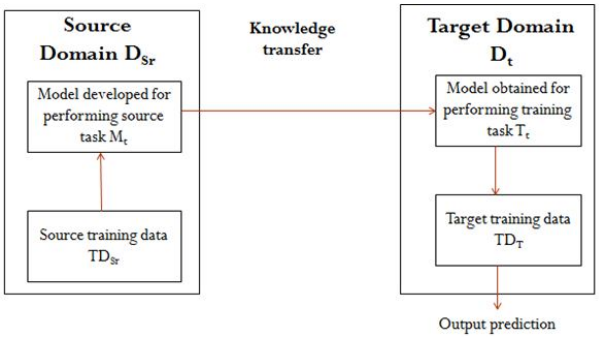

Figure 1.1: Transferring knowledge from a source domain to target domain

selection bias, and covariance shift. The dataset labels (presence/absence) of source and target datasets differ. Further, based on the type of problem, the TL type and TL settings are selected. In this manner there is no need to start from scratch, we can utilize the knowledge gained from a related task (source). In this work, the Inductive TL is performed. Thus, TL is integrated into the software engineering domain to improve the quality of future projects. With the help of TL existing projects can be reused with improved functionality in terms of code, and performance of the software.

1.4.3.2 Types of Transfer Learning

1. **Feature transfer**: In feature type TL, the relationship among the features of the source and target dataset is established in a similar space distribution. It synthesizes example data for model building.

2. **Instance-transfer**: It provides some example data for model building in the target dataset considering instances of source dataset.

3. **Parameter-transfer**: It provides parameter terms for existing models.

4. **Relational transfer**: Relational transfer establishes the relationship between source and target dataset.

1.4.3.3 Transfer Learning Settings

There are three different types of settings in TL based on the type of training and testing data as follows:

1. **Transductive TL:** In transductive TL, the model is trained on a source task and applied directly to the target task without additional training. The target task may have some labeled data available during inference. The model leverages the source task data it was trained on and the limited labeled data from the target task to make predictions. This type of setting is helpful for the limited amount of labeled data in the target task. Thus, the main goal is to improve prediction performance by leveraging knowledge from the source task.

2. **Inductive TL:** Inductive TL involves training the model on a source task and then fine-tuning it on the target task using the available labeled data. The model's parameters are updated or fine-tuned during training on the target task to adapt to its specific characteristics and requirements. It is useful when the target task contains a considerable amount of labeled data compared to source data. However, the model can learn features and patterns specific to a task through inductive TL.

3. **Unsupervised TL:** Using unsupervised TL, the source and target data lack labeled data, eliminating the need for supervision. This type of TL allows for easy knowledge transfer to the target task using a source dataset. Moreover, unsupervised TL becomes useful when no target data is available.

Each TL type, with its specific settings, presents a strategic advantage and disadvantage. This strategic use is crucial, as it allows for a advanced approach depending on the situation, data availability, task, data distribution, latent space, and the resources available for training and fine-tuning the prediction model, keeping your decision-making informed and strategic.

Advantages of Transfer Learning:

1. **Efficient Training:** Each TL type, with its specific settings, presents a strategic advantage and disadvantage. This strategic use is crucial, as it allows for a nuanced approach depending on the situation, data availability, task, data distribution, latent space, and the resources available for training and fine-tuning the prediction model, keeping your decision-making informed and strategic.

2. **Improved Generalization:** The requirement is the amount of labeled data and computational resources using pre-trained models with the help of TL. Thus, it is advantageous when data availability is expensive and resources are limited.

3. **Quick Model Deployment:** The use of a pre-trained model significantly accelerates the process of model deployment, particularly in urgent scenarios such as healthcare sector. This efficiency instills confidence in the reliability and effectiveness of TL in time-sensitive situations.

Applications of TL

TL becomes a ubiquitous technique across various domains, including:

1. **Software Engineering:** Task include development of defect prediction model, change prediction model [4], effort estimation [5, 6] used TL.

2. **Computer Vision:** Tasks include object detection, image segmentation, and visual question answering.

3. **Natural Language Processing:** Tasks include sentiment analysis, classification, and machine translation.

4. **Speech Recognition:** Tasks include speaker identification, emotion recognition, and automatic speech transcription. By harnessing the power of TL, researchers and practitioners can address the challenges associated with limited data and accelerate the development of effective ML models across diverse domains.

1.5 Literature Survey

The previous section discussed the importance of developing good quality software using predictive modeling. However, researchers, academicians, and industry experts are working hard to design good quality software considering the reusability of existing software's in the future. Thus, many researchers integrated the idea of cross-project with the prediction model. However, TL is one of the cross-project methodologies that efficiently utilizes existing software for future projects.

Systematic review plays a vital role in developing the solution for existing limitations regarding research gaps in the respective domain. This section discusses the cross-project methodologies explored in the existing literature. Further, considering the existing studies, the exploration of TL for SQPM is concerned.

1.5.1 Software Defect Prediction

Defect detection and removal of defects before final model deployment is an essential step in software engineering [7]. The performance of the SDP model is affected by many factors such as dataset, Feature Selection (FS) technique, and ML technique. Developers created a two-stage prediction model to detect defective modules [8]. A novel method is proposed to locate concept drift points for defect prediction [9]. The researchers concluded that they could predict the model's instability with a certain degree of accuracy without labeling newly entered data.

Researchers employed traditional ML methods, including Decision Tree (DT), Naive Bayes (NB), and Support Vector Machine (SVM), to analyze Genetic Algorithms (GA). Various performance measures such as precision, accuracy, recall, matthew's correlation coefficient, F-measure, and receiver operating characteristics are used. The applicability of GA is also analyzed for the selection of relevant features in SDP [10]. A five-stage framework developed [11] for SDP by selecting datasets and feature optimization using GA. Three ML classifiers are employed, followed by an ensemble voting technique to enhance predictive power. In the existing research, the authors focused on developing interpretable SDP models to aid test managers in identifying defect-prone modules.

A novel approach is designed for SDP using federated learning to remove the issues of data privacy and information security. The federated learning-based defect prediction model is deployed on docker, and performance is analyzed using open-source tools. The effect of dataset size, complexity, characteristics, language, and features such as metrics analyzed for SDP [12]. The reliability of cross-project prediction is also analyzed. It investigates the influence of size, complexity, and other code metrics on defect prediction

and evaluates the reliability of cross-project prediction. The models use ML techniques on PROMISE datasets to demonstrate interpretability through SHapley Additive exPlanations (SHAP) and Local Interpretable Model-Agnostic Explanations (LIME) techniques. The result showed that the contribution of static code metrics is high for prediction models, and the incorporation of explainability enhances the model's outcome. An SDP-CMPOA framework [13] was developed for SDP to improve the accuracy and aiming to improve the performance, and efficiency of defect prediction models. CMPOA focuses on addressing the dimensionality reduction through FS method. The authors utilize hybrid classifiers such as Improved Bi-Long Short Term Memory (LSTM) and Deep Max, as demonstrated on the SDP dataset. The effectiveness of SDP-CMPOA is evaluated against existing schemes across various measures, showcasing its efficacy in SDP. However, the authors analyzed the class imbalance issue with defect prediction by developing the Dual Ensemble SDP model (DE-SDP) [14]. Combined with a diverse ensemble, the Neural Network (NN) also improvises the performance. The learning-based approaches are explored for SDP considering FS, ML techniques, and performance measures in existing study [15]. In the existing studies, authors developed a novel approach for SDP that combined mutual information and Correlation Coefficient Weighted Class Association Rule Mining named CWCARM. However, CWCARM employs a cost-sensitive strategy to generate itemsets for determining the combination through mutual information weighted support [16].

1.5.2 Software Change Prediction

Software Change Prediction (SCP) focuses on predicting changes in software requirements for small-scale systems. It employs a probabilistic model integrating stakeholder input

via questionnaires into a Bayesian Network (BN) [17]. The authors also addressed the issues in industrial organizations due to limited dataset availability and small dataset size [18]. Due to this, it is only feasible to test some of the models through limited resources. We should actively predict specific attributes in advance, including whether defects are present or absent, the likelihood of changes required, and the necessary maintenance and effort. Early prediction of changes helps developers, allowing them to anticipate changes when resources are limited and likely to be affected. The researchers [19] have concluded in the existing study that more studies should be conducted for SCP considering projects developed in C++/Java/Python language. The existing models, utilizing the variable elimination method, accurately forecast the likelihood of revisions in the requirements document. However, the evaluation of the prediction model, including probabilities of low state revision, is 0.42, and high state revisions are 0.45. The performance of SCP can be utilized by integrating estimated change sets from impact analysis technique [20]. Existing studies introduce a framework to combine these approaches to improve cross-project capabilities.

1.5.3 Validation Methodologies for SQPM

1.5.3.1 Cross-Company Defect Prediction

The cross-company prediction [21] also plays a vital role in improving the quality of future software's. TL is explored by the researchers for cross-company prediction models, and SDP methods considered public datasets that belong to various companies. Thus, a modified NB algorithm developed using TL that leverage knowledge from one company's dataset to improve the performance of future datasets from different companies. In case

of cross-company [22], the company must use neighboring company data if a sufficient amount of data is unavailable. Furthermore, the exploration of Peter filter for cross-company prediction model, and its performance analysis done in the literature [23]. The approach is practical when scarcity of data exists in the nearest dataset; then, other company datasets are used to develop the prediction model. Metrics representation and TL are also explored for cross-company [24, 25].

1.5.3.2 Cross-Project Defect Prediction (CPDP)

The existing study conducted by the authors [26] reduced the gap between source and target datasets. Multisource Heterogeneous CPDP developed a multi-source TL algorithm to minimize the impact of negative transfers and upgrade the classifier's Performance. The authors used five datasets to evaluate the performance of MHCPDP. However, the researchers have provided a solution to multi-objective learning. The authors [27] developed a novel Multi-Objective NB (MONB) algorithm based on the Harmony Search Optimization (HSO) algorithm. The concept of clustering-based FS method is also integrated for cross-project [28]. Further, the parameters [29] influenced or affected the performance of CPDP models are identified.

The feasibility [30] of HetDP is also analyzed, considering various performance measures. The predictive performance of HetDP is compared with the model developed using TL. Furthermore, the cross-project model is developed using feature type TL. The learning is completed through features of two projects [31]. A two-phase TL approach for CPDP is developed [32]. However, the TL-based model aims to reduce the difference in data distribution among source and target datasets, enhancing prediction accuracy across different software projects. In the existing studies, the authors introduced a TL method that

considers correlation features and instance weights for improved defect prediction across diverse software projects [33]. The approach enhances prediction accuracy by leveraging knowledge from related projects. A novel CPSDP algorithm based on TL presents an innovative TL algorithm for CPDP. This approach utilizes TL techniques to effectively leverage understanding from source projects, enhancing prediction performance in target projects with limited data [34]. The problem of class imbalance is addressed alongwith data distribution through the proposed algorithm (TSboostDF) [35].

The handcrafted features and the semantic features mined from the Abstract Syntax Tree (AST) are joined with handcrafted features to form the joint features for CPDP [36]. The model was evaluated on 110 models developed with 11 open-source projects, and the proposed model outperformed the existing deep learning approach. The proposed end-to-end framework for SDP bypasses the need for feature extraction tools by visualizing programs such as images, applying a self-attention mechanism, TL, and utilizing pre-trained deep learning models [37]. Experiments demonstrate its efficacy in enhancing cross-project and Within-Project Defect Prediction (WPDP) across ten projects from the PROMISE dataset. The Adversarial Discriminative Convolutional Neural Network (ADCNN) addresses cross-project limitations by semantic feature extraction from source code [38]. However, ADCNN outperformed existing approaches in terms of F-measure, Are Under Curve (AUC), and PofB20 across ten benchmark projects.

1.5.3.3 Cross-Project Change Prediction

The cross-project change proneness prediction model is developed using ten datasets of Eclipse plugins. A GA-based technique was developed for cross-project. Random Forest (RF) performed best in within-project settings. Non-Ensemble Decision Trees (NEDT)

performed best for CP. After detailed analysis, it was observed that CP, alongwith GA, performed best, followed by WP.

1.5.3.4 TL for Defect Prediction

To improve the quality of software [39] using predictive modeling, the problem of data redundancy and data distribution differences are addressed by considering AUC and F1-score measures. In the existing study, the authors [40] showed the effectiveness of TL for CPDP and proposed Three Stage Weighting frameworks for Multi-Source TL. Thus, bellwether and weighted vote worked efficiently to select the source dataset, and 3SW-MSTL exists for four multi-source, single-source CPDP methods. Moreover, due to the differences in data distribution of different projects, the Performance of cross-project prediction models could be improved. Thus, the authors applied a state-of-the-art algorithm, Transfer Component Analysis (TCA), to differentiate between the features of the source and the target dataset. ; a novel TL algorithm was developed [41] by the researcher's named TCA+ (extended version of traditional TCA). Thus, it is observed that TCA+ performed better than TCA to improve the Performance of CPs for future or unseen data.

The problem of an imbalanced dataset alongwith cross-project is also addressed by TL [42]. A cost-sensitive boosting method is designed for CP, assigning weights to the instances considering class imbalance and data distribution issues. Thus, the cost-sensitive boosting algorithm outperformed other models. The prediction model's was developed using the cost-sensitive approach with TL, improving the software quality in reasonable cost and time. Earlier, TL [43–45] explored for defect prediction. However, the authors also analyzed the capability of effort prediction models for cross-company. Moreover, the fact that the organization's older data needs to be more relevant is not true. A survey on TL

is conducted to analyze its performance, capabilities, and work in the existing literature by eminent experts [46]. A Bagging Based TL (BETL) [47] algorithm was developed for cross-project. The BETL algorithm consists of three stages: Initiate, Update, and Integrate. However, the performance of the developed BETL algorithm is analyzed considering the University of California, Irvine (UCI) dataset, real-world data set, and text data set. Thus, the authors concluded that the BETL method can effectively label the unlabeled data in the target domain, which significantly enhances the performance of the target domain. TL combined with Low-Shot Classifier (TLLSC) for SDP [48]. The comparison showed that TLLSC outperformed in comparison to TCA+, Canonical Correlation Analysis Extended (CCA+), and Kernel CCA+ (KCCA+).

1.6 Vision and Focus of the Thesis

The prime vision of the work is to improve the performance of SQPM using various cross-project methodologies.

The thesis focuses on validating cross-project methodologies for improving the software quality of a prediction model. Development of effective cross-project prediction models for open-source software systems in this regard. This thesis endeavors to research predictive modeling using OO Metrics (OOM), Halstead metrics, different FS techniques, ML techniques, and TL. Keeping in mind the challenges of cross-project while developing the prediction models, the current study carefully designed and validated existing methods for producing efficient software in the future using cross-project. Thus, this study explicitly addressed the cross-project methodologies.

1.7 Objectives of the Thesis

Aggarwal and Singh [1] have emphasized the importance of efficient software by assuring the characteristics of good quality software. Developing an efficient, good-quality software system requires internal and external software characteristics. Also, the literature survey conducted by the researchers showed that the cost of producing efficient software prediction models has been increasing daily. It motivates researchers to think about how we can ensure software quality before its deployment at the customer site. Thus, the existing software with similar characteristics can be used in the future in order to reduce cost, time, and effort while at the same time increasing the reusability of software. Concerning the above problem, this thesis aims to develop effective prediction models to ensure a good quality of cross-project and efficient utilization of resources by employing cross-project. The summary of the objectives is as follows:

1. Perform extensive existing literature survey to study TL in cross-project with the following objectives:

    - Study of existing research publications that would help in understanding the process and procedure of TL. The systematic literature review also helps to understand the transfer of knowledge using the developed model.
    - An extensive study of existing literature to understand the concept of a prediction model using TL.
    - Study of various papers that used ML techniques as base learners in literature and their comparison with TL algorithms.

- The literature review also helps to identify various statistical test and validation methods used for TL.

2. Collection of datasets from different open-source software with the following objectives:

   - Collection of datasets from different projects for cross-project validation techniques.
   - The quality of projects enhanced by using the knowledge of similar projects.

3. Development of prediction models using CPV techniques

   - Development of prediction models for SQPM.
   - Development of new models for enhancing the quality of cross-project and cross-company software using various cross-validation techniques.
   - Development of efficient and accurate fault prediction model.
   - Development prediction models using TL to identify the defect and change-prone parts.
   - Development of predictive model using inter-version of projects. Explore different releases of the same project for prediction model development.
   - Development of prediction model to produce generalized results using Inter-project validation.
   - Exploration of Inter-project validation across different versions of a single software.

- Exploration of Inter-project validation across different software of similar nature i.e., build using the same programming language in the same interface, performing the same task but with different accuracy.

4. Exploring TL for SQPM.

   - Development of new techniques for SQPM using TL concept.
   - Identification of defect proneness and change-prone parts of cross-project using TL.
   - Using different statistical tests for validation of techniques developed for SQPM.
   - Explore feature-based and parameter-based TL for SQPM.
   - Combining software metrics as features for knowledge transfer.
   - Analyze effectiveness of techniques for identifying defects in cross-project with the help of various performance measures and statistical tests.

1.8 Overview of the Work

The main focus of the research done for this study is to improve the quality of cross-project models using TL. The concept of cross-project came due to the inability to provide sufficient data for training and testing in software engineering; the main focus is to produce good quality software by considering all the quality attributes of the software. TL plays a significant role in case of non-availability of sufficient data for training and testing. Following the guidelines of Kitchenham [49], we conducted a comprehensive systematic

literature review to thoroughly understand how existing research employs cross-project methodology, such as TL, in software engineering to improve software quality. TL is an ML technique where a model trained on one task is reused or adapted as the starting point as a model on second related task. When it comes to SQPM, TL offers several significant advantages. The prime aim of conducting the review is to understand the contribution of TL in the software engineering domain from the following aspects:

- The use of TL in software engineering.
- Study the usage of TL and its procedure in software engineering.
- To examine the dataset used in existing literature.
- To examine the performance of TL algorithms in software engineering.
- To examine the types of TL used in the existing literature.
- An extensive study of existing literature to understand the concept of a prediction model using TL.
- Study of existing literature concerning different software engineering domains.
- Study of existing literature that used ML techniques as base learners in literature and their comparison with TL algorithms.
- To examine the advantages of TL in software engineering.
- To examine the significance of TL in comparison to WP models.

- To examine the performance of TL algorithms in comparison to conventional ML algorithms.

According to [50], the next step is to extract relevant studies from various digital libraries with the help of a search string. Different search strings are combined to extract studies related to the topic. Furthermore, search strings are formed to extract empirical studies from libraries to extract relevant research papers. Systematically analyzing the data extracted from the studies, Research Questions (RQs) are answered.

The CPDP model is developed using TL. The predictive capability of ML algorithms is analyzed using an open-source dataset from the repository. Thus, the dataset is collected through Understand tool and dataset is prepared from the beginning for change prediction. Further, the performance of ML techniques is analyzed for CPDP and WPDP. The performance analysis of FS techniques for CPDP and WPDP is illustrated. A comparison is performed between filter, wrapper, and hybrid methods for FS in CPDP and WPDP. Thus, the authors analyze the search-based or embedded methods performed best for CPDP. The performance of techniques using WPDP, cross-validation, and 10-fold cross–validation methods is also analyzed. The search-based techniques, such as Grey Wolf Optimization (GWO) technique is analyzed with existing techniques. The performance of ANN, CNN, and RNN is also analyzed. In this work, the improvised ANN-based GWO CPDP model is developed, which performs better than traditional ML algorithm. Further, the experimental setup, methodology, and results are explained in detail from Chapter 2 to Chapter 9.

The analysis proved that TL performed at par with existing techniques. The Performance of the prediction model developed with and without TL was analyzed with different traditional datasets and open-source datasets. It aims to explore feature type TL for CPDP and CPCP to validate cross-project methodologies. Considering the availability of data and

overfitting issues in the industry, the model developed using feature type TL performs at par with models developed using traditional techniques. TL considers datasets of a similar domain. The models developed using modified TL algorithms and traditional algorithms also conclude that TL is effective for development of efficient prediction models.

1.9 Organization of the Thesis

This section presents the organization of the thesis. **Chapter 1** presents basic concepts of SQPM, cross-project methodologies, and TL in software engineering with the motivation of the thesis. **Chapter 2** discusses the research methodology followed in order to conduct research work. **Chapter 3** discusses a systematic review of existing studies and prevailing research gaps. **Chapter 4** presents the development of a defect-prediction model using feature type TL. **Chapter 5** discusses the validation of ML techniques for change prediction models develop using open-source dataset. Furthermore, **Chapter 6** presents the validation of FS techniques for cross-project models and compared the model performance using statistical techniques. Furthermore, **Chapter 7** develop defect prediction models to analyze the performance of cross-version and 10-fold cross-validation. **Chapter 8** presents Grid-Search ANN (GrANN) for heterogeneous cross-project defect prediction. Furthermore, **Chapter 9** presents improvised grid based GWO technique for SDP and develop an improvised CPDP model using ANN, CNN, and RNN with TL algorithms. **Chapter 10** summarizes the conclusion of the thesis. The brief description of each chapter is given below.

Chapter 2: The detailed description of the research methodology followed in this research work is discussed in this chapter alongwith a brief explanation of the dataset,

dependent and independent variables, cross–project, imbalanced dataset, open-source dataset, inter-version, 10-fold cross-validation, TL, ML techniques, data analysis methods, validation techniques used, performance measures used, data preprocessing steps, statistical test used, metrics matching analyzer, steps for the development of prediction model, TL techniques for software engineering domain are discussed. Furthermore, the effect of different FS techniques for cross–project, and optimized algorithms for cross-project prediction models are discussed in this chapter.

Chapter 3: A systematic review of work related to cross-project methodologies since 1991 to 2023 is presented in this chapter. However, RQs were formed to address various concerns related to SQPM, and cross-project methodologies. This chapter discusses the procedure of conducting systematic review in detail alongwith number of studies published, data extracted from collected studies, and studies collected after quality assessment criterion. The dataset, ML techniques, the independent and dependent variables, validation techniques, performance measures used for TL, statistical tests, and categories of TL are discussed in detail. Furthermore, the review results are compiled, analyzed, and research gaps are identified.

Chapter 4: This chapter presents a defect prediction model developed using ML techniques. A defect prediction model was developed using NASA and PROMISE repository dataset. IdTL is used for model development with ML techniques. The results are validated using statistical testing. The impact of FS on CPDP and WPDP is presented. The application of TL to develop an enhanced version of the defect prediction model is discussed. By leveraging knowledge from source domains, the model aims to improve performance on target domains. The study delves into methodologies for transferring knowledge between related datasets to optimize predictive capabilities. Feature type TL is used to establish

relationship between features of different datasets. Thus, metrics matching and metrics analyzer are used on a specified threshold to extract relevant features.

Chapter 5: This chapter presents the framework for developing a software change prediction model using an open-source dataset. The experiment presents the results using a dataset collected through Understand tool. Dataset collected with subsequent versions of each software namely Notepad++, CodeLite, and CodeBlocks. The dataset is prepared to study the impact of ML techniques on software change prediction. The researchers have not explored change prediction in the existing studies using TL. Thus, the predictive capability and efficiency of ML techniques with Feature type TL were analyzed. Further, the results are validated using statistical tests such as Friedman and Nemenyi tests. Heterogeneous cross-project change prediction perform at par with Within-Project Change Prediction (WPCP).

Chapter 6: This chapter presents the analysis of FS techniques for CPDP. Experimentation performed using defect dataset in order to evaluate the effectiveness of filter, wrapper, and swarm search based methods. In the previous work, it was concluded that FS plays important role in development of CPDP with TL. Thus, CPDP models are developed using filter methods such as ChiSquare, Correlation Attribute Evaluation, Gain Ratio, Information Gain, Relief Attribute Selection. Wrapper method used for predictive capability of CPDP models. Further, swarm search based methods such as Best First Search, Genetic Search, Greedy Stepwise Search, Harmony Search, Particle Swarm Optimization, Scatter Search used for development of CPDP models. The performance analyzed with development of CPDP models using feature type TL considering specified source and target dataset. The performance of developed models validated using statistical test.

Chapter 7: This chapter evaluates the effectiveness of cross-version and 10-fold

cross-validation for defect prediction using traditional ML techniques. The prediction model was developed with 14 different datasets of PROMISE with seven traditional ML techniques such as NB, Logistic Regression (LR), Multilayer perceptron (MLP), ADaBoost (ADB), Bagging (BNG), J8(DT), and RF. This research aims to analyze various datasets to compare the performance of these techniques, potentially shedding light on their strengths and limitations in practical software engineering scenarios. The study improves defect prediction methodologies by providing empirical evidence on the efficacy of cross-version and 10-fold cross-validation.

Chapter 8: This chapter presents an improvised technique based on swarm search methodology named Grid Search-Optimized Artificial Neural Network (ANN) for Het-CPDP using NASA and AEEEM datasets. In this research work, the aim was to improve the performance of the defect prediction model using the grid search warm search method. The performance of the model developed using traditional ANN is compared with the model developed using Particle Swarm Optimization (PSO), GrS, and evolutionary search using the AUC measure. The results are validated using a statistical test such as Friedman and Wilcoxon–signed rank test.

Chapter 9: This chapter presents an improvised CPDP model using GWO and ANN. Prediction model developed using NASA, and AEEEM dataset. The focus of this work is to improvise the performance of prediction models across diverse software projects. Model developed using GWO such as WPDP_GWO, 10-fold_GWO, and GSGWO_WPDP. The performance of TL-based algorithms is analyzed. Transfer GWO (TrGWO), GSGWO, and RFGWO were compared using AUC measure. The improvised CPDP model developed using ANN is also presented. The model developed using ANN is compared with the models developed transfer ANN, transfer ANNSMO, transfer ANNCSO, transfer ANNACO,

transfer RNN, transfer RNNSMO, transfer RNNCSO, transfer RNNACO, transfer CNN, transfer CNNSMO, transfer CNNCSO, transfer CNNACO. The results are validated using statistical test.

Chapter 10: This chapter summarizes conclusion of the research work and provides future directions.

Chapter 2

Research Methodology

2.1 Introduction

A systematic procedure needs to be followed to conduct effective research. We have followed systematic and sequential methodology to conduct empirical research with different datasets and techniques in this research. The main objective of using a specified method is to obtain an effective and efficient solution for the existing problem statement. In this chapter, the research methodology followed is discussed in detail. The organization of this chapter is as follows: Section 2.2 discusses the research process followed for completing the research work. Section 2.3 presents the definition of the research problem in more detail. Section 2.4 summarized the literature review conducted to complete the research work. Section 2.5 discusses the explanation of variables used in this section. Section 2.6 discussed data analysis methods or ML techniques for developing prediction models. Section 2.7 discussed the experimental framework followed in experimenting with

this research work, including the data collection procedure, dataset used, cross-validation methods, statistical tests, and performance measures.

2.2 Research Process

A systematic and prearranged series of actions necessary for investigating a research problem comprise the research process. The research methodology used in this thesis chapters is described in Figure 2.1. The following sections provide an explanation of each step in the research process.

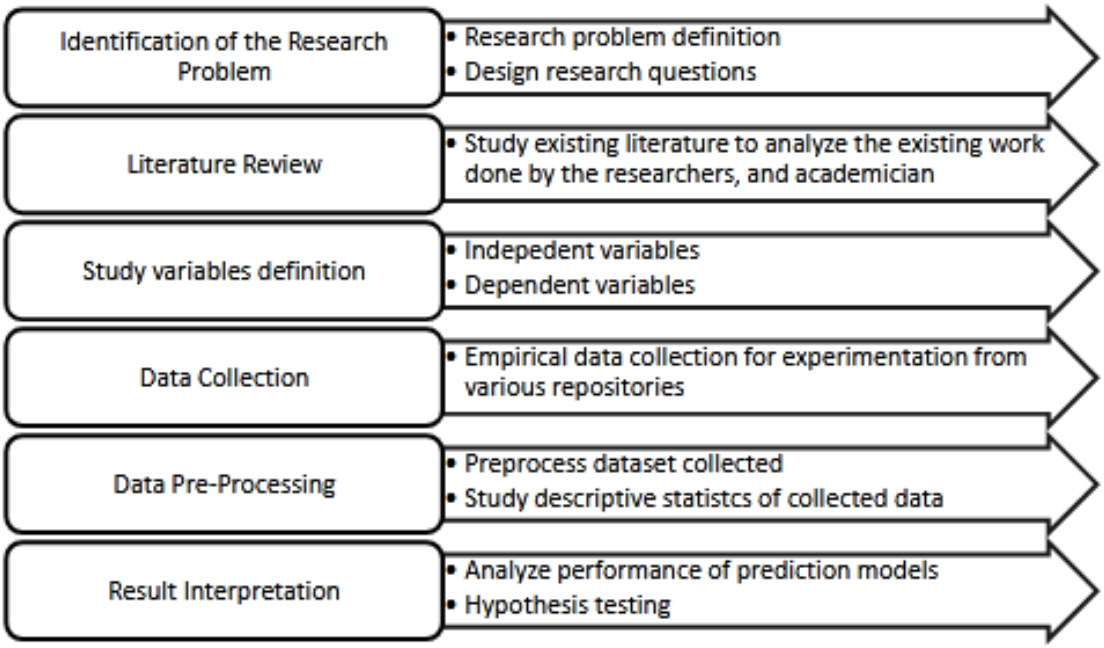

Figure 2.1: Research Process

2.3 Define Research Problem

In the research process, the primary step focused on the formulation of a research problem. The research problems are stated in the form of RQs. Furthermore, the experiments conducted in this work to determine the answers to RQs. The RQs answered in this thesis are as follows:

1. What is the current state of literature studies in the domain of TL in software engineering field?
2. What is the predictive capability of Cross-Project Change Prediction (CPCP) and CPDP models for open source dataset using TL?
3. What is the performance of cross-project prediction models using TL with various FS techniques?
4. What is the predictive capability of cross-project prediction models using optimization algorithms with TL algorithms?

2.4 Literature Survey

To comprehend the research problem, it is imperative to conduct a comprehensive review of previous studies through a literature survey. The review provides insight into the extent to which the research problem in the literature has been explored. However, in the existing literature [51–53] various prediction models were developed for identification of defect prone and change prone part of a project. These models have been developed

by considering the existing projects of similar characteristics, and existing version of the same project [54–56]. Thus, the relationship between the independent variables of available projects is established to develop prediction models for future projects. Hence, it enhances the reusability of existing projects in this way. The models developed using open-source dataset of various repositories. These prediction models help software developers, academicians, and industry experts to create efficient models for future unseen data using existing project data for training. It will help developers in reducing the defect prone and change prone part in the initial stage of software development life cycle models. Further, it is also effective in efficient utilization of available resources optimally to the software components with low maintainability. Therefore, it has been well recognized in the literature that developing effective software prediction models is crucial to improve software quality. Thus, it is concluded in the existing literature that development of effective CPDP is important, and it improves the software quality in various aspects such as functionality, maintainability, testability, reliability, and robustness.

2.5 Defining Variables

An empirical research involves two types of variables: the independent variable termed as predictor variable or input variables and the dependent variable are termed as response variable or output variables. The dependent variable is a variable that captures the researcher's interest. The research focuses on software change or defect prediction as the dependent variable. The dependent variable is affected by the independent factors, often known as predictor variables. The outcome variable can be forecasted based on the predictor variables. The independent variable must possess true independence, meaning it should be

unaffected by other variables. This thesis aims to construct CPDP and CPCP models that acquire knowledge of the independent variables of different projects in order to improve the software quality from the initial stage. The research work utilizes OOM as independent variables to reflect the diverse aspects of the software classes.

2.5.1 Independent Variables (OOM)

The literature has widely recognized the usefulness of OOM and Halstead metrics in constructing models for forecasting defective classes, change-prone classes, and reusability of a software.

The quality elements of a software considered such as cohesion, coupling, inheritance, encapsulation, and abstraction can be measured through OOM. Thus, it is important to carefully select the identifiers from the software/projects used for experimentation. The OOM used in the thesis includes the following:

- Chidamber and Kemerer (C&K) metric suite [57] contains six OO metrics, namely WMC, DIT, CBO, and RFC. C&K metrics are widely used in the literature [58–60].
- Quality Model for Object-Oriented Design (QMOOD) [61] metric suite is also validated in the thesis. The metrics contained in QMOOD metric suite are MOA, DAM, MFA, NPM, and CAM.
- Martin metrics [62] namely Ce and Ca have also been analyzed in the thesis. Other metrics used in the thesis are AMC, SLOC, LCOM3 (given by [63]), CBM, and IC.

Table 2.1: Independent Variables

Metric	**Definition**	**Source**
FAN IN	Maximum number of input parameters.	[57]
FAN OUT	Maximum number of output parameters.	[57]
Lack of COhesion in Methods (LCOM)	Number of methods in a class that is not related to themselves by sharing some of the class data.	[57]
Number of Attributes (NOA)	Count of attributes.	[57]
NOA Inherited (NOAI)	Count of attributes inherited from parent class.	[57]
Source Code Metric (SCM)	Metrics computed from source code.	[57]
Number of Methods (NOM)	Count of Methods.	[57]
NOM Inherited	Count of methods inherited from parent class.	[57]
Number of Private Attributes (NPrA)	Count of the number of private attributes defined in a class.	[57]
Number of Private Methods (NPrM)	Count of the number of private methods defined in a class	[57]
Number of Public Attributes (NPuA)	Count of the number of public attributes defined in a class.	[57]
Entropy of Change Metric (EOCM)	Artificial metrics computed from entropy.	[2]
Inheritance Coupling	Number of parent classes to which given class is coupled is called inheritance coupling of that class.	[2]

Branch Count (BRC)	Sum of branch count in the software.	[2]
Blank Line	Sum of blank lines in each module.	[2]
Cyclomatic Complexity (*v*(g))	Cyclomatic complexity of each module. $$v(g) = E - N + 2P \quad (2.1)$$ Here E= number of edges; N = number of nodes and P = number of connected components.	[2]
Design Complexity (*iv*(g))	Design complexity of every module.	[2]
Essential Complexity(*ev*(g))	Essential complexity of each module.	[2]
Num Operands (Nopn)	In Halstead metrics, n_2 is the count of unique operands, and N_2 is the count of the total occurrence of operands.	[2]
Num Operators (Nopt)	In Halstead metrics, n_1 is the count of unique operators, and N_1 is the count of total occurrences of operators.	[2]
Halstead Difficulty (*D*)	Ratio of the number of unique operators to the total number of operators in the program. $$D = \left(\frac{n_1}{2}\right) \times \left(\frac{N_2}{n_2}\right) \quad (2.2)$$	[2]
Comment and Code Line	Line of comments in each module.	[2]
Halstead Length (Le)	Sum of number of operator and operand occurrences in the program.	[2]

Response for Class (RFC)	The number of functions executed in response to a message received by an object of a class is called response for class.	[57]
Halstead program Time (T)	This is the estimated time required to implement the program, based on the program effort (E) and a constant value that depends on the programming language and development environment.	[2]
Halstead Volume(V)	Volume of each module. $$V = N * \log_2(n) \quad (2.3)$$ Here, N is program length, and n is vocabulary size.	[2]
Halstead Effort (B)	Measures the mental activity needed to translate the existing algorithm into implementation in the specified program language. $$E = \frac{V}{L} \quad (2.4)$$ Here, V is program volume and L is program level.	[2]
Line count	Quantitative counting of the source code, such as count of all lines.	[2]
Depth of Inheritance Tree (DIT)	It is defined as the length of the longest path in the inheritance hierarchy.	[57]

Number of Children (NOC)	It is defined as the number of immediately derived classes of a particular class.	[57]
Weighted Method per Class (WMC)	The sum of cyclomatic complexities of all methods of a class is called weighted method per class. $$WMC = \frac{\sum_{i=1}^{n} \text{TM}_i \times \text{Complexity}_i}{\text{Total Number of Methods}} \quad (2.5)$$ Here, TM is a number of methods.	[57]
Coupling between Objects (CBO)	This metrics shows the number of classes to which a specific class is coupled where the coupling can be due to data accesses, function calls, inheritance etc.	[57]
Afferent Coupling (Ca)	This metrics measures merely the number of classes that use a particular class.	[62]
Efferent Coupling (Ce)	Number of classes used by a specific class is called efferent coupling of that class.	[62]
Number of Public Methods (NPuM)	This is the count of the number of public methods defined in a class.	[61]
Data Access Metrics (DAM)	It is described as the number of protected or private attributes declared in a class divided by total number of attributes declared in that class.	[61]

Measure of Aggregation (MOA)	This metrics is a count of number of data fields in a class whose type is user-defined.	[61]
Measure of Functional Abstraction (MFA)	This metric defines the number of methods that are inherited by a specific class divided by the sum of methods accessible by that class.	[61]
Cohesion Among Methods of Class (CAM)	This metrics measure the relationship amongst the class methods. The association is found by their list of arguments and defined as: the sum of the number of unique argument types used by all of the class methods divided by product of total number of methods in the class and total count of unique argument type in that class.	[61]
Lines of Source code (SLOC)	It is defined as total number of lines in the binary code of a class.	[57]
Lack of cohesion (LCOM3)	LCOM3 is given as: $$LCOM3 = \frac{1}{n}(\sum_{i}^{n} f(pi)) - \frac{ma}{1-ma} \quad (2.6)$$ Here n= number of attributes in a class; ma = number of methods in a class and $f(pi)$ = number of functions that access an attribute.	[63]

These metrics describe different aspects of OO systems namely cohesion, coupling, size, inheritance, composition, and encapsulation etc. The metrics WMC, NPM, LOC, DAM, and AMC are indicators of the size of a class. The metrics CBO, RFC, Ca, Ce, IC, and CBM measure the coupling. The inheritance property is measured with the help of NOC, DIT, and MFC. The metrics LCOM, CAM, and LCOM3 are indicators of class cohesion, whereas MOA measures composition. The metrics that quantify the different characteristics of a class are regarded as internal quality attributes. The internal quality attributes used have significant relation with software maintainability [64–66]. For instance, the attributes WMC, NPM, LOC, DAM, and AMC measures the size of a class. If the size increases, code would likely be less maintainable, i.e., likely to require high maintainability effort [64]. Table 2.1 presented brief explanation of the above OO metrics. These metrics have been widely used by the researchers for predictive modeling in the domain of software engineering [60, 67–70].

2.5.2 Dependent Variable

In this thesis, software defect and software change proneness is the dependent variable. It is a binary variable i.e., there will be two values of software defect and change prone parts either Yes (1) or No (0).

2.6 Data Analysis Methods

In this thesis, various data analysis methods were used including ML, statistical, and optimization based methods. ML techniques were used in order to model the relationship between variables of different projects without relying on pre–determined results of a

model. In this thesis, the experiments used various ML classifiers such as LR, RF, Hybrid Voting (HV), MLP, ADB, DT, SVM, *K*-Nearest Neighbors (KNN), Quadratic Discriminant Analysis (QDA), Stochastic Gradient Descent (SGD), Sequential Minimal Optimization (SMO), BNG, LogiBoost (LB), NB, J48, ANN, CNN, RNN. Statistical methods focused on modelling the relationship between independent and dependent variables using mathematical concept such as mean, median, variance, and standard deviation of the features. The optimization techniques are used to optimize the results of the prediction model using specified domain. However, the models developed using different data analysis methods is further used for making predictions on unseen dataset in upcoming future. The data analysis methods are further discussed in more detail in Chapter 4, 5, 6, 7, 8, and 9.

2.7 Experimental Design Framework

The diagrammatic representation of the experimental designNis presented in Fig. 2.2. The various steps are described in detail in the following sections and subsequent chapter.

2.7.1 Empirical Data Collection

In this thesis, the data collection is performed through empirical procedure. Empirical procedure is a systematic way and process to gather data from various sources. The empirical data is used to analyze the experiment results and obtain answers to RQs. The data collected from different sources such as open–source software, industrial software, proprietary software, and academic projects. In this thesis, the dataset is collected from open–source repositories and open source software for validation of CPDP methodologies. The open-source projects dataset is used to validate existing CPDP methodologies for

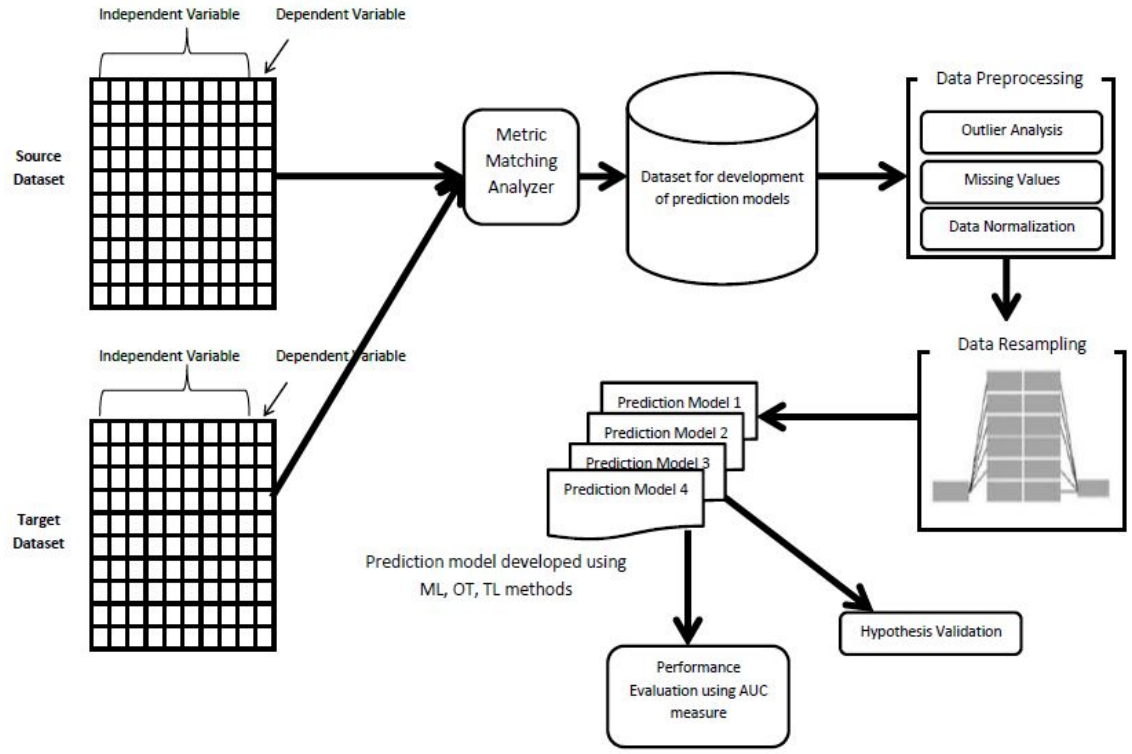

Figure 2.2: Experimental design for developing SDP Models

future project reusability. Moreover, in the existing research work, cross-project prediction models are not analyzed for change prediction, due to which our work emphasize on validation of CPDP models as well. The empirical defect dataset is also used in this thesis for development of CPCP model using TL techniques. Defect dataset can be easily downloaded from open source repository and GitHub. The software used for data collection from open–source software is discussed in this section. The open-source software is available at `https://scitools.com/`.

2.7.2 Metric Matching Analyzer

A metric matching analyzer is used in TL for software engineering aligns the metrics of a pre-trained model with the metrics of the target domain. This procedure facilitates the customization of the pre-trained model to suit the specific requirements and attributes of the new domain, hence improving its performance and efficacy in tasks like classification, regression, or clustering. By aligning the measurements, the model can enhance its ability to comprehend and extrapolate patterns in the new domain, promoting a more efficient transfer of information and learning.

2.7.3 Data Collection Procedure

The thesis considered two dataset types: defect dataset and software change dataset. Defect prediction models are developed using existing datasets such as NASA, AEEEM, PROMISE, and ReLink. Change prediction models are collected using open-source datasets collected through Understand tool.

2.7.3.1 Software Defect Dataset

The datasets are collected from NASA and PROMISE group. The dataset consists of independent variables, such as software metrics and process metrics. These variables are required for the training of a model and to complete the predictive model task. These software metrics consist of McCabe's cyclomatic metrics, C&K metrics, and other OOM. We have used eleven datasets of NASA and one dataset of the PROMISE group. All of these datasets have a different number of independent variables. These software metrics are characterized based on different measures such as cohesion, inheritance, coupling, the

complexity of a dataset, and LOC of a particular software.

We have used publicly available datasets on Github. NASA contains confidential datasets from NASA software company [30]. The PROMISE group comprise of defect datasets. These datasets are used for various free open-source software projects exist in various studies [30]. There are total 21 features in the PROMISE group dataset.

Table 2.2: Summary of NASA dataset

Group	**Dataset name**	**Total no. of instances**	**Defective Instances**	**No. of metrics**	**Granularity**
NASA	CM1	498	49 (9.84%)	21	Function
	KC1	2109	326 (15.46%)	21	
	KC2	522	107 (20.50%)	21	
	KC3	194	36 (18.56%)	39	
	MC1	1988	46 (2.31%)	38	
	MC2	125	44 (35.20%)	39	
	MW1	253	27 (10.67%)	37	
	PC1	1109	77 (6.94%)	21	
	PC2	745	16 (2.15%)	36	
	PC3	1077	134 (12.44%)	37	
	PC4	1458	178 (12.21%)	37	

Table 2.3: Summary of AEEEM dataset

S.No.	**Project**	**Type**	**No. of files**	**% of buggy files**	**No. of metrics**
1.	Equinox (P1_EQ)	OSGi framework	325	36.69	71
2.	Eclipse JDT Core (P2_JDT)	IDE development	997	20.66	71
3.	Apache Lucene (P3_LUC)	Search engine library	39,691	9.26	71
4.	Mylyn (P4_MYL)	Task management	1862	13.16	71
5.	Eclipse PDE UI (P5_PDE)	IDE development	1492	14.01	71

Table 2.4: Summary of ReLink dataset

S.No.	Project	Type	No. of files	% of buggy files	No. of metrics
1.	Apace (P6_AP)	Web Server	194	50.52	26
2.	Safe (P7_SA)	Security	56	39.29	26
3.	ZXing (P8_ZX)	Bar-code reader library	399	29.57	26

Table 2.5: Descriptive Statistics of PROMISE dataset

Project	Number of Modules	Number of Defective Modules	% of Defective Modules
ant_ver1.6	352	92	26.10%
ant_ver1.7	745	166	22.30%
camel_ver1.0	329	13	3.95%
camel_ver1.2	608	216	35.50%
camel_ver1.4	872	145	16.60%
camel_ver1.6	965	188	19.50%
ivy_ver2.0	352	40	11.36%
jedit_ver3.2	272	90	33.08%
jedit_ver4.0	306	75	24.50%
jedit_ver4.1	312	79	25.30%
jedit_ver4.2	367	48	13.10%
jedit_ver4.3	492	11	2.20%
log4j_ver1.0	135	34	25.20%
log4j_ver1.1	109	37	33.90%
log4j_ver1.2	205	189	92.20%
lucene_ver2.0	195	91	46.70%
lucene_ver2.2	247	144	58.30%
lucene_ver2.4	340	203	59.70%
poi_ver2.5	385	248	64.42%
poi_ver3.0	442	281	63.60%
synapse_ver1.2	266	86	32.33%
xalan_ver2.4	723	110	15.20%
xalan_ver2.5	803	387	48.20%
xalan_ver2.6	885	411	46.40%

xalan_ver2.7	909	898	98.80%
xerces_ver1.4	573	426	74.35%

2.7.3.2 Software Change Dataset

The dataset collection has been accomplished by a tool such as Understand SciTools `https://scitools.com/`. The summary of change prediction dataset used presented in Table 2.6.

Table 2.6: Summary of dataset used

Software	**Source code programming language and license**	**Release/Version**	**No. of metrics**	**No. of classes**	**Number of changed and unchanged classes and files**
Notepad++	C++/ GPLv2	Notepad++ 6.8.9	10	696	129
		Notepad++ 7.3		682	114
		Notepad++ 7.5.4		698	88
		Notepad++ 7.6.2		748	84
		Notepad++ 7.6.3		749	23
		Notepad++ 7.8.3		473	750
CodeBlocks	C++/ GNU GPLv3	CodeBlocks 10.05	7	1213	158
		CodeBlocks 13.12		1320	748
		CodeBlocks 20.03		2256	1097
CodeLite	C++ and C/ GNU GPL	CodeLite-2.9.0.4684	10	1187	341
		CodeLite-3.5.5375		1337	343
		CodeLite-5.0.6213		1869	735
		CodeLite-5.3		1925	441
		CodeLite-6.0.1		2050	408
		CodeLite-11.0		2688	519
		CodeLite-12.0		2790	546
		CodeLite-13.0		1149	2837
		CodeLite-14.0		3954	2335

2.7.3.3 Descriptive Statistics

Understand is a static code analysis tool aimed to cover complete code navigation, control flow graph generation, generation of metrics for a project, code comparison, and code reengineering for an array of programming languages like C, C++, Python, JAVA, Perl, .NET, Jovial, and ADA. Understand tool is provided by 'Scientific Tools In.'. This tool has been used in many industries, aerospace, defense for analyzing legacy code.

Understand as a tool can be integrated with many other existing tools that researchers, academicians, software developers, and experts use. Understand tool uses static source code for complete analysis and then it generates the metrics for each project in the form of a .csv file. These metrics reports consist of various metrics. Metrics indicate the relationship between values. The OOM of the dataset used for experimentation is extracted for 18 versions of Notepad++, CodeBlocks and CodeLite presented in Table 2.6.

In the existing study [22], the relationship between OOM and the change proneness of a class is discussed. The relationship between OOM and change proneness capability of a class helps in efficient utilization of resources in maintenance and testing phase. With the help of Understand tool the OOM of different versions of Notepad++, CodeBlocks and CodeLite was generated. After analyzing each project version with an updated version in Understand tool window, we have checked whether a particular module is changed or not in each succeeding version. In the end, we have a final file for each project version, considering checking if there is an update or not in the current version compared with the existing version. The module which is having any change is labeled with '1' and the non-changed module is labeled with '0'.

The descriptive statistics of several OOM for each dataset. Descriptive statistics

are helpful for assessing the attributes of the datasets utilized for experimentation. The subsequent statistical measures are presented for OOM.

- Minimum: The minimum value of an OO measure represents the lowest value of that metric in the dataset.
- Maximum: In the context of an OO measure, maximum refers to the lowest value of the metric inside the dataset.
- Mean: The Mean represents the average value of an OO measure in the dataset. The mean is calculated by dividing the sum of the OOM values in the dataset by the number of data points in the dataset.

The change report consists of the following columns:

- Name of the source file, classes.
- OOM data for each class.
- A binary variable indicates whether it is changed (1) or unchanged (0).

The descriptive statistics of the OOM for each dataset used in this study are discussed in this section. Descriptive statistics is different from inferential statistics. The descriptive statistics are discussed in Table [5.4–5.6]. In the Notepad dataset, for LOC the maximum value ranges from 8032 to 9077. LOC plays a significant role in identifying change density. Change density specifies the number of changes that occurred over a particular number of LOC. In CodeBlocks, the minimum value of LOC is 1 and the maximum value of LOC ranges from 13722 to 25091. In CodeLite, the minimum value of LOC is 1 and the range

for a maximum value of LOC is 7003 to 12107. The number of changes decreases if LOC increases [71].

Change density measures the number of changes that occur per LOC set. In all the three datasets used in this study, the minimum value of FAN IN is 0. The maximum value of FAN IN is 3 for all the datasets used in this study. The count of a superclass for a specific class impacts the number of changes. If superclass contains changes, then there is a possibility that the derived class is also change-prone. In this way, the no. of changes increases if the superclass is change prone. CBO indicates the interdependence between the instances of different classes. The minimum value of CBO for the datasets used in this study is 0. The maximum value of CBO for Notepad++ lies between 91 to 100. The maximum value of CBO for CodeBlocks lies in the range of 43 to 80. The maximum value of CBO for CodeLite ranges from 45 to 133. The highest value of CBO indicates a large number of changes. For Notepad++ 7.6.3 and Notepad++ 7.8.3 version, the maximum value of CBO is 13 100. For CodeBlocks 20.03, the maximum value of CBO is 80. For CodeLite–14.0, the maximum value of CBO is 133. The maximum coupling between classes will provide a large number of changes. Hence, the CodeLite–14.0 version is more change prone. NOC indicates the number of subclasses or derived classes for a particular class. The highest value of NOC indicates the lesser probability of change proneness. The maximum value of NOC ranges from 37 to 44 in Notepad++. In CodeBlocks, the maximum value of NOC ranges from 43 to 98. In CodeLite, the maximum value of NOC is 181. The maximum value of NOC is 181 for all versions of CodeLite used in this study. Thus, CodeLite has fewer numbers of changes in comparison to Notepad++ and CodeBlocks. WMC indicates the sum of all the methods for a class. For Notepad++, the maximum value of WMC is 427.

2.8 Results and Analysis

The results of the RQ have been summarized. In this study, the prediction model has been developed using feature–based TL for HetCPDP. In HetCPDP, the matching metrics have been selected using Spearman's correlation coefficient. The comparison of ML techniques used has been done in terms of AUC. AUC is used as one of the best performance measures for dealing with imbalanced data and noisy data [72].

2.8.1 Data Preprocessing

This section describes the descriptive statistics of OOM of the datasets and data preprocessing steps.

The standard deviation quantifies the dispersion or variability of the OO measure, providing insight into its central tendency. The low standard deviation values suggest that the metric values are closely clustered around the mean, whereas high standard deviation values indicate that the metric is more spread out or dispersed. The median is a statistical measure that reveals information about the distribution of classes in a given metric. When dealing with a dataset that contains outliers, the median is a more reliable measure of central tendency than the mean.

2.8.1.1 Removal of Common Classes not Changed

This thesis predicts defect and change by measuring changes in the subsequent version of a software. Consequently, following the extraction of data, we eliminated all classes from the datasets that remained unchanged between the prior version and the subsequent

release. In order to accomplish this, we conducted a comprehensive scan of all data points within the dataset and eliminated all data points where the value of the UPDATE variable was ”no”. We made this decision because the UPDATE variable's value signifies a modification in the shared class. If the UPDATE has a ”no” value, it indicates that there has been no modification in the prior version and the subsequent version. Further, comprehensive overview of the software projects utilized in the thesis, including their names, analyzed versions, number of common classes, number of modified common classes, and the percentage of change explained in further Chapter 4, 5, 6 7, 8, and 9.

2.8.1.2 Outlier Analysis

The dataset used in this thesis is analyzed by removing outliers. Thus, the data points having extreme variability from all the data points in the dataset are considered outliers. It is always better to remove such data points for the development of a prediction model that is effective and unbiased. We have used the functionality of Waikato Environment for Knowledge Analysis (WEKA) tool [73] for computation of Inter Quartile Range (IQR). IQR is computed as the difference between upper quartile (Q_3) and lower quartile (Q_1) i.e.,

$$IQR = Q_3 - Q_1 \tag{2.7}$$

An observation representing a class is classified as an outlier if any of the following conditions are met for any of the independent variables *x* of the class.

$$Q_3 + OF * IQR < x \leq Q_3 + EVF * IQR \tag{2.8}$$

$$Q_1 - EVF * IQR \leq x < Q_1 - OF * IQR \tag{2.9}$$

In Eqn. 2.8 and 2.9, EVF represents Extreme Values Factor and OF represents Outlier Factor. The default values of EVF and OF in the WEKA tool are 6.0 and 3.0, respectively. The outliers were removed from each dataset before further analysis.

2.8.2 Data Balancing

Synthetic Minority Over-sampling TEchnique (SMOTE) is a widely used method in ML to tackle the issue of class imbalance, especially in classification jobs. Imbalanced classes are frequently observed in real-world datasets, where one class (the minority class) has a much lower number of occurrences compared to the other classes (the majority class or classes). The presence of class imbalance can result in the development of biased models that exhibit at par performance when dealing with examples from the minority class. SMOTE resolves this problem by producing artificial samples for the under represented class. This is the operational process:

- SMOTE initially determines the occurrences that correspond to the minority class.
- The NN Selection (NNS) method involves identifying the *k* closest neighbors in the feature space for each instance of the minority class using the SMOTE algorithm. Usually, Euclidean distance is employed as a metric to quantify closeness.
- Synthetic Sample Generation: SMOTE generates synthetic samples by creating new data points along the line segments that connect the *k* nearest neighbors. This is achieved by selecting a point at random along the line segments. These synthetic examples are novel cases that pertain to the minority class but are not identical replicas of existing instances.

- Dataset balancing: By generating synthetic examples, the dataset achieves a more balanced distribution, with an increased presence of the minority class.

SMOTE addresses the class imbalance issue by generating artificial instances, hence avoid introducing bias into the dataset. By providing a more equitable distribution of the classes, this enables the model to enhance its performance, particularly in situations when the minority class is not adequately represented.

2.8.3 Prediction Model Development and Validation

In this thesis work, the CPDP, and CPCP models are developed for increasing the reusability of existing models and removing overfitting from the model. However, the performance of the prediction model is analyzed through the usage of historical data and open–source datasets. The data analysis techniques discussed in subsequent chapters are used to develop prediction models in supervising learning mode. The models are developed using different independent variables of various projects. Once a model is developed, its validation is done by providing the validation or test data to know that how the developed model would behave on the unknown data points. The only variables included in the validation data points are independent ones. The generated model is expected to predict the class label when it receives the test data point. Afterward, the projected label is compared to the actual label to see the accuracy of the model's predictions. The validation techniques used in this thesis are discussed in detail in Chapter 1.

- Cross-Validation

  The types of cross-validation techniques are discussed as follows:

  - Hold-out cross-validation
  - Leave–one–out cross-validation
  - K-Fold cross-validation
- Inter-Version Validation
- Cross-Project/ Company Validation
  - Transfer Learning
    * Instance transfer
    * Feature transfer
    * Parameter transfer
    * Relational transfer

2.8.4 Performance Measures

In this thesis, the performance of the prediction model is analyzed using AUC metric. The confusion matrix contains the class values in positives and negatives. There are four entries in the confusion matrix i.e., True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN). Various performance measures are derived from the confusion matrix to assess the performance of CPDP and CPCP models. One of the most used statistics for assessing how well binary classification algorithms perform is AUC. It assesses a model's capacity to discriminate between positive and negative classes over a range of thresholds.

Receiver Operating Characteristic (ROC) Curve: A graphical depiction called the ROC curve shows how well a binary classifier system can diagnose problems as its

discrimination threshold changes. At different threshold values, it shows the True Positive Rate (Sensitivity) versus the False Positive Rate (1 - Specificity). The percentage of real positive cases that the classifier properly identifies is called the True Positive Rate (TPR). The percentage of true negative instances that are mistakenly labeled as positive is called the False Positive Rate (FPR). This measure assesses a classifier's overall effectiveness across all threshold combinations. The area under the ROC curve is computed. An ideal classifier would have an AUC of 1, meaning it maintains the FPR at 0 (zero false positive rate) while achieving a TPR of 1 (100% sensitivity). A totally random classifier, on the other hand, would have an AUC of 0.5 since it would only outperform chance. An effective metric for comparing and evaluating models, AUC offers a single scalar value that sums up the classifier's performance across all feasible thresholds.

AUC values are a continuous measure of classifier performance, with higher values indicating greater performance. The model appears to be no more accurate than random guessing when the AUC is 0.5. AUC values ranging from 0.5 to 1 represent the model's accuracy in classifying instances in most cases. The model performs better when AUC is higher. When there is an imbalance in the class distribution, AUC is very helpful since it evaluates the model's capacity to rank positive instances higher than negative instances, independent of class prevalence.

Advantages: Adaptability to class imbalance and lack of need for a predetermined decision threshold make AUC a good choice for classifier evaluation for a range of operating points. It offers a comprehensive perspective on classifier performance and is less susceptible to changes in misclassification costs or class distribution variations. To sum up, AUC is an effective measure for evaluating binary classification models since it captures the prediction model capacity to distinguish between positive and negative

examples at different threshold values. It is frequently used in ML evaluation and model selection, as it offers insightful information about the model's overall performance.

2.8.5 Statistical Analysis

The purpose of a statistical test is to analyze the performance of various ML techniques used. Friedman's test is used to determine the significant difference among these techniques [74–76]. It is a non–parametric test. The normality distribution of the dataset is not necessary for the Friedman test. It assigns a rank to various techniques based on the datasets used. It takes two hypotheses for testing. The hypothesis designed for testing is as follows: Null Hypothesis (H_o): Significant difference does not exists among the performance of ML techniques used. Alternate Hypothesis (H_a): Significant difference exists among the performance of ML techniques used. The value of the Friedman measure is calculated using the following equation:

$$\chi^2 = \frac{12}{nk(k+1)} \sum_{i=1}^{k} {R_i}^2 - 3n(k+1) \tag{2.10}$$

where *R* stands for an average rank of individual techniques, *n* stands for a number of datasets, and *k* stands for the number of techniques used for comparison. The value of the Friedman measure is distributed over *k*–1 Degree of Freedom (DoF). If the value of the Friedman measure lies in a critical area (greater than the significance level i.e., 0.01 or 0.05 and *k*–1 DoF, then the null hypothesis is rejected and the alternate hypothesis is accepted. It concludes that there is a significant difference in the performance of ML techniques. Thus, the null hypothesis is accepted, and the alternate hypothesis is rejected, and concluded that there is no significant difference in the performance of ML techniques

used. All the techniques are individually ranked using Friedman's Individual Rank (FIR) using Eqn. 2.11.

$$\text{FIR} = \frac{C}{n} \tag{2.11}$$

where, *C* stands for the sum of rank corresponding to individual technique assigned to each dataset value, and *n* stands number of datasets. An individual rank has been assigned to each technique using Eqn. 2.11. The technique with the lowest rank is considered as worst technique and the technique with the highest rank is considered the best technique. If the result obtained using the Friedman test is found to be significant, and then it is verified whether the significant difference exists or not by conducting post hoc analysis using Wilcoxon–signed rank test and Nemenyi test. The Nemenyi test is used for post hoc analysis to compare all the techniques individually. Nemenyi test is used when the dataset size is equal. The Critical Distance (CD) value has been calculated for the Nemenyi test as follows:

$$CD = q_\alpha \sqrt{\frac{k(k+1)}{6n}} \tag{2.12}$$

where q_α stands for critical values, *k* stands for a number of techniques used for comparison, and *n* stands for a number of datasets. The computed *CD* values are compared with the difference between ranks assigned to individual techniques. If the *CD* value is less than the difference between the two techniques, then the two techniques have a significant difference at a significant level, i.e., α.

Chapter 3

Systematic Literature Review

3.1 Introduction

Dataset plays an important role in the development of efficient software. Thus, the selection of an appropriate methodology for the development of a prediction model is important. In CPDP, the dataset of different projects is employed to develop prediction models in the existing studies. Researchers also explored cross-language prediction models in the existing studies. The reusability of prediction models is increased through CPDP, and the challenge of lack of data is solved in this manner. The issue of data scarcity is resolved using TL in software engineering. To study CPDP in more detail, it is necessary to summarize the existing literature. The systematic review aimed to analyze and summarize existing studies in terms of dataset used, variables used, ML techniques used, TL techniques used, performance measure, cross-validation method, and statistical test used for prediction model development. The following RQs were investigated in this review:

- RQ1: Which quality attributes are used for TL?
- RQ2: Which ML techniques are used for TL?
- RQ3: What experimental settings have been used for TL?
  - RQ3.1: Which datasets have been used for TL?
  - RQ3.2: Which independent variables have been used for TL?
  - RQ3.3: Which algorithms have been used for TL?
  - RQ3.4: What validation techniques have been used for TL?
  - RQ3.5: What performance measure used for TL?
  - RQ3.6: What statistical test has been used for TL?
  - RQ3.7: Which category of TL has been used?
- RQ4: Which TL methods are effective using ML techniques?
- RQ5: What are various threats to validity for TL?
- RQ6: What are the advantages & disadvantages of various TL techniques?

The review analyzed the usage of TL in the software engineering domain to enhance software quality. This review provided existing research findings with future directions to conduct more research in this direction. Further, based on review findings, future directions are provided for researchers, academicians, and industry experts to consider the application of this work.

This review is carried out as per the guidelines of Kitchenham [50]. According to Kitchenham review methodology, it consists of three-phase: planning the review,

conducting the review, and presenting the review results. (1) Planning the review phase investigates the need to conduct a review, designing research questions that need to be answered in the review, developing a protocol for conducting the review, and evaluating the method. The RQs are main objective to analyze the existing studies, and it addresses the need of actual systematic review. Further, appropriate strategy or method is designed to extract studies from digital libraries, inclusion/exclusion criterion, quality assessment criterion, design data extraction forms, and data synthesize covered in review protocol. Relevant studies are extracted initially, further inclusion/exclusion criterions and quality assessment criterion determines candidate studies. The quality of each selected candidate study assessed, and assessment criterion in the form of quality questionnaire developed.(2) Conducting the review phase involves execution of search string for selection of relevant studies in data extraction forms, and data synthesize from relevant studies. (3) Presenting the review results involves presentation of review results in the form of pictorial or tabulated representation with discussion.

Organization of this chapter: Section 3.2 discussed review protocol. Section 3.3 presented the answers to the RQ. Furthermore, Section 3.4 presented the result discussion and future direction in more detail.

3.2 Review Protocol

The second phase of Kitchenham methodology consist of following steps such as selection of primary studies, inclusion and exclusion criterion, quality assessment criterion for analyzing the quality of candidate studies.

3.2.1 Search Strategy

To conduct review the selection of relevant studies with the review topic is primary objective. The initial search started with searching of keywords such as “software quality”, “transfer learning”, “cross-project”. After analyzing the initial search string, the advanced search string are formed for selection of most useful studies to conduct review.

((“Transfer” OR “transfer learning” OR “transfer knowledge” OR “knowledge transfer” OR “transfer of learning”) AND (“variables” OR “parameters”) AND (“machine learning” OR “support vector machine” OR “neural network” OR “ensemble learning” OR “random forest” OR “decision tree” OR “naive bayes” OR “CART” OR “bayesian network”) AND (“cross-project” OR “cross-company”) AND (“defect” OR “change” OR “effort” OR “maintenance” OR “software quality” OR “software quality improvise”) AND (“improved” OR “better” OR “enhanced”) AND (“validation” OR “empirical” OR “design” OR “development”) AND (“evolutionary” OR “search” OR “optimized” OR “heuristic” OR “particle swarm” OR “harmony search” OR “simulated annealing” OR “bat search” OR “swarm intelligence” OR “firefly search” OR “gravitational serach” OR “inclined planes sytem” OR “bio-inspired” OR “genetic algorithm” OR “Grey Wolf” OR “cuckoo serach” OR “ant colony” OR “artificial bee colony”) AND (“method” OR “technique” OR “algorithm” OR “variant” OR “model”) AND (“dataset” OR “database”) AND (“cross-validation” OR “hold-out validation”) AND (“statistically” OR “validated” OR “statistical” OR “statistical test” OR “paired test” OR “wilcoxon” OR “ANOVA”))

The formulation of above search strings are used to search for relevant studies from digital libraries such as Google Scholar, IEEE Xplore, Springer Link, ScienceDirect, Wiley Online Library, ACM Library, and Scopus. We have searched papers from January 1990

to March 2024. In this period, 122 studies are extracted from above the mentioned digital libraries. The primary studies are selected subject to inclusion/exclusion criterion, and quality assessment criterion discussed in Section 3.2.2 and 3.2.3.

3.2.2 Inclusion and Exclusion Criteria

The inclusion/exclusion criterion used mentioned below for selection/rejection of a study on the basis of RQ. Total 39 studies selected after applying inclusion/exclusion criterion.

Inclusion Criteria

- Experimental studies for TL using different ML techniques.
- Empirical studies relevant to the software engineering field.
- Empirical studies have a combination of ML and non-ML techniques.

Exclusion Criteria

- Empirical studies are not related to TL in software engineering.
- Empirical studies do not describe the experimental investigation.
- Empirical studies not provided the results of ML techniques for TL.
- Review studies.
- Empirical studies which are not written in the English language.
- Empirical studies have a similar author in the conference, and the existing version has been extended in the journal.
- Chapters of TL.

3.2.3 Quality Assessment Criteria

In quality assessment criterion, a questionnaire has been formed. These questionnaires are used to study the purpose and strength of selected primary studies.

Table 3.1: Quality Assessment Questionnaire

S. No.	Question	Yes	Partly	No
1	Does the stated aim of the research is clear?	25	7	7
2	Does the definition and usage of quality attributes is clear?	28	5	6
3	Does the explanation of the experimental setup is clear?	29	8	2
4	Does it specify the independent variables used?	21	13	5
5	Does the proper data size used?	28	3	8
6	Do the ML techniques are clearly defined?	27	7	5
7	Does the performance measure been clearly stated and used?	25	14	0
8	Does the validation techniques used in the study?	24	10	5
9	Does the comparative analysis been conducted (ML vs. TL)?	27	8	4
10	Do the stated results, findings, and conclusions are clear?	21	15	3
11	Does the study add some contribution to the literature?	28	7	4
12	Does the stated limitations or threats to validity are clear?	26	8	5
13	Does the study use a repeatable research methodology?	30	3	6

The quality assessment criteria were designed by considering the guidelines and suggestions provided in the existing studies [57]. We have used quality assessment criteria to assign a particular weight to every study. Table 3.1 presents quality evaluation criteria. We have decided on three parameters corresponding to each question, which are based on whether the particular study answers the question or not. If the study answers the question, then we tick mark corresponding to the yes parameter. If the study does not answer the question, then we tick the mark corresponding to no parameter. Every question has been assigned some rank as 1 (yes), 0.5 (partly), and 0 (no). The summation of values assigned to each question provides the final score corresponding to each study. The maximum and minimum score of every study is 13 and 0.

Table 3.2: Description of Primary Studies

Study#	Name	Reference	Study#	Name	Reference
SI1	Pratt (1992)	[77]	SI21	Feuz (2015)	[78]
SI2	do (2005)	[79]	SI22	liu (2017)	[80]
SI3	Raina (2006)	[81]	SI23	Yu (2017)	[82]
SI4	Mihalkova (2007)	[83]	SI24	Weiss (2017)	[84]
SI5	Pan (2008)	[85]	SI25	Pereira (2017)	[86]
SI6	Dai (2009)	[87]	SI26	Gargees (2017)	[88]
SI7	Li (2009)	[89]	SI27	Yan (2017)	[90]
SI8	Wan (2015)	[91]	SI28	Chen (2018)	[92]
SI9	Ma (2012)	[21]	SI29	Krishna (2018)	[93]
SI10	Long (2013)	[94]	SI30	Nam (2015)	[30]
SI11	Nam (2013)	[41]	SI31	Deshmukh (2018)	[95]
SI12	Zhou (2014)	[96]	SI32	Ying (2018)	[97]
SI13	Kocaguneli (2015)	[45]	SI33	Cui (2018)	[98]
SI14	Dillon (2014)	[99]	SI34	Chen (2019)	[100]
SI15	Qing (2015)	[31]	SI35	Tang (2021)	[35]
SI16	Jing (2015)	[24]	SI36	Bai (2022)	[40]
SI17	Cao (2015)	[101]	SI37	Wu (2021)	[102]
SI18	Krishna (2016)	[103]	SI38	Xu (2019)	[104]
SI19	Weiss (2016)	[105]	SI39	Du (2020)	[106]
SI20	Su (2016)	[107]			

3.3 Review Results

The results extracted from the primary studies are presented in this section.

3.3.1 Results Specific to RQ1

The quality attributes emphasized by the existing studies are discussed in order to answer RQ1. Some quality attributes, such as effectiveness, performance, reliability, effort, change, and defect. It has been observed that most of the studies focused on defect prediction.

Defect prediction in the software engineering field plays a vital role before deploying software at the end user site. Thus, software developers and the software tester team ensure the software is free from any kind of defect in the deployment stage. Developers need to take care of defect proneness in future projects with the help of existing datasets using TL for knowledge transfer in projects with similar data distribution and similar tasks. Fig. 3.1 shows the year-wise distribution of primary studies from January 1990 to March 2024. It was observed that TL in software engineering actually started in 2010, when there was a lack of data available for the training of prediction models.

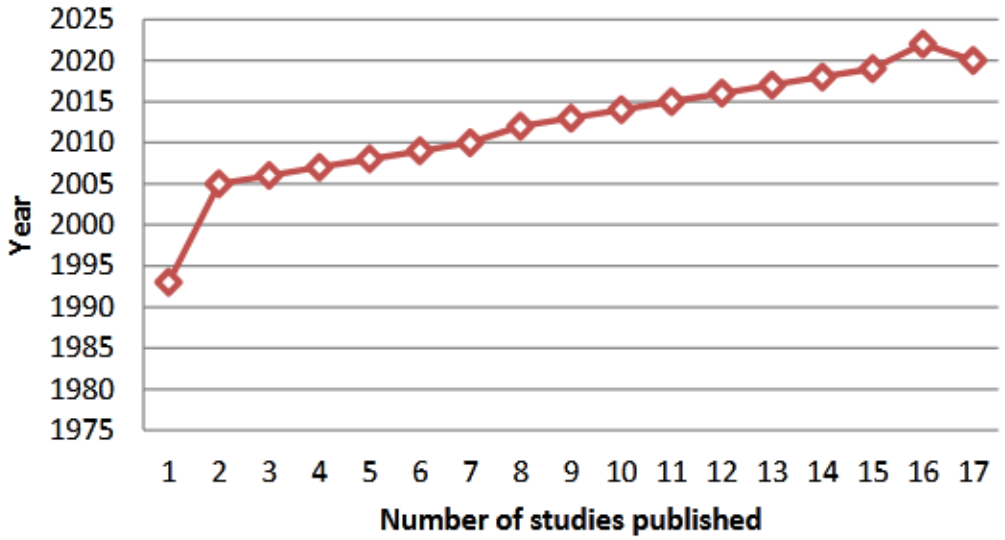

Figure 3.1: Year-wise distribution of primary studies

The most commonly used attribute out of all the quality attributes is performance. It has been used in 14 studies (SI1, SI3, SI4, SI10, SI11, SI19, SI20, SI21, SI22, SI24, SI25, SI26, SI27, SI34). The authors have analyzed the performance of an algorithm that has been developed or used in the study. The next frequently used attribute is effectiveness, which has been used in 10 (SI2, SI5, SI6, SI7, SI8, SI12, SI14, SI16, SI32, SI33) studies.

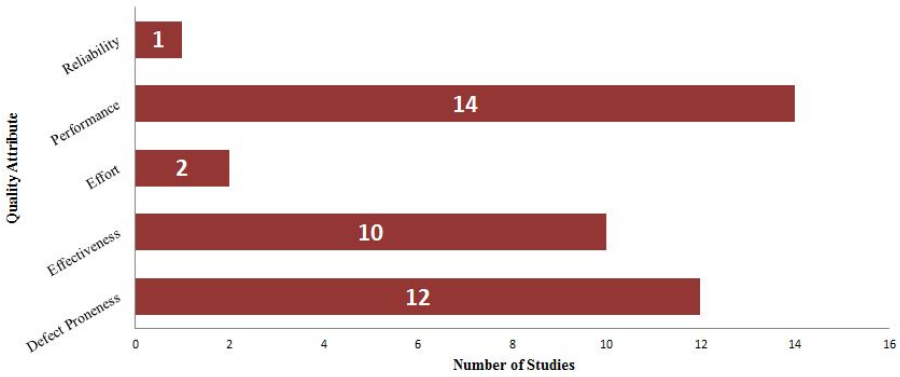

Figure 3.2: Quality attribute emphasized in the existing studies

The defect attributes have been used in 13 studies (SI9, SI15, SI17, SI18, SI23, SI28, SI29, SI30, SI35, SI36, SI37, SI38, SI39) out of the selected studies.

Table 3.3: Description of quality attributes

Quality attribute	**Description**
Effectiveness	This attribute can provide the desired output or the capability of providing the desired output.
Performance	This attribute provides the system output by doing some work for a particular period.
Reliability	This attribute is related to the characteristics that deal with the software potential to maintain its performance level under certain conditions in a specified period.
Effort	This quality attribute tells the reasonable amount of time required in developing a particular software (in terms of person-hours).
Defect Proneness	This quality attribute is defined as an error made in the source code or logic in the source code that can lead to crashing or can produce imprecise/ unpredicted outcomes.

The defect attribute is used to analyze the effect of TL on defect prediction in software

engineering. The authors have analyzed the effect of defect prediction using TL, whether it is predicted or not. The effort attribute has been used in 2 studies (SI13, SI39) out of 39 studies that have been used for this review. It has been checked that it is feasible to build transfer learners for effort estimation [82]. Developers are required to collect all the user requirements in the initial stage. Further, if any kind of change is requested by the customer, then it's feasibility must be checked and approved by the Change Control Board (CCB). Furthermore, the quality attributes identified in the primary studies are presented in Fig. 3.2.

3.3.2 Results Specific to RQ2

In the existing studies, the prediction model is developed using various ML techniques. The aim of using such techniques is to establish the relationship between independent and dependent variable. We have categorized techniques into three different categories such as ML, TL, and statistical methods. We divide the ML techniques used for developing prediction models into the following categories such as SVM, DT, Ensemble Learning (EL), Bayesian Learning (BL), KNN. The distribution of studies according to the above-mentioned techniques used for TL along with ML is presented in Table 3.4.

Table 3.4: Distribution of studies according to various techniques

Category of ML classifier	**Type**	**Percentage of studies**	**Number of studies**
SVM	TSVM: Transductive SVM, MSVMs: Multiclass SVM, LSVM: Linear SVM, SVDDD: Support Vector Domain Data Description, KNND: KNN Data Description	35.90	14
DT	CART C4.5	25.64	10
BL	WNBC: Weighted NB classifier, NB, BN	25.64	10
K-NN	NN	17.95	7

EL	RF, Value Aware Boosting with SVM, Stochastic Gradient Descent (SGD) Classifier, Gradient Boosting Classifier, ABD Classifier	28.21	11
Miscellaneous	SoR: Softmax Regression, LiR: Linear Regression, MTL: Multitask learning, Self-Training, LR, GBM: Graph-Based Methods, MR: Manifold Regularization	12.82	5

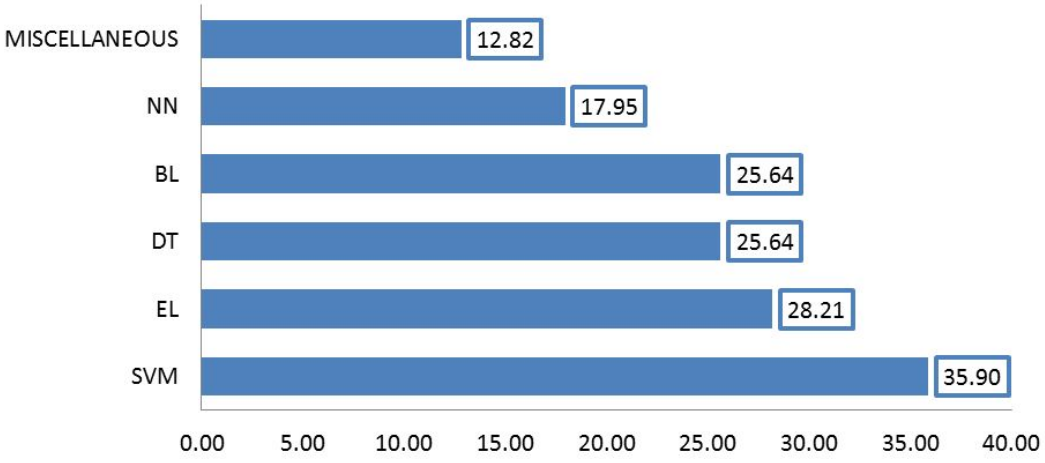

Figure 3.3: Distribution of studies (in terms of %) of ML techniques used

The percentage of studies used and a number of studies that used ML techniques are presented in Table 3.4. Out of all the ML techniques that have been mentioned in the above table, most of the techniques are from such categories as SVM, EL, DT, and BL examined in 35.90%, 28.21%, 25.64%, and 25.64% of studies, respectively. It has been observed that SVM is widely used with TL in 14 studies for software engineering (SI2, SI5, SI6, SI9, SI10, SI11, SI16, SI19, SI22, SI24, SI26, SI30, SI31, SI37), with LSVM, SVDD, KNND, MSVMs. Further, EL category ML techniques are used increasingly (SI16, SI17,

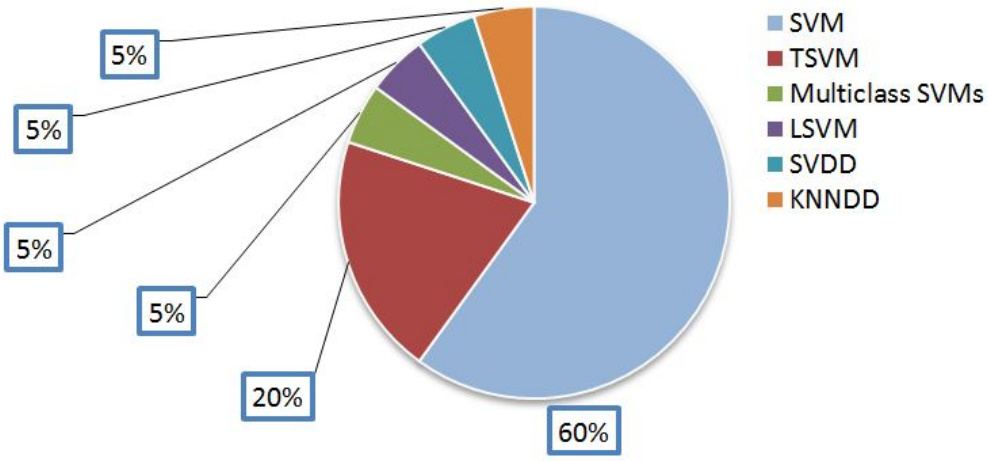

Figure 3.4: SVM categories

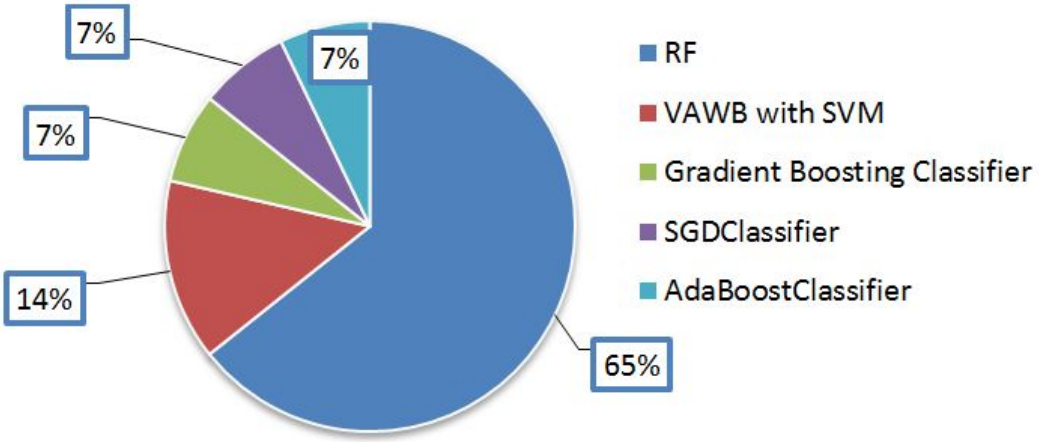

Figure 3.5: EL categories

SI18, SI19, SI24, SI29, SI30, SI34, SI36, SI37, SI38) in 11 studies with RF, VAWBSVM, ADB, SGD classifier, Gradient Boosting classifier. DT category is used in 10 studies (SI1, SI14, SI15, SI21, SI22, SI24, SI29, SI34, SI37, SI38) with C4.5 and CART variants.

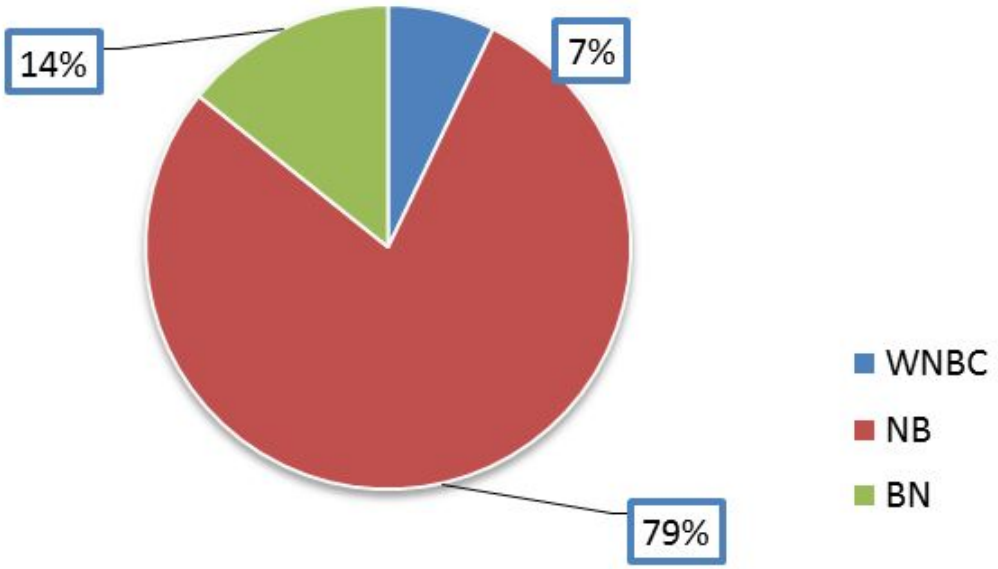

Figure 3.6: BL categories

Furthermore, the distribution of studies in terms of percentage is presented in Fig. 3.4, 3.5, 3.6, and 3.7. The ML techniques mentioned in Table 3.4 are used for TL.

3.3.3 Results Specific to RQ3

Identifying the datasets, independent variables, algorithms, validation techniques, performance measures, and statistical tests used for TL in the selected primary studies is important.

Dataset (RQ3.1)

Various types of datasets are used for TL studies. Fig. 3.8 presents the number and percentage of studies that used multiple types of datasets. All dataset possess different characteristics. Private datasets consist of data collected by researchers for conducting

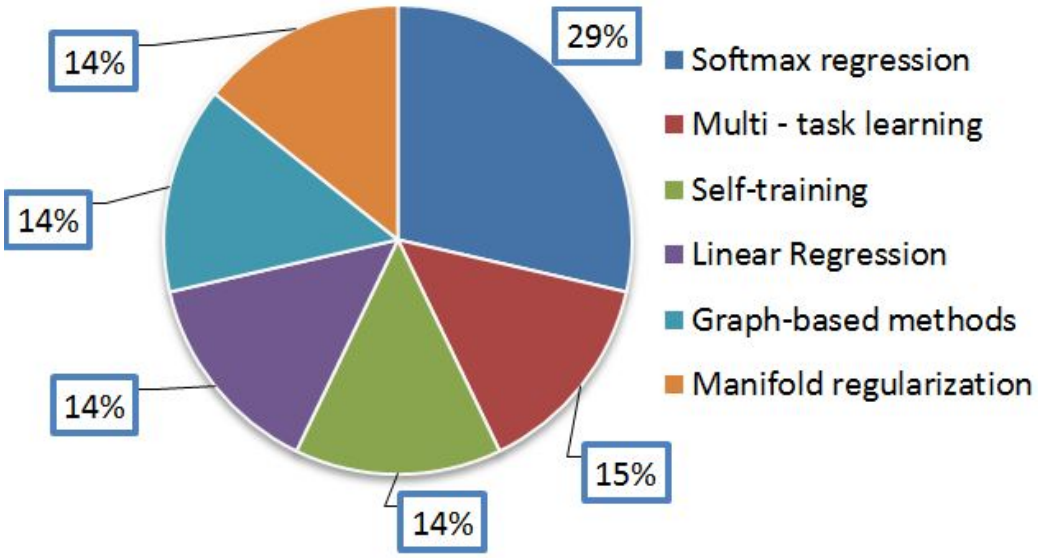

Figure 3.7: Division of sub-categories of ML techniques

specific studies and from other agencies for evaluation or research purposes. Private datasets are not distributed among researchers and due to this private datasets are not verified and repeatable by the researchers. Public datasets are freely available. Thus, it has been concluded that more proprietary and academic datasets must be used for future experimentation. The exhausted dataset does not provide efficient results in such cases. It is always advisable to use more industry-oriented datasets that help researchers to understand and study dataset in more detail.

The various categories of used datasets are as follows :

- AEEEM dataset: This dataset is collected by D'Ambros AEEEM dataset is a commonly used dataset concerning to a software defect. This dataset consists of various metrics such as change metrics, existing defects metrics, code metrics, the entropy

of changes metrics, and the entropy of source code metrics. It consists of 61 metrics and 5386 instances [59, 91, 93]. This dataset is used in 11% of the primary studies (SI11, SI16, SI17, SI18, SI29, SI30, SI34, SI36, SI38).

- MAGIC Gamma Telescope dataset: The MAGIC Gamma dataset, also known as MAG. This dataset is resourced from the repository of UCI ML. This dataset is in the form of binary classification, which has various instances and numerical value attributes. This dataset is used in 4% of the primary studies (SI19, SI24).

- MovieLens dataset: This dataset is collected by GroupLens. It is a movie rating dataset on a scale of 1 to 5. It provides the rating dataset that is available on the MovieLens web site. Users provide a rating for each movie during different time intervals. This dataset is used in 2% of the primary studies (SI7, SI8).

- NASA dataset: This dataset is publicaly available. NASA repository stores this dataset, and the NASA metrics data program maintains this dataset. All datasets in the NASA repository act as a particular NASA computer software or sub-part of software. This software consists of data regarding defect marking and metrics related to source code. Metrics related to source code consist of length, understandability, volume, vocabulary, and complexity, which are associated with software quality. This dataset is used in 15% of the primary studies (SI4, SI9, SI13, SI16, SI17, SI18, SI23, SI28, SI29, SI30, SI34, SI36, SI38).

- ReLink dataset: This dataset contains information regarding defects. The information stored in this is manually proven and improved. ReLink contains 26 complexity metrics. These metrics are used for defect forecasts. The ReLink dataset has differ-

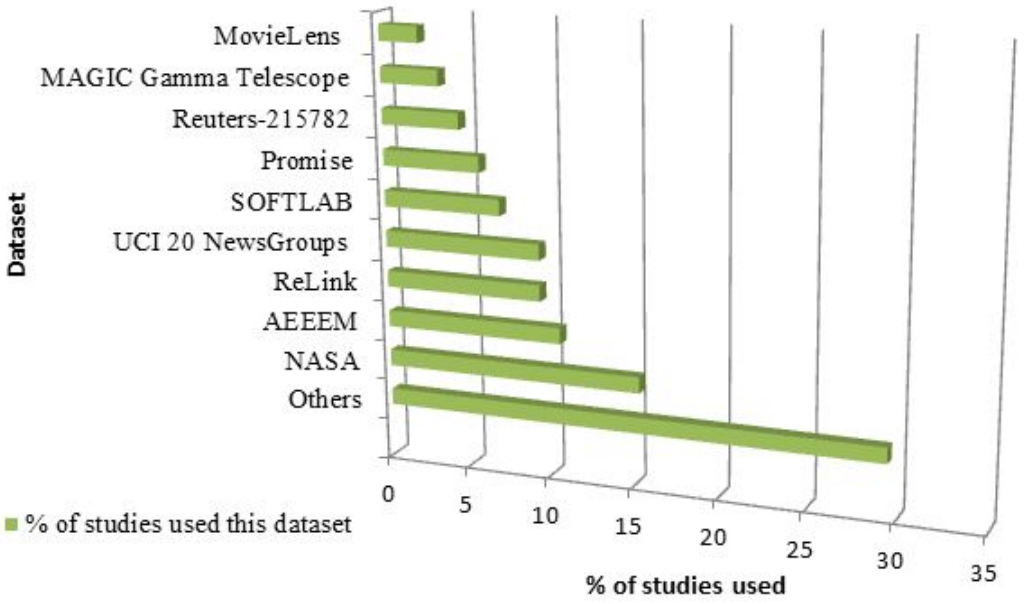

Figure 3.8: Dataset used

ent features like time interval, bug owner, change committer, and text similarity. It consists of total 26 features and 658 instances [94]. This dataset is used in 9% of the primary studies (SI11, SI16, SI17, SI18, SI29, SI30, P36, SI38).

- Reuters-215782: This dataset was collected by Carnegie Group, Inc. and Reuters, Ltd. during the period of developing the CONSTRUE text categorization system. This dataset is one of the commonly used datasets for text categorization. It is defined as the collection of various documents that are available on the reuters commercial newswire system. It has five top divisions and many subdivisions. This dataset is used in 5% of the primary studies (SI2, SI5, SI6, SI10).

Independent Variables (RQ3.2)

The independent variables used for each experimentation in further chapter discussed

and presented in Table 3.5. Various independent variables have been used by the selected primary studies like number of features, classes, OOM, Halstead metrics, and C&K metrics. It is observed that C&K metrics are mostly used.

Table 3.5: Independent variables used

Independent variables	**Study Identifier**	**Independent variables**	**Study Identifier**
Information Measure Metric (IMM)	SI1	Number of test samples	SI20
Number of classes in email	SI2	Performance metrics	SI21
Vocabulary of words and a summary of documents	SI3	Train Pivot Predictors	PS27
Eigenvector	SI6	Attributes	SI9, SI23
Regularization parameters, Number of feature clusters k, Number of nearest neighbors	SI10	C&K metrics	SI15, SI18, SI30, SI35
Defect prediction metrics	SI11	Line of Code (LOC)	SI30
Tradeoff parameter, Feature corruption probability p	SI12	Static code metrics	SI29
Number of instances, number of labels	SI14	OOM	SI11, SI16, SI17, SI18, SI29, SI30, SI34, SI36, SI38
Common metrics, Company-specific metrics	SI16	Halstead Metrics	SI4, SI9, SI13, SI16, SI17, SI18, SI23, SI28, SI29, SI30, SI34, SI36, SI38

Number of instance classes	SI17	Source code metrics	SI15, SI23, SI28, SI35, SI37
Test target size, number of labeled instances, DoF, p-value	SI19	Quality Model for Object-Oriented Design (QMOOD) metrics	SI35

Algorithms used for TL (RQ3.3)

The various algorithms used by the primary studies are discussed in this section and Table 3.6. The algorithms are based on the type of target and training data. Two studies have performed comparison among five TL methods using ML techniques as a base learner. Adaptation Regularization TL (ARTL), Geodesic Flow Kernel (GFK), TCA, Transfer Joint Matching (TJM) are the TL algorithms that have been used in two studies.

Table 3.6: Transfer learning algorithm used

Study Identifier	**TL algorithm**	**Description**
SI2	Task-clustering algorithm	This algorithm was used for text classification. There exists a linear text classification algorithm; it used inner product across a test document vector and parameter vector. In the task clustering algorithm, the tasks are grouped via the NN algorithm to facilitate knowledge transfer. Different parameter functions are used in this algorithm. The parameter function is obtained with the help of training data, and it has been used for testing data.

SI4	Find a legal mapping for a source clause	The study used the algorithm to find a mapping for a source clause. The authors have used the concept of TL in terms of transferring the mapping learned from source to target. There are two different types of mapping. One is global mapping, and the other is local mapping. In global mapping, mapping is established for each source predicate to a target predicate and used for the entire source translation. The other approach called local mapping, is to find the top mapping of each source clause individually.
SI5	Dimensionality Reduction algorithm (TL via Maximum Mean Discrepancy Embedding)	This algorithm is a two-step process. This algorithm minimized the dimensionality with the help of TL. It is designed to ensure active TL and the objective was to minimize the distance between data distributions across different domains.
SI9, SI29	Transfer NB (TNB)	This algorithm takes the set of labeled samples and unlabeled samples as input.
SI10, SI19	Graph co-regularized Collective Matrix tri-Factorization (GCMF)/ Graph Co-Regularization TL (GTL) algorithm	It uses any prior knowledge if available, and prior knowledge includes links in network mining.
SI12	Hybrid HetTL	This algorithm transfers features across different source and target domains.
SI19, SI20, SI22	ARTL algorithm	This algorithm performs the instant adaptation of different domains and classifier learning.
SI19, SI24, SI29, SI32, SI37	TCA algorithm	This algorithm explores the similar features between the training and target data.
SI19, SI24	TJM algorithm	The main aim of this algorithm is to decrease the marginal probabilities.

SI21	Feature Space Remapping (FSR)	It is a HetTL algorithm. It transforms features among the source and target data. It calculates meta-features and then computes the similarity between them.
SI27	Weight - Structural Corresponding Learning (SCL) algorithm	This algorithm finds out the important and unimportant features among the source and target domains.
SI22	Weighted-resampling-based TL algorithm	This algorithm perform several iterations. The main focus of this algorithm is to transfer weights assigned to the instances. In each iteration, a new source training dataset is created. The labeled data in the target dataset is also combined with the source training data set.
SI35	TL Oriented Minority Oversampling Technique based on Feature Weighting TNB	This algorithm transfers the features among the source and target data. The transferred features are selected based on their correlation with the predictor/ output.
SI36	3SW-MSTL	A novel method named 3SW-MSTL was developed for multi-source. In the first stage, it is used to select multiple source projects from multiple target projects considered as a training project. Further, KNN applied to obtain 14 reweighted training instances by minimizing the difference of marginal distributions between each selected source project, and target project. It is based on a difference between the conditional probability distribution of selected source projects and target projects, a multi-source data utilization scheme is employed for prediction model training.

SI38	Balanced Distribution Adaptation (BDA)	BDA considers both the marginal and the conditional distribution differences between the source and the target projects.
SI39	Transfer AdB	It is a supervised instance based domain adaptation algorithm and is mainly used for classification tasks. It is considered a reverse boosting concept.

Validation Techniques (RQ3.4)

The various types of cross-validation techniques used in the primary studies are discussed in this section. The objective of using cross-validation methods are to analyze the predictive capability of algorithms for prediction on unseen data. It also gives insights about the generalized capability of prediction model on independent dataset. The different validation techniques such as *K*-fold cross-validation, Leave-one-out cross-validation, and Hold-out validation presented in Table 3.7. The most commonly used validation technique is *K*-fold cross-validation.

K-fold cross-validation is used in fifteen studies (SI1, SI2, SI6, SI11, SI12, SI13, SI14, SI15, SI21, SI22, SI23, SI24, SI29, SI30, SI31) out of all the selected primary studies for the review. The *K*-fold cross-validation method helps in removing the overfitting and provides more reliable result. However, leave-one-out cross-validation is used in two studies (SI13, SI26), and hold-out cross-validation is used in one study (SI3). Hold-out validation method is advisable for large size dataset. The datasets used in the existing studies are of limited size, due to which authors used *K*-fold cross-validation method. The graphical representation of the count of studies that used validation techniques is presented in Fig. 3.9.

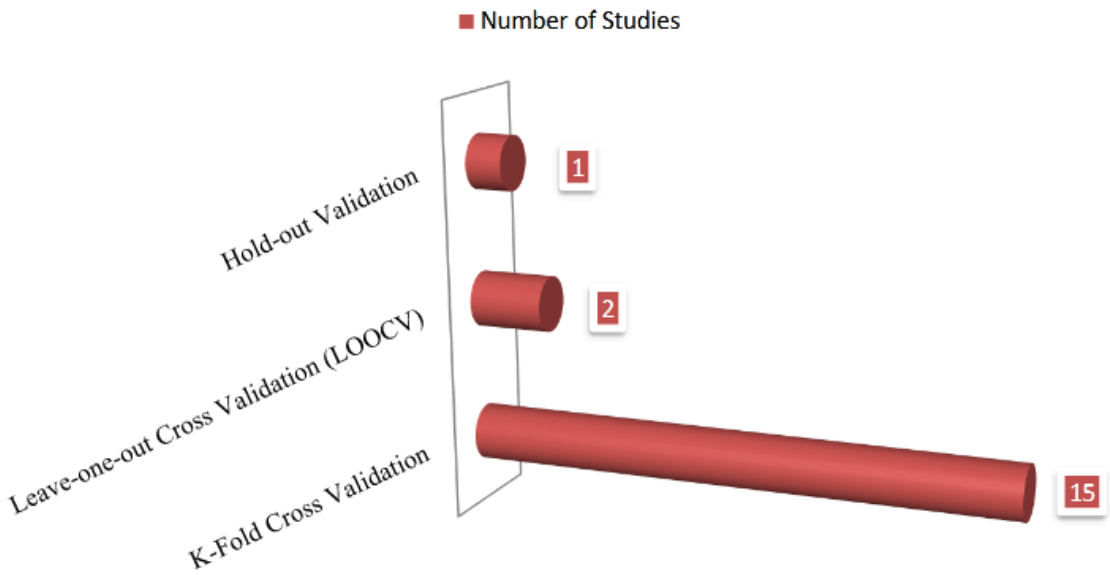

Figure 3.9: Validation techniques used

Table 3.7: Description of validation techniques used

Validation technique	**Description**
K–Fold cross–validation	In this validation technique, the original data is randomly divided into *K* identical-sized subsets of original data. Out of the *K* subsets, a single subset acts as verification data for performing testing, and the leftover *K*-1 subsets act as training data.
Hold–out cross–validation	This validation technique is a simple and commonly used cross-validation technique. In this technique, the dataset is categorized into two different sets, one dataset is used as a training set, and another dataset is used as a testing set. The training set is used to fit a function in the function approximator. The outcome values are predicted by function approximator using the testing set data, which is provided as an input to it.
Leave-one-out cross-validation	This validation technique is similar to *K*-fold cross-validation where *K* is equivalent to *N*, the number of data points in the set. It means that the function approximator is trained for all the data except one point and that one point is used for prediction.

Performance Measures (RQ3.5)

There are various metrics or measures used to analyze the performance of different models developed using TL. The evaluation measures play an important role in performing the comparison and evaluation of developed models using various TL and ML techniques. Table 3.8 represents the various evaluation measures, the theoretical description of the specified measures. The illustration of the count of studies for each specified evaluation metric is represented in Fig. 3.10. From the Fig. 3.10, it has been observed that accuracy is the widely used evaluation metrics (SI2, SI5, SI10, SI12, SI14, SI18, SI19, SI20, SI21, SI22, SI24, SI25, SI27, SI28, SI29, SI32, SI33), followed by Recall (SI9, SI11, SI16, SI18, SI21, SI27, SI28, SI29, SI35, SI36, SI38, SI39), F-measure (SI9, SI11, SI15, SI16, SI17, SI18, SI28, SI29, SI37, SI38, SI39), AUC measure (SI4, SI9,PS15, SI17, SI21, SI23, SI26, SI30, SI34, P36, SI38), Precision (SI11, SI14, SI18, SI28, SI29, SI39), FPR (SI9, SI16, SI18, SI29, SI35), and G-mean (SI36, SI38).

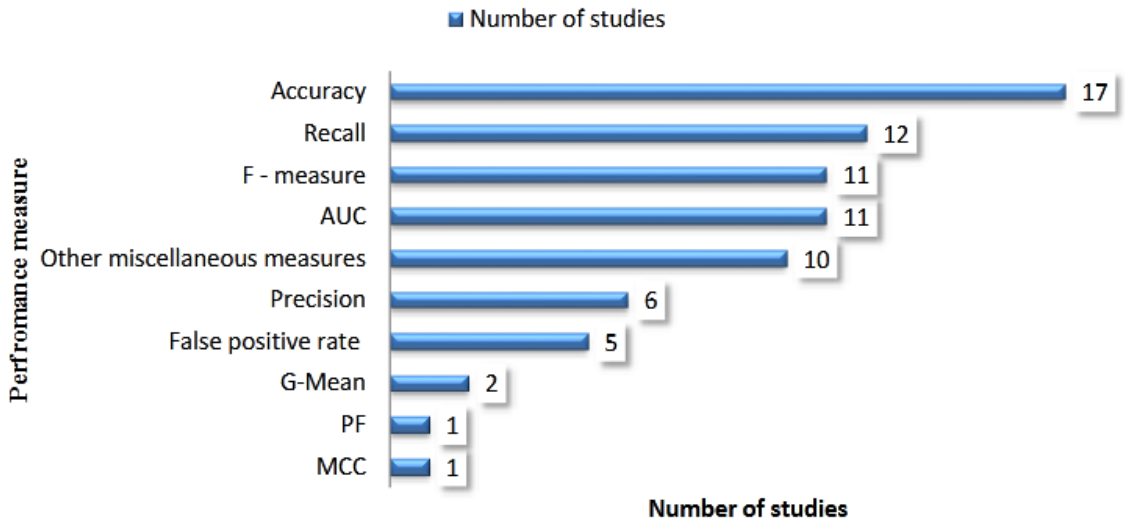

Figure 3.10: Performance measure used

Table 3.8: Description of performance measure

Evaluation Measures	**Description**
Accuracy	Accuracy defines a correlation between the correctly classified classes and the summation of all the classes. $$\frac{(TP+FN)}{(TP+TN+FN+FP)} \quad (3.1)$$
AUC	This performance metric tells us whether the model is capable of distinguishing between different classes. This metric or performance measure is used for binary classification. A model having an AUC value of 0.0 depicts the 100% incorrect predictions made by that model, and if the AUC is 1.0, that indicates the predictions made by the model are 100% right.
FPR	FPR is defined by the ratio of positive instances that are not correctly identified, which are originally classified as negative instances to the total original negative instances. $$\frac{(FP)}{(FP+TN)} \quad (3.2)$$
F-measure	F-measure is a weighted reciprocal of the arithmetic mean of the reciprocals of recall or sensitivity and precision. Its value depends on precision and recall value. If the value is less out of precision and recall, then it results in less F-measure value. $$\frac{((\alpha+1)*recall*p)}{(recall+\alpha*precision)} \quad (3.3)$$

Precision	Precision is defined as the proportion of actual positive instances that are correctly predicted to the overall predicted positive instances. It provides the count of correct positive predictions. The 100% precision value for any class A indicates that the instances associated with class A are correctly determined as a part of class A. It does not indicate anything about other instances that are associated with class A and predicted as incorrect instances. $$\frac{(TP)}{(TP+FP)} \quad (3.4)$$
Recall	The recall is defined by the negative instances (which are correctly classified) to the sum of real positive instances. The recall is also called the TPR or sensitivity. The recall value of 1.0 for any class, indicates that a total number of instances that are associated with that class are identified as a part of that class. $$\frac{(TP)}{(TP+FN)} \quad (3.5)$$
Matthews Correlation Coefficient (MCC)	MCC is used to identify the correlation coefficient between the actual and predicted binary classification. The perfect prediction means a +1 value of the coefficients, a random prediction is indicated by the value of coefficients as 0, and a value of -1 indicates complete disagreement between actual and predicted observation. $$\frac{(TP*TN-FP*FN)}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}} \quad (3.6)$$

G-Mean	G-mean measures the balance between the classification performances of both the majority and minority classes.
Other miscellaneous measures	They include absolute residual, AUCEC, CLL, Error rate, Error mean, Error median, MAE, MRE, MER, Mean Balanced Relative Error (MBRE), misclassification error, mean square error, RMSE, SA, UAR.

The performance metric which is rarely used are united in the miscellaneous category such as absolute residual, Area Under the Cost-Effectiveness Curve (AUCEC), Conditional log-likelihood (CLL), Error rate, Error mean, Error median, Mean Absolute Error (MAE), Magnitude of Relative Error (MRE), Magnitude of Error Relative to the Estimate (MER), Mean Balanced Relative Error (MBRE), Misclassification error, Mean square error, Root Mean Square Error (RMSE), Standardized Accuracy (SA), Unweighted Average Recall (UAR) (SI3, SI4, SI6, SI8, SI13, SI14, SI16, SI25, SI31, SI32).

Statistical test (RQ3.6)

This section describes the various statistical tests that have been used by the studies. These tests tell us about the significant difference between various distributions. Table 3.9 represents the type of statistical test, their description, and the study identifier in which they are used. In Fig. 3.11 we have represented graphically the statistical test and the number of studies in which they are used. The various tests that are used in the studies is One-way Analysis of Variance (ANOVA), Paired t-test, Wilcoxon test, Wilcoxon rank-sum test, Tukey's Honest Significant Difference test, Friedman test, Two-tailed T-test, Kolmogorov-Smirnov test (KS test).

Kruskal-Wallis H-Test is rarely used statistical test. Thus, from the observed data it is concluded that Wilcoxon test used in majority cases (SI10, SI11, SI13, SI23, SI34, SI36,

SI37), as it a non-parametric test and used to perform comparison among two independent samples. Furthermore, the Friedman test is used in the majority of studies (SI22, SI36, SI38), and it is used to compare multiple treatments. However, the limitation of the Friedman test is that it can be applied only when the minimum of treatments is 3. However, if the results of the Friedman test are to accept the alternate hypothesis, then a post-hoc analysis test must be performed. Thus, a comparison of two techniques must be performed using the Nemenyi test, Wilcoxon–signed rank test, and Bonferroni Dun test.

Table 3.9: Description of statistical test used

Statistical Test	**Study Identifier**	**Description**
One-way ANOVA	SI14, SI21	One - way ANOVA technique is used for the comparison of the mean of two or more than two samples. This technique applies to numeric data only.
ANOVA	SI19, SI24	ANOVA technique is used to check if there is a significant difference between the mean of two or more then two groups. It checks dependency between factors with the help of the mean comparison of different samples.
Kruskal-Wallis H-test	SI2, SI18	The Kruskal-Wallis H test is also called as one-way ANOVA on ranks. It is a rank-based non-parametric test that can be used to determine if there are statistically significant differences between two or more groups of an independent variable on a continuous or ordinal dependent variable. It is considered as extended version of the Mann-Whitney U test to perform comparison across more than two independent groups.

Paired t-test	SI20, SI32	The paired t-test is also known as the paired sample t-test and dependent sample t-test. It is a statistical test that is used to identify whether the mean difference between two sets of observations is zero.
Wilcoxon test	SI10, SI11, SI13, SI23, SI34, SI36, SI37	The Wilcoxon test has four different variants. It is a non-parametric test. One of the variants of the Wilcoxon test is the Wilcoxon–signed rank test. This test is used to compare two different samples which are related, matched samples, or parallel measurements over one sample to analyze the difference between their population mean ranks. Another variant is Wilcoxon–signed rank test, which is a non-parametric test that can be used to identify whether two dependent samples were selected from the populations having a similar distribution.
Wilcoxon rank-sum test	SI9, SI16, SI35	Wilcoxon rank-sum test also known as Mann-Whitney Wilcoxon, Mann-Whitney U test, or Wilcoxon Mann-Whitney test. Wilcoxon rank-sum test is a non-parametric test of no effect that is the value that is randomly selected form one population sample will be either less than or greater than a value that is randomly selected from another population sample. Non-parametric means it does not have any assumptions of gaussian distributions (normal distribution). This test applies to independent samples.
KS test	SI34, SI35	It is a non-parametric test which is used to test the equality of continuous or discontinuity
Friedman Test	SI22, SI36, SI38	It is a non-parametric test and it is an alternative measure, This test is used to test the difference across different groups when the target variable is of ordinal type.

Two-tailed T-test	SI22	In the two-tailed test, the critical area of the distribution is two-sided, it tests whether a sample is greater than or less than a certain range of values. Thus, it is used in null hypothesis testing.
Tukey's Honest Significant Difference (HSD)	SI19, SI24	Tukey's HSD test also known as Tukey's range test, Tukey's test, and Tukey method, is a one-step process of several comparison and statistical tests. This test can be applicable to unprocessed data or in combination with an ANOVA to find out the means that are different from each other.
Nemenyi test	SI38	This test is used as a post-hoc analysis test like Wilcoxon–signed rank test followed by the Friedman test. It is used to find out which groups are different. The hypothesis for Friedman test concerning Nemenyi tests as follows: • The null hypothesis (H_0): The mean value for each of the populations is equal. • The alternative hypothesis: (H_a): At least one population mean differs from the others.
Kendall tau-b rank correlation coefficient	SI16	It is used to find the strength and direction of association between two variables on an ordinal scale.
Bonferroni-Dunn test	SI36	It is used to perform comparison among multiple pairs of means (averages) among groups of data and is mostly used after applying statistical test for mean comparison such as ANOVA.
One-sided paired t-test	SI17	In a two-tailed test, the critical area of the distribution is one-sided, it tests whether a sample is greater than or less than a certain range of values, but not both.

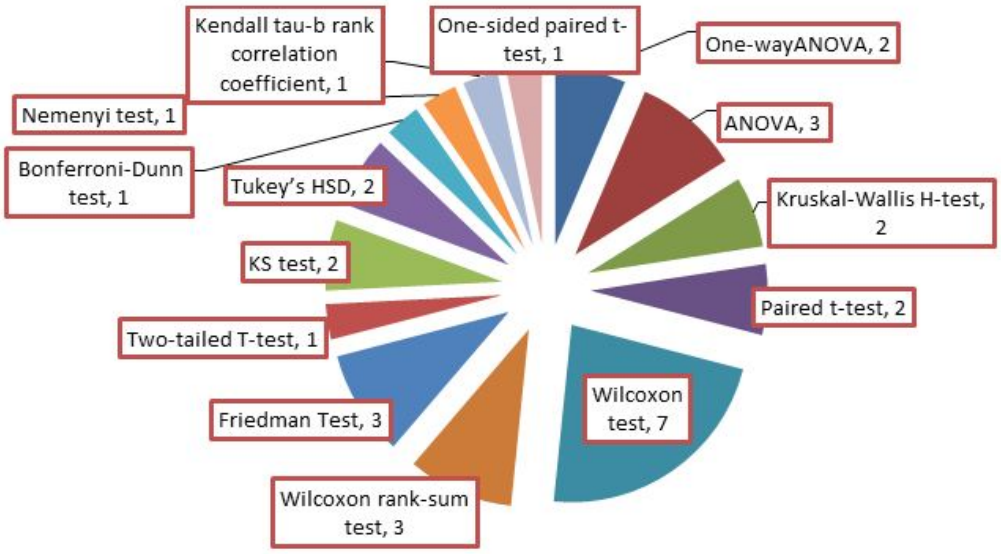

Figure 3.11: Statistical test used

Type of TL (RQ3.7)

This section describes the various categories of the TL method used in the studies. There are three different categories of TL methods, such as Transductive TL (TdTL), Inductive TL (IdTL), and Unsupervised TL (UnTL). These categories can also be termed TL as settings in which the TL algorithms have been performed. The TL categories and their settings are illustrated in Table 3.12. These categories differentiate from each other based on the type of source data, type of target data, source and target domain, and source and target task. It has been observed that most of the studies employed feature TL, and instance TL with IdTL. Moreover, relational knowledge and parameter transfer are also feasible with IdTL. However, in the existing studies authors explored feature representation, and instance transfer-based learning. The knowledge transfer is easy with the features of different projects. Feature transfer considers the features of the source and

target domain. Further, a correlation needs to be established between the features of both projects. Based on feature similarity, either direct features will be extracted, or a feature matching analyzer will be used if there is a huge amount of dissimilarity among the features of the source and target project. The parameter transfer is used when the algorithm used for transferring knowledge using default parameters value change, and the target project sets its algorithm parameter value according to the source project, this is accomplished for parameter TL. Hyperparameter optimization can also be employed for parameter TL. In relational knowledge transfer, a relationship needs to be established among the source project dataset, and using that prediction model must be designed. Furthermore, this prediction model would be used for knowledge transfer using the same methodology in the target dataset. In instance type transfer, the knowledge is shared based on the instances of the source project. The dataset must be preprocessed to apply instance TL. Thus, based on the analysis of existing literature, it has been concluded that feature transfer is more effective and efficient with TL in the software engineering domain.

We have observed that TdTL (34.28%) has been widely used among all the categories. IdTL (28.57%) has been used in only those studies that are related to multi-task learning and self-taught learning. The last category UnTL has not been used in any of the studies. Most of the studies considered the labeled data in the source domain while in the case of UnTL source domain labels are not available, and target domain labels are not available. Four different approaches correspond to these TL settings, such as instance transfer, feature-representation transfer, parameter transfer, and relational-knowledge transfer. For InTL setting, instance transfer is mostly used, and for TnTL, feature-representation transfer is mostly used. The count of studies for each approach corresponds to different TL settings presented in Fig. 3.12. Also, if a huge number of features are available, then a specified

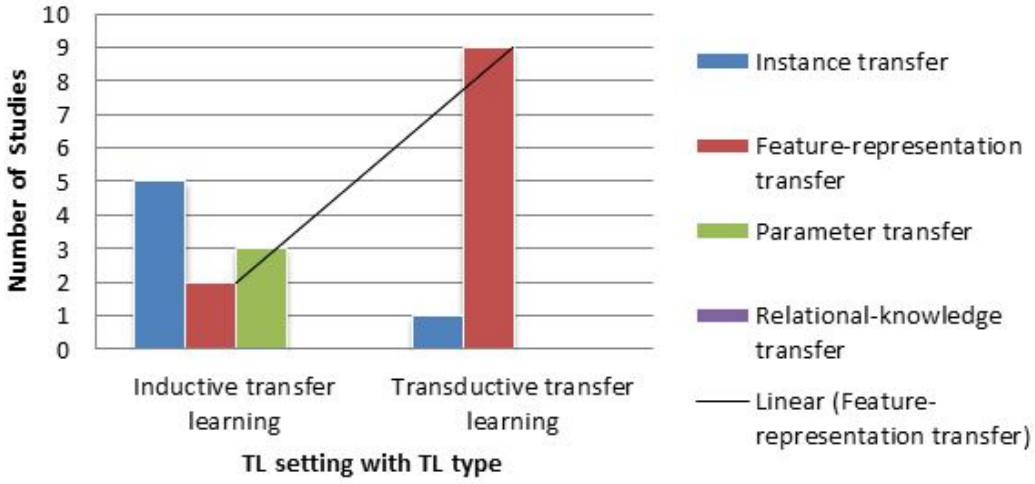

Figure 3.12: Type of transfer approach corresponding to different transfer learning settings

FS technique must be applied to select only relevant features.

3.3.4 Results Specific to RQ4

This section discusses the TL algorithm, which is effective against the various traditional learners. Traditional learners include ML techniques, which are compared with the proposed algorithm by different authors. These studies have used different datasets over which comparisons have been made. We have observed the values of accuracy, AUC, Recall, and F-measure for analyzing the performance of TL algorithms. However, these four metrics are mostly used in the existing studies. In the comparative analysis concerning existing studies, the combined dataset of results is collected and outliers are removed. Moreover, outliers lead to unbiased results corresponding to specified datasets. Thus, a boxplot is used to remove these outliers. Fig. 3.13 presents the distribution of studies concerning accuracy value corresponding to all the datasets majorly used. Fig. 3.14 and

3.15 present the distribution of studies concerning AUC value corresponding to all the datasets majorly used. Fig. 3.16 presents the distribution of studies concerning Recall value corresponding to all the datasets majorly used. Fig. 3.17 presents the distribution of studies concerning F-measure value corresponding to all the datasets majorly used. The descriptive statistics of all the performance measure with respect to TL techniques are presented in Table 3.10 including minimum, maximum, mean, median, and standard deviation measure values.

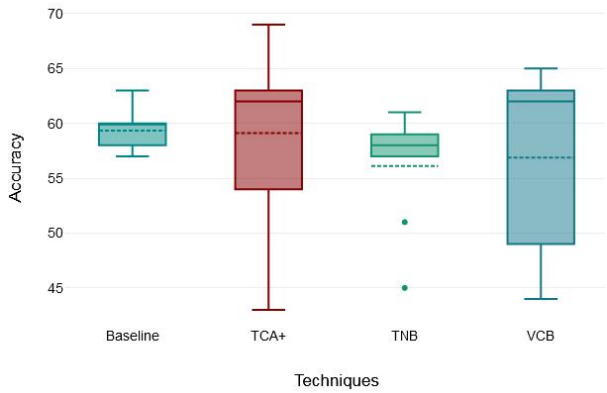

Figure 3.13: Dataset-wise accuracy for TL techniques used

In the study given by [108], experiments have been performed on various TL and ML algorithms on different datasets. The various TL algorithms that are used in the two studies are GTL, TCA, TJM, and GFK. These algorithms have been tested on five distortion profiles. It has been observed that the traditional ML algorithm that is RF performed best. GFK and TJM algorithms provided the worst result. The base classifier is same for both algorithms. Other base classifiers are flexible to noisy datasets, unlike 1-NN classifier, due to which these two algorithms performance results in the worst performance. When SVM is used as a base classifier, then TCA algorithm results in the worst performance.

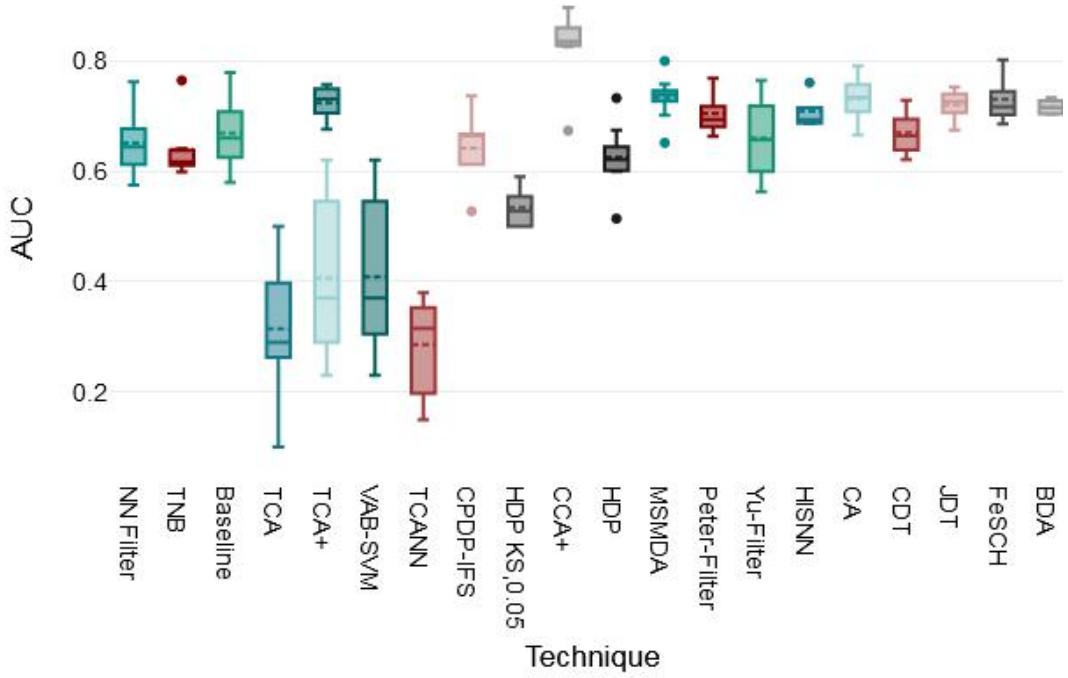

Figure 3.14: Dataset-wise AUC for TL techniques used

The performance of the ARTL algorithm is proved to be best in comparison with other TL algorithms that have been used in the study. The ARTL algorithm has attempted to resolve boundary and optional distribution differences, which can be a reason for the best performance of the ARTL algorithm. The overall conclusion of the study stated that the TCA algorithm performs best out of all the TL algorithms that have been used for comparison. The TJM algorithm is second best after the TCA algorithm. All of these algorithms perform best or worst on different distortions. The ARTL algorithm comes third after the TJM algorithm.

In the study given by [108], five different TL methods were compared on different datasets with a different statistical test against seven different base learners. The five TL

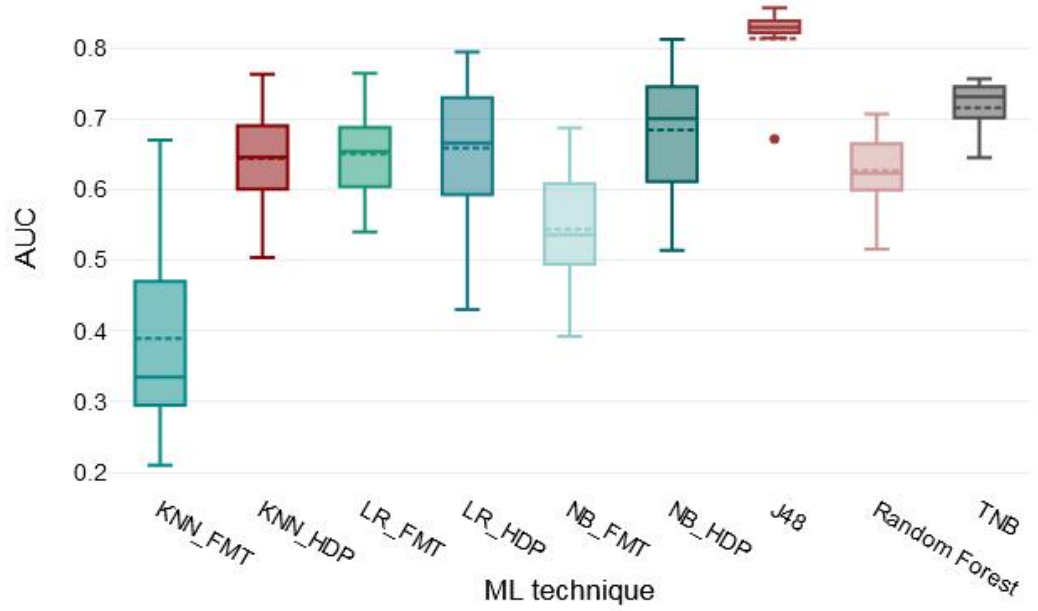

Figure 3.15: Dataset-wise AUC for ML techniques used

algorithms used are as follows: GFK, JDA, TJM, TKL, and TCA. The seven different base learners are RF, SVM, Discriminant Analysis, LR, 5NN, DT, and NB. AUC values computed for four base learners corresponding to the MAG, USPS, CCC, and CV datasets. In the next step, accuracy has been calculated for all algorithms corresponding to seven different distortion profiles. In the third step, accuracy has been computed over seven base learners corresponding to each TL algorithm. The best base learner for the TL algorithm individually investigated by Tukey's HSD test and assigned the HSD group for every accuracy value.

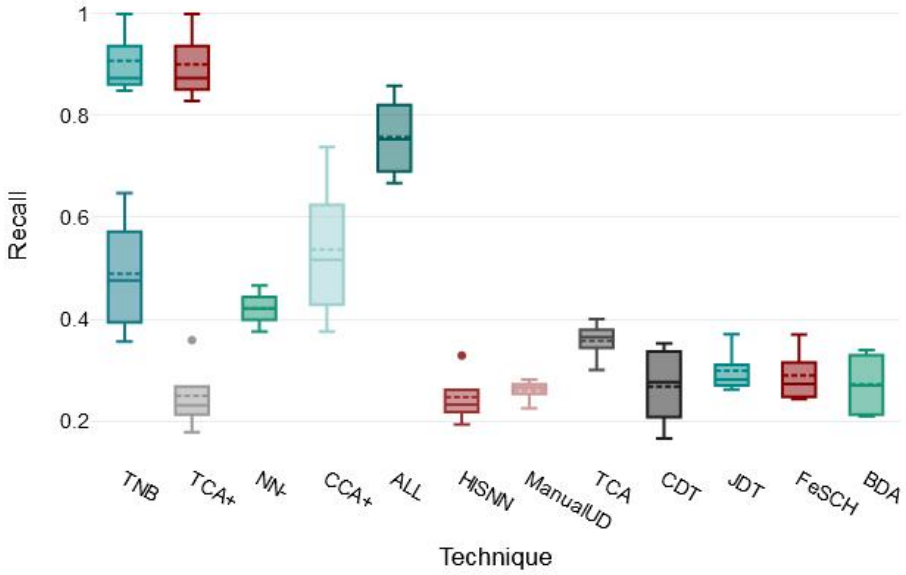

Figure 3.16: Dataset-wise Recall values for TL techniques used

3.3.5 Results Specific to RQ5

The threats to the validity for TL based on the similarity between domains, the kind of data used in the source, and the target domain are discussed in this section. Based on data distribution across different domains, [109] proposed a dimensionality reduction algorithm for effective TL when data is distributed across different domains. There exist different latent factors over different domains. TL uses labeled data from a similar learning task. TL uses available data for learning on the target data, which consists of both training and target test data. For TL, the source and target data have similarities, and there exists a relationship between them. It has been examined from all the primary studies that have been studied for this review that the data must be specified in the target and training domain based on the algorithm.

Table 3.10: Descriptive statistics of performance measure in the existing studies

Technique	**Performance Measure**	**Minimum**	**Maximum**	**Mean**	**Median ± Standard Deviation**
Baseline (LR)	AUC	0.5796	0.778	0.6688	0.6688 ± 0.06426
	Accuracy	57	63	59.333	59.333 ± 1.83356
	F-measure	0.1	0.69	0.3538	0.3538 ± 0.14655
BDA	AUC	0.651	0.704	0.6737	0.6737 ± 0.0191
	AUC	0.703	0.733	0.07165	0.07165 ± 0.0132382
	Recall	0.305	0.404	0.36175	0.36175 ± 0.036
CA	AUC	0.665	0.79	0.73075	0.73075 ± 0.0451
CC	AUC	1	6	2.698113	2.698113 ± 0.8813
	F-measure	0.1686	0.5097	0.334	0.334 ± 0.139
CCA+	AUC	0.6732	0.869	0.814	0.814 ± 0.0651
	Recall	0.67	0.86	0.76	0.76 ± 0.07778
	F-measure	0.55	0.84	0.712	0.712 ± 0.1042
CDT	AUC	0.621	0.728	0.669	0.669 ± 0.04063
	Recall	0.267	0.375	0.3035	0.3035 ± 0.0424
CPDP-CM	AUC	0.654	0.695	0.6628	0.6628 ± 0.01614
CPDP-IFS	AUC	0.527	0.736	0.6419	0.6419 ± 0.06944
FeSCH	AUC	0.685	0.801	0.72975	0.72975 ± 0.04371
	Recall	0.214	0.344	0.276	0.276 ± 0.0604
HDP	AUC	0.5139	0.721	0.6326	0.6326 ± 0.63125
HDP KS,0.05	AUC	0.5	0.59	0.5344	0.5344 ± 0.343
HISNN	AUC	0.686	0.76	0.708	0.708 ± 0.3045
	Recall	0.23	0.286	0.264	0.264 ± 0.02
J48	AUC	0.6712	0.8462	0.8033	0.8033 ± 0.0599
JDT	AUC	0.674	0.752	0.7195	0.7195 ± 0.029
	Recall	0.249	0.38	0.2995	0.2995 ± 0.0547
KNN_FMT	AUC	0.21	0.67	0.389	0.389 ± 0.1508
KNN_HDP	AUC	0.504	0.763	0.644	0.644 ± 0.06054
KNN_RM	AUC	0.467	1.758	0.6812	0.6812 ± 0.2474
LR_FMT	AUC	0.54	0.64	0.6498	0.6498 ± 0.0542
LR_HDP	AUC	0.43	0.794	0.65817	0.65817 ± 0.08938
LR_RM	AUC	0.414	1.826	0.7143	0.7143 ± 0.6725
ManualUD	AUC	1	1	1	1 ± 0
	Recall	0.222	0.451	0.345	0.345 ± 0.095
MSMDA	AUC	0.6511	0.7993	0.73126	0.73126 ± 0.04618
MTDP	AUC	1	5	3	3 ± 1.414214

NB_FMT	AUC	0.392	0.687	0.5433	0.5433 ± 0.07829
NB_HDP	AUC	0.514	0.812	0.6836	0.6836 ± 0.076
NB_RM	AUC	0.422	1.816	0.699	0.699 ± 0.26057
NN-	Recall	0.36	0.65	0.4925	0.4925 ± 0.1145
	F-measure	0.34	0.59	0.45	0.45 ± 0.09233
NN Filter	AUC	0.5745	0.7612	0.6508	0.6508 ± 0.05828
	F-measure	0.1407	0.5372	0.3239	0.3239 ± 0.14125
	AUC	0.66662	0.8361	0.79553	0.79553 ± 0.0584
	Recall	0.85	1	0.908	0.908 ± 0.0656
Peter-Filter	AUC	0.663	0.768	0.70425	0.70425 ± 0.0391
	Recall	0.235	0.325	0.27	0.27 ± 0.036
RF	AUC	0.5157	0.6564	0.6024	0.6024 ± 0.4403627
TCA	AUC	0.1	0.5	0.314	0.314 ± 0.10297
	Recall	0.183	0.363	0.254	0.254 ± 0.0665
	F-measure	0.23	0.72	0.45	0.45 ± 0.1498
TCA+	Accuracy	43	69	59.111	59.111 ± 7.4889
	AUC	0.676	0.757	0.7235	0.7235 ± 0.0314
	Recall	0.38	0.47	0.425	0.425 ± 0.045
	F-measure	0.23	0.72	0.454	0.454 ± 0.1484
	F-measure	0.31	0.36	0.336	0.336 ± 0.025
TCANN	AUC	0.15	0.38	0.2855	0.2855 ± 0.0794
	F-measure	0.21	0.76	0.4515	0.4515 ± 0.1826
TNB	Accuracy	45	61	56.111	56.111 ± 4.7
	AUC	0.5981	0.7641	0.6392714	0.6392714 ± 0.0528
	Recall	0.83	1	0.901	0.901 ± 0.0719
	F-measure	0.3	0.44	0.36	0.36 ± 0.0524
VABSVM	AUC	0.23	0.62	0.4085	0.4085 ± 0.1263
	F-measure	0.15	0.57	0.3234	0.3234 ± 0.1079
VCB	Accuracy	44	65	56.8889	56.8889 ± 7.578
YuFilter	AUC	0.563	0.764	0.66	0.66 ± 0.075
	Recall	0.198	0.333	0.2525	0.2525 ± 0.0499

3.3.6 Results Specific to RQ6

The advantages and disadvantages of the TL techniques supported by the studies are discussed in this section. Table 3.11 summarizes the advantages and disadvantages of TL

techniques.

Table 3.11: Advantages and Disadvantages of TL techniques

TL Technique	**Advantages and Disadvantages**
Effectiveness	This attribute has the ability to provide the desired output or the capability of providing the desired output.
Task-clustering	It provides an idea to achieve inductive transfer in classifier design with the help of labeled data from the related classification problems to solve a particular classification problem.
TNB	It improves the performance of the dataset collected from various companies or cross-company data.
Discriminability Based TL (DBT)	It has been demonstrated that the destination networks that are initialized via DBT learn much faster than networks that are initialized randomly. DBT indicates considerable and important learning speed improvement across randomly initialized networks. DBT is superior in comparison to literal transfer, and to directly use the destination network on the destination task.
GTL	The main focus of GTL is TdTL. In TdTL, the domain has generously labeled examples, while the destination domain consists of unlabeled examples only. GTL does not cover the latent features under various domains as the bridge to transfer knowledge simultaneously. It results in maximizing the empirical likelihood of all the domains and conserving the geometric structure in every domain.
Instance-based techniques	**Advantages:** These techniques are used for handling instances by removing the outliers, relevant filtering, or weighting of instances.
Distribution-based techniques	These techniques aim at managing the instance distribution for training and testing sets with the help of stratification, cost curves, and mixture models.
EL technique	**Advantages:** This technique performs better than a trained classifier using huge amount of labeled data in the destination domain.

TCA	**Advantages:** TCA learns a similar transfer component that comes under both domains such as the difference in the distribution of data across various domains. It can be reduced if it is projected on the subspace, and it conserves the various data properties. It is beneficial to use traditional ML methods in this subspace to train classification and regression over various domains. If two or more domains are associated with each other, then there may exist various similar components under them, due to which the partitioning of data across domains is to be distinguished.
Hybrid HeTL	**Advantages:** It is useful for transferring knowledge over various feature scopes and concurrently rectifying the data error on the transformed feature space. The performance of Hybrid HeTL is best and more stable when the size of parallel data is increased. The Hybrid HeTL is effective and robust for cross-language sentiment classification.
GA for Feature-Space Remapping (GAFSR) and Greedy Search for Feature-Space Remapping (GrFSR)	**Advantages:** These techniques are informed, supervised learning techniques. The benefit of FSR is that it can be applicable for both cases that are either informed or uninformed. The main advantage of GAFSR is that it achieves the best performance scores across all the metrics. **Disadvantages:** It takes more time to execute in comparison to IFS. In IFS, the computation count is low, but the performance score is high.
Stacking	**Advantages:** It is beneficial in terms of combining stacking with IFSR, and IFSR uses labeled data. **Disadvantage:** To train ensemble classifiers, it needs labeled data.
Canonical Correlation Analysis (CCA)	**Advantages:** It is an effective TL method. It is used to make the distribution among training and testing data of companies. CCA with CCDP is effective for HCCDP. CCA acts as a powerful tool in multivariate data analysis to establish the correlation between two different sets of variables.

FSR	**Advantages:** It can manage various feature spaces without using any co-occurrence data. This technique uses originally raw data which is already mapped on a feature space, and this is why this technique is also known as a remapping name. It requires a low amount of labeled data in its target domain. This labeled data is required to understand the relations to the training domain. It can increase the classification accuracy in the target domain by combining the relevant information from the training domain with the help of ensemble learners.
TNB	**Advantages:** TNB performs better for SOFTLAB dataset, and it does not perform better for NASA dataset. TNB works for both within-company as well as cross-company. The author focused on cross-company defect prediction. TNB outperforms naive bayes in the context of performance measures such as F-measure, AUC over within company, and cross-company defect prediction. **Disadvantages:** TNB is limited to a particular company dataset.
Voting EL	**Advantages:** It is defined as the simplest method for combining multiple classifiers. Bellwether can be used efficiently when the availability of historical data is limited or negligible. Due to a lack of historical data, developers try to get data from other projects. It has been examined that irrespective of the granularity of data, there exists a bellwether dataset that can be used for the training of defect prediction models. The bellwether does not require brief data mining methods to discover.

3.4 Discussion

We have studied and examined the various TL algorithms in the fields of AI, ML, and software engineering. A deep analysis followed by a sequence of systematic points and identified 39 primary studies during this period (1991-2024). Secondly, the quality

attributes that are focused on TL are discussed. Thirdly, the characteristics or experimental settings of the primary studies have been discussed based on the datasets, independent variables, TL algorithms, validation techniques, performance measures, and statistical tests. Fourth, we have analyzed the comparison of various TL techniques with traditional ML algorithms as a base learner. Lastly, the merits and demerits of TL techniques are summarized. The relevant outcomes obtained from the primary studies selected for this review are as follows:

- The quality attributes that have been used for TL are defect, effectiveness, performance, reliability, and effort. The most commonly used quality attribute is performance and effectiveness used in 32%, 23% of studies. There is no study conducted for change prediction using TL.
- The ML techniques were categorized into different classes such as SVM, EL, DT, BL, NB, and Miscellaneous. The mostly used ML techniques for TL were SVM, RF, and NB.
- The most commonly used dataset for performing experiments is NASA in the literature. The second and third most commonly used datasets is UCI 20 Newsgroups and AEEEM.
- The independent variables that have been used by various studies do not exhibit any relationship with each other.
- The algorithms that have been used by the selected studies differ and these algorithms depend on the type of training and target dataset.

- The validation technique that has been used in most of the primary studies is *K*-fold cross-validation. In *K*-fold cross-validation, the original dataset is used for both training as well as validation, and it uses every sample for validation exactly once.
- The performance measure that has been used by most studies is accuracy followed by AUC, F-measure, and recall.
- The TL categories used in the selected primary studies are IdTL and TdTL. UnTL has not been used in any study. The instance transfer setting has been mostly used in IdTL, and feature-representation transfer has been mostly used in TdTL.
- The outcome of systematic review corresponding to characteristics of TL, IdTL is mostly used in the existing study. The features and instances of different projects play an important role in transferring knowledge between two different projects. The performance of TL models was evaluated using the accuracy, adaptability, scalability, and generalization power of the predictive model. Thus, it has been analyzed that TL-based algorithms satisfy the above characteristics for the development of efficient software quality models for future projects.
- The comparison made using SVM as a base learner provides the best classification accuracy in comparison to other base learners. The comparison made with NB as a base learner provided the worst classification accuracy in most of the cases.
- These are some of the instructions for the researcher scholars, industry experts, and software developers to carry out future research on TL in software engineering:

1. More experiments should be carried out to increase the studies for TL in software

engineering. The number of studies using TL in software engineering must be increased to show the benefits of TL in software engineering.

2. The experimental settings should be specified in the study. Datasets used, independent variables used, performance measures, and the statistical test used in your study.

3. More studies should be carried out for change prediction using different types of TL.

4. To obtain more accurate, precise, and generalized results, more studies should be carried out across cross-project and cross-company prediction using TL.

5. More studies for TL should be carried out using ML techniques.

6. More studies should be carried out using different settings and approaches of TL. Each TL type should be explored corresponding to all the TL categories.

7. It is observed that there are very few studies that transfer knowledge using similar TL techniques or TL algorithms.

8. More studies should be conducted considering hybrid/evolutionary, and swarm-based algorithms.

9. More studies must be conducted to analyze the impact of FS techniques with TL.

10. More studies must be conducted with bio-inspired algorithms for TL.

11. The effectiveness of TL models must be analyzed with hyperparameter optimization.

Chapter 4

Cross-Project Defect Prediction Model using Transfer Learning

4.1 Introduction

In software engineering field, defect prediction is one of the important research area. In the last few years, defect prediction is exhaustively studied [30]. SDP aims to detect defect prone modules at different levels, such as functions, classes, and files. It improves software quality, reduces cost, and development time. It helps in software quality assurance in terms of software development effort, software testing or code review. It plays a significant role when an organization has limited resources. Over the past few years, researchers have developed various methods for defect prediction at an early stage.

TL is a new research area for transferring knowledge across different projects. The concept of TL is to learn some knowledge while solving some problems and apply that

knowledge to a similar problem. TL states that the knowledge extracted or learned from the existing code can be applied to the innovative system. TL concept has been used for DP. TL is used such that the model is trained on one dataset which can be also termed as pre-trained model. Now, pre-trained model is used to test another project of similar characteristics. Feature based TL is used such that the two training and testing dataset have a similar features to some extent. In order to select the matching features we have used algorithm for selection of matching pairs out of all the features for development of prediction model. TL is used to obtain generalized and unbiased results.

Nowadays, researchers, software experts, and academicians are working beyond WPDP. Moreover, in the upcoming years, the systems are designed with innovative ideas and techniques. The larger part of existing codes is used for the designing of innovative systems. However, using existing code, which is designed by different peoples with different techniques, was difficult to understand by other people. Thus, the concept of TL has emerged.

Data on various open source projects has been collected for development of prediction model. These software data consist of different types of software metrics and label such as defective or non-defective. Software metrics are defined as a quantitative measure. These metrics are a descriptive analysis of software data. There are various types of software metrics discussed in Chapter 2. The label indicates the presence of a defect in the project's data. Labels are used for the indication of defect presence.

The proposed models for defect prediction are designed for WPDP and HetDP. In WPDP, training and testing of model has been on the same project. Each instance in WPDP, represents particular module of a project. The presence of labels corresponding to each instance whether the module is defective or not, i.e., defect proneness tendency of a project

is present in the column of a project dataset. In WPDP, the prediction model is trained on some part of Project X, and testing of the prediction model is done on some part of Project X. However, WPDP has various limitations. WPDP can not be used when we have to make a defect prediction for completely new projects. In some other cases also, when different people were working on the same project for a longer duration, then many changes are introduced in the code. With the help of the previous code, we can not predict the defect for a newer version or with new changes in the code. To overcome this issue, researchers and academicians have started exploring CPDP.

CPDP takes two different projects for training and testing. In CPDP, Project A can be used for training, and Project B can be used for testing. In this way, CPDP can be used for projects having lack of historical data. Till now, most of the researchers and academician have explored CPDP for the projects that have same software metric; that limits the CPDP. Thus, HetDP approach has been proposed. HetDP allows prediction between two projects if they use different metrics. There exist various limitations of CPDP [30]. The open-source, available datasets consist of different metric sets. In heterogeneous data, all the projects have completely different metrics. For example, the datasets of AEEEM group consist of 61 software metrics and the NASA dataset in the PROMISE repository consist of 31 software metrics. There exist various studies in literature based on cross-project validation [3, 110, 111]. However, most of the authors have not developed models for generalized results. Thus, to achieve generalized results, we have developed a predictive model for defect prediction using different ML algorithms across different projects.

We have analyzed the HetDP approach for defect prediction across various projects with different metrics suite. Thus, using HetDP, researchers or experts can develop other

prediction models to predict defects. HetDP finds the matching metrics having same distribution from source (Project A) to target (Project B) dataset. For effective knowledge transfer, we have also demonstrated the size of a source and target dataset. To achieve our aim of improving the model for defect prediction using heterogeneous metrics, we have formulated the research questions. The RQs are as follows: RQ1: Does the FS is necessary for WPDP? RQ2: How does WPDP models performed using 10-fold cross-validation? RQ3: Which ML algorithm performs best for HetDP using TL? RQ4: How does HetDP models performed using 10-fold cross-validation in comparison to WPDP?

The main contribution of conducting experiment in this chapter is to analyze, experiment and assess the empirical evidence regarding (1) the importance of FS for WPDP (2) performance capability of ML techniques for HetDP (3) predictive performance capability of HetDP and WPDP in comparison to 10-fold cross validation (4) the applicability of CPDP. Defect prediction models are developed using different ML algorithms to produce generalized results using TL.

This chapter is organized as follows: Section 4.2 describes the research background with respect to experiment conducted in this chapter. Section 4.3 discusses the results of the chapter in the form of answers to RQs. Section 4.4 stated the key findings of the chapter. The results of this chapter are published in [112].

4.2 Research Background

The research background of this chapter is discussed in this section.

4.2.1 Dataset used

The dataset used for experimentation is referred in Chapter 2 Section 2.7.3.1.

We have discussed HetDP with different datasets of two different groups. In HetDP, we have considered two different datasets, one is source dataset, and another one is the target dataset, with different metrics. The instance is represented by a tuple of dataset, and metrics are represented by an attribute of a dataset. The last attribute of a dataset represents a label for the presence and absence of defects. In Figure 4.1, we have presented the framework for HetDP. In the first step, we have selected source (X) and target project (Y). The source and target dataset have a different metrics set. In the second step, we have selected metrics from both the projects. We have selected equal number of metrics from source and target dataset. In the next level, we find matching metrics for the target dataset into the source dataset. The matching metrics in the source and target dataset are found based on the similarity among them. The metrics matching performed with the help of a correlation analyzer. At the end of this process, we have a set of matched metrics from source and target dataset. The matched source dataset has been trained using different ML algorithms. This step results in a prediction model that can be further used for predicting labels of the target dataset.

The WPDP has been explained in Figure 4.2. WPDP considers a single project data due to lack of traditional data. In WPDP, we have divided a dataset in two different parts. One part of a dataset is training data and another part is testing data. In the second step, it builds a model for the training data, and it uses that model for predicting labels of test data. In the next subsections, we have explained the approach in more detail.

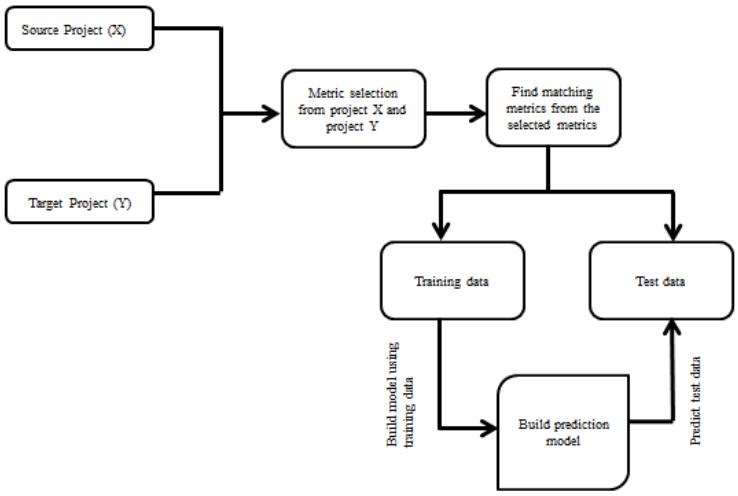

Figure 4.1: Framework for Heterogeneous Defect Prediction

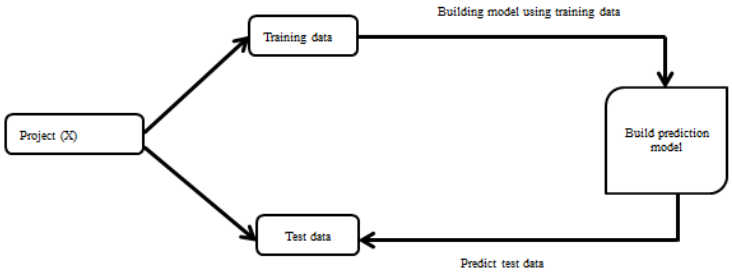

Figure 4.2: Framework for Within Project Defect Prediction

4.2.2 Data Collection

We have collected data from an open-source repository. NASA and PROMISE group dataset are used in the study. The dataset collected from these groups has different projects and attributes. These dataset consist of various software metrics. Each software metric provided with some numerical value corresponding to each module of a project. The last

column corresponding to each module of a project provides a label for defect proneness. The defect proneness attribute is considered as an dependent variable.

4.2.3 Development of Prediction Model

The ML algorithms used for experimentation are LR, RF, HV, MLP, ADB, and DT. These ML algorithms have been discussed below:

1. LR:- LR is named after a function called as sigmoid function. This function is interpreted by an S-shaped curve that takes a real-valued number and maps it to values in the range of 0 and 1, but it should not be at the extreme points. The coefficients of the LR algorithm are determined from training data.

2. RF:- The name of this technique itself specifies that it consists of various DT. These individual DT operate as groups, i.e., ensemble. In this technique, each RF redicts an output class label, and the class having majority counts represents the prediction model. RF provides an understandable model for researchers and academician. This algorithm is robust to noisy and missing values of the dataset.

3. HV:- In HV technique, various ML algorithms are combined to build a prediction model.

4. MLP:- This technique consists of multiple perceptrons. It is focused on creating robust algorithms and used those algorithms for building a prediction model. However, NN can learn the training data representation easily, and it easily predicts the label of the output variable. These networks can learn the mapping function, and they are proved as a global algorithm. The predictive capability of NN comes from

the hierarchical or multi-layered structure of the networks. It takes different features of input data over various scales and group them into high-order features.

5. ADB:- This classifier also makes a prediction. It is similar to the RF classifier. It applies multiple DT to individual samples, and it takes a combination of these predictions from each DT or majority trees in the classification problem. Each DT contributes a significant amount to the final prediction.

6. DT:- This technique works on the divide and conquer concept. It uses a heuristic approach known as recursive partitioning. This approach is based on the divide and conquer problem. It divides the original dataset into small groups, which are repeatedly split into even smaller groups. It solves each subproblem. In the end, the solution of all subproblems combined to produce solution to the actual problem. DT is easy to apply and easy for development of prediction models. It is not based on any assumption. It is not affected by the presence of outliers.

4.2.4 Cross-Validation Method

Cross-validation is a technique used for model evaluation by partitioning the given dataset into two parts [3]. One part is used for training, and another part is used for testing data. The data is divided into respective ratio and proportion using cross-validation method. The model is trained using training dataset. The prediction model is used for making predictions of testing data. The division of dataset into training and testing dataset depend on the type of cross-validation method used [113]. There exist three types of cross-validation methods, such as hold-out validation, K-fold cross-validation, and leave-one-out cross-validation. We have used K-fold cross-validation for conducting experiment.

In *K*-fold cross-validation, the dataset is divided into *K* different parts. *K*-1 parts are used for training the model, and the remaining part is used for testing. The value of *K* is considered as 10. In 10-fold cross-validation, the complete dataset is partitioned into ten parts, and each part is of equal size. Nine parts are randomly selected for training, while the remaining part, i.e., the tenth part, is used as a validation set. This process iterates for ten times. In the end, the results of each iteration would be combined. *K*-fold cross-validation method is mostly used by the researchers/academician and professionals in empirical studies [114].

4.2.5 Performance Metrics

There exist various performance metrics/ measures. These metrics/ measures are used to evaluate and analyze the performance of different ML algorithms. Different performance metrics exist for classification and regression problems. The selection of performance metrics/measures is very crucial. We have used AUC-ROC [115] performance metric. It helps in identifying the goodness of a developed prediction model. AUC-ROC curve is represented with a curve between recall and specificity. The value of AUC lies from 0 to 1. Depending on AUC values, we can predict how good or bad the developed model [115]. The significance of AUC corresponding to 0, 0.5, and 1 has been represented in Eqn. 4.1 given below.

$$AUC = \begin{cases} 0, \ the\ developed\ model\ wrongly\ classifies\ output\ class \\ 0.5, \ no\ discrimination\ between\ output\ classes \\ 1, \ the\ developed\ model\ perfectly\ classifies\ output\ class \end{cases} \tag{4.1}$$

4.2.6 Statistical Test

The statistical test is used for validation of results. The statistical test is used to check if there is any significant difference between the ML techniques used. Friedman test is used for independent groups. The data can be randomly selected from any distribution for the Friedman test [3]. This test does not consider any assumption; i.e., it is a non-parametric test. With the help of these, we have checked if there is any significant difference among the performance of ML algorithms used for HetDP. In the first step, we have designed two hypotheses: the null hypothesis H_o and the alternate hypothesis H_a. In the second step, we have obtained p_value at a significant level of 0.05. The null hypothesis is accepted if the p_value > 0.05. An alternate hypothesis is accepted if the p_value ≤ 0.05. The Wilcoxon–signed rank test performed pairwise comparison [116]. However, it is used to compare repeated measurements [117].

4.3 Results and Analysis

In this section, we have addressed the answers the RQs designed to experiment with open-source dataset. The results for the comparison of HetDP and WPDP are presented in this section.

4.3.1 Results Specific to RQ1

We have applied various FS techniques for selection of features from different projects. Chi-square method and Recursive Feature Elimination (RFE) method. We have observed that prediction model does not work efficiently after applying feature elimination method.

Moreover, the large number of features are reduced and irrelevant features are removed using FS technique. With FS techniques, we got AUC values near 0.5. AUC value of 0.5 does not make correct predictions. We have performed experiment without FS techniques. Based on the observation, the prediction model provided efficient result using FS technique. Hence, it is concluded that FS techniques are necessary for WPDP.

4.3.2 Results Specific to RQ2

We have used different ML algorithms to develop these predictive models. For WPDP, the AUC values are computed using different ML algorithms and 10-fold cross-validation method. The values of 12 datasets used for WPDP are shown in Table 4.1. All ML techniques performed significantly different based on the mathematical interpretation.

Table 4.1: AUC values for WPDP

Source dataset	**Target dataset**	**LR**	**RF**	**HV**	**MLP**	**ADB**	**DT**	**Average AUC value**
ant-1.7	ant-1.7	0.76	0.83	0.82	0.73	0.82	0.72	0.78
CM1	CM1	0.47	0.73	0.65	0.42	0.62	0.58	0.58
KC1	KC1	0.55	0.79	0.73	0.6	0.76	0.74	0.70
KC2	KC2	0.63	0.82	0.76	0.53	0.76	0.66	0.69
KC3	KC3	0.55	0.76	0.73	0.49	0.63	0.62	0.63
MC1	MC1	0.4	0.9	0.78	0.48	0.83	0.71	0.68
MC2	MC2	0.7	0.71	0.77	0.62	0.63	0.59	0.67
MW1	MW1	0.72	0.71	0.74	0.32	0.67	0.59	0.63
PC1	PC1	0.61	0.83	0.77	0.48	0.79	0.75	0.71
PC1	PC1	0.61	0.83	0.77	0.48	0.79	0.75	0.71
PC2	PC2	0.52	0.81	0.71	0.21	0.77	0.59	0.60
PC3	PC3	0.66	0.83	0.79	0.57	0.77	0.74	0.73
PC4	PC4	0.76	0.92	0.9	0.71	0.91	0.88	0.85

From the results obtained for WPDP, we have observed that RF technique performs better than

LR, HV, MLP, ADB, DT technique. RF works better for building a prediction model for WPDP as it takes a combination of individual trees. RF provides a binary classification of data points, but it also provides the probabilities for each instance belongs to defective and non-defective label. It is observed that AUC values are improved for all datasets. In some of the datasets, LR, and HV also performed efficiently.

4.3.3 Results Specific to RQ3

To answer this RQ, statistical test has been used. We have designed two hypotheses such as H_o and H_a.

H_o:There is no statistical difference between six ML algorithms.

H_a: There is a statistical difference between six ML algorithms.

Friedman test is used for testing the above hypothesis. The statistical test is used at a significance level of 0.05. The p_value obtained is 0.00 in this case. The value of Friedman's statistics is computed as 210.9. Hence, the p_value is less than 0.05; we reject H_o and accept H_a. In Figure 4.3, the Friedman mean rank corresponding to ML algorithms is presented. We have observed that the mean rank of RF is highest, i.e., 4.92. It implies that RF performs efficiently in comparison to other algorithms used in the study. ADB is the second-highest mean rank, i.e., 4.05, which implies that after RF, ADB is best performing ML algorithm for HetDP. We have also performed Wilcoxon–signed rank test to check the statistical difference among all the algorithms. The Wilcoxon–signed rank test is used to check if there is any statistical difference exist or not. The hypothesis testing is performed at 0.05 significance level. The p_value provided by Wilcoxon–signed rank test is 0.000. The computed p_value is less than the significance level. Thus, there is a statistical difference among the performance of ML algorithms. Hence, alternate hypothesis is accepted.

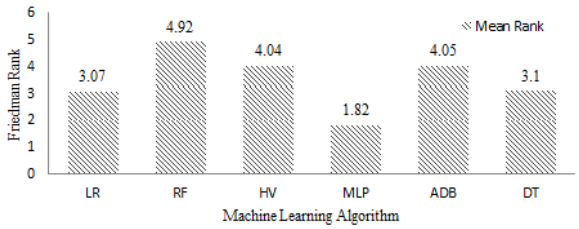

Figure 4.3: Friedman Mean Rank for ML Algorithm

4.3.4 Results Specific to RQ4

To answer this RQ, we have experimented on different datasets. In HetDP, the source and target dataset are different. This experiment has been observed at different cutoff_threshold for metrics matching. The metrics matching has been done by Spearman's correlation coefficient analyzer. In this analyzer, we have performed experiments by taking different threshold values, e.g., cut-off_threshold = 0.1, 0.05, 0.3, 0.9. We have observed in this analyzer that by changing the threshold to more than 0.05, it doesn't perform well. Thus, we have set the threshold value to 0.05. Different datasets are used as a target dataset. For which, we have executed our experiment iteratively, by setting different source and target dataset in each iteration. In Table [4.2-4.12] presented the AUC values corresponding to the ML algorithms used. The pairwise combination of the source and target dataset are presented separately in Table [4.2-4.12].

TL concept helps us in developing a generalized model. Generalized defect prediction model can be easily developed for industry or academic purpose. The software quality also improves with generalized prediction model. We have used existing software knowledge for building purpose. We can remove ambiguities in the new projects by using existed project's knowledge, patterns, design. It improves maintainability, understandability of a software. It would be easy to use prediction model for end-users. The feature-based TL method is used. The computed results were compared with existing studies also. AUC values for RF in HetDP and WPDP is 0.673 and 0.803. However,

the AUC values for RF in HetDP and WPDP in existing studies is 0.640 and 0.732. The AUC values for DT in HetDP and WPDP is 0.617 and 0.681. However, the AUC values for DT and WPDP in existing studies is 0.568 and 0.598. Hence, the computed AUC values indicated that RF and DT performed better for HetDP and WPDP based on the experiment performed.

Table 4.2: AUC values for HetDP (Source dataset:-11 different dataset, Target dataset:-ant-1.7)

Source dataset	**Target dataset**	**LR**	**RF**	**HV**	**MLP**	**ADB**	**DT**	**Average AUC value**
CM1	ant-1.7	0.67	0.71	0.69	0.57	0.72	0.67	0.67
KC1	ant-1.7	0.69	0.73	0.67	0.68	0.7	0.69	0.69
KC2	ant-1.7	0.68	0.72	0.67	0.58	0.67	0.7	0.67
KC3	ant-1.7	0.71	0.75	0.72	0.47	0.68	0.66	0.67
MC1	ant-1.7	0.61	0.72	0.66	0.54	0.69	0.66	0.65
MC2	ant-1.7	0.68	0.67	0.66	0.56	0.59	0.54	0.62
MW1	ant-1.7	0.65	0.63	0.64	0.56	0.61	0.6	0.62
PC1	ant-1.7	0.66	0.75	0.71	0.57	0.72	0.71	0.69
PC2	ant-1.7	0.68	0.73	0.7	0.62	0.72	0.69	0.69
PC3	ant-1.7	0.64	0.73	0.66	0.54	0.66	0.66	0.65
PC4	ant-1.7	0.63	0.73	0.67	0.56	0.71	0.69	0.67

Table 4.3: AUC values for HetDP (Source dataset:-10 different dataset, Target dataset:-CM1)

Source dataset	**Target dataset**	**LR**	**RF**	**HV**	**MLP**	**ADB**	**DT**	**Average AUC value**
ant-1.7	CM1	0.64	0.67	0.68	0.6	0.62	0.6	0.64
KC1	CM1	0.6	0.64	0.65	0.6	0.67	0.65	0.64
KC2	CM1	0.59	0.68	0.66	0.56	0.63	0.6	0.62
KC3	CM1	0.54	0.48	0.58	0.64	0.67	0.46	0.56
MC1	CM1	0.54	0.66	0.6	0.59	0.66	0.61	0.61
MW1	CM1	0.56	0.59	0.62	0.61	0.65	0.61	0.61

PC1	CM1	0.62	0.71	0.63	0.5	0.69	0.59	0.62
PC2	CM1	0.57	0.6	0.6	0.52	0.64	0.54	0.58
PC3	CM1	0.58	0.68	0.63	0.49	0.65	0.61	0.61
PC4	CM1	0.55	0.71	0.6	0.49	0.7	0.49	0.59

Table 4.4: AUC values for HetDP (Source dataset:-11 different dataset, Target dataset:-KC1)

Source dataset	**Target dataset**	**LR**	**RF**	**HV**	**MLP**	**ADB**	**DT**	**Average AUC value**
ant-1.7	KC1	0.68	0.76	0.69	0.75	0.74	0.68	0.72
CM1	KC1	0.64	0.69	0.65	0.61	0.71	0.65	0.66
KC2	KC1	0.61	0.77	0.72	0.57	0.76	0.69	0.69
KC3	KC1	0.71	0.71	0.72	0.6	0.78	0.72	0.71
MC1	KC1	0.59	0.75	0.69	0.62	0.75	0.73	0.69
MC2	KC1	0.54	0.69	0.69	0.51	0.63	0.65	0.62
MW1	KC1	0.64	0.74	0.71	0.56	0.68	0.64	0.66
PC1	KC1	0.67	0.75	0.68	0.62	0.72	0.74	0.7
PC2	KC1	0.68	0.73	0.72	0.61	0.74	0.68	0.69
PC3	KC1	0.62	0.74	0.71	0.55	0.74	0.69	0.68
PC4	KC1	0.62	0.75	0.7	0.62	0.74	0.75	0.7

Table 4.5: AUC values for HetDP (Source dataset:-11 different dataset, Target dataset:-KC2)

Source dataset	**Target dataset**	**LR**	**RF**	**HV**	**MLP**	**ADB**	**DT**	**Average AUC value**
ant-1.7	KC2	0.79	0.82	0.8	0.76	0.78	0.68	0.77
CM1	KC2	0.71	0.79	0.75	0.7	0.8	0.67	0.74
KC1	KC2	0.73	0.81	0.81	0.69	0.77	0.69	0.75
KC3	KC2	0.76	0.82	0.79	0.6	0.8	0.79	0.76
MC1	KC2	0.63	0.79	0.76	0.65	0.76	0.68	0.71
MC2	KC2	0.8	0.76	0.78	0.63	0.74	0.67	0.73
MW1	KC2	0.75	0.79	0.75	0.6	0.67	0.64	0.7
PC1	KC2	0.72	0.79	0.8	0.68	0.76	0.74	0.75

PC2	KC2	0.78	0.79	0.79	0.7	0.75	0.75	0.76
PC3	KC2	0.72	0.8	0.77	0.67	0.77	0.75	0.75
PC4	KC2	0.74	0.79	0.77	0.64	0.77	0.66	0.73

Table 4.6: AUC values for HetDP (Source dataset:-11 different dataset, Target dataset:-KC3)

Source dataset	**Target dataset**	**LR**	**RF**	**HV**	**MLP**	**ADB**	**DT**	**Average AUC value**
ant-1.7	KC3	0.78	0.84	0.79	0.8	0.77	0.74	0.79
CM1	KC3	0.54	0.58	0.6	0.5	0.47	0.58	0.55
KC1	KC3	0.56	0.61	0.58	0.45	0.45	0.44	0.52
KC2	KC3	0.63	0.64	0.65	0.49	0.6	0.51	0.59
MC1	KC3	0.59	0.62	0.64	0.48	0.56	0.55	0.57
MC2	KC3	0.51	0.57	0.46	0.39	0.52	0.49	0.49
MW1	KC3	0.66	0.63	0.6	0.51	0.69	0.42	0.59
PC1	KC3	0.66	0.6	0.64	0.46	0.59	0.53	0.58
PC2	KC3	0.63	0.57	0.66	0.56	0.55	0.46	0.57
PC3	KC3	0.46	0.54	0.52	0.51	0.58	0.52	0.52
PC4	KC3	0.59	0.52	0.56	0.6	0.56	0.52	0.56

Table 4.7: AUC values for HetDP (Source dataset:-9 different dataset, Target dataset:-MC1)

Source dataset	**Target dataset**	**LR**	**RF**	**HV**	**MLP**	**ADB**	**DT**	**Average AUC value**
ant-1.7	MC1	0.62	0.74	0.62	0.52	0.56	0.54	0.6
CM1	MC1	0.5	0.6	0.59	0.35	0.59	0.56	0.53
KC1	MC1	0.5	0.62	0.62	0.54	0.61	0.67	0.59
KC2	MC1	0.5	0.52	0.59	0.41	0.5	0.5	0.5
MW1	MC1	0.62	0.64	0.55	0.53	0.61	0.52	0.58
PC1	MC1	0.41	0.42	0.51	0.42	0.49	0.51	0.46
PC2	MC1	0.53	0.47	0.5	0.55	0.47	0.56	0.51
PC3	MC1	0.58	0.62	0.62	0.59	0.59	0.46	0.58
PC4	MC1	0.54	0.65	0.64	0.55	0.61	0.65	0.61

Table 4.8: AAUC values for HetDP (Source dataset:-11 different dataset, Target dataset:-MC2)

Source dataset	Target dataset	LR	RF	HV	MLP	ADB	DT	Average AUC value
ant-1.7	MC2	0.61	0.46	0.54	0.49	0.46	0.57	0.52
CM1	MC2	0.67	0.68	0.66	0.57	0.67	0.56	0.64
KC1	MC2	0.78	0.76	0.77	0.64	0.81	0.72	0.75
KC2	MC2	0.7	0.71	0.74	0.56	0.61	0.7	0.67
KC3	MC2	0.61	0.66	0.57	0.48	0.59	0.61	0.59
MC1	MC2	0.67	0.59	0.62	0.49	0.57	0.54	0.58
MW1	MC2	0.61	0.52	0.64	0.5	0.56	0.48	0.55
PC1	MC2	0.66	0.56	0.61	0.59	0.55	0.57	0.59
PC2	MC2	0.65	0.57	0.64	0.49	0.46	0.57	0.56
PC3	MC2	0.58	0.57	0.55	0.51	0.57	0.55	0.56
PC4	MC2	0.56	0.56	0.56	0.48	0.6	0.42	0.53

Table 4.9: AUC values for HetDP (Source dataset:-11 different dataset, Target dataset:-MW1

Source dataset	Target dataset	LR	RF	HV	MLP	ADB	DT	Average AUC value
ant-1.7	MW1	0.56	0.63	0.56	0.52	0.57	0.51	0.56
CM1	MW1	0.56	0.68	0.62	0.53	0.55	0.55	0.58
KC1	MW1	0.54	0.66	0.6	0.47	0.56	0.56	0.57
KC2	MW1	0.62	0.61	0.66	0.44	0.53	0.43	0.55
KC3	MW1	0.64	0.52	0.46	0.51	0.39	0.48	0.5
MC1	MW1	0.63	0.55	0.6	0.57	0.57	0.5	0.57
MC2	MW1	0.36	0.52	0.57	0.52	0.34	0.68	0.5
PC1	MW1	0.6	0.6	0.6	0.44	0.56	0.54	0.56
PC2	MW1	0.64	0.68	0.64	0.54	0.53	0.6	0.61
PC3	MW1	0.69	0.67	0.71	0.54	0.59	0.47	0.61
PC4	MW1	0.55	0.54	0.55	0.56	0.46	0.57	0.54

Table 4.10: AUC values for HetDP (Source dataset:-10 different dataset, Target dataset:- PC1

Source dataset	**Target dataset**	**LR**	**RF**	**HV**	**MLP**	**ADB**	**DT**	**Average AUC value**
ant-1.7	PC1	0.66	0.67	0.62	0.59	0.61	0.47	0.6
CM1	PC1	0.43	0.64	0.6	0.4	0.57	0.51	0.53
KC1	PC1	0.57	0.72	0.67	0.59	0.78	0.74	0.68
KC2	PC1	0.66	0.7	0.59	0.51	0.6	0.53	0.6
KC3	PC1	0.68	0.73	0.76	0.42	0.74	0.63	0.66
MC1	PC1	0.53	0.59	0.56	0.53	0.63	0.66	0.58
MW1	PC1	0.69	0.64	0.68	0.69	0.6	0.62	0.65
PC2	PC1	0.61	0.6	0.65	0.53	0.66	0.49	0.59
PC3	PC1	0.55	0.69	0.63	0.56	0.72	0.67	0.64
PC4	PC1	0.56	0.69	0.64	0.48	0.6	0.57	0.59

Table 4.11: AUC values for HetDP (Source dataset:= 8 different test, Target dataset:- PC2)

Source dataset	**Target dataset**	**LR**	**RF**	**HV**	**MLP**	**ADB**	**DT**	**Average AUC value**
ant-1.7	PC2	0.6	0.48	0.51	0.42	0.66	0.58	0.54
CM1	PC2	0.4	0.57	0.49	0.63	0.55	0.66	0.55
KC1	PC2	0.58	0.7	0.73	0.5	0.48	0.6	0.6
KC2	PC2	0.28	0.65	0.4	0.42	0.4	0.41	0.43
MC1	PC2	0.6	0.64	0.59	0.63	0.53	0.5	0.58
PC1	PC2	0.67	0.58	0.64	0.49	0.52	0.6	0.58
PC3	PC2	0.48	0.5	0.49	0.45	0.49	0.56	0.5
PC4	PC2	0.39	0.58	0.45	0.46	0.51	0.48	0.48

Table 4.12: AUC values for HetDP (Source dataset:= 11 different test, Target dataset:- PC1)

Source	Target	LR	RF	HV	MLP	ADB	DT	Average
ant-1.7	PC3	0.66	0.75	0.67	0.65	0.73	0.67	0.69
CM1	PC3	0.57	0.76	0.69	0.58	0.69	0.66	0.66
KC1	PC3	0.56	0.74	0.65	0.59	0.71	0.69	0.66
KC2	PC3	0.6	0.71	0.68	0.58	0.71	0.69	0.66
KC3	PC3	0.45	0.67	0.62	0.42	0.77	0.65	0.6
MC1	PC3	0.63	0.74	0.69	0.61	0.75	0.73	0.69
MC2	PC3	0.67	0.84	0.76	0.46	0.87	0.77	0.73
MW1	PC3	0.63	0.65	0.61	0.37	0.69	0.54	0.58
PC1	PC3	0.62	0.77	0.72	0.55	0.74	0.72	0.69
PC2	PC3	0.78	0.76	0.72	0.6	0.72	0.62	0.7
PC4	PC3	0.6	0.76	0.68	0.48	0.74	0.74	0.67

The feature-based TL method is used for HetDP. We have selected equal feature count from source and target projects, through which the defect prediction knowledge is transferred for testing of the target project. We have performed metrics matching from the selected features using Spearman's correlation coefficient. Based on the correlation coefficient among these features, we selected metrics for training and test data. The prediction model is developed using one project dataset for training, and we use that model for the prediction of labels in test data. In this manner, we have transferred the feature-based knowledge from source and target projects to build a prediction model. In order to analyze the effectiveness of HetDP, we have performed hypothesis testing. Hypothesis testing is performed using the Friedman test. These hypotheses are designed based on the effectiveness of HetDP and WPDP using R algorithm.

$\mathbf{H}_o$:HetDP performs better than WDP for RF

$\mathbf{H}_a$:WDP performs better than HetDP for RF

The Friedman test is used to test the above hypothesis. The statistical test was performed

at a 0.05 significance level. The obtained $p_$value is 0.001. The value of Friedman's statistics is computed as 12.0. Hence, the $p_$value is less than 0.05. Therefore, we reject H_o and accept H_a. Table 4.14, represents the Friedman mean rank for HetDP and WPDP corresponding to RF ML algorithm. It is observed that the mean rank of WPDP is highest, i.e., 2.00. Hence, RF performed efficiently for WPDP.

Table 4.13: Friedman Mean Rank for defect prediction methods using RF

DP Method	**Mean Rank**
RF for HetDP	1.00
RF for WPDP	2.00

4.4 Discussion

The datasets used for experimentation contain projects developed in various languages such as C, Java, and C++. To analyze the capability of defect prediction techniques, we have used dataset of two different groups, named as NASA and PROMISE. In the experimental procedure, firstly, we have developed a predictive model for predicting defects using different ML algorithms and 10-fold cross-validation for WPDP. In the next experiment, we have developed a predictive model using different ML algorithms and 10-fold cross-validation methods for HetDP. We have observed that the size of source dataset should be larger than 23%. It is observed that due to the presence of historical data in WPDP, it provides a better result. In this way, it would be helpful to identify the defects in the early stage before its delivery to end-users. Lastly, we have tested hypothesis using Friedman test. The designed hypothesis used to evaluate the significant difference between ML algorithms for HetDP. The result of Friedman test proves that there is a significant difference among all ML algorithms used. With our observation on Friedman's mean ranks, RF performs best for HetDP in our experiments. The hypothesis testing has been performed to analyze the performance of RF in

WPDP and HetDP. Thus, RF performs best out of all the six ML algorithms used. Moreover, we test this experiment with large datasets with different techniques. We also investigate the effect of TL on defect prediction models with different datasets considering optimization.

Chapter 5

Cross-Project Defect Prediction Model using Transfer Learning

5.1 Introduction

In software engineering field, defect prediction is one of the important research area. In the last few years, defect prediction is exhaustively studied [30]. SDP aims to detect defect prone modules at different levels, such as functions, classes, and files. It improves software quality, reduces cost, and development time. It helps in software quality assurance in terms of software development effort, software testing or code review. It plays a significant role when an organization has limited resources. Over the past few years, researchers have developed various methods for defect prediction at an early stage.

TL is a new research area for transferring knowledge across different projects. The concept of TL is to learn some knowledge while solving some problems and apply that

knowledge to a similar problem. TL states that the knowledge extracted or learned from the existing code can be applied to the innovative system. TL concept has been used for DP. TL is used such that the model is trained on one dataset which can be also termed as pre-trained model. Now, pre-trained model is used to test another project of similar characteristics. Feature based TL is used such that the two training and testing dataset have a similar features to some extent. In order to select the matching features we have used algorithm for selection of matching pairs out of all the features for development of prediction model. TL is used to obtain generalized and unbiased results.

Nowadays, researchers, software experts, and academicians are working beyond WPDP. Moreover, in the upcoming years, the systems are designed with innovative ideas and techniques. The larger part of existing codes is used for the designing of innovative systems. However, using existing code, which is designed by different peoples with different techniques, was difficult to understand by other people. Thus, the concept of TL has emerged.

Data on various open source projects has been collected for development of prediction model. These software data consist of different types of software metrics and label such as defective or non-defective. Software metrics are defined as a quantitative measure. These metrics are a descriptive analysis of software data. There are various types of software metrics discussed in Chapter 2. The label indicates the presence of a defect in the project's data. Labels are used for the indication of defect presence.

The proposed models for defect prediction are designed for WPDP and HetDP. In WPDP, training and testing of model has been on the same project. Each instance in WPDP, represents particular module of a project. The presence of labels corresponding to each instance whether the module is defective or not, i.e., defect proneness tendency of a project

is present in the column of a project dataset. In WPDP, the prediction model is trained on some part of Project X, and testing of the prediction model is done on some part of Project X. However, WPDP has various limitations. WPDP can not be used when we have to make a defect prediction for completely new projects. In some other cases also, when different people were working on the same project for a longer duration, then many changes are introduced in the code. With the help of the previous code, we can not predict the defect for a newer version or with new changes in the code. To overcome this issue, researchers and academicians have started exploring CPDP.

CPDP takes two different projects for training and testing. In CPDP, Project A can be used for training, and Project B can be used for testing. In this way, CPDP can be used for projects having lack of historical data. Till now, most of the researchers and academician have explored CPDP for the projects that have same software metric; that limits the CPDP. Thus, HetDP approach has been proposed. HetDP allows prediction between two projects if they use different metrics. There exist various limitations of CPDP [30]. The open-source, available datasets consist of different metric sets. In heterogeneous data, all the projects have completely different metrics. For example, the datasets of AEEEM group consist of 61 software metrics and the NASA dataset in the PROMISE repository consist of 31 software metrics. There exist various studies in literature based on cross-project validation [3, 110, 111]. However, most of the authors have not developed models for generalized results. Thus, to achieve generalized results, we have developed a predictive model for defect prediction using different ML algorithms across different projects.

We have analyzed the HetDP approach for defect prediction across various projects with different metrics suite. Thus, using HetDP, researchers or experts can develop other

prediction models to predict defects. HetDP finds the matching metrics having same distribution from source (Project A) to target (Project B) dataset. For effective knowledge transfer, we have also demonstrated the size of a source and target dataset. To achieve our aim of improving the model for defect prediction using heterogeneous metrics, we have formulated the research questions. The RQs are as follows: RQ1: Does the FS is necessary for WPDP? RQ2: How does WPDP models performed using 10-fold cross-validation? RQ3: Which ML algorithm performs best for HetDP using TL? RQ4: How does HetDP models performed using 10-fold cross-validation in comparison to WPDP?

The main contribution of conducting experiment in this chapter is to analyze, experiment and assess the empirical evidence regarding (1) the importance of FS for WPDP (2) performance capability of ML techniques for HetDP (3) predictive performance capability of HetDP and WPDP in comparison to 10-fold cross validation (4) the applicability of CPDP. Defect prediction models are developed using different ML algorithms to produce generalized results using TL.

This chapter is organized as follows: Section 4.2 describes the research background with respect to experiment conducted in this chapter. Section 4.3 discusses the results of the chapter in the form of answers to RQs. Section 4.4 stated the key findings of the chapter. The results of this chapter are published in [112].

5.2 Research Background

The research background of this chapter is discussed in this section.

5.2.1 Dataset used

The dataset used for experimentation is referred in Chapter 2 Section 2.7.3.1.

We have discussed HetDP with different datasets of two different groups. In HetDP, we have considered two different datasets, one is source dataset, and another one is the target dataset, with different metrics. The instance is represented by a tuple of dataset, and metrics are represented by an attribute of a dataset. The last attribute of a dataset represents a label for the presence and absence of defects. In Figure 4.1, we have presented the framework for HetDP. In the first step, we have selected source (X) and target project (Y). The source and target dataset have a different metrics set. In the second step, we have selected metrics from both the projects. We have selected equal number of metrics from source and target dataset. In the next level, we find matching metrics for the target dataset into the source dataset. The matching metrics in the source and target dataset are found based on the similarity among them. The metrics matching performed with the help of a correlation analyzer. At the end of this process, we have a set of matched metrics from source and target dataset. The matched source dataset has been trained using different ML algorithms. This step results in a prediction model that can be further used for predicting labels of the target dataset.

The WPDP has been explained in Figure 4.2. WPDP considers a single project data due to lack of traditional data. In WPDP, we have divided a dataset in two different parts. One part of a dataset is training data and another part is testing data. In the second step, it builds a model for the training data, and it uses that model for predicting labels of test data. In the next subsections, we have explained the approach in more detail.

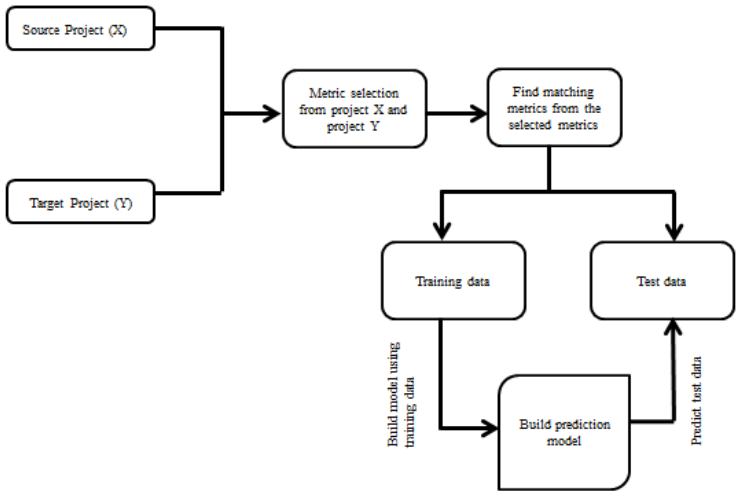

Figure 5.1: Framework for Heterogeneous Defect Prediction

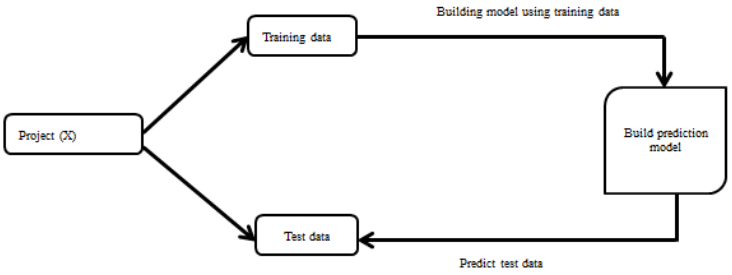

Figure 5.2: Framework for Within Project Defect Prediction

5.2.2 Data Collection

We have collected data from an open-source repository. NASA and PROMISE group dataset are used in the study. The dataset collected from these groups has different projects and attributes. These dataset consist of various software metrics. Each software metric provided with some numerical value corresponding to each module of a project. The last

column corresponding to each module of a project provides a label for defect proneness. The defect proneness attribute is considered as an dependent variable.

5.2.3 Development of Prediction Model

The ML algorithms used for experimentation are LR, RF, HV, MLP, ADB, and DT. These ML algorithms have been discussed below:

1. LR:- LR is named after a function called as sigmoid function. This function is interpreted by an S-shaped curve that takes a real-valued number and maps it to values in the range of 0 and 1, but it should not be at the extreme points. The coefficients of the LR algorithm are determined from training data.

2. RF:- The name of this technique itself specifies that it consists of various DT. These individual DT operate as groups, i.e., ensemble. In this technique, each RF redicts an output class label, and the class having majority counts represents the prediction model. RF provides an understandable model for researchers and academician. This algorithm is robust to noisy and missing values of the dataset.

3. HV:- In HV technique, various ML algorithms are combined to build a prediction model.

4. MLP:- This technique consists of multiple perceptrons. It is focused on creating robust algorithms and used those algorithms for building a prediction model. However, NN can learn the training data representation easily, and it easily predicts the label of the output variable. These networks can learn the mapping function, and they are proved as a global algorithm. The predictive capability of NN comes from

the hierarchical or multi-layered structure of the networks. It takes different features of input data over various scales and group them into high-order features.

5. ADB:- This classifier also makes a prediction. It is similar to the RF classifier. It applies multiple DT to individual samples, and it takes a combination of these predictions from each DT or majority trees in the classification problem. Each DT contributes a significant amount to the final prediction.

6. DT:- This technique works on the divide and conquer concept. It uses a heuristic approach known as recursive partitioning. This approach is based on the divide and conquer problem. It divides the original dataset into small groups, which are repeatedly split into even smaller groups. It solves each subproblem. In the end, the solution of all subproblems combined to produce solution to the actual problem. DT is easy to apply and easy for development of prediction models. It is not based on any assumption. It is not affected by the presence of outliers.

5.2.4 Cross-Validation Method

Cross-validation is a technique used for model evaluation by partitioning the given dataset into two parts [3]. One part is used for training, and another part is used for testing data. The data is divided into respective ratio and proportion using cross-validation method. The model is trained using training dataset. The prediction model is used for making predictions of testing data. The division of dataset into training and testing dataset depend on the type of cross-validation method used [113]. There exist three types of cross-validation methods, such as hold-out validation, K-fold cross-validation, and leave-one-out cross-validation. We have used K-fold cross-validation for conducting experiment.

In *K*-fold cross-validation, the dataset is divided into *K* different parts. *K*-1 parts are used for training the model, and the remaining part is used for testing. The value of *K* is considered as 10. In 10-fold cross-validation, the complete dataset is partitioned into ten parts, and each part is of equal size. Nine parts are randomly selected for training, while the remaining part, i.e., the tenth part, is used as a validation set. This process iterates for ten times. In the end, the results of each iteration would be combined. *K*-fold cross-validation method is mostly used by the researchers/academician and professionals in empirical studies [114].

5.2.5 Performance Metrics

There exist various performance metrics/ measures. These metrics/ measures are used to evaluate and analyze the performance of different ML algorithms. Different performance metrics exist for classification and regression problems. The selection of performance metrics/measures is very crucial. We have used AUC-ROC [115] performance metric. It helps in identifying the goodness of a developed prediction model. AUC-ROC curve is represented with a curve between recall and specificity. The value of AUC lies from 0 to 1. Depending on AUC values, we can predict how good or bad the developed model [115]. The significance of AUC corresponding to 0, 0.5, and 1 has been represented in Eqn. 4.1 given below.

$$AUC = \begin{cases} 0, \ the\ developed\ model\ wrongly\ classifies\ output\ class \\ 0.5, \ no\ discrimination\ between\ output\ classes \\ 1, \ the\ developed\ model\ perfectly\ classifies\ output\ class \end{cases} \tag{5.1}$$

5.2.6 Statistical Test

The statistical test is used for validation of results. The statistical test is used to check if there is any significant difference between the ML techniques used. Friedman test is used for independent groups. The data can be randomly selected from any distribution for the Friedman test [3]. This test does not consider any assumption; i.e., it is a non-parametric test. With the help of these, we have checked if there is any significant difference among the performance of ML algorithms used for HetDP. In the first step, we have designed two hypotheses: the null hypothesis H_o and the alternate hypothesis H_a. In the second step, we have obtained p_value at a significant level of 0.05. The null hypothesis is accepted if the p_value > 0.05. An alternate hypothesis is accepted if the p_value ≤ 0.05. The Wilcoxon–signed rank test performed pairwise comparison [116]. However, it is used to compare repeated measurements [117].

5.3 Results and Analysis

In this section, we have addressed the answers the RQs designed to experiment with open-source dataset. The results for the comparison of HetDP and WPDP are presented in this section.

5.3.1 Results Specific to RQ1

We have applied various FS techniques for selection of features from different projects. Chi-square method and Recursive Feature Elimination (RFE) method. We have observed that prediction model does not work efficiently after applying feature elimination method.

Moreover, the large number of features are reduced and irrelevant features are removed using FS technique. With FS techniques, we got AUC values near 0.5. AUC value of 0.5 does not make correct predictions. We have performed experiment without FS techniques. Based on the observation, the prediction model provided efficient result using FS technique. Hence, it is concluded that FS techniques are necessary for WPDP.

5.3.2 Results Specific to RQ2

We have used different ML algorithms to develop these predictive models. For WPDP, the AUC values are computed using different ML algorithms and 10-fold cross-validation method. The values of 12 datasets used for WPDP are shown in Table 4.1. All ML techniques performed significantly different based on the mathematical interpretation.

Table 5.1: AUC values for WPDP

Source dataset	**Target dataset**	**LR**	**RF**	**HV**	**MLP**	**ADB**	**DT**	**Average AUC value**
ant-1.7	ant-1.7	0.76	0.83	0.82	0.73	0.82	0.72	0.78
CM1	CM1	0.47	0.73	0.65	0.42	0.62	0.58	0.58
KC1	KC1	0.55	0.79	0.73	0.6	0.76	0.74	0.70
KC2	KC2	0.63	0.82	0.76	0.53	0.76	0.66	0.69
KC3	KC3	0.55	0.76	0.73	0.49	0.63	0.62	0.63
MC1	MC1	0.4	0.9	0.78	0.48	0.83	0.71	0.68
MC2	MC2	0.7	0.71	0.77	0.62	0.63	0.59	0.67
MW1	MW1	0.72	0.71	0.74	0.32	0.67	0.59	0.63
PC1	PC1	0.61	0.83	0.77	0.48	0.79	0.75	0.71
PC1	PC1	0.61	0.83	0.77	0.48	0.79	0.75	0.71
PC2	PC2	0.52	0.81	0.71	0.21	0.77	0.59	0.60
PC3	PC3	0.66	0.83	0.79	0.57	0.77	0.74	0.73
PC4	PC4	0.76	0.92	0.9	0.71	0.91	0.88	0.85

From the results obtained for WPDP, we have observed that RF technique performs better than

LR, HV, MLP, ADB, DT technique. RF works better for building a prediction model for WPDP as it takes a combination of individual trees. RF provides a binary classification of data points, but it also provides the probabilities for each instance belongs to defective and non-defective label. It is observed that AUC values are improved for all datasets. In some of the datasets, LR, and HV also performed efficiently.

5.3.3 Results Specific to RQ3

To answer this RQ, statistical test has been used. We have designed two hypotheses such as H_o and H_a.

H_o:There is no statistical difference between six ML algorithms.

H_a: There is a statistical difference between six ML algorithms.

Friedman test is used for testing the above hypothesis. The statistical test is used at a significance level of 0.05. The p_value obtained is 0.00 in this case. The value of Friedman's statistics is computed as 210.9. Hence, the p_value is less than 0.05; we reject H_o and accept H_a. In Figure 4.3, the Friedman mean rank corresponding to ML algorithms is presented. We have observed that the mean rank of RF is highest, i.e., 4.92. It implies that RF performs efficiently in comparison to other algorithms used in the study. ADB is the second-highest mean rank, i.e., 4.05, which implies that after RF, ADB is best performing ML algorithm for HetDP. We have also performed Wilcoxon–signed rank test to check the statistical difference among all the algorithms. The Wilcoxon–signed rank test is used to check if there is any statistical difference exist or not. The hypothesis testing is performed at 0.05 significance level. The p_value provided by Wilcoxon–signed rank test is 0.000. The computed p_value is less than the significance level. Thus, there is a statistical difference among the performance of ML algorithms. Hence, alternate hypothesis is accepted.

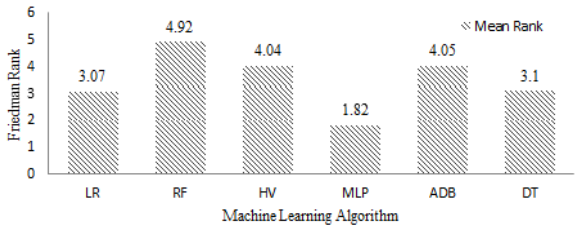

Figure 5.3: Friedman Mean Rank for ML Algorithm

5.3.4 Results Specific to RQ4

To answer this RQ, we have experimented on different datasets. In HetDP, the source and target dataset are different. This experiment has been observed at different cutoff_threshold for metrics matching. The metrics matching has been done by Spearman's correlation coefficient analyzer. In this analyzer, we have performed experiments by taking different threshold values, e.g., cut-off_threshold = 0.1, 0.05, 0.3, 0.9. We have observed in this analyzer that by changing the threshold to more than 0.05, it doesn't perform well. Thus, we have set the threshold value to 0.05. Different datasets are used as a target dataset. For which, we have executed our experiment iteratively, by setting different source and target dataset in each iteration. In Table [4.2-4.12] presented the AUC values corresponding to the ML algorithms used. The pairwise combination of the source and target dataset are presented separately in Table [4.2-4.12].

TL concept helps us in developing a generalized model. Generalized defect prediction model can be easily developed for industry or academic purpose. The software quality also improves with generalized prediction model. We have used existing software knowledge for building purpose. We can remove ambiguities in the new projects by using existed project's knowledge, patterns, design. It improves maintainability, understandability of a software. It would be easy to use prediction model for end-users. The feature-based TL method is used. The computed results were compared with existing studies also. AUC values for RF in HetDP and WPDP is 0.673 and 0.803. However,

the AUC values for RF in HetDP and WPDP in existing studies is 0.640 and 0.732. The AUC values for DT in HetDP and WPDP is 0.617 and 0.681. However, the AUC values for DT and WPDP in existing studies is 0.568 and 0.598. Hence, the computed AUC values indicated that RF and DT performed better for HetDP and WPDP based on the experiment performed.

Table 5.2: AUC values for HetDP (Source dataset:-11 different dataset, Target dataset:-ant-1.7)

Source dataset	**Target dataset**	**LR**	**RF**	**HV**	**MLP**	**ADB**	**DT**	**Average AUC value**
CM1	ant-1.7	0.67	0.71	0.69	0.57	0.72	0.67	0.67
KC1	ant-1.7	0.69	0.73	0.67	0.68	0.7	0.69	0.69
KC2	ant-1.7	0.68	0.72	0.67	0.58	0.67	0.7	0.67
KC3	ant-1.7	0.71	0.75	0.72	0.47	0.68	0.66	0.67
MC1	ant-1.7	0.61	0.72	0.66	0.54	0.69	0.66	0.65
MC2	ant-1.7	0.68	0.67	0.66	0.56	0.59	0.54	0.62
MW1	ant-1.7	0.65	0.63	0.64	0.56	0.61	0.6	0.62
PC1	ant-1.7	0.66	0.75	0.71	0.57	0.72	0.71	0.69
PC2	ant-1.7	0.68	0.73	0.7	0.62	0.72	0.69	0.69
PC3	ant-1.7	0.64	0.73	0.66	0.54	0.66	0.66	0.65
PC4	ant-1.7	0.63	0.73	0.67	0.56	0.71	0.69	0.67

Table 5.3: AUC values for HetDP (Source dataset:-10 different dataset, Target dataset:-CM1)

Source dataset	**Target dataset**	**LR**	**RF**	**HV**	**MLP**	**ADB**	**DT**	**Average AUC value**
ant-1.7	CM1	0.64	0.67	0.68	0.6	0.62	0.6	0.64
KC1	CM1	0.6	0.64	0.65	0.6	0.67	0.65	0.64
KC2	CM1	0.59	0.68	0.66	0.56	0.63	0.6	0.62
KC3	CM1	0.54	0.48	0.58	0.64	0.67	0.46	0.56
MC1	CM1	0.54	0.66	0.6	0.59	0.66	0.61	0.61
MW1	CM1	0.56	0.59	0.62	0.61	0.65	0.61	0.61

PC1	CM1	0.62	0.71	0.63	0.5	0.69	0.59	0.62
PC2	CM1	0.57	0.6	0.6	0.52	0.64	0.54	0.58
PC3	CM1	0.58	0.68	0.63	0.49	0.65	0.61	0.61
PC4	CM1	0.55	0.71	0.6	0.49	0.7	0.49	0.59

Table 5.4: AUC values for HetDP (Source dataset:-11 different dataset, Target dataset:-KC1)

Source dataset	**Target dataset**	**LR**	**RF**	**HV**	**MLP**	**ADB**	**DT**	**Average AUC value**
ant-1.7	KC1	0.68	0.76	0.69	0.75	0.74	0.68	0.72
CM1	KC1	0.64	0.69	0.65	0.61	0.71	0.65	0.66
KC2	KC1	0.61	0.77	0.72	0.57	0.76	0.69	0.69
KC3	KC1	0.71	0.71	0.72	0.6	0.78	0.72	0.71
MC1	KC1	0.59	0.75	0.69	0.62	0.75	0.73	0.69
MC2	KC1	0.54	0.69	0.69	0.51	0.63	0.65	0.62
MW1	KC1	0.64	0.74	0.71	0.56	0.68	0.64	0.66
PC1	KC1	0.67	0.75	0.68	0.62	0.72	0.74	0.7
PC2	KC1	0.68	0.73	0.72	0.61	0.74	0.68	0.69
PC3	KC1	0.62	0.74	0.71	0.55	0.74	0.69	0.68
PC4	KC1	0.62	0.75	0.7	0.62	0.74	0.75	0.7

Table 5.5: AUC values for HetDP (Source dataset:-11 different dataset, Target dataset:-KC2)

Source dataset	**Target dataset**	**LR**	**RF**	**HV**	**MLP**	**ADB**	**DT**	**Average AUC value**
ant-1.7	KC2	0.79	0.82	0.8	0.76	0.78	0.68	0.77
CM1	KC2	0.71	0.79	0.75	0.7	0.8	0.67	0.74
KC1	KC2	0.73	0.81	0.81	0.69	0.77	0.69	0.75
KC3	KC2	0.76	0.82	0.79	0.6	0.8	0.79	0.76
MC1	KC2	0.63	0.79	0.76	0.65	0.76	0.68	0.71
MC2	KC2	0.8	0.76	0.78	0.63	0.74	0.67	0.73
MW1	KC2	0.75	0.79	0.75	0.6	0.67	0.64	0.7
PC1	KC2	0.72	0.79	0.8	0.68	0.76	0.74	0.75

PC2	KC2	0.78	0.79	0.79	0.7	0.75	0.75	0.76
PC3	KC2	0.72	0.8	0.77	0.67	0.77	0.75	0.75
PC4	KC2	0.74	0.79	0.77	0.64	0.77	0.66	0.73

Table 5.6: AUC values for HetDP (Source dataset:-11 different dataset, Target dataset:-KC3)

Source dataset	**Target dataset**	**LR**	**RF**	**HV**	**MLP**	**ADB**	**DT**	**Average AUC value**
ant-1.7	KC3	0.78	0.84	0.79	0.8	0.77	0.74	0.79
CM1	KC3	0.54	0.58	0.6	0.5	0.47	0.58	0.55
KC1	KC3	0.56	0.61	0.58	0.45	0.45	0.44	0.52
KC2	KC3	0.63	0.64	0.65	0.49	0.6	0.51	0.59
MC1	KC3	0.59	0.62	0.64	0.48	0.56	0.55	0.57
MC2	KC3	0.51	0.57	0.46	0.39	0.52	0.49	0.49
MW1	KC3	0.66	0.63	0.6	0.51	0.69	0.42	0.59
PC1	KC3	0.66	0.6	0.64	0.46	0.59	0.53	0.58
PC2	KC3	0.63	0.57	0.66	0.56	0.55	0.46	0.57
PC3	KC3	0.46	0.54	0.52	0.51	0.58	0.52	0.52
PC4	KC3	0.59	0.52	0.56	0.6	0.56	0.52	0.56

Table 5.7: AUC values for HetDP (Source dataset:-9 different dataset, Target dataset:-MC1)

Source dataset	**Target dataset**	**LR**	**RF**	**HV**	**MLP**	**ADB**	**DT**	**Average AUC value**
ant-1.7	MC1	0.62	0.74	0.62	0.52	0.56	0.54	0.6
CM1	MC1	0.5	0.6	0.59	0.35	0.59	0.56	0.53
KC1	MC1	0.5	0.62	0.62	0.54	0.61	0.67	0.59
KC2	MC1	0.5	0.52	0.59	0.41	0.5	0.5	0.5
MW1	MC1	0.62	0.64	0.55	0.53	0.61	0.52	0.58
PC1	MC1	0.41	0.42	0.51	0.42	0.49	0.51	0.46
PC2	MC1	0.53	0.47	0.5	0.55	0.47	0.56	0.51
PC3	MC1	0.58	0.62	0.62	0.59	0.59	0.46	0.58
PC4	MC1	0.54	0.65	0.64	0.55	0.61	0.65	0.61

Table 5.8: AAUC values for HetDP (Source dataset:-11 different dataset, Target dataset:-MC2)

Source dataset	**Target dataset**	**LR**	**RF**	**HV**	**MLP**	**ADB**	**DT**	**Average AUC value**
ant-1.7	MC2	0.61	0.46	0.54	0.49	0.46	0.57	0.52
CM1	MC2	0.67	0.68	0.66	0.57	0.67	0.56	0.64
KC1	MC2	0.78	0.76	0.77	0.64	0.81	0.72	0.75
KC2	MC2	0.7	0.71	0.74	0.56	0.61	0.7	0.67
KC3	MC2	0.61	0.66	0.57	0.48	0.59	0.61	0.59
MC1	MC2	0.67	0.59	0.62	0.49	0.57	0.54	0.58
MW1	MC2	0.61	0.52	0.64	0.5	0.56	0.48	0.55
PC1	MC2	0.66	0.56	0.61	0.59	0.55	0.57	0.59
PC2	MC2	0.65	0.57	0.64	0.49	0.46	0.57	0.56
PC3	MC2	0.58	0.57	0.55	0.51	0.57	0.55	0.56
PC4	MC2	0.56	0.56	0.56	0.48	0.6	0.42	0.53

Table 5.9: AUC values for HetDP (Source dataset:-11 different dataset, Target dataset:-MW1

Source dataset	**Target dataset**	**LR**	**RF**	**HV**	**MLP**	**ADB**	**DT**	**Average AUC value**
ant-1.7	MW1	0.56	0.63	0.56	0.52	0.57	0.51	0.56
CM1	MW1	0.56	0.68	0.62	0.53	0.55	0.55	0.58
KC1	MW1	0.54	0.66	0.6	0.47	0.56	0.56	0.57
KC2	MW1	0.62	0.61	0.66	0.44	0.53	0.43	0.55
KC3	MW1	0.64	0.52	0.46	0.51	0.39	0.48	0.5
MC1	MW1	0.63	0.55	0.6	0.57	0.57	0.5	0.57
MC2	MW1	0.36	0.52	0.57	0.52	0.34	0.68	0.5
PC1	MW1	0.6	0.6	0.6	0.44	0.56	0.54	0.56
PC2	MW1	0.64	0.68	0.64	0.54	0.53	0.6	0.61
PC3	MW1	0.69	0.67	0.71	0.54	0.59	0.47	0.61
PC4	MW1	0.55	0.54	0.55	0.56	0.46	0.57	0.54

Table 5.10: AUC values for HetDP (Source dataset:-10 different dataset, Target dataset:- PC1

Source dataset	**Target dataset**	**LR**	**RF**	**HV**	**MLP**	**ADB**	**DT**	**Average AUC value**
ant-1.7	PC1	0.66	0.67	0.62	0.59	0.61	0.47	0.6
CM1	PC1	0.43	0.64	0.6	0.4	0.57	0.51	0.53
KC1	PC1	0.57	0.72	0.67	0.59	0.78	0.74	0.68
KC2	PC1	0.66	0.7	0.59	0.51	0.6	0.53	0.6
KC3	PC1	0.68	0.73	0.76	0.42	0.74	0.63	0.66
MC1	PC1	0.53	0.59	0.56	0.53	0.63	0.66	0.58
MW1	PC1	0.69	0.64	0.68	0.69	0.6	0.62	0.65
PC2	PC1	0.61	0.6	0.65	0.53	0.66	0.49	0.59
PC3	PC1	0.55	0.69	0.63	0.56	0.72	0.67	0.64
PC4	PC1	0.56	0.69	0.64	0.48	0.6	0.57	0.59

Table 5.11: AUC values for HetDP (Source dataset:= 8 different test, Target dataset:- PC2)

Source dataset	**Target dataset**	**LR**	**RF**	**HV**	**MLP**	**ADB**	**DT**	**Average AUC value**
ant-1.7	PC2	0.6	0.48	0.51	0.42	0.66	0.58	0.54
CM1	PC2	0.4	0.57	0.49	0.63	0.55	0.66	0.55
KC1	PC2	0.58	0.7	0.73	0.5	0.48	0.6	0.6
KC2	PC2	0.28	0.65	0.4	0.42	0.4	0.41	0.43
MC1	PC2	0.6	0.64	0.59	0.63	0.53	0.5	0.58
PC1	PC2	0.67	0.58	0.64	0.49	0.52	0.6	0.58
PC3	PC2	0.48	0.5	0.49	0.45	0.49	0.56	0.5
PC4	PC2	0.39	0.58	0.45	0.46	0.51	0.48	0.48

Table 5.12: AUC values for HetDP (Source dataset:= 11 different test, Target dataset:- PC1)

Source	Target	LR	RF	HV	MLP	ADB	DT	Average
ant-1.7	PC3	0.66	0.75	0.67	0.65	0.73	0.67	0.69
CM1	PC3	0.57	0.76	0.69	0.58	0.69	0.66	0.66
KC1	PC3	0.56	0.74	0.65	0.59	0.71	0.69	0.66
KC2	PC3	0.6	0.71	0.68	0.58	0.71	0.69	0.66
KC3	PC3	0.45	0.67	0.62	0.42	0.77	0.65	0.6
MC1	PC3	0.63	0.74	0.69	0.61	0.75	0.73	0.69
MC2	PC3	0.67	0.84	0.76	0.46	0.87	0.77	0.73
MW1	PC3	0.63	0.65	0.61	0.37	0.69	0.54	0.58
PC1	PC3	0.62	0.77	0.72	0.55	0.74	0.72	0.69
PC2	PC3	0.78	0.76	0.72	0.6	0.72	0.62	0.7
PC4	PC3	0.6	0.76	0.68	0.48	0.74	0.74	0.67

The feature-based TL method is used for HetDP. We have selected equal feature count from source and target projects, through which the defect prediction knowledge is transferred for testing of the target project. We have performed metrics matching from the selected features using Spearman's correlation coefficient. Based on the correlation coefficient among these features, we selected metrics for training and test data. The prediction model is developed using one project dataset for training, and we use that model for the prediction of labels in test data. In this manner, we have transferred the feature-based knowledge from source and target projects to build a prediction model. In order to analyze the effectiveness of HetDP, we have performed hypothesis testing. Hypothesis testing is performed using the Friedman test. These hypotheses are designed based on the effectiveness of HetDP and WPDP using R algorithm.

$\mathbf{H}_o$:HetDP performs better than WDP for RF

$\mathbf{H}_a$:WDP performs better than HetDP for RF

The Friedman test is used to test the above hypothesis. The statistical test was performed

at a 0.05 significance level. The obtained $p_$value is 0.001. The value of Friedman's statistics is computed as 12.0. Hence, the $p_$value is less than 0.05. Therefore, we reject H_o and accept H_a. Table 4.14, represents the Friedman mean rank for HetDP and WPDP corresponding to RF ML algorithm. It is observed that the mean rank of WPDP is highest, i.e., 2.00. Hence, RF performed efficiently for WPDP.

Table 5.13: Friedman Mean Rank for defect prediction methods using RF

DP Method	**Mean Rank**
RF for HetDP	1.00
RF for WPDP	2.00

5.4 Discussion

The datasets used for experimentation contain projects developed in various languages such as C, Java, and C++. To analyze the capability of defect prediction techniques, we have used dataset of two different groups, named as NASA and PROMISE. In the experimental procedure, firstly, we have developed a predictive model for predicting defects using different ML algorithms and 10-fold cross-validation for WPDP. In the next experiment, we have developed a predictive model using different ML algorithms and 10-fold cross-validation methods for HetDP. We have observed that the size of source dataset should be larger than 23%. It is observed that due to the presence of historical data in WPDP, it provides a better result. In this way, it would be helpful to identify the defects in the early stage before its delivery to end-users. Lastly, we have tested hypothesis using Friedman test. The designed hypothesis used to evaluate the significant difference between ML algorithms for HetDP. The result of Friedman test proves that there is a significant difference among all ML algorithms used. With our observation on Friedman's mean ranks, RF performs best for HetDP in our experiments. The hypothesis testing has been performed to analyze the performance of RF in

WPDP and HetDP. Thus, RF performs best out of all the six ML algorithms used. Moreover, we test this experiment with large datasets with different techniques. We also investigate the effect of TL on defect prediction models with different datasets considering optimization.

Chapter 6

Empirical validation of machine learning techniques for heterogeneous cross–project change prediction and within–project change prediction

6.1 Introduction

SCP focuses on identification of change proneness in the initial phase of software development [118]. It reduces the amount of time required for maintenance and testing phase of the software. The prediction model helps in software quality assurance. Software quality mainly aims at the design of software and conformance to that design. Thus, software quality assurance is accomplished using change prediction models. The changes present in

the modules of software impact the software quality, due to which change–prone modules first need to be identified. Using prediction models, change–prone modules can be easily identified in the future projects, in this manner the quality of upcoming software can be improved. But, in some cases, the project does not contain sufficient data. However, it results in a lack of training data for the prediction model.

The change prediction works successfully for projects having a large quantity of data. Moreover, the usage of historical data or empirical data of the same project is not feasible. The process of predicting changes in one project using historical and empirical data from other projects is known as CPCP. In this chapter, performance of ML techniques analyzed for HetCPCP and WPCP using open-source projects [119]. ML techniques can be used to predict the impact of changes on software quality. The size of the software is increasing day by day due to which the need of developing more stable and secure software arise. Thus, many companies employing ML techniques to develop efficient software in terms of improvising the quality of future software, by reducing the cost and effort. In the existing studies, many ML algorithms are used to detect bugs in order to find out the amount of change between two different versions of a project using SVM, Bayesian Belief Network (BBN), NB, and DT. ML techniques are useful in designing the prediction model for improvising the software quality change prediction among different projects of similar characteristics. The changes in the existing software helps to improvise the quality of future software in terms of reliability, maintenance, functionality, robustness, and efficiency. Furthermore, ML techniques help in analyzing these changes on the basis of performance of prediction model.

With the help of data collected from various projects and historical repositories, change prediction can be easily employed for new projects. There are various studies exist in

the literature which showed that software repositories dataset can be used for change prediction for future releases. In some of the studies, it has been shown that the data of the same project can be used for change prediction. Thus, there is a lack of training data for creating models due to which other projects are required for change prediction.

CPCP does not give feasible solution in every case. For CPCP, author considered different versions of a project or software. In cross-project we have two different projects for training and testing. If both the projects i.e., training and testing projects have different attributes then it is called as HetCP [120]. In HetCP, the attributes space is different and it is non–overlapping in both training/source and testing/target projects. The relationship between attribute space of both the projects need to be established. The correlation between the attribute space of both the projects should be computed, it will result in reducing the gap across their attribute space as well as the dimensions of their attributes. The features are taken as predictors in this chapter.

Feature type transfer is conducted by transferring the features of one project for making predictions in a new project [46]. If the attributes are not related to each other in the projects, then the prediction model results in poor performance. If the attributes and features are similar in both the projects, then it is called a Homogeneous Cross Project (HoCP). In HoCP, we have similar attributes through which it is feasible to estimate the correlation between attributes. The main issue is to identify the particular problem, in which CPCP is applicable. We have to identify the relationship between the attributes or features of training and testing projects. There are various studies exist in the literature for CPCP. In some of the studies, the prediction model are developed using various techniques such as ML and statistical based techniques [3]. The performance of developed models can be estimated using various performance metrics [121]. The empirical validation of the

dataset of other projects such as Notepad++, CodeBlocks, and CodeLite has been done for predicting change prone classes in future projects in the experiment conducted. AUC is used for analyzing the performance of various ML classifiers.

In the existing studies, the authors have analyzed various ways of data distribution affecting the accuracy of CPCP models. CPCP model consider different projects for training and testing of model development and model prediction. Furthermore, it has been observed that various aspects are considered based on which data distribution impact the model accuracy such as different project size, project complexity, imbalanced dataset, data preprocessing, project development practices and process, temporal distribution, and domain shift. In CPCP model, domain shift occurs when source and target projects have different characteristics. Moreover, domain shift leads to decreasing predictive capability and fails to generalize the results for new domain. The domain of different projects features also plays key role in changing the performance of CPCP. Furthermore, imbalanced dataset affect the accuracy of CPCP model by providing biased results towards majority class. The practices, programming language, standards, and process followed for development of different projects also affect accuracy of CPCPM. Furthermore, generalization of CPCP models fails on different project considering project size, and complexity. The temporal changes with respect to development practices also affect the accuracy of CPCP as models does not adapt evolving changes easily.

TL is useful in transferring some knowledge from one project to another project or new project by creating a prediction model [122]. There are various methods of TL such as feature transfer, instance transfer, relational knowledge transfer, and parameter transfer [46]. NN is integrated with TL which result in higher accuracy and efficiency for finger printing[123]. In existing study, we have used the feature transfer method. The features

of one project are transferred for change prediction in future/ new projects for the feature transfer method [46]. In this chapter, we have created dataset by collecting metrics of various software. The created dataset has been used for the analysis of the HetCPCP and WPCP approaches. The feature type TL has been used. If there is a lack of training data in future, then we can use some other projects for change prediction using HetCPCP. Thus, using HetCPCP, researchers, experts, or software developers can develop other prediction models to predict changes in future projects. With the help of TL, the knowledge has been transferred from one project to another project for identification of change prone parts.

The change prediction model is developed using ML techniques in order to improve software quality and reducing maintenance effort. Combination of ML with data assimilation provides optimized results for improving software quality [124]. ML is useful for designing prediction models with better performance and numerical simulation. Prediction models developed using ML uses large amount of data. In order to analyze the capability of CPCP and WPCP analyzed using open source projects. ML techniques gives better insights. Moreover, in upcoming future there will be huge amount of data for analysis for which ML techniques are easily adaptable. Traditional methods or techniques are not easily adaptable to changes in the future dataset.

To achieve our aim of analyzing various ML techniques for change prediction, we have formulated the RQs:

- RQ1: What is the predictive capability of ML techniques for change prediction in WPCP and HetCPCP?
- RQ2: Which ML technique performed better than other techniques for WPCP?
- RQ3: Which ML technique performed better than other techniques for HetCPCP?

Aiming at the above RQs, we conducted an experiment on 18 different versions of 3 software (CodeBlocks, Notepad++ and CodeLite) and we have considered 6 different versions of Notepad++, 3 different versions of CodeBlocks and 9 different versions of CodeLite. Experiment considering both CPCP and WPCP with all the possible combinations. However, for CPCP source and target dataset have different data distribution for which metrics matcher is used to extract the matching features from both source and target projects. It has been analyzed that the CPCP can be employed for future projects in order to check change prone parts of a project and improvement of software quality.

Organization of this chapter: Section 5.2 presented empirical data collection. Section 5.3 presented the research background. Section 5.4 discussed research methodology. Section 5.5 presented the answers to RQs. Section 5.6 summarized conlusion. The results of this chapter are published in [125].

6.2 Empirical Data Collection

The dataset used for experimentation is referred in Chapter 2 Section 2.7.3.2.

6.3 Research Background

We have followed systematic steps to experiment. The concept of TL is used for HetCPCP. In HetCP, two different versions of two different projects were considered. We learn some knowledge from one project and use that knowledge to develop a prediction model. The prediction model will be used for making predictions and testing on another project. For learning knowledge, feature-based TL used. On the basis of first project features, a

prediction model is designed. The following steps have been followed for developing a prediction model:

6.3.1 Selection of Dataset

In this study, we have followed systematic steps to experiment. In this study, we have used the concept of TL for HetCPCP. In HetCP, two different versions of two different projects were considered. We learn some knowledge from one project and use that knowledge to develop a prediction model. The prediction model will be used for making predictions and testing on another project. For learning knowledge, feature–based TL used in this chapter. On the basis of first project features, a prediction model is designed.

6.3.2 Data Preprocessing

The metrics value in all the datasets is varying. Normalization is required to bring the dataset to a common range. Data has been preprocessed using min–max normalization. In min–max normalization, the minimum value of an independent attribute in the dataset is 0. The maximum value of an independent attribute in the dataset is made as 1. All the other values of an attribute are converted into the range of 0 to 1.

6.3.3 Imbalanced Dataset

SMOTE [126] to balance our imbalanced dataset. The imbalanced dataset suffers from overfitting. Due to overfitting, the model learns some irrelevant noise. With the help of SMOTE, the imbalance dataset can be converted into a balanced dataset. In SMOTE,

random samples are created for the minority class. Thus, application of SMOTE provides the equivalent number of minority and majority class samples.

6.3.4 FS Technique

FS plays an important role in the performance of our model. We need to extract relevant features. Thus, the usage of irrelevant features provides us with low accuracy. FS helps in reducing overfitting and training time, and it improves accuracy. We have used the univariate selection technique. It selects features based on their correlation with the output variable or target variable. From univariate selection techniques, we have used the chi–square test for FS. In the chi–square test, we find out the relationship between independent and dependent variables are established. The SelectKBest library of Python is used to select relevant features out of feature set [127]. Chi–square calculates the deviation of expected count (E_i) from observed count (O_i). The formula for chi–square is given below:

$$\chi^2 = \sum_{i=1}^{n} \frac{(O_i - E_i)^2}{E_i} \tag{6.1}$$

where, *i* stand for a DoF, *O* stands for observed count, and *E* stands for expected count

6.3.5 ML Classifier

To develop a prediction model, we have used various ML classifiers. In the existing software development process, ML techniques can be integrated on the basis of problem (classification/regression), data analysis, dataset characteristics, data visualization, data modeling, and data validation. However, in the study conducted by authors, ML techniques

combined into existing software development process for predictive analytics considering unseen software. In the study, ML techniques are integrated to change prediction for cross-project (different projects for training and testing) and WP (same project for training and testing). Thus, ML based predictive model identify whether software is change prone or not to improvise the quality of future projects. However, integration of ML techniques with the traditional software development process helps in various aspects such as defect removal, effort estimation, and software maintainability. ML classifiers are used such as LR, RF, SVM, HV, KNN, MLP, ADB, DT, and QDA. A prediction model is designed using specified classifiers. The prediction model is developed using the dataset of one project. Later, the prediction model will be used for testing and predictions on another dataset.

Table 6.1: ML techniques used

Category	**ML Technique**	**Description**
Statistical classifier	LR	LR is used for classification problems. LR is used for analyzing the predictive capability of an algorithm. It is developed based on the concept of probability. It uses a cost function that is complex and known as the sigmoid function. The hypothesis of the sigmoid function depicts its value between 0 to 1. Thus, LR is not known as linear regression.
	QDA	The statistical techniques used for classification. QDA differs from Linear Discriminant Analysis (LDA). In LDA, an assumption of equal covariance for each class is used. QDA computes individual covariance for each class. However, QDA is a better classifier to find out non-linear boundaries among the class.

DT	DT	Used for classification and regression. It starts from the root node and ends at a leaf node. At each level, the best variable is selected to split into two child nodes. To decide the best node at each level of a tree, checks for the possibility out of all possible nodes left at that level. It has three different measures for attribute selection such as information gain, Gini index, and gain ratio. DT algorithm aims at reducing the average impurity at each level.
SVM	SVM	Used for classification problems. Designs an N-dimensional hyperplane. SVM modeling aims to find the optimal hyperplane. The optimal hyperplane divides all the data points into classes concerning the output variable label. The data points that belong to one class of output labels lie on one side of a plane, and the data points that belong to another class of output labels lie on another side of the hyperplane. If the hyperplane is non-linear, then SVM prefers to use kernel function. Various types of kernel functions exist such as polynomial, linear, and sigmoid.
NN	MLP	It is a group of feed-forward ANN. It consists of multiple perceptron. It consists of three layers of nodes such as input layer, hidden layer, and output layer. All the nodes consist of neurons with activation functions except input layer nodes. Multiple layers and non-linear activation function differentiate it from linear perceptron. MLP can be used for both types of problems that is classification and regression depending upon the type of activation function used in it. Used for the non-linearly separable classification of data.
EL	RF	It consists of various DTs. Each tree in RF is considered as an ensemble. The output provided by each DT is a prediction label corresponding to the output class. The class with the majority voting is considered for the model's prediction.

	ADB	One of the most successful algorithms for classification problems from EL. Used to combine atleast two weak classifiers into one strong classifier. Works fast in comparison to other ML classifiers.
Others	KNN	It stores all the training instances and the new unlabeled testing instance will be classified based on a similarity measure. Used for continuous output variables and categorical output variables. In this chapter, we have used a categorical output variable. For categorical output variables, KNN uses hamming distance to measure similarity. If the dataset consists of a mixture of values, then it standardized the values between 0 to 1. However, KNN has a very high predictive power.

6.3.6 Cross–Validation method

Cross-validation is a technique that is used for model evaluation. For cross-validation, division of data is important to evaluate the performance of a model on unseen data. It divides the data into two parts, that is training and testing data. The data through which the model is trained, named as training data. With the help of a trained model, we will make predictions on new unseen data called as test data. The dataset is divided into specified ratios depending on the type of cross–validation method used. In K–Fold cross–validation, the value of K indicates the number of parts in which the dataset is divided. After division, the training of the model uses K parts, and K–1 parts are used for validation and testing [57]. This process will iterate for K times. In the end, the result of each iteration is combined. We have selected the value of K as ten for experiment. However, 10–fold cross–validation provides better model estimation with low bias and low computational cost.

6.3.7 Performance Measure

AUC is used for performance evaluation. A graph plotted over TP vs. FP rate is depicted by ROC curve. This curve is also considered as a plot between sensitivity and 1–specificity. FP rate is defined as the ratio of negative instances that were incorrectly classified. However, if the value of AUC is closer to 1, then it would be verified that the classification has been performed correctly.

6.3.8 Statistical Test

In this experiment the Friedman test, alongwith Wilcoxon–signed rank test and Nemeneyi test used.

6.4 Research Methodology

The research methodology used to conduct this study has been discussed in this section. We have developed a change prediction model for HetCPCP and WPCP using OOM.

6.4.1 WPCP

The steps followed for this approach are discussed below:

- The metrics values of various projects have been collected using Understand tool [128].
- The dataset consists of dependent and independent attributes. The dependent attribute which is the change label indicates the presence and absence of a change.

- SMOTE applied to balance the dataset.
- Appropriate FS technique that such as chi–square test used to select relevant features only.
- Prediction model build using various ML techniques.
- Performance of the prediction model evaluated using AUC measure.

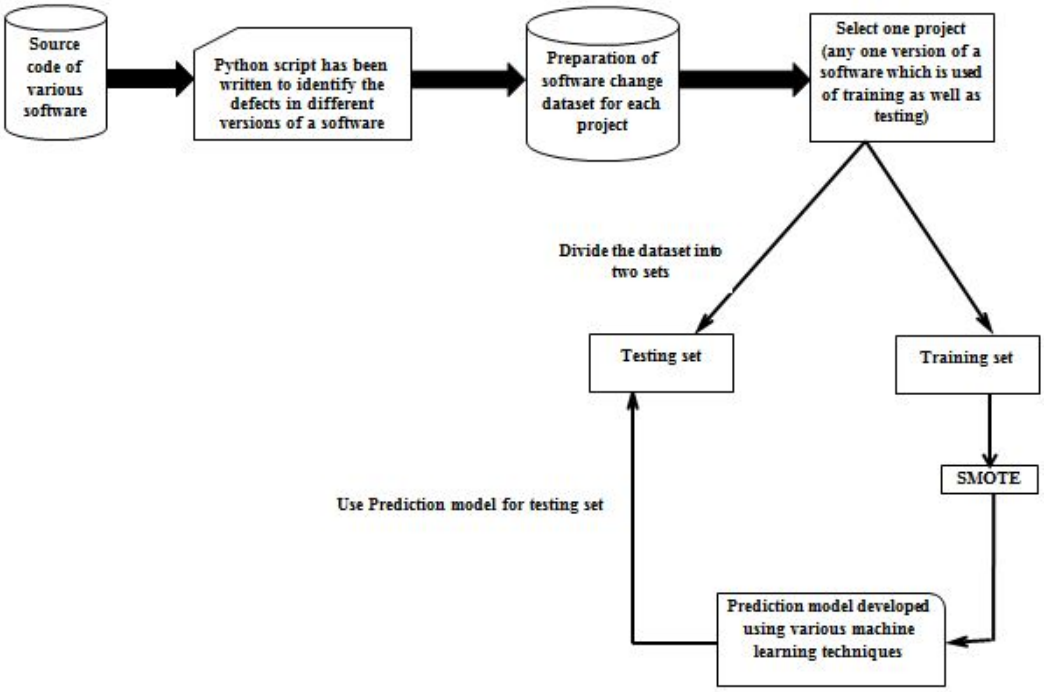

Figure 6.1: Architecture of approach used for within-project change prediction

6.4.2 HetCPCP

The steps followed for this approach are discussed below:

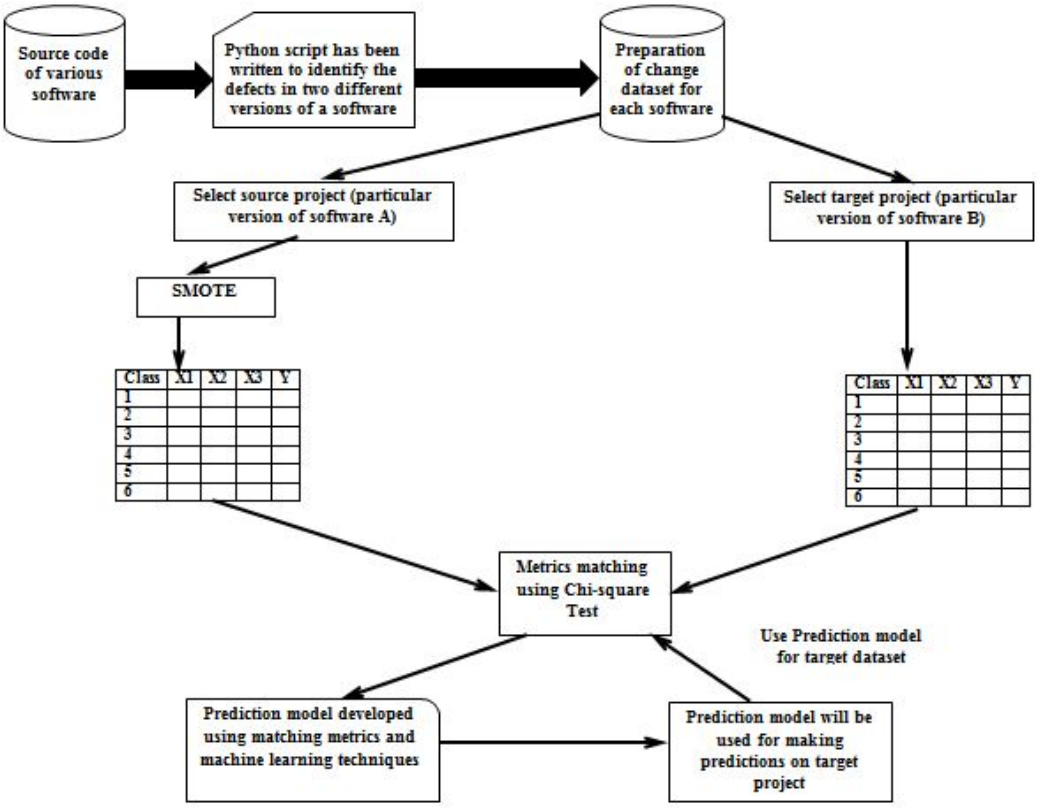

Figure 6.2: Architecture of approach used for heterogeneous cross-project change prediction

- The metrics values of various projects collected using Understand tool [128].
- Two different projects have been selected to experiment such as the source and target dataset.
- The dataset consists of dependent and independent attributes. The dependent attribute which is the change label indicates the presence and absence of a change.
- A sampling technique has been applied to balance the dataset.

- Metrics matching used as an analyzer for extracting matching metrics from source and target projects.
- Appropriate FS technique such chi–square test has been used for FS.
- The prediction model build using various ML techniques.
- The prediction model used for predicting the target dataset based on the knowledge learned from the source dataset.
- Performance of a prediction model evaluated using AUC measure.

6.5 Results and Analysis

6.5.1 Results Specific to RQ1

The AUC was used to measure the performance of various ML algorithms for HetCPCP and WPCP. The AUC values corresponding to various ML technique and dataset has been represented in Table [5.2–5.3]. Table 5.2 presents the AUC values of WPCP for various techniques and eighteen datasets used in this chapter. For WPCP, we have used the same project for training and testing. Thus, we have analyzed the performance of nine techniques for WPCP. The AUC value for the SVM technique is 0.9 for the 61.0% dataset. These 61% dataset consist of Notepad++ and CodeLite. The predictive efficiency of the HV technique is 0.91 AUC for the Notepad++ dataset. The predictive performance of KNN technique is 0.94 AUC for Notepad++ 7.5.4 to Notepad++ 7.6.3. The predictive performance of other techniques is in the range of 0.6 to 0.89 AUC. The effective performance efficiency shown in Table [5.4–5.5].

The predictive performance of HetCPDP has been analyzed in Table 5.3. In HetCPCP, we have selected different source and target datasets. The predictive performance of most of the techniques using TL is 0.6 for HetCPCP. The predictive performance of CodeBlocks 13.12 is greater than 0.71 for some specific source datasets. The source dataset that provided effective predictive performance for CodeBlocks 13.12 dataset are Notepad++ 7.5.4, Notepad++ 7.6.2, Notepad++ 7.6.3, Notepad++ 7.8.3, CodeLite–2.9.0.4684, CodeLite–3.5.5375, CodeLite–5.3, CodeLite–13.0, and CodeLite–14.0.

Table 6.2: Analysis of predictive perform of ML techniques for WPCP

S.No.	**Dataset**		**ML Algorithm**								
	Source Dataset	Target Dataset	LR	RF	SVM	HV	KNN	MLP	AB	DT	QDA
1	Notepad++ 6.8.9	Notepad++ 6.8.9	0.63	0.83	0.93	0.92	0.87	0.7	0.74	0.76	0.63
2	Notepad++ 7.3	Notepad++ 7.3	0.78	0.89	0.97	0.93	0.9	0.78	0.83	0.85	0.78
3	Notepad++ 7.5.4	Notepad++ 7.5.4	0.75	0.9	0.99	0.94	0.94	0.8	0.85	0.85	0.74
4	Notepad++ 7.6.2	Notepad++ 7.6.2	0.73	0.88	0.97	0.94	0.93	0.77	0.79	0.81	0.66
5	Notepad++ 7.6.3	Notepad++ 7.6.3	0.68	0.95	0.99	0.96	0.97	0.79	0.94	0.88	0.5
6	Notepad++ 7.8.3	Notepad++ 7.8.3	0.71	0.74	0.78	0.77	0.72	0.71	0.76	0.68	0.61
7	CodeBlocks 10.05	CodeBlocks 10.05	0.65	0.74	0.84	0.82	0.76	0.68	0.69	0.69	0.64
8	CodeBlocks 13.12	CodeBlocks 13.12	0.76	0.79	0.84	0.84	0.76	0.76	0.77	0.75	0.72
9	CodeBlocks 20.03	CodeBlocks 20.03	0.69	0.73	0.79	0.8	0.71	0.69	0.71	0.71	0.65
10	CodeLite-2.9.0.4684	CodeLite-2.9.0.4684	0.75	0.84	0.93	0.9	0.84	0.76	0.81	0.8	0.7

11	CodeLite-3.5.5375	CodeLite-3.5.5375	0.73	0.85	0.94	0.91	0.85	0.79	0.83	0.83	0.67
12	CodeLite-5.0.6213	CodeLite-5.0.6213	0.62	0.74	0.8	0.8	0.7	0.61	0.65	0.72	0.62
13	CodeLite-5.3	CodeLite-5.3	0.71	0.8	0.93	0.89	0.84	0.73	0.76	0.76	0.66
14	CodeLite-6.0.1	CodeLite-6.0.1	0.7	0.77	0.92	0.89	0.87	0.73	0.75	0.74	0.66
15	CodeLite-11.0	CodeLite-11.0	0.68	0.81	0.95	0.91	0.89	0.71	0.75	0.79	0.64
16	CodeLite-12.0	CodeLite-12.0	0.68	0.78	0.92	0.9	0.85	0.66	0.74	0.74	0.64
17	CodeLite-13.0	CodeLite-13.0	0.72	0.8	0.85	0.83	0.79	0.74	0.78	0.78	0.68
18	CodeLite-14.0	CodeLite-14.0	0.64	0.7	0.69	0.72	0.67	0.66	0.68	0.69	0.63

Table 6.3: Analysis of predictive performance of ML techniques for HetCPCP

S.No.	**Dataset**		**ML Algorithm**								
	Source Dataset	Target Dataset	LR	RF	SVM	HV	KNN	MLP	AB	DT	QDA
1	Notepad++ 6.8.9	CodeBlocks 10.05	0.65	0.69	0.59	0.66	0.6	0.62	0.69	0.67	0.59
2	Notepad++ 7.3	CodeBlocks 10.05	0.65	0.66	0.54	0.63	0.56	0.59	0.65	0.64	0.6
3	Notepad++ 7.5.4	CodeBlocks 10.05	0.65	0.64	0.57	0.63	0.6	0.61	0.65	0.63	0.6
4	Notepad++ 7.6.2	CodeBlocks 10.05	0.63	0.66	0.56	0.62	0.59	0.64	0.65	0.64	0.55
5	Notepad++ 7.6.3	CodeBlocks 10.05	0.63	0.64	0.56	0.63	0.58	0.62	0.65	0.64	0.58

6	Notepad++ 7.8.3	CodeBlocks 10.05	0.61	0.6	0.53	0.63	0.57	0.58	0.6	0.61	0.55
7	Notepad++ 6.8.9	CodeBlocks 13.12	0.71	0.71	0.61	0.69	0.62	0.7	0.72	0.7	0.56
8	Notepad++ 7.3	CodeBlocks 13.12	0.69	0.7	0.58	0.68	0.62	0.69	0.7	0.69	0.57
9	Notepad++ 7.5.4	CodeBlocks 13.12	0.7	0.72	0.62	0.69	0.64	0.68	0.71	0.7	0.6
10	Notepad++ 7.6.2	CodeBlocks 13.12	0.71	0.72	0.58	0.59	0.63	0.7	0.72	0.71	0.56
11	Notepad++ 7.6.3	CodeBlocks 13.12	0.71	0.73	0.57	0.69	0.64	0.71	0.71	0.73	0.57
12	Notepad++ 7.8.3	CodeBlocks 13.12	0.72	0.72	0.71	0.72	0.67	0.71	0.71	0.69	0.58
13	Notepad++ 6.8.9	CodeBlocks 20.03	0.59	0.59	0.53	0.54	0.53	0.6	0.58	0.57	0.53
14	Notepad++ 7.3	CodeBlocks 20.03	0.65	0.68	0.58	0.64	0.58	0.62	0.66	0.64	0.58
15	Notepad++ 7.5.4	CodeBlocks 20.03	0.62	0.64	0.58	0.61	0.56	0.64	0.63	0.61	0.57
16	Notepad++ 7.6.2	CodeBlocks 20.03	0.64	0.66	0.59	0.62	0.57	0.62	0.64	0.64	0.53
17	Notepad++ 7.6.3	CodeBlocks 20.03	0.62	0.66	0.58	0.6	0.57	0.6	0.64	0.64	0.56
18	Notepad++ 7.8.3	CodeBlocks 20.03	0.61	0.63	0.51	0.56	0.56	0.61	0.62	0.58	0.51
19	CodeLite-2.9.0.4684	CodeBlocks 10.05	0.64	0.65	0.58	0.57	0.58	0.63	0.64	0.63	0.61
20	CodeLite-3.5.5375	CodeBlocks 10.05	0.63	0.63	0.54	0.6	0.59	0.6	0.64	0.62	0.56
21	CodeLite-5.0.6213	CodeBlocks 10.05	0.65	0.63	0.57	0.62	0.58	0.6	0.65	0.62	0.61

22	CodeLite-5.3	CodeBlocks 10.05	0.65	0.67	0.55	0.64	0.59	0.62	0.67	0.65	0.61
23	CodeLite-6.0.1	CodeBlocks 10.05	0.64	0.64	0.55	0.61	0.59	0.6	0.64	0.64	0.58
24	CodeLite-11.0	CodeBlocks 10.05	0.63	0.63	0.54	0.6	0.58	0.59	0.64	0.63	0.56
25	CodeLite-12.0	CodeBlocks 10.05	0.65	0.69	0.59	0.68	0.62	0.57	0.69	0.66	0.64
26	CodeLite-13.0	CodeBlocks 10.05	0.64	0.65	0.58	0.62	0.6	0.61	0.66	0.66	0.61
27	CodeLite-14.0	CodeBlocks 10.05	0.65	0.68	0.54	0.65	0.58	0.65	0.68	0.67	0.59
28	CodeLite-2.9.0.4684	CodeBlocks 13.12	0.71	0.73	0.59	0.68	0.63	0.7	0.73	0.72	0.58
29	CodeLite-3.5.5375	CodeBlocks 13.12	0.71	0.72	0.6	0.7	0.63	0.71	0.71	0.69	0.57
30	CodeLite-5.0.6213	CodeBlocks 13.12	0.69	0.71	0.6	0.69	0.62	0.69	0.7	0.67	0.56
31	CodeLite-5.3	CodeBlocks 13.12	0.71	0.72	0.58	0.71	0.65	0.71	0.71	0.71	0.59
32	CodeLite-6.0.1	CodeBlocks 13.12	0.68	0.7	0.6	0.68	0.61	0.68	0.7	0.68	0.55
33	CodeLite-11.0	CodeBlocks 13.12	0.69	0.71	0.59	0.67	0.63	0.7	0.7	0.69	0.58
34	CodeLite-12.0	CodeBlocks 13.12	0.7	0.71	0.58	0.69	0.63	0.7	0.72	0.7	0.61
35	CodeLite-13.0	CodeBlocks 13.12	0.71	0.73	0.56	0.68	0.63	0.71	0.72	0.7	0.55
36	CodeLite-14.0	CodeBlocks 13.12	0.7	0.73	0.58	0.68	0.6	0.7	0.72	0.71	0.57
37	CodeLite-2.9.0.4684	CodeBlocks 20.03	0.66	0.68	0.56	0.63	0.61	0.65	0.66	0.65	0.55

38	CodeLite-3.5.5375	CodeBlocks 20.03	0.66	0.67	0.57	0.64	0.6	0.65	0.67	0.65	0.53
39	CodeLite-5.0.6213	CodeBlocks 20.03	0.63	0.64	0.56	0.6	0.55	0.61	0.63	0.62	0.54
40	CodeLite-5.3	CodeBlocks 20.03	0.65	0.68	0.58	0.63	0.6	0.67	0.67	0.68	0.54
41	CodeLite-6.0.1	CodeBlocks 20.03	0.64	0.66	0.57	0.62	0.59	0.63	0.67	0.65	0.55
42	CodeLite-11.0	CodeBlocks 20.03	0.64	0.66	0.59	0.63	0.59	0.62	0.66	0.64	0.56
43	CodeLite-12.0	CodeBlocks 20.03	0.62	0.65	0.54	0.61	0.57	0.62	0.64	0.62	0.55
44	CodeLite-13.0	CodeBlocks 20.03	0.66	0.68	0.58	0.62	0.58	0.66	0.67	0.66	0.56
45	CodeLite-14.0	CodeBlocks 20.03	0.64	0.67	0.51	0.61	0.59	0.65	0.66	0.64	0.54

6.5.2 Results Specific to RQ2

To analyze the performance of ML techniques, statistical tests are used. With the help of statistical differences, we have checked the statistical difference among ML techniques. According to the existing studies [129] non–parametric statistical tests are used when the normality of data is not known or homogeneity of variance in the data sample. Friedman test used to analyze the performance of nine ML techniques on eighteen different datasets. In the Friedman test, the value of significance level is 0.05 or 5% and the value of the DoF is eight (i.e., for nine ML techniques). The null hypothesis for the Friedman test states that the performance of ML techniques is not significantly different. The alternate hypothesis for the Friedman test states that there is a significant difference among the performance

of ML techniques. Here, the $\chi_{tabulated}$ obtained from the chi–square table corresponds to DoF (8) and significance level (0.05 or 5%). The value of $\chi_{tabulated}$ is 15.51. The value of $\chi_{calculated}$ is computed using Friedman's test with eighteen data samples and the value of the DoF is 8. The value of $\chi_{calculated}$ is 132.173. Thus, the value of $\chi_{calculated}$ is greater than the $\chi_{tabulated}$ i.e., the computed value lies in the range of the tabulated value. Hence, the null hypothesis is rejected and alternate hypothesis is accepted. Thus, it is concluded that there exists a significant difference among the performance of ML techniques.

The ranking of ML techniques for WPCP is computed. Table 5.4 presents the rank of each technique. Higher mean rank indicates better will be the performance of that technique. Thus, SVM performed best for WPCP in this chapter and HV is the second best technique.

Table 6.4: Mean ranks of ML techniques using Friedman test on AUC

ML Technique	**Mean Rank**
SVM	8.75
HV	8.08
RF	6.39
KNN	6.31
AB	4.72
DT	4.47
MLP	2.83
LR	2.28
QDA	1.17

Friedman test results in accepting the alternate hypothesis, then we proceed for post-hoc analysis using Nemenyi test. The statistical difference among ML techniques has been investigated using Nemenyi test. The Critical Difference (CD) value is computed as 2.832 with a number of datasets (18) and the number of ML techniques (9). In the next step, the rank difference has been calculated for every technique with every other technique.

The result obtained using Nemenyi test presented in Table 5.5. The rank difference which is greater than the CD is highlighted. It has been observed that out of 36 pairs of ML techniques, 18 pairs (highlighted entries) are found to be significantly different based on their performance. Hence, the performance of 18 pairs out of 36 pairs was found to be significantly different from other pairs, i.e., the performance of 50% of pairs is significantly different. Hence, the pairwise comparison of ML technique concluded that SVM and HV performed better than AB, DT, MLP, LR, and QDA. However, RF performed better than MLP, LR, and QDA. Thus, based on post-hoc analysis results, SVM and HV outperformed other techniques.

Table 6.5: Computation and comparison of rank difference for ML techniques (WPCP)

CD=2.832	**SVM**	**HV**	**RF**	**KNN**	**AB**	**DT**	**MLP**	**LR**	**QDA**
SVM	*	0.67	2.36	2.44	4.03	4.47	5.92	6.47	7.58
HV	*	*	1.69	1.77	3.36	3.61	5.25	5.8	6.91
RF	*	*	*	0.08	1.67	1.92	3.56	4.11	5.22
KNN	*	*	*	*	1.59	1.84	3.48	4.03	5.14
AB	*	*	*	*	*	0.25	1.89	2.44	3.55
DT	*	*	*	*	*	*	1.64	2.19	3.3
MLP	*	*	*	*	*	*	*	0.55	1.66
LR	*	*	*	*	*	*	*	*	1.11
QDA	*	*	*	*	*	*	*	*	*

6.5.3 Results Specific to RQ3

To analyze the performance of ML techniques using statistical tests for HetCPCP. The Friedman test has been used to analyze the performance of nine ML techniques on forty–five different pairs (with a combination of 15 source datasets and 3 target datasets). In the Friedman test, the value of significance level is 0.05 or 5% and the value of DoF is eight (i.e., for nine ML techniques). The null hypothesis for the Friedman test states

that the performance of ML techniques is not significantly different for HetCPCP. The alternate hypothesis for the Friedman test states that there is a significant difference in the performance of ML techniques for HetCPCP. The $\chi_{tabulated}$ obtained from the chi-square table corresponds to the DoF (8) and significance level (0.05 or 5%). The value of $\chi_{tabulated}$ is 15.51. The value of $\chi_{calculated}$ computed using Friedman's test equation with 18 data samples and the value of the DoF is 8. The value of $\chi_{calculated}$ is 302.653. Thus, the value of $\chi_{calculated}$ is greater than the $\chi_{tabulated}$ i.e., the computed value lies in the range of the tabulated value. Hence, the null hypothesis is rejected. Thus, it is concluded that there exists a significant difference in the performance of ML techniques. Table 5.6 presents the rank of each individual technique. However, RF performed best for HCPCP and ADB act as second best technique for HetCPCP. Friedman test results in accepting the alternate hypothesis, then we proceed for post hoc analysis using the Nemenyi test. The statistical difference among techniques has been investigated using the Nemenyi test. The CD value is computed as 2.276 with several datasets with combinations of 15 source datasets and 3 target datasets.

Table 6.6: Mean ranks of ML techniques using Friedman test on AUC

ML Technique	**Mean Rank**
RF	8.37
AB	7.84
LR	6.47
DT	6.14
MLP	5.2
HV	4.71
KNN	2.69
SVM	1.82
QDA	1.76

The result obtained using the Nemenyi test presented in Table 5.7. It has been observed

that out of 36 pairs of ML techniques, 21 pairs are found to be significantly different based on their performance. Hence, the performance of 21 pairs out of 36 pairs was found to be significantly different from other pairs, i.e., the performance of 58.34% of pairs is significantly different. From the pairwise comparison of the ML technique, it has been observed that RF and ADB performed better than MLP, HV, KNN, SVM, and QDA. Hence, based on post hoc analysis results, RF and AB outperformed other techniques.

Table 6.7: Computation and Comparison of rank difference for ML techniques (HCPCP)

CD=2.832	**RF**	**AB**	**LR**	**DT**	**MLP**	**HV**	**KNN**	**SVM**	**QDA**
RF	*	0.53	1.9	2.23	3.17	3.66	5.68	6.55	6.61
AB	*	*	1.37	1.7	2.64	3.13	5.15	6.02	6.08
LR	*	*	*	0.33	1.27	1.76	3.78	4.65	4.71
DT	*	*	*	*	0.94	1.43	3.45	4.32	4.38
MLP	*	*	*	*	*	0.49	2.51	3.38	3.44
HV	*	*	*	*	*	*	2.02	2.89	2.95
KNN	*	*	*	*	*	*	*	0.87	0.93
SVM	*	*	*	*	*	*	*	*	0.06
QDA	*	*	*	*	*	*	*	*	*

6.6 Discussion

The experiment conducted in this chapter to empirically validate the effect of ML techniques for WPCP and HetCPCP. The dataset used in this chapter is collected from open–source software. An empirical comparison of nine ML techniques on eighteen different datasets has been performed. The prediction model is developed using OOM. The most commonly used OOM is provided as input to the prediction model, and the change proneness label is considered as an output of the prediction model. The performance of ML techniques analyzed using the AUC performance measure. The significant difference among various ML techniques analyzed using the Friedman test with post hoc analysis

used Nemenyi test. The Nemenyi test is used to check whether statistical differences exist between the pair of various ML techniques. Some of the major findings of this study are as follows:

- SMOTE used to remove noisy data and to balance the data samples for each label of the output class.
- A matching analyzer is used to find the common metrics between the source and target dataset for HetCPDP.
- For WPCP, SVM and HV performed better than other seven ML techniques used.
- For HetCPDP, RF and ADB performed better than other seven ML techniques used.

Hence, the prediction model build using TL for HetCPCP can be used for upcoming versions of existing software. The developed prediction model helps in identifying the changes and effectively removing change prone parts in the early phases of software development life cycle, for upcoming versions of existing software. The outcomes of the experiment conducted in this chapter are demonstrated and exploratory. Thus, the change prediction depends on the characteristics of the dataset, density of change data, and classifiers used. Change prediction is useful in enhancing the software quality by identifying the highly change prone models in the early stages. Change prediction models enhance the software maintainability by identifying and modifying the change prone modules. After the identification of changed prone modules, updates are required to be done that directly impact the software maintenance. However, there might also be other factors that affect the generalization of results. We plan to validate the results on the

industry dataset in future work. Thus, the impact of other types of TL in future studies will be investigated.

Chapter 7

Empirical Validation of FS Techniques for Cross-Project Defect Prediction

7.1 Introduction

Nowadays, prediction model plays important role in ML. In software engineering, SDP is very crucial domain. To ensure better quality of a software, the defects should be identified in the early stage. It aims at identification of defects at the initial stage of development [130]. Defect prediction plays important role in improving the quality of a software in terms of maintainability, functionality, reliability and testability. In order to release a defect free software, every developer must check it from the criterion to maintain quality of a software in upcoming years also. For designing defect prediction model, it must be ensured that a sufficient amount of training data is used for designing prediction model. Moreover, the predictive model does not perform efficiently for future data. Prediction

models are useful for making prediction on unseen data or future data. However, the unseen data must have similar characteristics to use pre-designed prediction model. In the existing studies, it has been demonstrated that the SDP models are trained with sufficient amount of training dataset work efficiently for testing on the same dataset having similar distribution [131]. However, such type of prediction modes come under the category of WP. In WP, the training and testing of the prediction model is accomplished using same dataset. Moreover, the collection of sufficient amount of training data is a serious issue. In recent years, CPDP gained popularity among researchers, academicians and developers. The idea of CPDP emerged from availability of sufficient amount of training data. In CPDP, two different projects are used for training and testing. One project is considered as a source project for training and another project considered as testing project. The features of both the projects must be checked. If the features of source and target projects are same then it is termed as HoCP. If the features of both the projects are different, then it is termed as HetCP [55]. In case of HetCP, both source and target projects may contain different attributes. In order to design prediction models, the attributes must be correlated. Furthermore, the matching metrics are selected based on the amount of correlation between different pairs of source and target dataset attributes. Moreover, TL work when a prediction model is used for testing on some other project which are having similar characterizes as of training data [46]. Moreover, prediction model learned through the knowledge provided by source project. There are various types of TL settings based on the domain of source and target dataset. However, TL works perfectly if we want to use our pre-trained model for future projects. In prediction model, before starting with model development, we have to preprocess our dataset. In the next phase we extract relevant features of our dataset using various techniques that is termed as FS. techniques. These techniques [46] are categorized

into three methods such as filter method, wrapper method, and swarm search based method. Various FS selection techniques are available under each of the above method. FS helps in selecting the features which are relevant for model development with respect to output variable [132]. The main aim of conducting this study is to analyze the impact of various FS techniques for CPDP. Each and every method has a different criterion, different threshold, different parameter settings, and aspect to select relevant features. In today's era, the speed of data warehouse and database accumulation is increasing rapidly, dimensionality reduction considered as a major problem [133]. In further years, dimensionality reduction become a major problem for ML algorithms. Negative effects also observed due to scaling of dimensionality of a dataset. The presence of irrelevant and redundant features add noise to the dataset, that leads to a problem of overfitting [134]. Thus, data mining algorithms are developed in various areas such as text mining, bioinformatics, medicine, image processing, engineering, financial estimation, and sustainability. However, the significance depends on the ability to convert large amount of a data into acceptable form. Thus, it will help in knowledge discovery, increase understandability of dataset, make dataset more easy to analyze and predictable. The cause of the defects in a software module are difficult to determine [135]. Various studies exist for WPDP for impact of FS and feature reduction techniques [72], various cross-validation techniques are applied on WPDP [136]. In some of the existing studies, benchmark ML based models were also proposed for WPDP such as LR, SVM, NN, KNN, DT, NB etc. In this study, the impact of each FS technique is analyzed statistically using filter, wrapper, and swarm search based methods. The importance of the three methods of FS techniques are studied in order to improve software quality in terms of maintenance and efforts required. AEEEM and ReLink dataset are considered for experiemntation of CPDP. The projects used in this study are having similar

metrics with respect to AEEEM and ReLink. AEEEM and ReLink projects are used in pairwise combination of source and target pairs. The source and target pairs are created by combining different projects. The experiment performed for 20 pairs of AEEEM and 6 pairs of ReLink dataset. The comparison performed between different FS methods alongwith the impact on CPDP. The experiment conducted using WEKA 3.9.6. In existing studies, FS methods are not analyzed with respect to CPDP. Filter methods work on the concept of ranker while wrapper and swarm search based methods working is based on the principle of greediness. This experiment performed to evaluate the effectiveness of different FS methods in conjuction with OOM for CPDP. Furthermore, this study explored the importance of each FS method individually. Furthermore, the study also explored different settings of source and target projects with respect to the methods. Furthermore, the ML classifiers are used for building a prediction model. The results of all the predicted models are statistically validated for determining the best FS method for CPDP. RQs addressed in this study are:

- RQ1: What is the predictive capability of filter methods for CPDP with respect to AEEEM and ReLink dataset?
- RQ2: What is the predictive capability of swarm search based methods of FS techniques for CPDP?
- RQ3: What is the predictive capability of filter, wrapper and swarm search based methods for CPDP?

This chapter is organized as follows: Section 6.2 discussed experimental design and framework for conducting empirical experiment. Section 6.3 presented the results and

analysis of the experiment. Finally, Section 6.4 states the discussion of the results. The results of this chapter are published in [137].

7.2 Experimental Design

The dataset characteristics, variables of the dataset, dataset sources are discussed in this section.

7.2.1 Dataset

The dataset used for experimentation is referred in Chapter 2 Section 2.7.3.1.

7.2.2 Data Preprocessing

In this study, the dataset is preprocessed before using it in model development phase. For data preprocessing, missing values are handled, outliers are removed, null values are handled, and normalization of dataset techniques are performed. Firstly, the missing values or null values are replaced with the mean of the remaining values in the column. Thus, imputation of missing values by mean prevents information loss. Secondly, the outliers are handled using imputations. Normalization is a technique of converting data to a specified range. Furthermore, normalization techniques are helpful for designing prediction model, statistical model, forecasting model, clustering model, and classification model. Thus, normalization is performed by converting all the numerical values in [0,1] considering the data used for normalization.

7.2.3 Imbalance Dataset

In classification problem, when instances of one target class is more than the other class instances, this problem is termed as imbalanced dataset problem. There are various techniques exist to deal with imbalance dataset such as SMOTE, oversampling, undersampling, and cost-sensitive learning. SMOTE is used in this study for balancing dataset. SMOTE is a technique, that oversample the instances of minority class. Furthermore, new instances of minority class are synthesized from existing data only.

7.2.4 Filter Methods

7.2.4.1 ChiSquare (FS1)

It selects the relevant features based on ranking criterion. Furthermore, chi-square test considers independence of two events. Thus, if the value of chi-square is higher, then the attribute is more depending on the outcome variable. Hence, attribute will be selected by chi-square test.

7.2.4.2 Correlation Attribute Evaluation (FS2)

It selects the relevant attributes or features of a dataset using correlation matrix. The correlation matrix predicts the amount of correlation between independent variable using Pearson's correlation coefficient.

7.2.4.3 Gain Ratio (FS3)

Gain ratio selects the relevant attributes by using ranker. It assign ranks to the features of high-dimension. Furthermore, the features with low ranks are removed and create reduced set of features. However, the K-anonymization privacy preservation techniques is used for both original set of features and reduced set of features.

7.2.4.4 Information Gain (FS4)

Information gain also use ranker for FS. Information gain provides the amount of useful information provided by each attribute with respect to the outcome variable. It computes the gain of the attributes with respect to the target variable. However, the feature with high gain is selected.

7.2.4.5 Relief Attribute Selection (FS5)

Relief method is also based on ranker criterion. It selects the relevant features according to the differentiation between instances that are near to each other. However, Relief method was designed for binary classification problems including non-metric data is discrete or numerical features.

7.2.5 Wrapper Method

The selection of relevant features using wrapper method follow specific classifier that is used to develop model. Unlike filter method wrapper method doesn't use ranker. Wrapper method is based on principle of greediness. It evaluates all the possible combinations

for selection of relevant attributes against the evaluation criterion. The commonly used wrapper methods are forward selection, backward selection, and bi-directional elimination.

7.2.6 Swarm Search Methods

We have used six different swarm search methods are used for FS.

7.2.6.1 Best First Search (FS7)

BFS selects the attributes from an attribute set using search space. It selects the attributes by using greedy hill climbing augmented with a backtracking facility. Best first starts with empty set of attributes and start searching for attributes in a specific direction by considering all possible combinations (additions and deletions).

7.2.6.2 Genetic Search (FS8)

GENS can be used for FS and hyperparameter optimization. It finds out the optimal set of attributes on the basis of evolution.

7.2.6.3 Greedy Stepwise Search (FS9)

GRSS selects the attributes using forward and backward search through the space of attributes subset. Starting criterion is either zero or all attributes are arbitrary point in the feature space. Stopping criterion is when the performance of evaluation criterion starts decreasing after completion of addition/ deletions.

7.2.6.4 Harmony Search (FS10)

It is inspired by harmony improvisation process. Harmony search method is considered as global optimization algorithm. In harmony search, at every iteration harmonies are generated termed as solution. Furthermore, these solutions is stored in harmony memory. Harmony search method uses five parameters, three mandatory parameters, and two optional parameters.

7.2.6.5 Particle Swarm Optimization (PSO)(FS11)

PSO method is an optimization approach based on the behavioral study of animals/birds. Furthermore, PSO performed better for FS with respect to other techniques, Moreover, PSO has a capability of searching through large space, computationally less expensive, and require few parameters. Traditional PSOapproach has some limitations. However, a new variant is developed PSO(4-2) and binary PSO limit overcomes the limitations of traditional PSO approach.

7.2.6.6 Scatter Search V1 (FS12)

Scatter search methods uses attribute subset for scatter search through attribute space. However, it is starting with many significant attributes subsets and stops when the output is higher than a specified threshold or no further chances of improvement.

7.2.7 ML Classifiers

In this study, five ML classifiers are used for designing CPDP such as SGD, SMO, ADB, BNG, and LB.

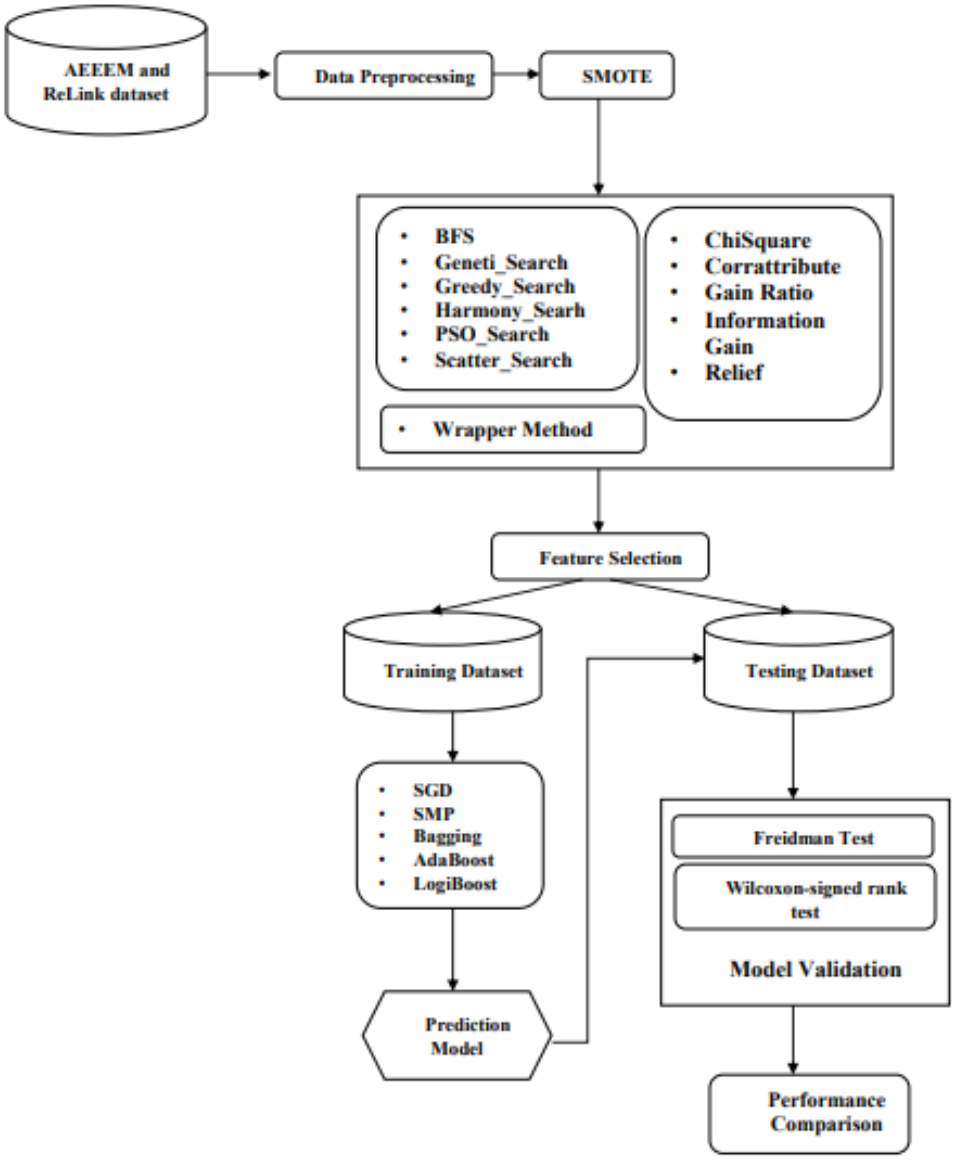

Figure 7.1: Proposed methodology

7.2.8 Statistical test

Friedman and Wilcoxon–signed rank test [72] are used for measuring the difference among the performance of various FS techniques for CPDP. The results are validated using Friedman and Wilcoxon–sign rank test. However, Wilcoxon–sign rank test is used as a post-hoc test for prediction.

7.3 Results and Analysis

The results and analysis are discussed in this section in detail. We have collected different projects of AEEEM and ReLink for experimentation. Furthermore, for performing experiment all eight projects are combined to form different pairs for CPDP. However, these pairs are formed considering the size of training dataset is lower. Moreover, the model is not more efficient, if the amount of training data is less than the testing data. Initially, 26 pairs are formed in this study, after extracting only those pairs having source data larger than the target data 13 pairs are selected. Furthermore, 12 methods are used for FS categorized as 5 filter method, 1 wrapper subset method, and 6 swarm search based methods. Furthermore, five ML classifiers are used for development of prediction model with selected features. In this study, 780 models were designed considering possible pairs of source and target dataset. The performance of each model is analyzed according to ML classifier for each FS method. However, the performance metric used in this study is AUC. AUC provides more generalized and unbiased result in case of imbalanced dataset. This section interprets the results and answers to the RQs discussed in Section 6.1.

7.3.1 Results Specific to RQ1

In this study, CPDP models are developed successfully in conjuction with software metrics. Various filter methods are used for selection of relevant attributes. Thus, filter methods are used with ranker in order to select relevant features based on ranking. Null Hypothesis (H_o): There is no significant difference among the performance of filter methods used for FS. Alternate Hypothesis (H_a): There is a significant difference among the performance of filter methods used for FS. The predictive performance of filter methods was analyzed using Friedman test for 325 CPDP models. Furthermore, the results obtained using filter methods are validated using Friedman test [138] at a significance level of 0.05. However, for 5 FS techniques and DoF as 4, the obtained chi-square value is 7.520, and *p*-value as 0.111. Moreover, *p*-value is greater than significance level that is 0.05. Thus, hypothesis testing proved that there is no significant difference among the performance of filter methods for five ML classifier for identification of defects in CP. Hence, H_o is accepted stated that all filter methods performed similar for FS with respect to HoCP. The results are presented in Table 6.1.

Table 7.1: Average AUC values for filter methods and ML classifiers for CPDP (13 pairs of AEEEM and ReLink dataset)

Filter Method ML Classifier	**FS1**	**FS2**	**FS3**	**FS4**	**FS5**
SGD	0.59	0.59	0.54	0.60	0.54
SMO	0.61	0.61	0.60	0.63	0.58
AB	0.75	0.75	0.75	0.68	0.65
BNG	0.64	0.64	0.66	0.62	0.63
LB	0.72	0.71	0.72	0.67	0.63

7.3.2 Results Specific to RQ2

To answer this RQ, various swarm search based methods are used for selection of relevant attributes in this study. Thus, filter methods are used with correlation based FS in order to select relevant features. Null Hypothesis (H_o): There is no significant difference among the performance of six swarm search methods used for FS. Alternate Hypothesis (H_a): There is a significant difference among the performance of six swarm search methods used for FS. The predictive performance of swarm search based methods was analyzed using Friedman test for 390 CPDP models. Furthermore, the results obtained using filter methods are validated using Friedman test at a significance level of 0.05. However, for 5 FS techniques and DoF as 4, the obtained chi-square value is 7.353, and *p*-value as 0.196. Moreover, *p*-value is greater than significance level that is 0.05. Thus, hypothesis testing proved that there is no significant difference among the performance of six swarm search based methods used with five ML classifier for identification of defects in CP. Hence, H_o is accepted stated that all the swarm search methods performed similar for FS with respect to HoCP. The results are presented in Table 6.2.

Table 7.2: Average AUC values for swam search based methods and ML classifiers for CPDP (13 pairs of AEEEM and ReLink dataset)

Filter MethodML Classifier	**FS7**	**FS8**	**FS9**	**FS10**	**FS11**	**FS12**
SGD	0.58	0.55	0.58	0.59	0.58	0.59
SMO	0.62	0.61	0.62	0.64	0.61	0.61
ADB	0.73	0.74	0.73	0.73	0.73	0.74
BNG	0.70	0.72	0.70	0.69	0.69	0.70
LB	0.75	0.73	0.75	0.72	0.69	0.76

Table 7.3: Wilcoxon–signed rank test result for analyzing performance of filter, wrapper, and swarm search based FS methods (18 pairs not significant))

Filter, Wrapper, and Swarm Search based FS Pairs	**Asymp. Sig. (2-tail)**
FS5 - FS2	0.043
FS5 - FS3	0.039
FS7 - FS5	0.042
FS8 - FS5	0.042
FS9 - FS5	0.042
FS10 - FS5	0.042
FS11 - FS5	0.041
FS12 - FS5	0.043
FS7 - FS6	0.038
FS8 - FS6	0.041
FS9 - FS6	0.038
FS10 - FS6	0.042
FS11 - FS6	0.042
FS12 - FS6	0.042
FS9 - FS7	0.025
FS11 - FS7	0.042
FS11 - FS9	0.042
FS12 - FS11	0.042

7.3.3 Results Specific to RQ3

To answer this RQ, the performance of models developed using filter, wrapper, and swarm search based methods are analyzed. Furthermore, the performance of 780 CPDP models are compared altogether using Friedman test at a significance level of 0.05. H_o: There is no significant difference among the performance of filter, wrapper, and swarm search methods used for FS. H_a: There is a significant difference among the performance of filter, wrapper, and swarm search methods used for FS. However, for 12 FS techniques and DoF as 11, the obtained chi-square value is 23.185, and *p*-value as 0.017. However, *p*-value is less than significance level that is 0.05. Furthermore, Friedman test resulted in acceptance of alternate hypothesis, that is there is a significant difference among the performance of all three methods and 12 techniques used for FS. Thus, H_a is accepted stated that the performance of filter, wrapper, and swarm search based FS methods differ significantly for HoCP. Furthermore, the results are validated using post-hoc test using Wilcoxon–signed

rank test. Wilcoxon–signed rank test performed pairwise comparison among all 12 techniques that formed 66 combinational pairs. Furthermore, out of 66 pairs, the Wilcoxon–signed rank test resulted in acceptance of null hypothesis for 18 pairs presented in Table 6.3. Thus, there is no significant difference among the performance of 18 pairs of FS techniques. However, for remaining 48 pairs Wilcoxon–signed rank tests resulted in acceptance of the alternate hypothesis presented in Table 6.4. Hence, there is a significant difference among the performance of 48 combinational pairs of FS techniques. Thus, the performance of swarm search based methods is comparatively better than filter and wrapper methods.

Table 7.4: Wilcoxon–signed rank test result for analyzing performance of filter, wrapper, and swarm search based FS methods (48 pairs not significant)

Filter, Wrapper, and Swarm Search FS Pairs	**Asymp. Sig. (2-tail)**	**Filter, Wrapper, and Swarm Search FS Pairs**	**Asymp. Sig. (2-tail)**
FS2 - FS1	0.686	FS9 - FS3	0.078
FS3 - FS1	0.892	FS10 - FS3	0.279
FS4 - FS1	0.180	FS11 - FS3	1.000
FS5 - FS1	0.129	FS12 - FS3	0.078
FS6 - FS1	0.684	FS5 - FS4	0.129
FS7 - FS1	0.225	FS6 - FS4	0.684
FS9 - FS1	0.225	FS8 - FS4	0.892
FS10 - FS1	0.080	FS9 - FS4	0.225
FS11 - FS1	1.000	FS10 - FS4	0.176
FS12 - FS1	0.225	FS11 - FS4	1.000
FS3 - FS2	0.336	FS12 - FS4	0.225
FS6 - FS2	0.102	FS8 - FS7	0.498
FS7 - FS2	0.414	FS10 - FS7	0.680
FS8 - FS2	1.000	FS12 - FS7	0.157
FS9 - FS2	0.223	FS9 - FS8	0.276
FS10 - FS2	0.500	FS10 - FS8	0.686
FS11 - FS2	0.343	FS11 - FS8	0.223
FS12 - FS2	0.225	FS12 - FS8	0.104
FS4 - FS3	0.890	FS10 - FS9	0.396
FS6 - FS3	0.102	FS12 - FS9	0.492
FS7 - FS3	0.276	FS11 - FS10	0.223

7.4 Discussion

The objective of conducting this study was to analyze the effectiveness of FS techniques for CPDP. However, in this study, HoCP model is designed, that considered source and target projects with similar features. In this study, three types of methods are explored for CPDP (i) Filter method (ii) Wrapper subset evaluation method and (iii) Swarm search based methods. The results are evaluated using five ML classifiers in order to detect defects in future projects with similar features to enhance maintainability, reliability, and robustness of software. Friedman and Wilcoxon–signed rank test were used for statistical validation.

Five filter methods are compared based on ranking criterion. However, all the filter methods performed on same scale. However, for wrapper attribute subset evaluation, principal of greediness is used to find out the global optimal subset of attributes with specified ML classifier. However, wrapper method performed better than filter method in most of the pairs. Furthermore, six swarm search based methods are used for FS. However, the swarm methods search for the best attribute through attributes pairs search space and stop selecting the attribute according to stopping criterion. Furthermore, swarm based methods outperformed with respect to filter and wrapper method. Hence, for conducting further experiments for CPDP swarm search based methods are preferable.

Chapter 8

Empirical Validation of Cross-Version and 10-fold Validation for Defect Prediction Models

8.1 Introduction

The software project size is increasing everyday, it directly impacts the cost and complexity of software. Due to which the defects in the software are indispensable [139, 140]. The detection and correction of defects before the beta release of the software is not feasible to assure the quality of software. The defect prediction focuses on the identification of the defect-prone modules and files of a software project. Defect prediction helps researchers, practitioners, and testers to test suspicious modules on priority. The prediction models developed in existing studies are supervised classification models that are developed using

labeled data of a project and can be used for prediction on unlabeled dataset of same project that is termed as WPDP. There is a lack of data availability due to which an efficient and reliable defect prediction model could not be developed using WPDP. The model development using WPDP requires huge amount of training data to overcome this issue. To resolve this limitation, an emerging concept of TL is employed. In TL [46], a prediction model is developed by utilizing the labeled data of other projects that is termed is CPDP. The performance of the model developed using the CPDP concept is not effective, due to data distribution differences across various projects. Existing studies in the literature created a prediction model by labeled data of earlier versions and predict the unlabeled module'data of upcoming or future versions of the same project that is termed as Cross-Version Defect Prediction (CVDP) [141, 142]. In the upgraded version of any software some of the modules would be deleted, some of the modules would be added, and some additional functionality would be added. But, in terms of data distribution difference, the differences across multiple versions of the same project are low in comparison to data distribution difference across cross-projects because the upcoming version carries a huge amount of information or data from the existing version of that software [143]. Consider Project X.1, Project X.2, and Project Y. If Project X.1 is used for training and testing then it comes under WPDP. If Project X.1 is used for training and Project X.2 is used for testing then it comes under CVDP. If Project X is used for training and Project Y is used for testing, then it comes under CPDP. This study demonstrated the applicability of CVDP. However, various studies exist in the literature for change and defect prediction using CVDP [110, 111, 144], but no study discussed the applicability of CVDP for generalization. However, the investigation of CVDP is aimed for generalizability. In the literature, existing studies developed defect prediction models using either CPDP or WPDP. But to enhance

the generalizability of results and removal of bias in the study analyzed the performance of cross-version and 10-fold cross-validation. The following RQs are formulated which are as follows:

- RQ1: What is the predictive capability of defect prediction models developed using cross-version validation?
- RQ2: What is the predictive capability of the defect prediction model developed using 10-fold cross-validation?
- RQ3: What is the feasibility of implementing cross-version in comparison to the performance of model developed using 10-fold cross-validation i.e. to what extent cross-version validation is feasible?

In this study, the SDP model is developed using various ML techniques. The generalizability of the prediction model is analyzed using cross-version and 10-fold cross-validation.

The organization of this chapter is as follows: Section 7.2 describes the framework of the experiment, Section 7.3 presented the analysis. Section 7.4 discussed the conclusion. The results of this chapter are published in [145].

8.2 Experimental Framework

The experimental framework is discussed in this section.

8.2.1 Empirical Dataset

The dataset used for experimentation is referred in Chapter 2 Section 2.7.3.1.

8.2.2 Model Development

The prediction model developed using ML techniques are used for developing prediction model such as NB, LR, MLP, ADB, BNG, J48, and RF.

8.2.3 Performance Measure

The predictive performance of defect prediction model developed in this study is analyzed based on the prevalent performance measure. AUC is used as a performance measure. AUC ROC curve can be calculated and provides a single score to summarize the plot that can be used to compare models.

8.2.4 Validation Method

In this study, to develop defect prediction model, a 10-fold cross-validation is used. During the model development process using training dataset, the dataset is divided into ten parts of identical size. For validation, randomly nine parts are used for training purpose and tenth part is used for testing. This process will repeat iteratively ten times for validation.

8.3 Results and Analysis

This section presents the answers to the RQs of this chapter discussed in Chapter 7.1 and their analysis.

8.3.1 Results Specific to RQ1

To answer this RQ, we have experimented using seven ML techniques on different versions of all 8 projects out of 10 projects, 2 projects have one version only. We have created prediction model for all the 30 combinations and the AUC values have been noted down in Table 7.1. From the AUC values, it has been observed that in some of the cases CVDP is outperformed. The results are statistically validated using the Friedman test followed by post-hoc analysis that is Wilcoxon–signed rank test. For Friedman test, we have formed and tested the following hypothesis.

Null Hypothesis (H_{o1}):- There is no significant difference among the performance of all ML techniques for CVDP.

Alternate Hypothesis (H_{a1}): There is a significant difference among the performance of all ML techniques for CVDP.

The ranks obtained using Friedman test of all the ML techniques are shown in Table 7.2. The *p*-value obtained using the Friedman test is 0.000 which is less than the significance value. Hence, null hypothesis (H_o) is rejected and alternate hypothesis (H_a)is accepted.

Table 8.1: AUC values for cross-version defect prediction

Techniques Used	**NB**	**LR**	**MLP**	**ADB**	**BNG**	**J48**	**RF**
ant_ver1.6 → ant_ver1.7	0.779	0.778	0.75	0.8	0.826	0.679	0.803
camel_ver1.0 → camel_ver1.2	0.49	0.586	0.56	0.589	0.558	0.544	0.574
camel_ver1.0 → camel_ver1.4	0.482	0.644	0.636	0.631	0.644	0.565	0.677
camel_ver1.0 → camel_ver1.6	0.552	0.63	0.618	0.639	0.638	0.552	0.651
camel_ver1.2 → camel_ver1.4	0.687	0.68	0.666	0.668	0.761	0.654	0.765
camel_ver1.2 → camel_ver1.6	0.642	0.589	0.593	0.619	0.656	0.621	0.677
camel_ver1.4 → camel_ver1.6	0.633	0.628	0.679	0.64	0.68	0.632	0.682
jedit_ver3.2 → jedit_ver4.0	0.751	0.776	0.788	0.755	0.8	0.756	0.824
jedit_ver3.2 → jedit_ver4.1	0.759	0.81	0.788	0.775	0.795	0.727	0.813
jedit_ver3.2 → jedit_ver4.2	0.785	0.805	0.787	0.804	0.825	0.738	0.833

jedit_ver3.2 → jedit_ver4.3	0.608	0.64	0.589	0.575	0.586	0.531	0.625
jedit_ver4.0 → jedit_ver4.1	0.799	0.811	0.763	0.808	0.839	0.667	0.836
jedit_ver4.0 → jedit_ver4.2	0.826	0.824	0.764	0.83	0.823	0.655	0.852
jedit_ver4.0 → jedit_ver4.3	0.628	0.702	0.672	0.645	0.631	0.611	0.617
jedit_ver4.1 → jedit_ver4.2	0.837	0.872	0.797	0.817	0.842	0.784	0.858
jedit_ver4.1 → jedit_ver4.3	0.629	0.669	0.643	0.586	0.599	0.606	0.688
jedit_ver4.2 → jedit_ver4.3	0.624	0.648	0.595	0.594	0.686	0.759	0.706
log4j_ver1.0 → log4j_ver1.1	0.84	0.802	0.802	0.803	0.783	0.748	0.785
log4j_ver1.0 → log4j_ver1.2	0.621	0.728	0.708	0.66	0.694	0.586	0.666
log4j_ver1.1 → log4j_ver1.2	0.635	0.626	0.593	0.64	0.504	0.413	0.516
lucene_ver2.0 → lucene_ver2.2	0.64	0.641	0.615	0.666	0.657	0.564	0.658
lucene_ver2.0 → lucene_ver2.4	0.68	0.704	0.65	0.671	0.702	0.636	0.699
lucene_ver2.2 → lucene_ver2.4	0.72	0.678	0.658	0.701	0.648	0.658	0.639
poi_ver2.5 → poi_ver3.0	0.766	0.677	0.597	0.613	0.614	0.548	0.619
xalan_ver2.4 → xalan_ver2.5	0.584	0.619	0.567	0.632	0.646	0.472	0.619
xalan_ver2.4 → xalan_ver2.6	0.756	0.702	0.586	0.707	0.693	0.46	0.69
xalan_ver2.4 → xalan_ver2.7	0.79	0.762	0.528	0.6	0.778	0.743	0.808
xalan_ver2.5 → xalan_ver2.6	0.581	0.635	0.71	0.688	0.725	0.672	0.688
xalan_ver2.5 → xalan_ver2.7	0.705	0.661	0.768	0.826	0.821	0.722	0.883
xalan_ver2.6 → xalan_ver2.7	0.835	0.841	0.778	0.9	0.903	0.868	0.908

Table 8.2: Mean Rank of machine learning techniques

ML Technique	**Mean Rank**
J48	1.93
MLP	3.23
NB	3.65
ADB	4.22
LR	4.58
BNG	4.82
RF	5.57

Table 8.3: Wilcoxon–signed rank test for cross-version defect prediction

Technique	***p*-value**	**Difference (S/NS)**
LR - NB	0.361	NS
MLP - NB	0.361	NS

ABABOOST - NB	0.361	NS
BAGGING - NB	0.045	NS
J48 - NB	0.009	NS
RANDOMFOREST - NB	0.002	S
MLP - LR	0.001	S
ADABOOST - LOG	1.000	NS
BAGGING - LOG	1.000	NS
J48 - LOG	0.002	S
RANDOMFOREST - LOG	0.265	NS
ABABOOST - MLP	0.100	NS
BAGGING - MLP	0.018	NS
J48 - MLP	0.000	S
RANDOMFOREST - MLP	0.002	S
BAGGING - ABABOOST	0.100	NS
J48 - ABABOOST	0.001	S
RANDOMFOREST - ABABOOST	0.026	NS
J48 - BAGGING	0.000	S
RANDOMFOREST - BAGGING	0.045	NS
RANDOMFOREST - J48	0.000	S

Wilcoxon–signed rank test is performed to validate the alternate hypothesis of the Friedman test. The Wilcoxon–signed rank test results are shown in Table 7.3. All ML techniques are paired with other techniques, in such a manner, total 21 pairs are formed. The *S* denotes a significant difference and *NS* denotes no significant difference among that pair. If the *p*-value is greater than the significance level (0.05), then it is indicated that there is no significant difference among that pair. From Table 7.3, it has been observed that RF performed better than NB in CVDP. MLP and J48 performed better than LR. J48 and RF performed better than MLP. J48 performed better than BNG and ADB. Hence, RF performed better than J48 all other ML techniques for CVPD. Thus, RF can be used for CVDP in upcoming projects with multiple versions of the same project.

Therefore, for RQ1 it is concluded that CVDP models developed with the applications

of J48, MLP, NB, ADB, LR, BNG and RF provided AUC in the range of 0.49 to 0.908 for overall 30 pairs. RF is outperformed in comparison to all other ML techniques used in a study for prediction of defect prone modules based on the Friedman test result.

8.3.2 Results Specific to RQ2

To answer this RQ, the performance of SDP is evaluated and analyzed on the same training and testing dataset. The performance of the evaluated model is noted down in Table 7.4 for all 26 datasets. NB does not perform better in comparison to other ML techniques. The AUC values obtained using NB are low. The prediction model is developed using 10-fold cross-validation for all datasets using 7 ML techniques. To validate the results Friedman test used followed by post-hoc analysis that is Wilcoxon–signed rank test. The Friedman test is performed with a significance level of 0.05 and 6 DoF. The hypothesis created and tested for the Friedman test is as follows:

Table 8.4: AUC values for 10-fold cross-validation

Techniques Used	**NB**	**LR**	**MLP**	**ADB**	**BNG**	**J48**	**RF**
ant_ver1.6 → ant_ver1.6	0.823	0.844	0.807	0.82	0.882	0.82	0.904
ant_ver1.7 → ant_ver1.7	0.814	0.823	0.8	0.843	0.88	0.738	0.912
camel_ver1.0 → camel_ver1.0	0.834	0.736	0.735	0.883	0.825	0.68	0.863
camel_ver1.2 → camel_ver1.2	0.59	0.654	0.666	0.649	0.805	0.714	0.853
camel_ver1.4 → camel_ver1.4	0.678	0.678	0.637	0.652	0.701	0.592	0.73
camel_ver1.6 → camel_ver1.6	0.69	0.72	0.735	0.724	0.855	0.762	0.885
ivy_ver2.0 → ivy_ver2.0	0.809	0.827	0.803	0.83	0.879	0.8	0.916
jedit_ver3.2 → jedit_ver3.2	0.781	0.855	0.864	0.851	0.879	0.803	0.925
jedit_ver4.0 → jedit_ver4.0	0.769	0.811	0.832	0.813	0.88	0.804	0.914
jedit_ver4.1 → jedit_ver4.1	0.795	0.848	0.829	0.79	0.87	0.794	0.905
jedit_ver4.2 → jedit_ver4.2	0.845	0.864	0.838	0.873	0.906	0.772	0.928
jedit_ver4.3 → jedit_ver4.3	0.707	0.75	0.819	0.896	0.855	0.719	0.92
log4j_ver1.0 → log4j_ver1.0	0.863	0.821	0.846	0.84	0.861	0.767	0.903
log4j_ver1.1 → log4j_ver1.1	0.832	0.877	0.872	0.904	0.882	0.788	0.914
log4j_ver1.2 → log4j_ver1.2	0.792	0.81	0.851	0.823	0.887	0.798	0.936
lucene_ver2.0 → lucene_ver2.0	0.75	0.765	0.743	0.771	0.826	0.712	0.863

lucene_ver2.2 → lucene_ver2.2	0.624	0.643	0.706	0.699	0.757	0.687	0.826
lucene_ver2.4 → lucene_ver2.4	0.756	0.792	0.81	0.77	0.85	0.694	0.893
poi_ver2.5 → poi_ver2.5	0.773	0.849	0.879	0.888	0.918	0.875	0.944
poi_ver3.0 → poi_ver3.0	0.8	0.824	0.818	0.883	0.906	0.831	0.932
synapse_ver1.2 → synapse_ver1.2	0.768	0.789	0.81	0.823	0.843	0.757	0.892
xalan_ver2.4 → xalan_ver2.4	0.757	0.801	0.812	0.815	0.881	0.766	0.916
xalan_ver2.5 → xalan_ver2.5	0.604	0.695	0.719	0.71	0.838	0.759	0.89
xalan_ver2.6 → xalan_ver2.6	0.781	0.805	0.821	0.828	0.9	0.814	0.924
xalan_ver2.7 → xalan_ver2.7	0.925	0.933	0.822	0.93	0.894	0.811	0.958
xerces_ver1.4 → xerces_ver1.4	0.858	0.931	0.943	0.954	0.967	0.901	0.974

Null Hypothesis (H_{o2}):- For 10-fold cross-validation, there is no significant difference in the efficiency of all ML techniques.

Alternate Hypothesis (H_{a2}): For 10-fold cross-validation, there is a significant difference in the efficiency of all ML techniques.

The ranks obtained using Friedman test of all the ML techniques are shown in Table 7.5. The computed *p*-value obtained using the Friedman test is 0.000 which is less than the significance value. Hence, H_o is rejected and alternate H_a FAL THE is accepted.

Wilcoxon–signed rank test is performed to validate the alternate hypothesis of the Friedman test. Table 7.6 shows the Wilcoxon–signed rank test results. All the techniques are paired with other techniques, in such a manner, total 21 pairs are formed. The *S* denotes a significant difference and *NS* denotes no significant difference among that pair. If the *p*-value is greater than the significance level (0.05), then it indicates that there is no significant difference among that pair. From Table 7.6, it has been observed that LR, ADB, BNG, RF performed better than NB. BNG and RF performed better than LR. BNG and RF performed better than MLP. BNG, J48, and RF performed better than ADB. J48 and RF performed better than BNG. RF performed better than J48. Hence, RF outperformed

all other ML techniques for 10-fold cross-validation.

Table 8.5: Mean Rank of ML Techniques

ML Technique	**Mean Rank**
NB	2.28
J48	2.28
LR	3.25
MLP	3.37
AB	4.23
BAGGING	5.70
RF	6.88

Table 8.6: Wilcoxon–signed rank test for 10-fold cross-validation

Technique	***p*-value**	**Difference (S/NS)**
LR - NB	0.000	S
MLP - NB	0.201	NS
AB - NB	0.001	S
Bagging - NB	0.000	S
J48 - NB	0.855	NS
RF - NB	0.000	S
MLP - LR	0.584	NS
AB - LR	0.018	NS
Bagging - LR	0.000	S
J48 - LR	0.100	NS
RF - LR	0.000	S
AB - MLP	0.201	NS
Bagging - MLP	0.000	S
J48 - MLP	0.006	NS
RF - MLP	0.000	S
Bagging - AB	0.001	S
J48 - AB	0.001	S
RF - AB	0.000	S
J48 - Bagging	0.000	S
RF - Bagging	0.000	S
RF - J48	0.000	S

Therefore, we concluded that 10-fold cross-validation, defect prediction models devel-

oped with the applications of J48, MLP, NB, ADB, LR, BNG and RF gave AUC in the range of 0.59 to 0.974 for overall 26 pairs. RF outperformed in comparison to all other ML techniques used in this chapter for the prediction of defect-prone modules based Friedman test result.

8.3.3 Results Specific to RQ3

The performance of prediction models developed using various ML techniques for cross-version validation should not vary significantly from the model developed using 10-fold cross-validation by applying same ML techniques to research the applicability of cross-version methodology to recognize defective modules of the software. Cross-version validation applicability to defect prediction models, is limited due to the significant amount of difference. The following hypothesis is established and tested to prove it:

Null Hypothesis (H_{o3}): The performance of CVDP and the defect prediction model produced using 10-fold cross-validation are not significantly different.

Alternate Hypothesis (H_{a3}): The performance of CVDP and the defect prediction model established using 10-fold cross-validation differs significantly.

To determine the feasibility of cross-version validation, the findings are statistically validated. On the results of seven ML techniques, we used pairwise Wilcoxon test with a significance level of 0.05. The performance of seven ML techniques is analyzed individually for both cross-version and 10-fold cross-validation with respective AUC values.

The pair-wise Wilcoxon test results are presented in Table 7.6 for cross-version and 10-fold cross-validation. Table 7.7 depicts the *p*-value obtained from the pair-wise Wilcoxon test of seven ML techniques for cross-version and 10-fold cross-validation. From Table

7.7 it has been observed that the p-value is less than that of significance level (0.05). Therefore, the null hypothesis (H_{o3}) is rejected and the alternate hypothesis is accepted. Thus, there exists a significant difference among the performance of cross-version and 10-fold cross-validation. The answer of RQ3 is concluded such that: the performance of defect prediction modules developed using cross-version and 10-fold cross-validation differ significantly. Thus, it has been observed that usage of existing versions of software for upcoming software is not feasible in comparison to 10-fold cross-validation.

Table 8.7: Wilcoxon test on CVDP and model developed using 10-fold cross-validation

Technique	***p*-value**
NB^{CV} - NB^{TF}	0.002
LR^{CV} - LR^{TF}	0.000
MLP^{CV} - MLP^{TF}	0.000
AB^{CV} - AB^{TF}	0.000
$Bagging^{CV}$ - $Bagging^{TF}$	0.000
$J48^{CV}$ - $J48^{TF}$	0.000
RF^{CV} - RF^{TF}	0.000

8.4 Discussion

The feasibility of cross-version is analyzed using 26 versions of 10 different projects. Two experiments are conducted in this chapter. In the first experiment, the defect prediction model is developed using cross-version validation method. The defect prediction model is developed using 10-fold cross-validation in the second experiment, and the performance of this model is compared to that of a model developed using cross-version. The results are validated using statistical test. The results obtained from paired Wilcoxon test proved that it is not feasible to implement the CVDP model in comparison to the model developed

using 10-fold cross-validation.

Cross-version helps in improving the quality of software by using existing versions of software to identify defects in the updated version of that project. It has been observed that the training and testing dataset of the same project is efficient instead of using different versions of a project as a training and testing dataset.

Chapter 9

Optimized Artificial Neural Network for Heterogeneous Cross-Project Defect Prediction using Grid Search

9.1 Introduction

Nowadays, SDP is a key domain for efficient utilization of resources in a reasonable amount of time. The identification of defective modules resulted in the efficient utilization of resources for academics and industrial workplaces. In ML, the prediction model holds significant importance. In the past years, the defect prediction model trained using empirical dataset for the identification of defects in the test data [146]. The defect prediction models can be developed considering two situations, one is WPDP, and another is CPDP. Types of defect prediction models differ based on the projects used for training and testing

[147]. In the context of WPDP, the prediction model is trained and tested using same project. However, WPDP can be further used for the identification of defects in the upcoming versions of same project and helps in enhancing the functionality of upcoming projects. Nowadays, data is exhausted for experiments in the upcoming era. Due to this researchers are facing issues with the availability of data. The constraint of the limited amount of data availability can be removed by considering different projects for the development of prediction models.

CPDP utilizes different projects for model development and testing. The prediction model designed for CPDP is based on the concept of TL. TL is based on knowledge transfer learned from one task to another task. The knowledge is transferred using different mechanisms on the basis of two different projects characteristics [54]. The source and target domain considers the same project [148] for WPDP. However, CPDP consider different projects for source and target data. TL is one of the methodology to conduct CPDP. Further, TL helps in improving the software quality for future projects with similar feature distribution. The source and target projects consist of different features that is termed as heterogeneous TL.

The study conducted to analyze the effectiveness of ANN for CPDP and grid search optimization with ANN for CPDP. ANN is efficient for learning a compact representation of different behaviors rapidly. They are flexible in managing various behaviors sequentially. One of the advantage of using ANN is to use it without relearning the synaptic weights or strengths. The CPDP considered two scenarios such as HoCPDP and HetCPDP. Both scenarios vary depending on the feature types in the training and testing project. In HoCPDP, the training and target projects consist of similar features. In HetCPDP, the training and testing dataset contain different features with some similarities. Moreover,

there must be some relationship between the training and testing dataset when few features are considered. In case of HetCPDP, the idea of TL emerged [149]. The main aim of integrating TL for CPDP is to enhance the quality of software. The software quality depends on various parameters depending on the types of features considered for experimentation. The software quality attributes are categorized into two categories based on functional and non-functional requirements. Further, the quality attributes consider various measures for quality estimation using software metrics. In existing research, the researchers considered traditional ML algorithms for analyzing the efficiency of CPDP models.

The experiment is conducted using AEEEM and NASA datasets. In this chapter, ANN is used for CPDP grid based model development. This study also identified the grid based ANN model efficiency with optimization algorithms for CPDP. The grid based ANN model performed better in comparison to traditional ANN model for CPDP. The RQs addressed in this study are: RQ1: What is a predictive capability of ANN for CPDP considering AEEEM and NASA projects? RQ2: What is the predictive capability of the grid search ANN model for the AEEEM and NASA dataset?

The chapter is organized in the following manner: Section 8.2 explains the research methodology. Section 8.3 presented result and analysis. Section 8.4 presented conclusion and future direction.

9.2 Research Methodology

The research methodology followed to conduct the experiment discussed in this chapter.

9.2.1 Dataset

The dataset used for experimentation is referred in Chapter 2 Section 2.7.3.1.

9.2.2 Data Preprocessing

The dataset used in this study is preprocessed using various data preprocessing techniques. Thus, the techniques used in this study are handling missing values, outliers removal, null values handling, and normalization of a dataset is performed.

9.2.3 Optimization Algorithm

In this chapter, three optimization algorithms are used such GrS, PSO, and Evolutionary Search (EVS). GrS for ANN architecture starts with a small network and iteratively leads with the selection of neurons and hidden layers till final output. PSO used as an optimization technique for training ANN. The basic idea is to use PSO to search for the optimal set of weights and biases for ANN that minimizes the error between the predicted outputs and actual outputs. EVS used to find out optimal set of weights and biases to minimize the error between predicted and actual outputs.

9.2.4 Techniques used for model development

In this chapter, traditional ANN with default parameter settings and ANN with optimization algorithms such as GS, PSO, and EV under varying settings of ANN parameters were used for the development of prediction models.

9.3 Results and Analysis

This section presents the results and analysis of answers to the RQs. In this section, the answers to RQs are discussed in detail. The experiment conducted in this experiment used two projects of AEEEM and eight projects of NASA. The projects used for experimentation consist of different features such as Halstead metrics, and OOM. To perform HetCPDP using TL, the setting of selected projects has been fixed. AEEEM projects are used for training of prediction model and NASA projects are used for testing of developed prediction model. Further, 16 pairs are formed for experimentation. In this study, 112 models were designed considering possible pairs of source and target datasets. The performance of each pair is analyzed according to the selected classifier with default and grid search parameter settings. However, the performance of prediction model analyzed using AUC metric. AUC provides more generalized and unbiased results in the case of imbalanced dataset. This section interprets the results and answers to the RQs discussed in Section 8.1.

9.3.1 Results Specific to RQ1

In this study, CPDP models are developed using various projects with different features. CPDP is developed using ANN with default parameter settings. In default parameter settings, batch size = 100, learning rate = 0.3, momentum = 0.2, epochs (no.) = 500. The traditional ANN performance does perform well with default parameter settings. During experimentation, we have to see features of both training and testing dataset must have the same vector. If the feature vector space is not matching, then TL concept is not applicable. Since, both source and target projects have similar features considering

OOM. Thus, the challenging task was to extract relevant features from both the projects. Moreover, training and testing projects consist of different features. The results obtained using ANN for CPDP are summarized in Table 8.1. Based on covariance among the features of both projects, specified feature pairs were selected for designing the prediction model. Further, the prediction model is used for defect prediction in upcoming projects with similar characteristics. ANN based prediction model can be improved using grid search optimization.

Table 9.1: AUC values for ANN and grid search ANN model for CPDP

Source Dataset → Target Dataset	**ANN**	**ANN+PSO**	**ANN+EVS**	**ANN+GrS**
EQ → CM1	0.5	0.79	0.74	0.79
EQ → KC1	0.57	0.74	0.7	0.73
EQ → KC3	0.65	0.64	0.65	0.64
EQ → MC2	0.65	0.78	0.76	0.79
EQ → MW1	0.47	0.72	0.7	0.69
EQ → PC1	0.54	0.69	0.65	0.66
EQ → PC2	0.53	0.65	0.63	0.67
EQ → PC3	0.59	0.73	0.72	0.75
Lucene → CM1	0.49	0.72	0.7	0.78
Lucene → KC1	0.53	0.68	0.72	0.7
Lucene → KC3	0.57	0.7	0.67	0.7
Lucene → MC2	0.69	0.65	0.69	0.67
Lucene → MW1	0.5	0.69	0.67	0.69
Lucene → PC1	0.2	0.68	0.67	0.64
Lucene → PC2	0.3	0.78	0.72	0.79
Lucene → PC3	0.5	0.71	0.7	0.74

9.3.2 Results Specific to RQ2

To answer this RQ, the default parameters of ANN were changed using grid search approach. In the grid search approach, the model has been trained multiple times for varying sets of parameter values. In the default parameter setting, the learning rate was 0.3, which is used for training large and complex datasets. However, the dataset is not much larger in this study. Thus, the experiment has been conducted for varying sets of learning values, which provides the best performance of the ANN model at a learning rate of 0.01. In optimized ANN the number of hidden layers are updated with number of neurons in each hidden layer. In ANN, we have used the logistic activation function for experimentation. The performance of the grid search ANN model for CPDP was 0.79. In some of the pairs, it was 0.3 also depending on the size of the training and testing dataset. ANN ensures that the size of the training and testing dataset is same. Furthermore, statistical analysis has been done to analyze the performance of techniques statistically. The null hypothesis states that there is no significant difference in the performance of four techniques. The alternate hypothesis says that there is a significant difference in the performance of four prediction models. Thus, based on the Friedman test result at a 0.05 significance level, the computed chi-square value is 16.016. The mean rank of four techniques are represented in Table 8.2. Hence, an alternate hypothesis is accepted such that there is statistical difference among the performance of four prediction model. The next step is to analyze the performance of each technique individually using post-hoc analysis such as Wilcoxon–signed rank test. Furthermore, six combinational pairs are formed for Wilcoxon–signed rank test.

The hypothesis for the further comparison among six pairs are as follows: Null Hypothesis (H_{o1}): There is no significant difference among the performance of ANN_PSO and ANN. Alternate Hypothesis (H_{a1}): There is a significant difference among the perfor-

mance of ANN_PSO and ANN. Thus, the Asymp. Sig. (2-tailed) is 0.003 at at significance level 0.05. Hence, Ha1 is accepted. (H_{o2}): There is no significant difference among the performance of ANN_EVS and ANN. (H_{a2}): There is a significant difference among the performance of ANN_EVS and ANN. Thus, the Asymp. Sig. (2-tailed) is 0.003 at at significance level 0.05. Hence, Ha2 is accepted. (H_{o3}): There is no significant difference among the performance of ANN_GrSand ANN. (H_{a3}): There is a significant difference among the performance of ANN_GrSand ANN. Thus, the Asymp. Sig. (2-tailed) is 0.003 at significance level 0.05. Hence, (H_{a3}) is accepted. (H_{o4}): There is no significant difference among the performance of ANN_EVS and ANN_PSO. (H_{a4}): There is a significant difference among the performance of ANN_EVS and ANN_PSO. Thus, the Asymp. Sig. (2-tailed) is 0.105 at significance level 0.05. Hence, (H_o4) is accepted. (H_{o5}): There is no significant difference among the performance of ANN_GrSand ANN_PSO. (H_{a5}): There is a significant difference among the performance of ANN_GrSand ANN_PSO. Thus, the Asymp. Sig. (2-tailed) is 0.472 at at significance level 0.05. Hence, (H_o5) is accepted. (H_{o6}): There is no significant difference among the performance of ANN_GrSand ANN_EVS. Ha6: There is a significant difference among the performance of ANN_GrSand ANN_EVS. Thus, the Asymp. Sig. (2-tailed) is 0.027 at at significance level 0.05. Hence, (H_{o6}) is accepted. However, it has been observed that ANN+GrS outperformed in comparison to other models.

Table 9.2: Model Mean Ranks

Model	**Mean Rank**
ANN	1.38
ANN_PSO	2.92
ANN_EVS	2.46
ANN_GrS	3.23

9.4 Discussion

The main aim of conducting experiment is to analyze the efficiency and evaluate the performance of ANN with ANN grid search model for CPDP. HetCPDP is developed in this study considering projects from different repositories. In the source dataset, the AEEEM dataset was specified, while NASA datasets were specified in the target dataset. The number of features differs in both, the CPDP model is designed considering similarity among features to some extent. The experiment performed in this study showed that traditional ANN performance was not much significant. Grid based ANN performed better than traditional ANN considering the parameters setting using a grid based approach. Grid search ANN model worked better by considering more number of neurons and hidden layers for CPDP. Grid search ANN model outperformed ANN based on no. of neurons in the hidden layer, no. of hidden layers, and learning rate value. Grid search provides an effective combination of hyperparameters. Moreover, grid search is expensive, especially for large networks with many hyperparameters. In future work, random search or bayesian optimization may be used as an alternative to grid search with deep learning. The grid search ANN model can be tested further with more optimization algorithms for HetCPDP and HoCPDP.

Chapter 10

Development of Prediction Model using Grid Search Grey Wolf Optimization and Optimized Artificial Neural Network

10.1 Introduction

In today's era SDP is one of the important research area in the software engineering field. SDP helps in developing a more suitable prediction model with a reasonable amount of resources and in a reasonable amount of time. The main aim of developing SDP model is to release defect free software to the end user. In order to minimize the failure of software's, the SDP models helps in maintaining the good quality of software by considering various

quality attributes. During software quality life cycle phase the defects are identified in each phase. Software quality is also ensured using defect prediction models considering relevant quality attribute and domain. Researchers, academicians, and industry experts are focused on developing defect prediction model using Traditional ML, Deep Learning (DL), and Artificial Intelligence (AI).

In this study, defect prediction models are developed based on different criterion. Firstly WPDP, and secondly CPDP. In WPDP, the same project is used for training and testing. Based on the specified criterion, the dataset is divided into 80:20 or 70:30 that is amount of training set: amount of testing set ratio. Furthermore, the managerial implications is to improvise the software quality using different measures. Thus, using experimental settings of this study such as testability, usability, reusability, interoperability, functionality of the system can be improvised in future projects. However, the study aims at identification, and removal of defects with reasonable amount of sources, and minimal amount of time.

In CPDP, different projects are used for training and testing dataset. However, in today's era most of the researchers prefer to use CPDP approach. Due to limitation of data, CPDP provides more effective results. Based on the types of features in training and testing dataset, CPDP is categorized in two sub-categories. The first sub-category is HetCPDP, and second sub-category is HoCPDP. Homogeneous CPDP is based on similar feature distribution and space among the training and testing dataset. Moreover, HetCPDP is based on different space and feature distribution among the training and testing dataset. However, in the existing GWO there are various demerits, such as low predictive capability, slow convergence, and local search. Thus, improvised or modified GWO is developed in this study. The modified GWO technique is developed in combination with RF.

Defect prediction models are helpful in developing an efficient, reliable, and robust

software. In the existing research, the defect prediction models are developed using various ML algorithms. In this study, three models are developed for WPDP first is WPDP_GWO, second is 10-fold GWO model, and last is GrS_GWO. In the existing research, the GWO is analyzed for CPDP, but not for WPDP. Thus, it was observed that GrS_GWO performed by optimizing the hyperparameters.

In today's era, everyone wants efficient and reliable software. The software developed in earlier years was complex with less efficiency. The complexity of software increases with software size which leads to an increase in the number of defects. However, the number of defects can be minimized considering the removal of all defects from scratch is impossible [136]. ML can be used to develop an efficient model for the detection, and removal of defects in the software. Software development slows down due to the presence of defects in the software. Software testing is important in the software development life cycle phases. The presence of defects in the software affects the delivery of software, development cost, time, resource utilization, and security vulnerability. In upcoming years, software engineering emerge to produce improvised software considering quality, process, time, and cost. The objective of software engineering discipline is to create software that enhances the quality of software in terms of its capacity to be reused, maintained, ported, made robust, and eliminated defects.

In the existing literature, defect prediction is studied considering two scenarios. Firstly, no single prediction technique dominates. Secondly, the interpretation of different prediction models hampers the usage of various datasets, preprocessing techniques, validation schemes, performance measures, and statistical tests. However, such differences in prediction models are overcome by increasing the reusability of existing prediction models. Moreover, it concludes that the researchers, industry experts, and academicians are not

able to select a specified technique. Defect prediction aims at the identification of defects in every phase of software development life cycle and the final testing of software before its deployment at the user's site.

In traditional research, researchers focused on the traditional method of developing a defect prediction model. However, the temporal analysis indicated the availability of training data is very low in the defect prediction domain. Thus, the cross-project domain emerged with the non-availability of training data. In the traditional prediction model, the single dataset is used for training and testing using a few validation methods for specified studies. In the upcoming era, the focus of researchers is to increase the usability of existing models. Thus, existing models are used iteratively for defect prediction.

Defect prediction models can be constructed using many methods, including WPDP and CPDP. There are two distinct types of defect prediction models that vary depending on the projects used for training and testing [150]. In WPDP, the same project (X) can be used for training as well as for testing. However, in CPDP, different projects (Y and Z) are required for training and testing. In CPDP the training and testing dataset have similarities in specified domain and distribution. CPDP can be done in two ways based on the characteristics and company of the project. Cross-company defect prediction discussed in Chapter 1. However, if two different projects of the same company are used for the development of a prediction model named intra-project defect prediction. Furthermore, the CPDP is also categorized based on the types of features used in the training and testing dataset. However, the idea of TL emerged from CPDP. CPDP worked on the basics of TL, four different types of TL exist. Thus, the settings of TL are based on the characteristics and domain distribution of the training and testing dataset. It transfers knowledge learned from one task to a future task. The significance of TL is further used in developing more

efficient prediction models. TL helps in transferring knowledge between different projects having similar features, and data distribution. If the features are completely different in both projects, then developers are required to establish relationship among training and testing dataset considering specified criteria.

This study focused on the identification and removal of defects before the final deployment of the software at the customer site. However, when multiple versions of a project need to be released, then CVDP [151] plays an important role. Furthermore, in the existing studies, authors have experienced that using multiple versions of the same project as a training dataset results in an efficient prediction model. Thus, different versions of the same project are used as training datasets in most of the studies. Researchers have experimented with Just-In-Time (JIT) prediction. However, CPDP using JIT resulted in better performance, while WPDP does not perform well, and the EL idea also worked well considering the historical data of several projects [152].

In this study, authors have analyzed the impact of ANN, RNN, and CNN in combination with various optimization and transfer variants. Three optimization algorithms are used in this study such as Ant Colony Optimization (ACO), Cat Swarm Optimization (CSO), and GWO. Further, the performance of all the prediction models compared with traditional ML algorithms such as RF, KNN, and NB.

The performance was analyzed by evaluating the effectiveness of different methodologies using AEEEM and NASA repository datasets. The authors of the current study have employed ANN to perform CPDP by utilizing three different techniques: GrS, PSO, and EVS. Moreover, transfer techniques are employed to compare these models. The study utilizes datasets from AEEEM and NASA repositories for experimental purpose. In the existing study [153], Transfer ANN (TANN) was employed to create a grid-based model

for CPDP. This study also determined the effectiveness of the grid-based ANN model when combined with optimization methods for the purpose of cross-platform development and performance. In this study, different types of NN are explored for CPDP. TANN, Transfer RNN (TRNN), and Transfer Convolutional NN (TCNN) are used for conducting CPDP experiments. Further, the performance of ANN, RNN, and CNN is first analyzed with traditional algorithm. In the next step, traditional ANN, RNN, and CNN are optimized using SpMO, CSO, and ACO algorithms. To analyze the effectiveness of traditional NN, each NN is compared with the optimized variant. Further, all the NN and their optimized NN variants are compared interchangeably using statistical tests. In this study, three optimized algorithms of all the NN are developed such as ANN_SpMO, ANN_CSO, ANN_ACO, RNN_SpMO, RNN_CSO, RNN_ACO, CNN_SpMO, CNN_CSO, and CNN_ACO. Further, traditional ML algorithms such as Transfer RF (TRF), Transfer K-NN, and Transfer NB (TNB) performance evaluated for CPDP using optimization algorithm. Thus, the variants of TRF, TNB, and TKNN compared with efficient variants of NN.

The following RQs answered in this study:

- RQ1: What is the predictive capability of WPDP, 10-fold_GWO, and WPDP_GSGWO?
- RQ2: What is the predictive capability of TGWO, GSGWO, and RFGWO for InPDP?
- RQ3: What is the predictive capability of TANN, TANN_SpMO, TANN_CSO, TANN_ACO, TRNN, TRNN_SpMO, TRNN_CSO, TRNN_ACO, TCNN, TCNN_SpMO, TCNN_CSO, TCNN_ACO CPDP model for AEEEM and NASA dataset?
- RQ4: What is the predictive capability of TRF, TRF_SpMO, TRF_CSO, TRF_ACO,

TNB, TNB_SpMO, TNB_CSO, TNB_ACO, TKNN, TKNN_SpMO, TKNN_CSO, TKNN_ACO CPDP model for AEEEM and NASA dataset?

The main contribution of this study is as follows:

- Analysis of WPDP with 10-fold cross-validation, and GWO algorithm.
- Analysis of TGWO, GrSGWO, and RFGWO, CPDP, and Intra–Project Defect Prediction (InPDP).
- Efficiency of GWO for defect prediction models.
- Improvising software quality using GWO in the domain of defect prediction model.

The organization of this chapter is as follows: Section 9.2 discussed research methodology. Section 9.3 presented results. Section 9.4 summarized conclusion.

10.2 Research Methodology

This section describes the research methodology followed in this chapter.

The experiment performed using traditional ANN, RNN, and CNN are used with default parameter settings, and ANN, RNN, and CNN with optimization algorithms such as ACO, CSO, and SpMO under varying settings used for the development of prediction models. Furthermore, additional three traditional ML techniques are also used such as RF, NB, and KNN with a transformed version named TRF, TNB, and TKNN.

1. RF: It is a supervised ML algorithm used to solve real-world classification problems. However, in general terms, the forest consists of numerous trees, and the quantity of

trees increases the robustness of the forest. However, the count of trees increases the accuracy of the classifier, and problem-solving ability. Furthermore, RF consists of subsets of trees, and it takes the average of all subset trees accuracy to improve the performance of the model. The concept of EL is used in combining multiple subtrees to solve complex problems by taking an average of all subtrees. The group of multiple models is termed an ensemble. EL is categorized into two categories such as BNG and Boosting. BNG involves generating multiple training subsets from the original training data by randomly selecting samples with replacement. The final output is determined by a majority vote process. Boosting is a technique that combines weak learners to create a powerful learner. This is achieved by sequentially building models, such as Adaboost and XGBoost, in a way that maximizes accuracy. Further, the TRF classifier is developed based on an ensemble of multiple TL approaches, ensuring coverage of similarity among source and target datasets in a broad domain.

2. NB: It is a supervised ML algorithm and is used for classification problems such as text classification. However, NB is different from other discriminative classifiers such as LR, that doesn't learn highly important features to differentiate among output classes. NB is based on Bayes theorem and is also termed as Baye's rule. The concept of conditional probability is required for the NB algorithm. The conditional probability signifies the probability of an event given that some other event occurred at that time. However, NB classifier works under the assumption that predictions in NB are conditionally independent, or unrelated to other features in the model. The classification algorithm performs efficiently with independent probabilities. Further,

the TNB model is proposed by integrating the TL idea with NB to balance the distribution among source and target projects.

3. KNN: KNN is a widely used technique for classification and regression problems. KNN focused on labeling data based on similarity. Thus, it segregates similar data points with similar labels. Furthermore, KNN works by storing the entire training dataset for reference, During the training phase, it computes the distance between data points using euclidean distance. In the next step, based on the computed distance, it identifies data points or KNN. However, for the classification task, KNN assigns a common class label to K neighbors as the predicted label for the input data point. In the regression task, the average or weighted average distance is calculated to find out the KNN to predict the value of the input data point. The performance of KNN is affected by the value of K and the distance measures. However, KNN is used for both classification and regression tasks with simplicity and effectiveness in different scenarios.

4. ANN: ANN algorithm is named after biologically inspired field the brain. It is a computational network developed based on a biologicalNN considered brain architecture. It is similar to human brain architecture as the neurons are interconnected to each other in our brain, ANN also contains neurons that are linked with each other, and these neurons are termed as nodes/units. These units or nodes are connected in a sequence of layers. ANN mainly consists of three layers such as input layer, the hidden layer, and the output layer. The input layer accepts all the inputs in different formats by the programmer. The hidden layer is intermediate between the input, and output layers and performs all the computations to find out hidden patterns, and

features. The output layer provides the actual result after the transformation of input neurons through hidden layer. ANN takes input and computes weights including bias and transfer function. It computes the weighted total, is passed as an input to an activation function to compute output. However, there are various activation functions available based on the type of task. ANN is represented as a weighted directed graph. There exists an association between the input and output neurons as the directed edges with weights. ANN receives input signals from external sources in the form of a pattern and image in the form of a vector. In the next step, each input signal is multiplied by the corresponding weights and these weights represent interconnection between neurons inside the ANN.

The total of weighted inputs lies in the range of 0 to positive infinity and total weighted inputs are passed through various activation functions. The set of transfer functions to compute the desired output is referred to as the activation function. However, there are linear and non-linear sets of activation functions. The commonly used activation functions are binary, linear, and tan hyperbolic sigmoidal activation functions.

5. RNN: RNN is one of the NN types that fed output from the previous step as input to the current step. RNN is useful when previous output is required to make future predictions. In traditional ANN, input, and output are independent of each other. However, RNN has a hidden state that stores information about a sequence. The hidden state is also termed a memory state since it keeps information about the previous input. The same parameters are used in every step as the same task needs to be performed everytime on the hidden layer to provide the same output.

However, RNN reduces the complexity of parameters, unlike other NN. In RNN, the fundamental processing unit is the Recurrent Unit not termed a Recurrent Neuron. This recurrent unit can maintain a hidden state, by allowing the RNN to keep information about sequential dependencies by remembering previous inputs while processing. Long terms dependencies can also be handled using LSTM, and gated recurrent unit. There exist four types of RNN based on the number of inputs and outputs in the network such as one to one, one-to-many, many-to-one, and many-to-many. RNNs have the same input and output layers. Moreover, the difference arises in the way information flows from input to output. The hidden state is computed for every input X_i. Moreover, RNNs suffer from gradient vanishing, and exploding problems including training RNNs is a challenging task. The large sequences can not be processed in the case of Relu or tanh used as an activation function. Furthermore, advanced versions of RNN are developed to deal with the limitations of RNN. The advanced versions of RNN are such as Bidirectional Neural Network (BiNN), and LSTM.

6. CNN: CNN is an extended version of ANN. CNN is used extraction of features from the grid-like matrix dataset. CNN is most commonly used for image recognition and classification tasks. It consists of an input layer, convolutional layer, pooling layer, and fully connected layer. CNN is built upon layers of convolutional operations. A set of learnable filters is provided as input or convolves across the input. The convolution operation included arithmetic operation summation including element-wise multiplication of filter with input. In CNN, a non-linear activation function is applied to produce non-linearity in the network. Furthermore, pooling layers are

used to reduce computational complexity. In the next step, fully connected layers are used which take the flattened output of the last convolutional or pooling layer. The last convolutional layer is further processed through one or more traditionally fully connected layers same as traditional ANN. The CNN training phase uses various gradient descent optimization algorithms and network weights are adjusted during the training phase. The parameters are adjusted to reduce the difference between predicted output, and actual labels in the training data. The difference between predicted, and actual output is reduced with the help of the loss function. Furthermore, overfitting is reduced by integrating regularization, and dropout techniques. Regularization penalizes large weights in the network. Dropout focused on disabling a fraction of neurons during training to a repeated adaptation of features. However, it has been observed that CNN is highly effective for computer vision tasks such as image classification, object detection, human activity recognition, and segmentation as it learns hierarchical patterns, and features from raw pixel data.

10.2.1 Optimization Algorithm

Three optimization algorithms are used such as ACO, CSO, and SpMO to optimize ANN, RNN, and CNN. The three optimization algorithms are also used with traditional ML algorithms to analyze the performance of traditional ML algorithms with TL-based algorithms.

1. ACO: ACO is based on a swarm intelligence algorithm similar to particle swarm optimization. ACO was introduced in the 1990s by Marco Dorigo. The aim behind the swarm intelligence algorithm is to find the optimal solution by analyzing the

behavior of insects, ants, and bees. The name of ACO indicates that it is based on the behavior of ant colonies. Ants communicate through pheromones, which a chemical secreted by the ants on the soil, and ants from the same colony can smell pheromones and follow the same path. However, ants use the shortest path to reach to get the food from the colony. Furthermore, ants secret the pheromones while going for food, and at the same time, other ants follow the same path. The increase in the amount of ants leads to an increase in pheromone secretion, and the secretion of pheromone decreases on other paths. Thus, the shortest path plays an important role in getting the food from the source to the colony. Consider a graph with vertex (V) and edges (E). Further, the source vertex and destination vertex are termed as V_s (ant colony) and V_d (food source). Edges E_1, and E_2 with L_1, and L_2 assigned to both. Pheromones are denoted by R_1 and R. Furthermore, it is evident that if R_1 greater than R_2, then the chances of choosing E_1 are higher. Thus, while returning through the shortest path named E_i, the pheromone value will be updated for the corresponding path. The updation is based on the path length and evaporation rate of pheromone.

2. CSO: CSO is also based on swarm intelligence algorithms. Similar to ACO which is based on ants behavior, CSO is based on the behavior of cats in the real world. According to research, there are a variety of cat species categorized from lions to cheetahs, and from tigers to domestic cats. However, these species live in different environments. Moreover, the behavioral attributes are similar for the cat species. Considering CSO, there are two different states of cat such as seeking mode, and tracking mode. The cats are inactive in seeking mode, which indicates looking

around the surroundings or thinking of moving to another place. The cats are active in tracking mode, which indicates they are changing their current position.

In CSO, every cat has its dimensions in its solution space, with velocity v_i, d that indicates the state of a cat in either seeking mode or tracking mode and provides the fitness value that represents the accommodation of cats to the fitness function. However, to ensure that cats must be in both modes, CSO focused on cats that spend most of their time in an inactive state. Thus, a ratio is defined that indicates how much time either of these modes is taken into account by the cats while performing CSO, a ratio termed a Mixture Ratio (MR). However, the value of MR must be very low for the cats spending most of their time in an inactive state.

3. SpMO: The study of social species foraging behavior has long been important for the creation of optimization algorithms. A global optimization system called Spider Monkey Optimization (SpMO) was developed after studying the fission-fusion social structure of spider monkeys when they forage. Self-organization and division of labor are two core ideas of swarm intelligence that SpMO masterfully illustrates. Recent years have seen a rise in the use of SpMO, a swarm intelligence-based algorithm, for a variety of engineering optimization issues.

10.3 Results and Analysis

The performance of models developed using GSO and ANN are discussed in this section.

10.3.1 Results Specific to RQ1

In this study, defect prediction models are developed for analyzing the efficiency of WPDP. Hence, different versions of same projects are used for training and testing. In order to answer RQ, the experiment is aimed at WPDP. In WPDP, the same project is used for training and testing based on some specified criterion. To address this RQ answer, 12 different projects or versions of NASA dataset are used for experimentation. Three different models are constructed in the study. In the first model, that is WPDP same dataset is used for training and testing in order to avoid lack of data problem.

10.3.2 Results Specific to RQ2

In order to answer this RQ, specified combinations of source and target project is considered for TL. On the basis of number of variables and types of variables in the source and target project specified datasets are used to analyze the predictive capability of TGWO, GSGWO, and RFGWO. The results are presented in Table 9.1. However, out of three models RFGWO outperformed TGWO and GSGWO. It has been observed that RF performed best after optimizing hyperparameters through GWO. In the existing research, it has been observed that grid search performed best for homogeneous CPDP, where both training and testing dataset has similar variables. Furthermore, statistical test validated the results through hypothesis testing. Further, Friedman test concluded that alternate hypothesis is accepted such as there is a significant difference among the performance of TGWO, GSGWO, and RFGWO prediction models. Moreover, it has been observed that null hypothesis is accepted for TGWO and GSGWO. However, alternate hypothesis is accepted between RFGWO and GSGWO pair, and RFWO and TGWO. Thus, RFGWO performed

best out of all the prediction models used for CPDP in this study. Hyperparameter tuning optimizes the parameters to some extent in order to improvise the performance of predictive model. Hence, RFGWO is best for InPDP. Furthermore, graphical representation of AUC values for all the pairs of source and target dataset used for InPDP using TGWO, GSGWO, and RFGWO.

Table 10.1: AUC values for TGWO, GSGWO, and RFGWO for InPDP

Training Dataset → Testing Dataset	**TGWO**	**GSGWO**	**RFGWO**
JM1 → KC1	0.62	0.61	0.64
MC1 → PC5	0.67	0.67	0.69
PC4 → PC3	0.70	0.75	0.75
PC4 → PC1	0.70	0.68	0.77
PC4 → CM1	0.57	0.52	0.57
PC4 → MW1	0.40	0.40	0.45
PC3 → PC1	0.86	0.66	0.86
PC3 → CM1	0.63	0.63	0.65
PC3 → MW1	0.61	0.63	0.64
PC1 → CM1	0.73	0.73	0.75

10.3.3 Results Specific to RQ3

This study provides a comprehensive analysis of the RQs and their corresponding responses. The experiment utilized two projects from AEEEM and eight projects from NASA. The experimental projects incorporate many features, including Halstead metrics and OOM. The selection of projects for HetCPDP has been finalized, and TL is being used in the process. AEEEM projects are utilized to train prediction models, while NASA programs are employed to test the produced prediction models. In addition, a total of 16 couples are created specifically for the purpose of conducting experiments. For this investigation, a

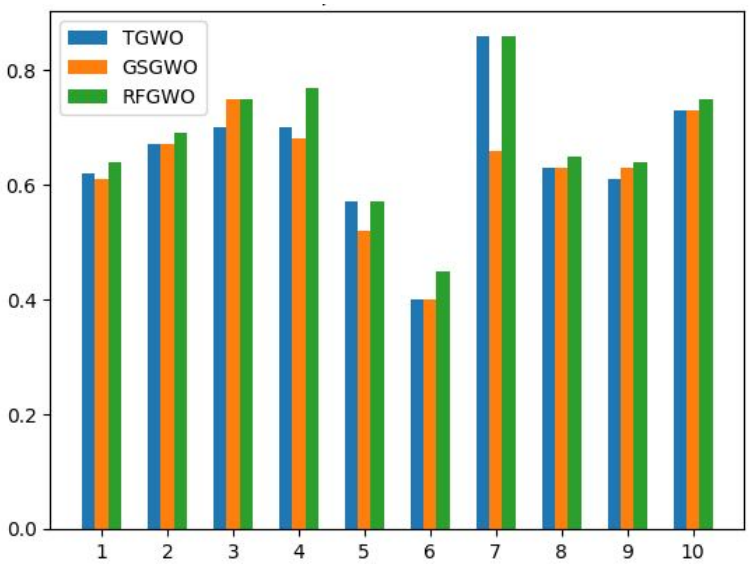

Figure 10.1: Performance comparison of intra-project defect prediction models

total of 112 models were created, taking into account all potential combinations of source and target datasets. The performance of each pair is evaluated based on the chosen classifier using default and grid search parameter configurations. Nevertheless, the evaluation of the prediction models was conducted using the AUC metric. The AUC metric yields more generalized and unbiased outcomes when dealing with imbalanced datasets. This part provides an analysis and addresses the RQs stat Section 1.

Table 10.2: AUC values of the prediction model developed using ANN, RNN, and CNN

Source Dataset → Target Dataset	**ANN SpMO**	**ANN CSO**	**ANN ACO**	**TANN**	**RNN SpMO**	**RNN CSO**	**RNN ACO**	**TRNN**	**CNN SpMO**	**CNN CSO**	**CNN ACO**	**TCNN**
Mylyn → EQ	0.77	0.82	0.78	0.81	0.52	0.67	0.57	0.68	0.91	0.92	0.91	0.91

Mylyn → Lucene	0.94	0.95	0.95	0.95	0.60	0.60	0.68	0.66	0.90	0.92	0.90	0.91
Mylyn → PDE	0.94	0.93	0.93	0.96	0.53	0.65	0.53	0.73	0.90	0.91	0.91	0.91
Mylyn → JDT	0.96	0.95	0.94	0.95	0.44	0.68	0.75	0.54	0.92	0.90	0.92	0.91
PDE → EQ	0.94	0.93	0.92	0.91	0.46	0.68	0.69	0.65	0.88	0.88	0.86	0.88
PDE → Lucene	0.94	0.93	0.94	0.94	0.60	0.62	0.57	0.70	0.85	0.88	0.89	0.87
PDE → JDT	0.93	0.93	0.93	0.92	0.67	0.50	0.60	0.59	0.85	0.88	0.88	0.87
JDT → EQ	0.92	0.91	0.94	0.92	0.70	0.56	0.57	0.52	0.87	0.88	0.88	0.88
JDT → Lucene	0.88	0.91	0.90	0.93	0.64	0.61	0.61	0.68	0.87	0.86	0.89	0.89
Lucene → EQ	0.98	0.98	0.98	0.98	0.55	0.58	0.65	0.64	0.93	0.89	0.91	0.90
CM1 → MW1	0.46	0.32	0.55	0.34	0.52	0.63	0.55	0.58	0.73	0.66	0.66	0.70
PC1 → CM1	0.67	0.32	0.76	0.84	0.68	0.66	0.46	0.70	0.71	0.69	0.74	0.51
PC1 → MW1	0.78	0.72	0.77	0.78	0.60	0.67	0.64	0.66	0.82	0.85	0.74	0.73
PC3 → CM1	0.79	0.34	0.35	0.58	0.67	0.42	0.63	0.57	0.64	0.84	0.50	0.79
PC3 → MW1	0.40	0.44	0.82	0.47	0.48	0.49	0.27	0.26	0.30	0.75	0.66	0.50
PC3 → PC1	0.35	0.82	0.37	0.61	0.51	0.40	0.60	0.69	0.53	0.58	0.53	0.56
PC4 → CM1	0.87	0.90	0.85	0.89	0.62	0.44	0.47	0.58	0.77	0.82	0.87	0.79
PC4 → MW1	0.89	0.75	0.79	0.77	0.62	0.71	0.60	0.35	0.82	0.80	0.76	0.73
PC4 → PC1	0.75	0.79	0.86	0.86	0.64	0.65	0.64	0.62	0.88	0.82	0.72	0.79
PC4 → PC3	0.75	0.73	0.90	0.83	0.61	0.64	0.51	0.43	0.83	0.77	0.88	0.81

To answer this RQ, the model has been developed using 12 different combinations with 3 ML techniques and 3 optimization techniques. Thus in total 12 combinations were formed such as ANN_SpMO, ANN_CSO, ANN_ACO, TANN, RNN_SpMO, RNN_CSO, RNN_ACO, TRNN, CNN_SpMO, SNN_CSO, SNN_ACO, TCNN. Further, the AUC performance measure is used to analyze the performance of these models.

However, the results are validated using a statistical test such as the Friedman test and post hoc analysis completed through the Wilcoxon–signed rank test. However, the Friedman test is performed using N = 20 (combinations of source and target dataset), with 252 models, and 12 different techniques. Through statistical test, it has been analyzed that there is a significant difference among the performance of models developed by 12 different pairs. Friedman's test result provided a computed chi-square value of 82.846 and a p–value of 0.000 at a significance level of 0.005. Thus, the performance of all the 252 models differs significantly.

Furthermore, the performance of each model is analyzed with all other pairs through post hoc analysis test such as the Wilcoxon–signed rank test. Thus, it has been observed that TANN and its optimized models with SpMO, CSO, and ACO outperformed in comparison to RNN and CNN with variants. Thus, it is analyzed that TANN with ACO can be used for CPDP. Also, ANN_ACO and ANN_ANN_SpMO executive time are less in comparison to RNN and CNN-optimized models. TANN, ANN_ACO, ANN_SpMO execution time was 50 sec, 1 min, 48 sec. Hence, in future projects for improvising the software of defect prediction models, time complexity needs to be considered for experimenting with empirical data.

Table 10.3: Friedman test mean rank

Model	**Mean Rank**
ANN_SpMO	8.35
ANN_CSO	7.68
ANN_ACO	9.10
TANN	9.35
RNN_SpMO	3.73
RNN_CSO	3.65
RNN_ACO	3.65
TRNN	3.75
CNN_SpMO	7.28
CNN_CSO	7.63
CNN_ACO	7.13
TCNN	6.73

10.3.4 Results Specific to RQ4

To answer this RQ, the author has considered three different traditional ML algorithms with three different optimization techniques. The three traditional ML algorithms use such as RF, KNN, and NB. To perform CPDP, a total of 20 different pairs were formed with a dataset of different projects. However, it has been observed that KNB and NN do not outperform RF. Further, the results were analyzed using the Friedman test with a DoF is 11, and N = 12. The Friedman test results are demonstrated. The chi-square value obtained is 158.127 and the p-value is 0.000. However, RF takes more computational time and NB takes less time. Thus, it has analyzed that there is a significant difference in the performance of these techniques. To answer this RQ, the default parameters of ANN were changed using a grid search approach. Moreover, it has been concluded that NB performs better for multinomial datasets. Thus, RF produce more better and stable results. Furthermore, to see the final best algorithm technique post–doc analysis test is performed

using the Wilcoxon–signed rank test. Wilcoxon–signed rank test compares two related samples at a time to find out the conclusion among those variables. Wilcoxon–signed rank test showed that TRF and optimized TRF performed better than TNB, TNB_SpMO, TNB_CSO, TNB_ACO, NN, TKNN_SpMO, TKNN_CSO, TKNN_ACO, TKNN.

Table 10.4: AUC values of traditional machine learning techniques such as random forest, naive bayes, K-nearest neighbor

SRDataset → TR-Dataset	**RF SpMO**	**RF CSO**	**RF ACO**	**KNN CSO**	**KNN ACO**	**TKNN**	**TNB SpMO**	**TNB CSO**	**TNB ACO**	**TNB**
Mylyn → EQ	0.80	0.79	0.80	0.70	0.70	0.63	0.69	0.69	0.69	0.60
Mylyn →Lucene	0.80	0.81	0.80	0.70	0.70	0.56	0.69	0.69	0.69	0.58
Mylyn → PDE	0.81	0.82	0.80	0.70	0.70	0.63	0.69	0.69	0.69	0.70
Mylyn → JDT	0.83	0.80	0.81	0.70	0.70	0.63	0.69	0.69	0.69	0.59
PDE → EQ	0.82	0.82	0.83	0.69	0.69	0.59	0.60	0.60	0.60	0.57
PDE → Lucene	0.81	0.82	0.82	0.69	0.69	0.61	0.60	0.60	0.60	0.57
PDE → JDT	0.82	0.80	0.81	0.69	0.69	0.73	0.60	0.60	0.60	0.59
JDT → EQ	0.87	0.88	0.87	0.81	0.81	0.67	0.57	0.57	0.57	0.51
JDT → Lucene	0.85	0.88	0.85	0.81	0.81	0.58	0.57	0.57	0.57	0.59
Lucene → EQ	0.77	0.82	0.79	0.71	0.71	0.66	0.60	0.60	0.60	0.46

CM1→ MW1	0.63	0.63	0.63	0.44	0.44	0.54	0.33	0.33	0.33	0.23
PC1 → CM1	0.91	0.94	0.92	0.70	0.70	0.60	0.14	0.14	0.14	0.41
PC1 → MW1	0.95	0.93	0.94	0.70	0.70	0.55	0.14	0.14	0.14	0.19
PC3 → CM1	0.81	0.79	0.80	0.61	0.61	0.60	0.53	0.53	0.53	0.45
PC3 → MW1	0.81	0.79	0.80	0.61	0.61	0.67	0.53	0.53	0.53	0.51
PC3 → PC1	0.81	0.79	0.80	0.61	0.61	0.72	0.53	0.53	0.53	0.44
PC4 → CM1	0.96	0.95	0.95	0.57	0.57	0.55	0.65	0.65	0.65	0.60
PC4 → MW1	0.96	0.95	0.95	0.56	0.56	0.60	0.65	0.65	0.65	0.37
PC4 → PC1	0.96	0.95	0.95	0.57	0.57	0.61	0.65	0.65	0.65	0.70
PC4 → PC3	0.96	0.95	0.95	0.57	0.57	0.58	0.65	0.65	0.65	0.65

Table 10.5: Friedman test mean rank

Model	**Mean Rank**
TRF	8.45
TRF_SpMO	11.08
TRF_CSO	10.73
TRF_ACO	10.70
TMB_SpMO	6.15
TNB_CSO	6.08
TNB_ACO	6.08
TKNN	4.50
TNB_SpMO	4.40
TNB_CSO	3.65

TNB_ACO	3.50
TNB	2.70

10.4 Discussion

The aim of conducting this study is analyze the predictive capability of GWO for defect prediction models with different variants. In this study, NASA dataset is considered with 12 projects. However, for WPDP 12 projects are utilized for experimentation in order to observe the efficiency of WPDP_GWO, 10fold_GWO, and GS_GWO for WPDP. Hence, it has been concluded that the grid search results in better performance in comparison of other models for WPDP. In case limited amount of data availability, WPDP can be used with GSGWO. The hyperparameters of the traditional algorithms are tuned using GWO. Furthermore, for intra-project defect prediction, different combinations of source and target projects are considered for developing models using different versions of same dataset. Thus, GS_GWO tuned following hyper–parameters such as classifier_learning_rate: 0.01, classifier max_depth, classifier estimators, feature_selection_k. The values of all the hyper–parameters varied on the each project characteristics. Hence, the developed model is useful in the lack of data availability scenario for predictive modeling.

The predictive capability of CPDP models was analyzed using ANN, RNN, and CNN with optimization algorithms. Each optimization algorithm performs differently based on specific characteristics. In this study, three optimization algorithms were used such as ACO, SpMO, and CSO. We have created different combinations considering the NASA and AEEEM dataset for developing CPDP. It is observed that TANN outperformed CNN and RNN. Thus, in future projects, researchers must use TANN for the development

of CPDP by reducing overfitting issues from the WPDP model. The performance of traditional ML techniques was analyzed with TL algorithms and optimization algorithms. The randomization nature of RF while growing the tress plays a pivotal role in the formal performance. However, it has been observed that EL performs best in comparison to other algorithms. TRF and its variants performed better in comparison to TNB, and TKNN. The issue of the non-availability of data for training and testing is solved through the development of prediction models using TL.

Chapter 11

Conclusion

11.1 Summary of the Work

In today's world, the technology is updating everyday. Due to this, traditional software is outdated. The quality of conventional software ensures it's continuous usage considering various factors such as functionality, scalability, testability, maintainability, usability, reliability, and robustness. Thus, to deliver good quality software to the end user, developers must always check and ensure the software quality concerning the software development life cycle. The software requires continuous modifications and adaptability to new technology in the real world. However, many changes occur in the software over time due to various issues such as additional functionality, correction of existing errors, an increment in several features, and change in output. These changes lead to the occurrence of defects. It is not possible to make every change or modification in the software. The change approved by the change control board must only be implemented by the development team, and its

feasibility should be checked by the members before deployment. Moreover, modifications and updates in the software may lead to defects.

The primary aim of this thesis is to develop efficient CPDP using HeCP and HoCP with TL. The efficiency of SDP models increased when TL used for model development with traditional datasets. However, the usability of conventional datasets can be analyzed using TL by developing defect prediction models. Further, this thesis also validated the FS techniques for HeCPDP and HoCPDP. The efficiency of FS techniques analyzed for CPDP using traditional ML classifiers. The filter, wrapper, and swarm methods are compared to get efficient results for CPDP. The applicability of CNN, RNN, and ANN is also analyzed for CPDP using TL with traditional ML classifiers. The hyperparameter optimization is performed for ANN using TL for CPDP. The limitations of WPDP are overcome through the work done in this thesis by removing overfitting issues using CPDP. The thesis also evaluates the effectiveness of CPDP and 10-fold cross-validation. This thesis also validates the open-source datasets through traditional ML classifiers for CPCP. The applicability of CPCP is also analyzed. The number of existing studies using TL in SDP could be much higher. The open-source project datasets aid in the easy replicability and generalizability of the results. However, the descriptive statistics of open-source datasets used in this thesis are mentioned. The data preprocessing techniques, alongwith outlier detection, are described. Performance measures used for performance evaluation of prediction models are described in this thesis. The statistical models used to analyze the experiment results performed in this thesis statistically are described in detail.

A systematic literature review is performed to analyze the work done in the existing studies concerning CPDP and CPDP with HeCP and HoCP. In the review, 39 studies were reviewed specifically in software engineering from 1990 to 2023. Further, criteria are

designed to extract the relevant studies for data synthesis and extraction. The RQs are designed to extract relevant data from the collected studies. The RQs addressed answers related to types of the dataset used, independent variables used, experimental settings, ML techniques used, performance measures used, TL types used, statistical methods used to develop predictive models, threats to validity, merits, and demerits of TL techniques. Further, it has been observed that defect is used in most studies as a quality attribute. In the existing literature, the most commonly explored dataset for TL using CPDP is NASA; other datasets, such as AEEEM and ReLink, have yet to be explored. The open-source dataset has yet to be examined or investigated in the existing studies. The change prediction study is not conducted in the literature using TL. Most of the existing studies used AUC metrics for the analysis of prediction models in the existing studies. IdTL and TdTl are primarily used in existing studies concerning the TL category. The features play an essential role in the development of efficient prediction models. Thus, in the work done in this thesis, the selection of appropriate FS for HeCP and HoCP is explored.

An experiment was conducted to discover the existing literature's research gaps. The CPDP prediction model was developed using TL with feature type transfer. The dataset of different languages for experimentation, such as C, Java, and C++ is used. NASA and PROMISE datasets are used to analyze the predictive ability of ML techniques for WPDP and 10-fold cross-validation. Thus, due to the overfitting issue, WPDP performed well. Further, the performance of CPDP and 10-fold cross-validation is statistically validated. The observed results suggested that the size of the source dataset should be more significant than 23%. Hypothetically, results are validated using the Friedman's and Wilcoxon-signed rank tests. Thus, RF performs best out of all ML algorithms used for experimentation.

Regarding SCP, the CPCP models are developed to validate the results of ML tech-

niques with TL empirically. Thus, an open-source dataset was collected for empirical validation of TL. An empirical study is conducted to validate the performance of ML techniques with 18 open-source datasets. The OOM as an independent variable is extracted from these dataset source codes through Understand tool. SMOTE is used to balance the dataset before model development. The results are statistically validated using the Friedman and Nemenyi test. A metric matching analyzer estimates the matching pair of metrics between different datasets. A metric matching analyzer is also used to establish a relationship between dataset variables in the spatial domain. It is observed that for WPCP, SVM, and HV performed better than the other seven ML techniques used in this study. For HetCPDP, RF and ADB performed better than this study's other seven ML techniques. Thus, the prediction model can identify changes in the subsequent software version and different projects in the early phases of software development. Estimation of changes can help in the prediction of software quality in terms of usability, functionality, maintainability, and reliability.

The performance of different FS techniques was analyzed for HoCPDP models. Filter, wrapper, and swarm search-based methods select relevant features. The prediction models are developed using five ML classifiers to identify defects in the future projects. The results of prediction models developed using ML classifier and FS techniques statistically validated using Friedman and Wilcoxon–signed-rank tests were used for statistical validation. In filter methods, five FS techniques are used: correlation coefficient, Chi-square, gain ratio, relief attribute selection, and information gain based on ranking criteria. However, all filter methods are performed on the same level. The wrapper method worked on the greediness approach. Six swarm search methods are used for CPDP: best-first search, genetic search, greedy stepwise search, harmony search, scatter search, and PSO. Thus, using swarm

search FS techniques resulted in the efficient performance of HoCPDP models based on the grid search approach. The cross-version prediction model helps identify defects in the subsequent versions of an identical project. Using a cross-version approach, the performance and quality of future projects can be improved. The feasibility of cross-version was analyzed in comparison with 10-fold cross-validation. It is observed that practical implementation of different versions in terms of defects is not feasible in large datasets. The dataset of the same project is helpful for training and testing compared to the other versions using TL. The experiment was conducted to analyze the efficiency and evaluate the performance of ANN for CPDP. HetCPDP was developed in this study, considering projects from different repositories. The AEEEM dataset was specified in the source dataset, while NASA datasets were specified in the target dataset. The number of features differs in both, and the CPDP model is designed considering the similarity among features to some extent. The experiment performed in this study showed that traditional ANN performance could have been more significant. Grid-based ANN performed better than conventional ANN, considering the parameters setting using a grid-based approach. Grid search ANN model worked better with more neurons and hidden layers for CPDP. Grid search ANN model outperformed ANN based on number of neurons in the hidden layer, number of hidden layers, and learning rate value. Grid search provides an effective combination of hyperparameters. Moreover, grid search is expensive, especially for large networks with many hyperparameters.

Prediction models were also developed with GWO for defect prediction models and variants. However, the performance of WPDP was analyzed with WPDP_GWO, 10fold_GWO, and GS_GWO for WPDP. Hence, it has been concluded that the grid search results in better performance than other models for WPDP. In case of limited data availabil-

ity, WPDP can be used with GSGWO. The hyperparameters of the traditional algorithms are tuned using GWO. Furthermore, for intra-project defect prediction, different combinations of source and target projects are considered for developing models using different versions of the same dataset. Thus, GS_GWO tuned few hyper–parameters such as classifier_learning_rate: 0.01, classifier max_depth, classifier estimators, feature_selection_k. The values of all the hyperparameters varied on each project's characteristics. Hence, the developed model is useful when a sufficient amount of data is not available for the development of predictive modeling.

11.2 Application of the Work

The work conducted in this thesis would help software practitioners, academicians, developers, researchers, and industry experts in the following ways:

- A systematic approach and methodology can be used to develop effective prediction models in CPDP, HeCPDP, and HoCPDP.
- The effectiveness of ML techniques analyzed for CPDP, CPCP, HeCPDP, HoCPDP, HeCPCP, and HoCPCP, alongwith TL techniques for identification of effective methods to detect defects using historical data.
- Researchers, Software Practitioners, and Academicians can efficiently select the FS valid for CPDP and CPCP.
- The work done in the thesis is useful for predicting defects in the development cycle and allows proactive debugging and quality assurance. Early defect detection in

future projects using TL. It utilizes existing projects within the company or from similar projects in the open-source community.

- CPDP model will be adapted by the legacy system using domain adaptation techniques. It helps in improving the maintenance of legacy systems and resource allocation.
- Large organizations efficiently manage multiple projects using CPDP models to predict defect-prone areas, adapt models as new projects start, and allocate resources based on predicted defect density.
- The open-source project uses community data and TL to improve software reliability and attract more contributors by predicting defects from similar projects and adjusting CPDP models.
- CPDP models can be used for real-time defect predictions, enhancing deployment stability by reducing production defects and improving usage of feature type TL.
- Startups can improve software quality by applying CPDP models trained on public datasets or partner organizations, fine-tuning them for specific projects, and achieving robust defect prediction with minimal initial data.
- A software development team uses defect prediction insights to analyze processes, identify patterns, and refine development practices. This results in reduced defects and more efficient cycles. Challenges include data privacy, feature alignment, model adaptation, and regular evaluation to ensure compliance with regulations, maintain consistency, and enhance prediction accuracy.

11.3 Future Work

The experiment conducted in this thesis can be evaluated and analyzed for a sizeable open-source dataset in the real world. Further, the work performed in this thesis can be analyzed for datasets with different languages, such as cross-language defect prediction. In the future, various types of TL settings will be explored. TL can also be examined for regression problems and unsupervised learning in future prediction models. However, the researchers may empirically investigate the empirical results for the addition of evidence on the basis of which the work's applicability is assured.

Further, the researchers may explore search-based and hybrid techniques must be explored for CPDP and CPCP along with TL algorithms. The researchers may investigate the performance of more optimization algorithms to validate the performance of CPDP and CPCP using TL and traditional ML algorithms with optimization algorithms. The researchers may explore relational knowledge in the future. Thus, the work done in this thesis scan can be replicated for more generalized results and strengthen findings.

Appendices

Descriptive Statistics of Datasets

The descriptive statistics of datasets used in this thesis as given below.

Descriptive Statistics Notepad++ 6.8.9 dataset

Metric	Minimum	Maximum	Mean
LOC	1	8978	118.07
FAN IN	0	3	0.32
CBO	0	91	2.02
NOC	0	37	0.31
WMC	0	427	26.20
RFC	0	256	10.38
DIT	0	4	.48
LCOM	0	100	38.73
NIM	0	252	9.99
NIV	0	192	3.56

Descriptive Statistics Notepad++ 7.3 dataset

Metric	Minimum	Maximum	Mean
LOC	0	9077	112.02
FAN IN	0	3	0.32
CBO	0	92	1.92
NOC	0	38	0.32
WMC	0	427	26.36
RFC	0	256	10.41
DIT	0	4	0.49
LCOM	0	100	39.50
NIM	0	252	10.02
NIV	0	75	3.18

Descriptive Statistics Notepad++ 7.5.4 dataset

Metric	Minimum	Maximum	Mean
LOC	1	8332	111.99
FAN IN	0	3	0.32
CBO	0	96	1.85
NOC	0	42	0.32
WMC	0	427	27.57
RFC	0	256	10.61
DIT	0	4	0.50
LCOM	0	100	40.21
NIM	0	252	10.22
NIV	0	75	3.11

Descriptive Statistics Notepad++ 7.6.2 dataset

Metric	Minimum	Maximum	Mean
LOC	1	8540	153.94
FAN IN	0	3	.31
CBO	0	98	1.78
NOC	0	44	0.31
WMC	0	427	26.10
RFC	0	256	10.25
DIT	0	4	.48
LCOM	0	100	40.45
NIM	0	252	9.84
NIV	0	75	2.99

Descriptive Statistics Notepad++ 7.6.3 dataset

Metric	Minimum	Maximum	Mean
LOC	1	8748	155.56
FAN IN	0	3	0.31
CBO	0	100	1.77
NOC	0	44	0.31
WMC	0	427	26.53
RFC	0	256	10.32
DIT	0	4	0.48
LCOM	0	100	41.01
NIM	0	252	9.91
NIV	0	75	3.03

Descriptive Statistics Notepad++ 7.8.3 dataset

Metric	Minimum	Maximum	Mean
LOC	1	8748	155.56
FAN IN	0	3	0.31
CBO	0	100	1.77
NOC	0	44	.31
WMC	0	427	26.53
RFC	0	256	10.32
DIT	0	4	.48
LCOM	0	100	41.01
NIM	0	252	9.91
NIV	0	75	3.03

Descriptive Statistics CodeBlocks 10.05 dataset

Metric	Minimum	Maximum	Mean
LOC	1.00	24970.00	330.1657
FAN IN	0.00	2.00	0.3619
CBO	0.00	43.00	2.6950
NOC	0.00	43.00	0.3512
WMC	0.00	536.00	30.6603
RFC	0.00	473.00	13.8021
DIT	0.00	4.00	0.5540

Descriptive Statistics CodeBlocks 13.12 dataset

Metric	Minimum	Maximum	Mean
LOC	1.00	13722.00	.4854
FAN IN	0	2	0.44
CBO	0	51	3.01
NOC	0	90	0.43
WMC	0	637	31.09
RFC	0	627	14.27
DIT	0	4	0.65

Descriptive Statistics CodeBlocks 20.03 dataset

Metric	Minimum	Maximum	Mean
LOC	1.00	13722.00	140.4854
FAN IN	0.00	3.00	0.3985
CBO	0.00	80.00	2.8027
NOC	0.00	98.00	0.3914
WMC	0.00	777.00	34.1365
RFC	0.00	727.00	14.3090
DIT	0.00	4.00	0.6006

Descriptive Statistics CodeLite-2.9.0.4684 dataset

Metric	Minimum	Maximum	Mean
LOC	1	7003	123.33
FAN IN	0	2	0.47
CBO	0	45	2.19
NOC	0	181	0.47
WMC	0	727	16.88
RFC	0	514	10.97
DIT	0	3	0.48
LCOM	0	100	34.61
NIM	0	514	10.80
NIV	0	97	2.09

Descriptive Statistics CodeLite-3.5.5375 dataset

Metric	Minimum	Maximum	Mean
LOC	1	7246	122.33
FAN IN	0	2	0.47
CBO	0	49	2.20
NOC	0	181	0.47
WMC	0	742	16.74
RFC	0	516	10.95
DIT	0	3	0.47
LCOM	0	100	35.75
NIM	0	516	10.77
NIV	0	109	2.15

Descriptive Statistics CodeLite-5.0.6213 dataset

Metric	Minimum	Maximum	Mean
LOC	1	7646	118.24
FAN IN	0	2	0.45
CBO	0	52	2.03
NOC	0	181	0.45
WMC	0	762	19.86
RFC	0	518	10.70
DIT	0	5	0.50
LCOM	0	100	41.73
NIM	0	518	10.53
NIV	0	155	2.26

Descriptive Statistics CodeLite-5.3 dataset

Metric	Minimum	Maximum	Mean
LOC	1 7273	117.19	
FAN IN	0	2	0.45
CBO	0	50	2.05
NOC	0	181	0.45
WMC	0	252	18.19
RFC	0 241	10.24	
DIT	0	5	0.50
LCOM	0	100	35.07
NIM	0	239	10.10
NIV	0	179	2.11

Descriptive Statistics CodeLite-6.0.1 dataset

Metric	Minimum	Maximum	Mean
LOC	1	7273	114.84
FAN IN	0	2	0.43
CBO	0	50	2.07
NOC	0	181	0.43
WMC	0	259	17.78
RFC	0	245	10.14
DIT	0	5	0.48
LCOM	0	100	35.06
NIM	0	243	10.00
NIV	0	179	2.29

Descriptive Statistics CodeLite-11.0 dataset

Metric	Minimum	Maximum	Mean
LOC	1	7940	134.55
FAIN IN	0	2	0.43
CBO	0	63	2.71
NOC	0	181	0.43
WMC	0	323	20.64
RFC	0	296	12.67
DIT	0	5	0.47
LCOM	0	100	38.14
NIM	0	294	12.48
NIV	0	179	2.20

Descriptive Statistics CodeLite-12.0 dataset

Metric	Minimum	Maximum	Mean
LOC	1	12107	141.39
FAN IN	0	2	0.43
CBO	0	63	2.73
NOC	0	181	0.43
WMC	0	326	20.92
RFC	0	297	12.83
DIT	0	5	0.47
LCOM	0	100	37.65
NIM	0	295	12.66
NIV	0	179	2.27

Descriptive Statistics CodeLite-13.0 dataset

Metric	Minimum	Maximum	Mean
LOC	1	12107	140.67
FAN IN	0	2	0.43
CBO	0	60	2.71
NOC	0	181	0.43
WMC	0	330	20.87
RFC	0	300	12.86
DIT	0	5	0.48
LCOM	0	100	37.56
NIM	0	298	12.68
NIV	0	179	2.25

Descriptive Statistics CodeLite-14.0 dataset

Metric	Minimum	Maximum	Mean
LOC	1	12107	121.88
FAN IN	0	3	0.44
CBO	0	133	2.61
NOC	0	181	0.43
WMC	0 3	41	25.96
RFC	0	312	11.75
DIT	0	5	0.53
LCOM	0	100	37.60
NIM	0	310	11.56
NIV	0	179	1.92

Bibliography

[1] K. Aggarwal, *Software engineering*. New Age International, 2005.

[2] Y. Singh and R. Malhotra, *Object-oriented software engineering*. PHI Learning Pvt. Ltd., 2012.

[3] R. Malhotra, "An empirical framework for defect prediction using machine learning techniques with android software," *Applied Soft Computing*, vol. 49, pp. 1034–1050, 2016.

[4] L. Kumar, R. K. Behera, S. Rath, and A. Sureka, "Transfer learning for cross-project change-proneness prediction in object-oriented software systems: A feasibility analysis," *ACM SIGSOFT Software Engineering Notes*, vol. 42, no. 3, pp. 1–11, 2017.

[5] M. CHENG, G.-q. WU, and M.-t. YUAN, "Transfer learning for software defect prediction," *ACTA ELECTONICA SINICA*, vol. 44, no. 1, p. 115, 2016.

[6] S. Tong, Q. He, Y. Chen, Y. Yang, and B. Shen, "Heterogeneous cross-company effort estimation through transfer learning," in *2016 23rd Asia-Pacific Software Engineering Conference (APSEC)*. IEEE, 2016, pp. 169–176.

[7] T. Sharma, A. Jatain, S. Bhaskar, and K. Pabreja, "Original research article an empirical analysis of feature selection techniques for software defect prediction," *Journal of Autonomous Intelligence*, vol. 7, no. 3, 2024.

[8] M. Ali, T. Mazhar, Y. Arif, S. Al-Otaibi, Y. Y. Ghadi, T. Shahzad, M. A. Khan, and H. Hamam, "Software defect prediction using an intelligent ensemble-based model," *IEEE Access*, 2024.

[9] A. Nikanjam *et al.*, "Interpretation conclusion stability of software defect prediction over time," 2024.

[10] M. Ali, "Optimizing software defect prediction: A genetic algorithm based comparative analysis."

[11] M. Ali, T. Mazhar, A. Al-Rasheed, T. Shahzad, Y. Y. Ghadi, and M. A. Khan, "Enhancing software defect prediction: a framework with improved feature selection and ensemble machine learning," *PeerJ Computer Science*, vol. 10, p. e1860, 2024.

[12] S. Haldar and L. F. Capretz, "Interpretable software defect prediction from project effort and static code metrics," *Computers*, vol. 13, no. 2, p. 52, 2024.

[13] C. Shyamala, S. Mohana, M. Ambika, and K. Gomathi, "Hybrid deep architecture for software defect prediction with improved feature set," *Multimedia Tools and Applications*, pp. 1–36, 2024.

[14] J. Chen, J. Xu, S. Cai, X. Wang, H. Chen, and Z. Li, "Software defect prediction approach based on a diversity ensemble combined with neural network," *IEEE Transactions on Reliability*, 2024.

[15] D. Rai and J. A. Prashant, "Insights of effectivity analysis of learning-based approaches towards software defect prediction," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 14, no. 2, pp. 1916–1927, 2024.

[16] W. Wu, S. Wang, B. Liu, Y. Shao, and W. Xie, "A novel software defect prediction approach via weighted classification based on association rule mining," *Engineering Applications of Artificial Intelligence*, vol. 129, p. 107622, 2024.

[17] R. Fatima, F. Zeshan, A. Ahmad, M. Hamid, I. Filali, A. A. Alhussan, and H. A. Abdallah, "Requirement change prediction model for small software systems," *Computers*, vol. 12, no. 8, p. 164, 2023.

[18] R. Malhotra and A. J. Bansal, "Software change prediction: A literature review," *International Journal of Computer Applications in Technology*, vol. 54, no. 4, pp. 240–256, 2016.

[19] R. Malhotra and M. Khanna, "Software change prediction: A systematic review and future guidelines," *e-Informatica Software Engineering Journal*, vol. 13, no. 1, 2019.

[20] H. Kagdi and J. I. Maletic, "Software-change prediction: Estimated+ actual," in *2006 Second International IEEE Workshop on Software Evolvability (SE'06)*. IEEE, 2006, pp. 38–43.

[21] Y. Ma, G. Luo, X. Zeng, and A. Chen, "Transfer learning for cross-company software defect prediction," *Information and Software Technology*, vol. 54, no. 3, pp. 248–256, 2012.

[22] B. Turhan, T. Menzies, A. B. Bener, and J. Di Stefano, "On the relative value of cross-company and within-company data for defect prediction," *Empirical Software Engineering*, vol. 14, pp. 540–578, 2009.

[23] F. Peters, T. Menzies, and A. Marcus, "Better cross company defect prediction," in *2013 10th Working Conference on Mining Software Repositories (MSR)*. IEEE, 2013, pp. 409–418.

[24] X. Jing, F. Wu, X. Dong, F. Qi, and B. Xu, "Heterogeneous cross-company defect prediction by unified metric representation and cca-based transfer learning," in *Proceedings of the 2015 10th joint meeting on foundations of software engineering*, 2015, pp. 496–507.

[25] F. Peters, T. Menzies, L. Gong, and H. Zhang, "Balancing privacy and utility in cross-company defect prediction," *IEEE Transactions on Software Engineering*, vol. 39, no. 8, pp. 1054–1068, 2013.

[26] J. Wu, Y. Wu, N. Niu, and M. Zhou, "MHCPDP: Multi-source heterogeneous cross-project defect prediction via multi-source transfer learning and autoencoder," *Software Quality Journal*, vol. 29, no. 2, pp. 405–430, 2021.

[27] D. Ryu and J. Baik, "Effective multi-objective naïve bayes learning for cross-project defect prediction," *Applied Soft Computing*, vol. 49, pp. 1062–1077, 2016.

[28] C. Ni, W.-S. Liu, X. Chen, Q. Gu, D.-X. Chen, and Q.-G. Huang, "A cluster based feature selection method for cross-project software defect prediction," *Journal of Computer Science and Technology*, vol. 32, pp. 1090–1107, 2017.

[29] T. Zimmermann, N. Nagappan, H. Gall, E. Giger, and B. Murphy, "Cross-project defect prediction: a large scale experiment on data vs. domain vs. process," in *Proceedings of the 7th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering*, 2009, pp. 91–100.

[30] J. Nam and S. Kim, "Heterogeneous defect prediction," in *Proceedings of the 2015 10th joint meeting on foundations of software engineering*, 2015, pp. 508–519.

[31] H. Qing, L. Biwen, S. Beijun, and Y. Xia, "Cross-project software defect prediction using feature-based transfer learning," in *Proceedings of the 7th Asia-Pacific Symposium on Internetware*, 2015, pp. 74–82.

[32] W. Liu, Y. Zhu, X. Chen, Q. Gu, X. Wang, and S. Gu, "S^2LMMD: Cross-project software defect prediction via statement semantic learning and maximum mean discrepancy," in *2021 28th Asia-Pacific Software Engineering Conference (APSEC)*. IEEE, 2021, pp. 369–379.

[33] Q. Zou, L. Lu, S. Qiu, X. Gu, and Z. Cai, "Correlation feature and instance weights transfer learning for cross project software defect prediction," *IET Software*, vol. 15, no. 1, pp. 55–74, 2021.

[34] C. Liu, D. Yang, X. Xia, M. Yan, and X. Zhang, "A two-phase transfer learning model for cross-project defect prediction," *Information and Software Technology*, vol. 107, pp. 125–136, 2019.

[35] S. Tang, S. Huang, C. Zheng, E. Liu, C. Zong, and Y. Ding, "A novel cross-project

software defect prediction algorithm based on transfer learning," *Tsinghua Science and Technology*, vol. 27, no. 1, pp. 41–57, 2021.

[36] J. Deng, L. Lu, S. Qiu, and Y. Ou, "A suitable ast node granularity and multi-kernel transfer convolutional neural network for cross-project defect prediction," *IEEE Access*, vol. 8, pp. 66 647–66 661, 2020.

[37] J. Chen, K. Hu, Y. Yu, Z. Chen, Q. Xuan, Y. Liu, and V. Filkov, "Software visualization and deep transfer learning for effective software defect prediction," in *Proceedings of the ACM/IEEE 42nd international conference on software engineering*, 2020, pp. 578–589.

[38] L. Sheng, L. Lu, and J. Lin, "An adversarial discriminative convolutional neural network for cross-project defect prediction," *IEEE Access*, vol. 8, pp. 55 241–55 253, 2020.

[39] J. Chen, X. Wang, S. Cai, J. Xu, J. Chen, and H. Chen, "A software defect prediction method with metric compensation based on feature selection and transfer learning," *Frontiers of Information Technology & Electronic Engineering*, vol. 23, no. 5, pp. 715–731, 2022.

[40] J. Bai, J. Jia, and L. F. Capretz, "A three-stage transfer learning framework for multi-source cross-project software defect prediction," *Information and Software Technology*, vol. 150, p. 106985, 2022.

[41] J. Nam, S. J. Pan, and S. Kim, "Transfer defect learning," in *2013 35th international conference on software engineering (ICSE)*. IEEE, 2013, pp. 382–391.

[42] D. Ryu, J.-I. Jang, and J. Baik, "A transfer cost-sensitive boosting approach for cross-project defect prediction," *Software Quality Journal*, vol. 25, pp. 235–272, 2017.

[43] A. Arnold, R. Nallapati, and W. W. Cohen, "A comparative study of methods for transductive transfer learning," in *Seventh IEEE international conference on data mining workshops (ICDMW 2007)*. IEEE, 2007, pp. 77–82.

[44] A. Corazza, S. Di Martino, F. Ferrucci, C. Gravino, and F. Sarro, "From function points to cosmic-a transfer learning approach for effort estimation," in *Product-Focused Software Process Improvement: 16th International Conference, PROFES 2015, Bolzano, Italy, December 2-4, 2015, Proceedings 16*. Springer, 2015, pp. 251–267.

[45] E. Kocaguneli, T. Menzies, and E. Mendes, "Transfer learning in effort estimation," *Empirical Software Engineering*, vol. 20, pp. 813–843, 2015.

[46] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2009.

[47] X. Liu, G. Wang, Z. Cai, and H. Zhang, "Bagging based ensemble transfer learning," *Journal of Ambient Intelligence and Humanized Computing*, vol. 7, pp. 29–36, 2016.

[48] V. Suhag, S. K. Dubey, and B. K. Sharma, "Transfer learning based low shot classifier for software defect prediction." *Journal of Information Systems Engineering & Business Intelligence*, vol. 9, no. 2, 2023.

[49] B. Kitchenham, O. P. Brereton, D. Budgen, M. Turner, J. Bailey, and S. Linkman, "Systematic literature reviews in software engineering–a systematic literature review," *Information and software technology*, vol. 51, no. 1, pp. 7–15, 2009.

[50] B. A. Kitchenham, P. Rialette, B. David, O. P. Brereton, M. Turner, M. Niazi, and S. Linkman, "Systematic literature reviews in software engineering a tertiary study," *Information and Software Technology*, vol. 52, no. 8, pp. pp. 792–805, 2010.

[51] C. Ni, X. Xia, D. Lo, X. Chen, and Q. Gu, "Revisiting supervised and unsupervised methods for effort-aware cross-project defect prediction," *IEEE Transactions on Software Engineering*, vol. 48, no. 3, pp. 786–802, 2020.

[52] X.-Y. Jing, H. Chen, and B. Xu, "Cross-project defect prediction," in *Intelligent Software Defect Prediction*. Springer, 2024, pp. 35–63.

[53] Y. Z. Bala, P. A. Samat, K. Y. Sharif, and N. Manshor, "Cross-project software defect prediction through multiple learning," *Bulletin of Electrical Engineering and Informatics*, vol. 13, no. 3, pp. 2027–2035, 2024.

[54] S. Hosseini, B. Turhan, and D. Gunarathna, "A systematic literature review and meta-analysis on cross project defect prediction," *IEEE Transactions on Software Engineering*, vol. 45, no. 2, pp. 111–147, 2017.

[55] S. Pal and A. Sillitti, "Cross-project defect prediction: a literature review," *IEEE Access*, vol. 10, pp. 118 697–118 717, 2022.

[56] A. Panichella, R. Oliveto, and A. De Lucia, "Cross-project defect prediction models: L'union fait la force," in *2014 Software Evolution Week-IEEE Conference on*

Software Maintenance, Reengineering, and Reverse Engineering (CSMR-WCRE). IEEE, 2014, pp. 164–173.

[57] S. R. Chidamber and C. F. Kemerer, "A metrics suite for object oriented design," *IEEE Transactions on software engineering*, vol. 20, no. 6, pp. 476–493, 1994.

[58] L. Kumar, S. Lal, and L. B. Murthy, "Estimation of maintainability parameters for object-oriented software using hybrid neural network and class level metrics," *International Journal of System Assurance Engineering and Management*, vol. 10, no. 5, pp. 1234–1264, 2019.

[59] K. El Emam, W. Melo, and J. C. Machado, "The prediction of faulty classes using object-oriented design metrics," *Journal of systems and software*, vol. 56, no. 1, pp. 63–75, 2001.

[60] T. Gyimothy, R. Ferenc, and I. Siket, "Empirical validation of object-oriented metrics on open source software for fault prediction," *IEEE Transactions on Software engineering*, vol. 31, no. 10, pp. 897–910, 2005.

[61] J. Bansiya and C. Davis, "A hierarchical model for object-oriented design quality assessment," *IEEE Transactions on Software Engineering*, vol. 28, no. 1, pp. pp. 4-17, 2002.

[62] R. C. Martin, *Agile software development: principles, patterns, and practices*. Prentice Hall, 2002.

[63] B. Henderson-Sellers, *Object-oriented metrics: measures of complexity*. Prentice-Hall, Inc., 1995.

[64] J. Al Dallal, "Object-oriented class maintainability prediction using internal quality attributes," *Information and Software Technology*, vol. 55, no. 11, pp. 2028–2048, 2013.

[65] L.-j. Wang, X.-x. Hu, Z.-y. Ning, and W.-h. Ke, "Predicting object-oriented software maintainability using projection pursuit regression," in *2009 First International Conference on Information Science and Engineering*. IEEE, 2009, pp. 3827–3830.

[66] V. C. Koten and A. R. Gray, "An application of bayesian network for predicting object-oriented software maintainability," *Information and Software Technology*, vol. 48, no. 1, pp. pp. 59-67, 2006.

[67] Y. Singh, A. Kaur, and R. Malhotra, "Empirical validation of object-oriented metrics for predicting fault proneness models," *Software quality journal*, vol. 18, no. 1, p. 3, 2010.

[68] H. M. Olague, L. H. Etzkorn, S. Gholston, and S. Quattlebaum, "Empirical validation of three software metrics suites to predict fault-proneness of object-oriented classes developed using highly iterative or agile software development processes," *IEEE Transactions on software Engineering*, vol. 33, no. 6, pp. 402–419, 2007.

[69] S. Kpodjedo, F. Ricca, P. Galinier, Y.-G. Guéhéneuc, and G. Antoniol, "Design evolution metrics for defect prediction in object oriented systems," *Empirical Software Engineering*, vol. 16, no. 1, pp. 141–175, 2011.

[70] M. O. Elish and M. Al-Rahman Al-Khiaty, "A suite of metrics for quantifying historical changes to predict future change-prone classes in object-oriented software," *Journal of Software: Evolution and Process*, vol. 25, no. 5, pp. 407–437, 2013.

[71] H. Zhang, "An investigation of the relationships between lines of code and defects," in *2009 IEEE international conference on software maintenance*. IEEE, 2009, pp. 274–283.

[72] S. Lessmann, B. Baesens, C. Mues, and S. Pietsch, "Benchmarking classification models for software defect prediction: A proposed framework and novel findings," *IEEE transactions on software engineering*, vol. 34, no. 4, pp. 485–496, 2008.

[73] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: an update," *ACM SIGKDD explorations newsletter*, vol. 11, no. 1, pp. 10–18, 2009.

[74] X.-Y. Jing, F. Wu, X. Dong, and B. Xu, "An improved sda based defect prediction framework for both within-project and cross-project class-imbalance problems," *IEEE Transactions on Software Engineering*, vol. 43, no. 4, pp. 321–339, 2016.

[75] D. W. Zimmerman and B. D. Zumbo, "Relative power of the wilcoxon test, the friedman test, and repeated-measures anova on ranks," *The Journal of Experimental Education*, vol. 62, no. 1, pp. 75–86, 1993.

[76] M. R. Sheldon, M. J. Fillyaw, and W. D. Thompson, "The use and interpretation of the friedman test in the analysis of ordinal-scale data in repeated measures designs," *Physiotherapy Research International*, vol. 1, no. 4, pp. 221–228, 1996.

[77] L. Y. Pratt, "Discriminability-based transfer between neural networks," *Advances in neural information processing systems*, vol. 5, 1992.

[78] K. D. Feuz and D. J. Cook, "Transfer learning across feature-rich heterogeneous feature spaces via feature-space remapping (fsr)," *ACM transactions on intelligent systems and technology (TIST)*, vol. 6, no. 1, pp. 1–27, 2015.

[79] C. B. Do and A. Y. Ng, "Transfer learning for text classification," *Advances in neural information processing systems*, vol. 18, 2005.

[80] X. Liu, Z. Liu, G. Wang, Z. Cai, and H. Zhang, "Ensemble transfer learning algorithm," *Ieee Access*, vol. 6, pp. 2389–2396, 2017.

[81] R. Raina, A. Y. Ng, and D. Koller, "Constructing informative priors using transfer learning," in *Proceedings of the 23rd international conference on Machine learning*, 2006, pp. 713–720.

[82] Q. Yu, S. Jiang, and Y. Zhang, "A feature matching and transfer approach for cross-company defect prediction," *Journal of Systems and Software*, vol. 132, pp. 366–378, 2017.

[83] L. Mihalkova, T. Huynh, and R. J. Mooney, "Mapping and revising markov logic networks for transfer learning," in *Aaai*, vol. 7, 2007, pp. 608–614.

[84] K. Weiss and T. Khoshgoftaar, "Evaluation of transfer learning algorithms using different base learners," in *2017 IEEE 29th International Conference on Tools with Artificial Intelligence (ICTAI)*. IEEE, 2017, pp. 187–196.

[85] S. J. Pan, J. T. Kwok, Q. Yang *et al.*, "Transfer learning via dimensionality reduction." in *AAAI*, vol. 8, 2008, pp. 677–682.

[86] F. L. F. Pereira, F. D. dos Santos Lima, L. G. de Moura Leite, J. P. P. Gomes, and J. de Castro Machado, "Transfer learning for bayesian networks with application on hard disk drives failure prediction," in *2017 Brazilian Conference on Intelligent Systems (BRACIS)*. IEEE, 2017, pp. 228–233.

[87] W. Dai, O. Jin, G.-R. Xue, Q. Yang, and Y. Yu, "Eigentransfer: a unified framework for transfer learning," in *Proceedings of the 26th Annual International Conference on Machine Learning*, 2009, pp. 193–200.

[88] R. Gargees, J. Keller, and M. Popescu, "Early illness recognition in older adults using transfer learning," in *2017 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*. IEEE, 2017, pp. 1012–1016.

[89] B. Li, Q. Yang, and X. Xue, "Transfer learning for collaborative filtering via a rating-matrix generative model," in *Proceedings of the 26th annual international conference on machine learning*, 2009, pp. 617–624.

[90] S. Yan, B. Shen, W. Mo, and N. Li, "Transfer learning for cross-platform software crowdsourcing recommendation," in *2017 24th Asia-Pacific Software Engineering Conference (APSEC)*. IEEE, 2017, pp. 269–278.

[91] J. Wan, X. Wang, Y. Yin, and R. Zhou, "Transfer learning in collaborative filtering for sparsity reduction via feature tags learning model," *Computer Science and Technology*, 2015.

[92] Y. Chen and X. Ding, "Research on cross-project software defect prediction based on transfer learning," in *AIP Conference Proceedings*, vol. 1955, no. 1. AIP Publishing, 2018.

[93] R. Krishna and T. Menzies, "Bellwethers: A baseline method for transfer learning," *IEEE Transactions on Software Engineering*, vol. 45, no. 11, pp. 1081–1105, 2018.

[94] M. Long, J. Wang, G. Ding, D. Shen, and Q. Yang, "Transfer learning with graph co-regularization," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 7, pp. 1805–1818, 2013.

[95] A. A. Deshmukh and E. Laftchiev, "Semi-supervised transfer learning using marginal predictors," in *2018 IEEE Data Science Workshop (DSW)*. IEEE, 2018, pp. 160–164.

[96] J. Zhou, S. Pan, I. Tsang, and Y. Yan, "Hybrid heterogeneous transfer learning through deep learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 28, no. 1, 2014.

[97] W. Ying, Y. Zhang, J. Huang, and Q. Yang, "Transfer learning via learning to transfer," in *International conference on machine learning*. PMLR, 2018, pp. 5085–5094.

[98] Y. Cui, Y. Song, C. Sun, A. How-ard, and S. Belongie, "Large scale fine-grained categorization and do-main-specific transfer learning," in *Proceedings of the IEEE conference on com-puter vision and pattern recognition*, 2018, pp. 4109–4118.

[99] K. Dillon Feuz and D. J. Cook, "Heterogeneous transfer learning for activity recognition using heuristic search techniques," *International journal of pervasive computing and communications*, vol. 10, no. 4, pp. 393–418, 2014.

[100] J. Chen, Y. Yang, K. Hu, Q. Xuan, Y. Liu, and C. Yang, "Multiview transfer learning for software defect prediction," *IEEE Access*, vol. 7, pp. 8901–8916, 2019.

[101] Q. Cao, Q. Sun, Q. Cao, and H. Tan, "Software defect prediction via transfer learning based neural network," in *2015 First international conference on reliability systems engineering (ICRSE)*. IEEE, 2015, pp. 1–10.

[102] K. Li, Z. Xiang, T. Chen, S. Wang, and K. C. Tan, "Understanding the automated parameter optimization on transfer learning for cpdp: An empirical study," *arXiv preprint arXiv:2002.03148*, 2020.

[103] R. Krishna, T. Menzies, and W. Fu, "Too much automation? the bellwether effect and its implications for transfer learning," in *Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering*, 2016, pp. 122–131.

[104] Z. Xu, S. Pang, T. Zhang, X.-P. Luo, J. Liu, Y.-T. Tang, X. Yu, and L. Xue, "Cross project defect prediction via balanced distribution adaptation based transfer learning," *Journal of Computer Science and Technology*, vol. 34, pp. 1039–1062, 2019.

[105] K. R. Weiss and T. M. Khoshgoftaar, "An investigation of transfer learning and traditional machine learning algorithms," in *2016 IEEE 28th International Conference on Tools with Artificial Intelligence (ICTAI)*. IEEE, 2016, pp. 283–290.

[106] X. Du, Z. Zhou, B. Yin, and G. Xiao, "Cross-project bug type prediction based on transfer learning," *Software Quality Journal*, vol. 28, pp. 39–57, 2020.

[107] K.-M. Su, W. D. Hairston, and K. A. Rob-bins, "Adaptive thresholding and reweighting to improve domain transfer learning for unbalanced data with appli-cations to

eeg imbalance," in *2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE, 2016, pp. 320–325.

[108] K. R. Weiss and T. M. Khoshgoftaar, "Detection of phishing webpages using heterogeneous transfer learning," in *2017 IEEE 3rd International Conference on Collaboration and Internet Computing (CIC)*. IEEE, 2017, pp. 190–197.

[109] Y. Xu, S. J. Pan, H. Xiong, Q. Wu, R. Luo, H. Min, and H. Song, "A unified framework for metric transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 6, pp. 1158–1171, 2017.

[110] R. Malhotra and A. J. Bansal, "Cross project change prediction using open source projects," in *2014 International conference on advances in computing, communications and informatics (ICACCI)*. IEEE, 2014, pp. 201–207.

[111] Z. He, F. Shu, Y. Yang, M. Li, and Q. Wang, "An investigation on the feasibility of cross-project defect prediction," *Automated Software Engineering*, vol. 19, pp. 167–199, 2012.

[112] R. Malhotra and S. Meena, "Defect prediction model using transfer learning," *Soft Computing*, vol. 26, no. 10, pp. 4713–4726, 2022.

[113] R. Kohavi *et al.*, "A study of cross-validation and bootstrap for accuracy estimation and model selection," in *Ijcai*, vol. 14, no. 2. Montreal, Canada, 1995, pp. 1137–1145.

[114] G. Forman and M. Scholz, "Apples-to-apples in cross-validation studies: pitfalls in

classifier performance measurement," *Acm Sigkdd Explorations Newsletter*, vol. 12, no. 1, pp. 49–57, 2010.

[115] M. Bekkar, H. K. Djemaa, and T. A. Alitouche, "Evaluation measures for models assessment over imbalanced data sets," *J Inf Eng Appl*, vol. 3, no. 10, 2013.

[116] A. Benavoli, G. Corani, and F. Mangili, "Should we really use post-hoc tests based on mean-ranks?" *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 152–161, 2016.

[117] S. Taheri and G. Hesamian, "A generalization of the wilcoxon signed-rank test and its applications," *Statistical Papers*, vol. 54, pp. 457–470, 2013.

[118] R. Malhotra and M. Khanna, "An empirical study to evaluate the relationship of object-oriented metrics and change proneness." *Int. Arab J. Inf. Technol.*, vol. 15, no. 6, pp. 1016–1023, 2018.

[119] J. Hemanth, M. Bathia, and O. Geman, *Data Visualization and Knowledge Engineering*. Springer, 2021, vol. 209.

[120] O. Day and T. M. Khoshgoftaar, "A survey on heterogeneous transfer learning," *Journal of Big Data*, vol. 4, pp. 1–42, 2017.

[121] S. Liu, X. Chen, W. Liu, J. Chen, Q. Gu, and D. Chen, "Fecar: A feature selection framework for software defect prediction," in *2014 IEEE 38th annual computer software and applications conference*. IEEE, 2014, pp. 426–435.

[122] H. Mou and J. Yu, "Transfer learning with dwt based clustering for blood pressure estimation of multiple patients," *Journal of Computational Science*, vol. 64, p. 101865, 2022.

[123] B. Morawska, P. Lipinski, K. Lichy, and K. Adamkiewicz, "Transfer learning-based uwb indoor localization using mht-mdc and clusterization-based sparse fingerprinting," *Journal of Computational Science*, vol. 61, p. 101654, 2022.

[124] C. Buizza, C. Q. Casas, P. Nadler, J. Mack, S. Marrone, Z. Titus, C. Le Cornec, E. Heylen, T. Dur, L. B. Ruiz *et al.*, "Data learning: Integrating data assimilation and machine learning," *Journal of Computational Science*, vol. 58, p. 101525, 2022.

[125] R. Malhotra and S. Meena, "Empirical validation of machine learning techniques for heterogeneous cross-project change prediction and within-project change prediction," *Journal of Computational Science*, p. 102230, 2024.

[126] N. U. Maulidevi, K. Surendro *et al.*, "Smote-lof for noise identification in imbalanced data classification," *Journal of King Saud University-Computer and Information Sciences*, vol. 34, no. 6, pp. 3413–3423, 2022.

[127] X. Chen, Z. Yuan, Z. Cui, D. Zhang, and X. Ju, "Empirical studies on the impact of filter-based ranking feature selection on security vulnerability prediction," *IET Software*, vol. 15, no. 1, pp. 75–89, 2021.

[128] D. K. Kim, "Finding bad code smells with neural network models," *International Journal of Electrical and Computer Engineering*, vol. 7, no. 6, p. 3613, 2017.

[129] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *The Journal of Machine learning research*, vol. 7, pp. 1–30, 2006.

[130] S. Herbold, "Crosspare: a tool for benchmarking cross-project defect predictions," in *2015 30th IEEE/ACM International Conference on Automated Software Engineering Workshop (ASEW)*. IEEE, 2015, pp. 90–96.

[131] J. Zou, Z. Li, X. Liu, and H. Tong, "Mscpdplab: A matlab toolbox for transfer learn-ing based multi-source cross-project defect prediction," *SoftwareX*, vol. 21, p. 101286, 2023.

[132] U. S. Bhutamapuram and R. Sadam, "With-in-project defect prediction using boot-strap aggregation based diverse ensemble learning tech-nique," *Journal of King Saud University-Computer and Information Sciences*, vol. 34, no. 10, pp. 8675–8691, 2022.

[133] H. Liu and L. Yu, "Toward integrating feature selection algorithms for classification and clustering," *IEEE Transactions on knowledge and data engi-neering*, vol. 17, no. 4, pp. 491–502, 2005.

[134] D. M. Hawkins, "The problem of overfitting," *Journal of chemical information and computer sciences*, vol. 44, no. 1, pp. 1–12, 2004.

[135] G. Czibula, Z. Marian, and I. G. Czibula, "Software defect prediction using relational as-sociation rule mining," *Information Sciences*, vol. 264, pp. 260–278, 2014.

[136] C. Tantithamthavorn, S. McIntosh, A. E. Hassan, and K. Matsumoto, "An empirical

comparison of model validation techniques for defect prediction models," *IEEE Transactions on Software Engineering*, vol. 43, no. 1, pp. 1–18, 2016.

[137] R. Malhotra and S. Meena, "Empirical validation of feature selection techniques for cross-project defect prediction," *International Journal of System Assurance Engineering and Management*, pp. 1–13, 2023.

[138] Z. Ma, G. Wu, P. N. Suganthan, A. Song, and Q. Luo, "Performance assessment and exhaustive listing of 500+ nature-inspired metaheuristic algorithms," *Swarm and Evolutionary Computation*, vol. 77, p. 101248, 2023.

[139] N. E. Fenton and N. Ohlsson, "Quantitative analysis of faults and failures in a complex software system," *IEEE Transactions on Software engineering*, vol. 26, no. 8, pp. 797–814, 2000.

[140] J. C. Knight, "Safety critical systems: challenges and directions," in *Proceedings of the 24th international conference on software engineering*, 2002, pp. 547–550.

[141] E. Arisholm and L. C. Briand, "Predicting fault-prone components in a java legacy system," in *Proceedings of the 2006 ACM/IEEE international symposium on Empirical software engineering*, 2006, pp. 8–17.

[142] P. He, B. Li, X. Liu, J. Chen, and Y. Ma, "An empirical study on software defect prediction with a simplified metric set," *Information and Software Technology*, vol. 59, pp. 170–190, 2015.

[143] H. K. Dam, T. Tran, T. Pham, S. W. Ng, J. Grundy, and A. Ghose, "Automatic

feature learning for predicting vulnerable software components," *IEEE Transactions on Software Engineering*, vol. 47, no. 1, pp. 67–85, 2018.

[144] R. Malhotra and M. Khanna, "Inter project validation for change proneness prediction using object oriented metrics," *Software engineering: An International Journal*, vol. 3, no. 1, pp. 21–31, 2013.

[145] R. Malhotra and S. Meena, "Empirical validation of cross-version and 10-fold cross-validation for defect prediction," in *2021 Second International Conference on Electronics and Sustainable Communication Systems (ICESC)*. IEEE, 2021, pp. 431–438.

[146] H. Tong, W. Lu, W. Xing, and S. Wang, "Array: adaptive triple feature-weighted transfer naive bayes for cross-project defect prediction," *Journal of Systems and Software*, vol. 202, p. 111721, 2023.

[147] U. S. Ba and R. Sadama, "How far does the predictive decision impact the software project? the cost, service time, and failure analysis from a cross-project defect prediction model."

[148] Y. Liu, T. M. Khoshgoftaar, and N. Seliya, "Evolutionary optimization of software quality modeling with multiple repositories," *IEEE Transactions on Software Engineering*, vol. 36, no. 6, pp. 852–864, 2010.

[149] X. Zong, G. Li, S. Zheng, H. Zou, H. Yu, and S. Gao, "Heterogeneous cross-project defect prediction via optimal transport," *IEEE Access*, vol. 11, pp. 12 015–12 030, 2023.

[150] U. Sharma and R. Sadam, "How far does the predictive decision impact the software project? the cost, service time, and failure analysis from a cross-project defect prediction model," *Journal of Systems and Software*, vol. 195, p. 111522, 2023.

[151] S. Amasaki, "Cross-version defect prediction using cross-project defect prediction approaches: Does it work?" in *Proceedings of the 14th International Conference on predictive models and data analytics in software engineering*, 2018, pp. 32–41.

[152] T. Fukushima, Y. Kamei, S. McIntosh, K. Yamashita, and N. Ubayashi, "An empirical study of just-in-time defect prediction using cross-project models," in *Proceedings of the 11th working conference on mining software repositories*, 2014, pp. 172–181.

[153] R. Malhotra, V. Gupta, and M. Khanna, "Applicability of inter project validation for determination of change prone classes," *Int. J. Comput. Appl*, pp. 0975–8887, 2014.

Supervisor's Biography

Prof. Ruchika Malhotra
Professor & HoD
Department of Software Engineering
Delhi Technological University
Email: ruchikamalhotra@dtu.ac.in

Educational Qualifications:

Postdoc (Indiana University-Purdue University Indianapolis, USA), Ph.D (Computer Applications)

Prof. Ruchika Malhotra is Head of Department and Professor in the Department of Software Engineering, Delhi Technological University, Delhi, India. She served as Associate Dean in Industrial Research and Development, Delhi Technological University from August 2018 to 2022. She was awarded with prestigious Raman Fellowship for pursuing Postdoctoral research in Indiana University Purdue University Indianapolis USA. She received her master's and doctorate degree in software engineering from the University School of Information Technology, Guru Gobind Singh Indraprastha University, Delhi, India. She has received IBM Faculty Award 2013. She has been ranked amongst the World's top 2% scientist by Stanford University report, USA, for her research in the field of 'Artificial Intelligence and Image Processing' in 2020, 2021, 2022, and 2023. She is recipient of Commendable Research Award (in 2018, 2019, 2020, 2021, 2022, and 2023) by Delhi Technological University. Her H-index is 37 as reported by Google Scholar. She is author of book titled 'Empirical Research in Software Engineering' published by CRC press and co-author of a book on Object-Oriented Software Engineering published by PHI Learning. She has published more than 245 research papers in international journals and conferences. Her research interests are in software testing, improving software quality, statistical and adaptive prediction models, software metrics and the definition and validation of software metrics.

Author's Biography

Shweta Meena
Research Scholar
Assistant Professor
Department of Software Engineering
Delhi Technological University
Email: shwetameena@dtu.ac.in

Educational Qualifications:
M.Tech. (SWE), B.Tech. (IT)

Shweta Meena is currently pursuing her doctoral degree from Delhi Technological University. She is currently working as Assistant Professor in the Department of Software Engineering, Delhi Technological University. She completed her master's degree in Software Engineering in 2018 from Delhi Technological University, Delhi, India. She received her bachelor degree in Information Technology in 2016 from the University School of Information Technology, Guru Gobind Singh Indraprastha University, Delhi, India. She is recipient of Commendable Research Award in 2023 by Delhi Technological University. She has been served as a reviewer in various international conference and journal. She has published various research papers in international conferences and journal. Her research interests are in software quality improvement, software testing, predictive modelling and data analysis.

DEPARTMENT OF SOFTWARE ENGINEERING

DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)

Bawana Road, Delhi-110042

PLAGIARISM VERIFICATION

Tittle of the Thesis: Empirical Validation of Cross-Project Methodologies for Software Quality Predictive Modeling

Total Pages: Name of the Scholar: Shweta Meena

Supervisor: (1)Prof. Ruchika Malhotra

Department of Software Engineering

This is to report that the above thesis was scanned for similarity detection. Process and outcome is given below:

Software used: Turnitin Similarity Index: 8% Total Word Count:

Date:

Candidates Signature **Signature of Supervisor**