

# **IDENTIFICATION OF DEFECTS IN A SOFTWARE USING MACHINE LEARNING TECHNIQUES**

A DISSERTATION

SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR  
THE AWARD OF DEGREE  
OF  
MASTER OF TECHNOLOGY  
IN  
SOFTWARE ENGINEERING

Submitted by:

**NISHTHAA**  
**2K20/SWE/15**

Under the supervision of

**Dr. RUCHIKA MALHOTRA**  
Head of Department (Software Engineering)

Associate Dean IRD, DTU



**DEPARTMENT OF SOFTWARE ENGINEERING**  
**DELHI TECHNOLOGICAL UNIVERSITY**  
(Formerly Delhi College of Engineering)  
Bawana Road, Delhi-110042

MAY, 2022

**DEPARTMENT OF SOFTWARE ENGINEERING**

**DELHI TECHNOLOGICAL UNIVERSITY**

(Formerly Delhi College of Engineering)

Bawana Road, Delhi - 110042

**CANDIDATE'S DECLARATION**

I, Nishthaa, Roll No. 2K20/SWE/15 student of M. Tech (Software Engineering) hereby declare that the project Dissertation titled "**Identification Of Defects in a Software Using Machine Learning Techniques**" which is submitted by me to the Department of Software Engineering, Delhi Technological University, Delhi in partial fulfilment of the requirement for the award of the degree of Master of Technology, is original and not copied from any source without proper citation. This work has not previously formed the basis for the award of and Degree, Diploma Associateship, Fellowship or other similar title or recognition.

Place: Delhi

Date:

*Nishthaa*

NISHTHAA

**DEPARTMENT OF SOFTWARE ENGINEERING**

DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)

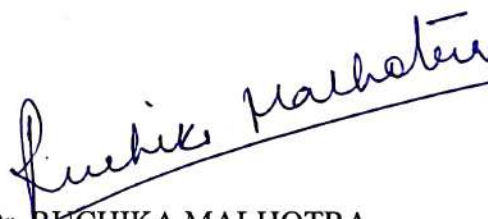
Bawana Road, Delhi - 110042

**CERTIFICATE**

I hereby certify that the Project Dissertation titled "**Identification Of Defects in a Software Using Machine Learning Techniques**" which is submitted by Nishthaa, 2K20/SWE/15 Department of Software Engineering, Delhi Technological University, Delhi in partial fulfilment of the requirement for the award of the degree of Master of Technology, is a record of the project work carried out by the student under my supervision. To the best of my knowledge, this work has not been submitted in part or full for any Degree or Diploma to this University or elsewhere.

Place: Delhi

Date:



Dr. RUCHIKA MALHOTRA

**SUPERVISOR**

Professor,

Associate Dean IRD, DTU

Department of Software Engineering,

Delhi Technological University

(Formerly Delhi College of Engineering)

Bawana Road, Delhi - 110042

## ACKNOWLEDGMENT

The success of this project requires the assistance and input of numerous people and the organisation. I am grateful to everyone who helped in shaping the result of the project.

I express my sincere thanks to **Dr. Ruchika Malhotra**, my project guide, for providing me with the opportunity to undertake this project under her guidance. Her constant support and encouragement have made me realise that it is the process of learning which weighs more than the end result. I am highly indebted to the panel faculties during all the progress evaluations for their guidance, constant supervision and for motivating me to complete my work. They helped me throughout with new ideas, provided information necessary and pushed me to complete the work.

I also thank all my fellow students and my family for their continued support.

*Nishtha*

**NISHTHAA**

## ABSTRACT

The end users who are using the software and its products is vastly increased when compared to the earlier days. As we are seeing that the technology has evolved a lot and it has delivered an extraordinary technology named artificial intelligence. Identifying defects in a software in the current time can be held with Software Development Life Cycle (SDLC) and it stays a fundamental and crucial task. In the present days, a few instances of flawed and non-defective modules are used to construct prediction models which utilize machine learning techniques. To address the software modules, software metrics were used as input to these machine learning algorithms. In order to detect the defects in a software, few powerful machine learning techniques are implemented and in existing system the algorithm named Random Forest (RF) gives an adequate accuracy. But we need to identify the defects in a software using machine learning methods with better model which must give some improved accuracy when compared with RF. So here in this study we are using extra trees classifier and hybrid model to identify the defects in a software.

## CONTENTS

<b>Candidate's Declaration</b>	ii
<b>Certificate</b>	iii
<b>Acknowledgement</b>	iv
<b>Abstract</b>	v
<b>Contents</b>	vi
<b>List of Figures</b>	ix
<b>List of Tables</b>	x
<b>List of Abbreviations</b>	xi
<b>CHAPTER 1 INTRODUCTION</b>	<b>1</b>
1.1 General	1
1.2 Problem Formulation	3
1.3 Objectives of the Project	4
1.4 Organisation/Dissertation of Thesis	4
<b>CHAPTER 2 LITERATURE REVIEW</b>	<b>5</b>
2.1 Individual Classifiers For Defect Prediction	5
2.2 Feature Selection For Software Defect Prediction	6
2.3 Homogeneous and Heterogeneous Ensemble	7
<b>CHAPTER 3 THEORETICAL CONCEPTS</b>	<b>11</b>
3.1 Features	11
3.2 Data PreProcessing	12
3.2.1 Importing data into a dataset	13

3.2.2	Evaluation of Test-Train Split	15
3.3	Feature Selection	16
3.4	Classification	18
3.5	Ensemble Techniques	19
3.6	Software Defect Prediction	19
3.7	SDLC-Software Development Life Cycle	23
3.8	Practicability Investigation	25
3.8.1	Technical Practicability	25
3.8.2	Social Practicability	26
3.8.3	Economic Practicability	26
3.9	System Requirements Specification	27
3.9.1	Non-functional and functional requirements	27
3.9.2	Non-functional necessities	27
3.9.3	Functional necessities	28
<b>CHAPTER 4 EXPERIMENTAL DESIGN</b>		<b>29</b>
4.1	Independent and Dependent Variable	29
4.2	Empirical Data Collection	32
<b>CHAPTER 5 WORKING AND ANALYSIS</b>		<b>33</b>
5.1	Dataset	33
5.2	Dataset Pre-processing	34
5.3	Feature Selection	34
5.4	Classification and Ensemble Techniques	35
5.5	ML Techniques	36
5.5.1	CatBoost	36
5.5.2	Extra Trees Classifier Algorithm	38
5.5.3	Gradient Boosting	40
5.5.4	Light GBM	40
5.5.5	Hybrid Model	41
5.6	Performance Evaluation Measure	42

5.6.1 Accuracy	42
5.6.2 Precision	42
5.6.3 Recall	42
5.7 Probability Score	43
5.7.1 Probability Analysis Software	44
5.7.2 Probability Analysis Process	45
<b>CHAPTER 6 RESULTS</b>	<b>46</b>
<b>CHAPTER 7 CONCLUSIONS AND FUTURE SCOPE</b>	<b>54</b>
<b>References</b>	<b>56</b>



**LIST OF FIGURES**

2.1	ML Techniques	8
3.1	Data Variable Types	11
3.2	Feature Selection	17
3.3	Representation of Waterfall Model	23
5.1	Proposed Methodology	35
5.2	Catboost Algorithm	38
5.3	Basic Indicators for defect Prediction	43
5.4	Probability Score	45
6.1	Upload Dataset	51
6.2	View Dataset	51
6.3	Pre-processing	52
6.4	Model Trained with Hybrid Model	52
6.5	Model Trained with Extra Trees Classifier	53
6.6	Predicting Defected Software	53

## LIST OF TABLES

2.1 Summary of Related Work	9
6.1 Results obtained for jm1 Dataset	46
6.2 Results obtained for cm1 Dataset	47
6.3 Results obtained for mw1 Dataset	48
6.4 Comparison between other's work	49

### **LIST OF ABBREVIATIONS**

- 1 ML: Machine Learning
- 2 SVM: Support Vector Machine
- 3 lgbm: Light Gradient Boosting Machine
- 4 RF Classifier: Random Forest Classifier
- 5 SDP: Software Defect Prediction
- 6 LOG: Logistic Regression
- 7 RBF: Radial Basis Function Network
- 8 BN: Bayesian Network
- 9 DTree: Decision Tree
- 10 DTables: Decision Tables
- 11 SDLC: Software Development Life Cycle
- 12 GOSS: Gradient Boosting one sided sampler
- 13 SMOTE: Synthetic Minority OverSampling Technique
- 14 WPDP: Within Project Defect Prediction

# CHAPTER 1

## INTRODUCTION

### 1.1 GENERAL

For improving the reliability rate of software, few measuring techniques are used. Software Defect Prediction (SDP) has a lot of test experience in computer science for finding flaws. Mostly in the present scenario, the curiosity amongst the individuals has increased a lot because the majority of the devices contain software programs which have become important to its customers because it includes appealing features, and buyers want to access them without having to learn anything. Yet, the focus of interest was that it evolved into a communal requirement whereby individuals can connect and share knowledge. In the recent decade, people have been focusing on application frames, where performance enhancement is seen as the most important aspect of client functioning. Software performance has remained a perplexing subject, yielding insufficient outcomes for commercial and facilities services, owing to the clear tremendous growth of utilization. Throughout the growth phase, firms frequently use fault diagnostic patterns and similar methodologies to help in anticipating defects, estimating effort, assessing software dependability, risk assessment, and so on. With a given huge dataset, a controlled machine learning prediction computation is employed. Following that, the algorithm learns with the help of the training sample and develops instructions in order to predict the class name for every newer data set. Using math equations to construct and improve the indicator work is one of the learning stages. There is a certain intake esteem and a specific yield esteem in that interaction training set. A generally

known result is used to assess the correctness of a typical Machine Learning (ML) calculation.

Some concepts may be to establish a gathering of associates at a specified location for casual contact amongst data users. The quality of the Software may be improved by forecasting the failure areas. The detection of defects is one type of generating methodologies that many researchers tend to use early in the contact process to identify problematic frames such as units or categories. It is accomplished by classifying components into two: one is defective case and other is non-defective case. Several methodologies have been deployed to recognize the defective modules, some of them include: CatBoost, XGBoost, SVM, Light Gradient Boosting, RF classifier and Hybrid algorithm.

Each classifying item (i.e.) connection intermediary to the training dataset class mark and the attributes, is placed down on the classifier technique and analyzed using the target order equations. Such parameters will also be used to choose the names of future data classes. These complicated data may be classified in this way by using categorization algorithms and classifiers. Identifying software problems, discovering the defect, and acknowledging it is difficult work for specialists. The main purpose is to divide the software data into defective and non-defective datasets as a paradigm for identifying issues. The input software dataset is supplied to the classifier in this method, and the client knows the real class values. Prior to this graph, metric strategies centered on need and setup yielded long-term outcomes. Regardless, the design of methodologies and the accuracy of forecasts remain a challenge that must be addressed.

Predicting the Software defects always aim towards the forecasting of defect-prone software systems with the deployment of a few essential elements of the software project. It's usually done by creating forecasting models for known projects using project attributes reinforced with defective data, and then using these forecasting models to anticipate defects for unknown projects. SDP is based on the idea that if a project is created in a defect-prone environment, every module created

in a similar environment with comparable job characteristics would be troublesome as well. The reason for predicting the software defects is done to anticipate defect affected software modules based on a set of baseline software project characteristics. Creating a classification algorithm for a known project utilizing project attributes supplemented with incorrect information, and then applying the prediction system to new projects to foresee issues, is a common method. If a program generated in a given environment causes flaws, then every component created in a restricted sequence with identical project parts would produce errors as well. The RF ensemble technique performs well, but our major aim is to improve the accuracy gained and more precisely detect the defective software. As a result, we devised a method that uses a hybrid algorithm and an extra trees classifier to provide more accuracy than previous ensemble methods.

## **1.2 PROBLEM FORMULATION**

SDP is a software testing method that can help increase programme reliability. It has a lot of experience in computer science testing and flaw detection. Customers have been curious about software programmes in this case because they have attractive features that they wish to utilise without having to understand anything. However, what piqued people's interest was how it blossomed into a collective need for people to connect and share information. In the recent decade, people have been focusing on application frames, where performance improvement is seen as the most important aspect of client functioning. Software performance remains a difficult subject that produces insufficient outcomes for commercial and facilities services, owing to the clear tremendous growth of utilisation. Organizations frequently use fault diagnostic patterns, and similar methodologies aid in anticipating defects, calculating effort, evaluating software reliability, risk assessment, and other tasks throughout the growth phase.

### **1.3 OBJECTIVES OF THE PROJECT**

A regulated ML prediction calculation is utilised with a specified large dataset. Following that, the algorithm takes what it has learned from its trained set, in order to create instructions to guess the class name in the fresh set of data. One of the learning stages is using math equations to develop and improve the indication work. In that interaction training set, there is a certain intake esteem and a specific yield esteem. To measure the accuracy of a typical ML computation, a well-known result is employed. Predicting failure locations helps improve software quality. Defect detection is a technique for developing models that can be utilised early in the contact process to identify incorrect frames such as units or categories. It's done by determining whether or not a component is prone to defects. CatBoost, XGBoost, SVM, Light Gradient Boosting, RF classifier, and Hybrid algorithm are just a few of the methods that have been used to identify the damaged modules.

### **1.4 DISSERTATION OF THESIS**

This thesis is classified into various chapters. Chapter 2 describes the related work in the area of study. Chapter 3 gives detailed explanation of all the theoretical concepts required. Chapter 4 briefly describes the experimental design and empirical data collection . Chapter 5 contains all the working and analysis. Chapter 6 outlines the results achieved for our proposed model on all the data sets used. Chapter 7 describes the conclusion and future scope of the project.

## **CHAPTER 2**

### **LITERATURE REVIEW**

In this section, information related to the variety of research papers concentrating on software defect prediction using different methodologies, challenges present in defect prediction, and which techniques can perform better, etc. has been listed. This section provides a brief insight into the previous works done in the field of software defect prediction.

Many researchers have been trying to build software defect prediction models using different defect prediction techniques which deliver better performance, but most of them use static code metrics as independent variables, and few of them use feature selection to analyze metrics.

#### **2.1 INDIVIDUAL CLASSIFIERS FOR DEFECT PREDICTION**

Logistic Regression (LOG), Support Vector Machines (SVM), Radial Basis Function Network (RBF), Multi-Layer Perceptron (MLP), Bayesian Network (BN), Decision Tree (DTree), and Decision Tables(DTables), are some of the classifiers, that can be used to develop a defect prediction model. However, past studies have shown that there is no obvious winner when it comes to predicting defect proneness. Their accuracy is determined by the predictor we chose and the dataset we used.



Lessman et al. [23] experimented with 22 classification models. The top 17 models were statistically similar to each other on ten publicly available software development data sets from the NASA repository. Later Shepperd et al. [24] found that the NASA dataset used was noisy and biased.

## 2.2 FEATURE SELECTION FOR SOFTWARE DEFECT PREDICTION

Feature selection is the process of selecting  $N$  features from the original set of features to decrease the dimensionality of the feature pool and boost prediction model performance while reducing cost and time to create it.

The various Feature Selection methods, are listed below:

- a. **Wrapper Method:** These methods carry out the selection process keeping in mind the classification algorithms that are going to be used. Wrapper methods evaluate the variable subset using the predictor as a black box and the predictor performance as the objective function.[25].
- b. **Filter Method:** These methods carry out the selection process without considering the algorithm used for classification. Because they function independently of the classification process, filter methods are faster than wrapper methods and result in superior generalisation.
- c. **Embedded method:** These methods encompass the benefits of both the wrapper and filter methods by including interaction of features, while keeping the computational cost low. Embedded methods are iterative because they take care of each iteration of the model training process and carefully extracts those features that contribute the most to the training for a particular iteration [26].

### 2.3 HOMOGENEOUS AND HETEROGENEOUS ENSEMBLE

Ensemble models have been demonstrated exceptionally successfully to improve the precision and the presentation of the models. An ensemble consists of a set of individually trained classifiers, such as neural networks or decision trees, whose predictions are combined when classifying new instances [27]. Ensemble takes place in two steps:

- a. Model Training: Training each individual classifier with the same training dataset but using different subsets.
- b. Model Combination: Combining the power of all the trained classifiers using one of the combining techniques (Averaging or Voting).

The two types of ensemble techniques, are as follows:

- a. Homogeneous Ensemble: It consists of classifiers having a single-type base learner. Example bagging or boosting.
- b. Heterogeneous Ensemble: It consists of classifiers having different base learning algorithms. Example stacking or voting of bagged classifiers.

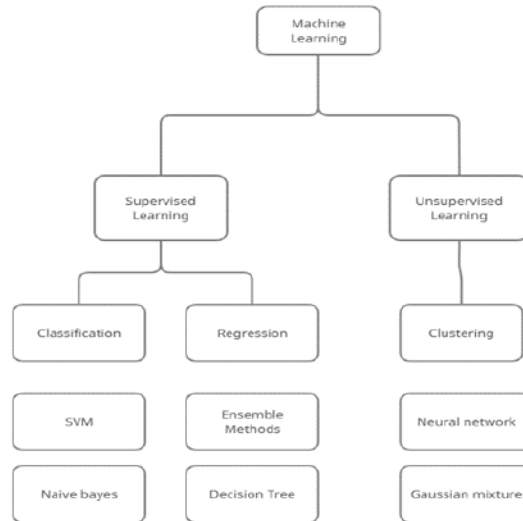


Figure 2.1. ML Techniques

Zhou et al. [28] make an empirical comparison consisting of 32 feature selection methods and the result of this study shows that feature selection algorithms significantly improve the defect prediction performance. Also, Wrapper and filter feature selection algorithms give the best result compared to clustering-based, but they tend to take more time to select features.

Shivaji et al. [29] conducted research showing that the performance of defect prediction models is increased by eliminating 90% of the original features. Also, NOVAKOVIC et al. [30] compares the performance of 5 filter feature selection algorithms concluding that to rank the algorithms based on their performance, one will need to keep indices to check, the best feature subset is chosen, and a larger dataset with additional classifiers is required.

The following table also summarizes some of the related work close to our proposed work

**TABLE 2.1** A summary of related research.

Author	Proposed	Finding/Outcomes
Guisheng Fan, Xuyang Diao, and Huiqun Yu [1]	They proposed using an RNN-based architecture to detect faults. DP-ARNN, in particular, parses abstract syntax trees and returns data in the form of vectors	The test findings show that DP-ARNN improves the F1-measure by 14% on average, and the area under the curves also improves
Ebubeogu Amarachukwu Felix, and Sai Peck Lee [2]	Using predictor variables obtained from defect acceleration, such as defect density, defect velocity, and defect introduction time, this paper presents a method for identifying the connection of each predictor variable with the number of defects.	The mean fault speed is substantially positively associated with the number of defects, with a correlation value of 0.98. As a result, it has been demonstrated that this strategy can be used to improve the efficacy of software design processes by serving as a blueprint for program testing

Zhou Xu, Jin Liu,  
Xiapu Luo,  
Zijiang Yang,  
Yifeng Zhang,  
Peipei Yuan,  
Yutian Tang, and  
Tao Zhang [3]

Kernel Principal Component Analysis and Weighted Extreme Learning Machine are two techniques used in the classifier architecture. Propose the KPWE defect prediction framework, which integrates two techniques: Kernel Principal Component Analysis (KPCA) and Weighted Extreme Learning Machine (WELM) (WELM)

KPWE outperforms 41 benchmark approaches, encompassing seven fundamental classifications using Kernel Principal Component Analysis, five versions of KPWE, 8 representational classification methods with Weighted [Extreme Learning Machine](#), and 21 unbalanced learning techniques

Zhou Xu, Shuai Pang, Tao Zhang, Xia-Pu Luo, Jin Liu, Yu-Tian Tang, Xiao Yu, and Lei Xue [4]

Cross-project defect prediction systems based on transfer learning are now in use. In general, such solutions aim to lessen the discrepancies in information dispersion between the two programs.

In comparison to 12 baseline approaches, balanced distribution adaptation obtains average gains of 23.8%, 12.5%, 11.5%, 4.7%, 34.2%, and 33.7% across datasets

Ruchika Malhotra,  
Shine Kamal [5]

This paper examines the performance of machine learning classifiers for software defect prediction on twelve imbalanced National Aeronautics and Space Administration datasets using sampling techniques and cost sensitive classifiers

The performance of machine learning techniques improved dramatically when oversampling approaches were used to address the uneven nature of datasets

Ruchika Malhotra,  
Shine Kamal [6]

Adaptive Synthetic (ADASYN), SPIDER, and Safe-Level-SMOTE are three of the unexplored oversampling strategies used in this work to create a tool

The findings obtained utilising three machine learning algorithms, namely random forest, J48, and naive bayes, demonstrate that the approaches utilised in this work are better than SMOTE in the majority of cases for the Object Oriented (OO) defect dataset in terms of ROC and recall

## CHAPTER 3

### THEORETICAL CONCEPTS

#### 3.1 FEATURES

A feature is an individual measurable property, or characteristic of a phenomenon being observed [38]. Data objects are described by many features, which captures the essence of the object under consideration.

Overview of Data Variable Types

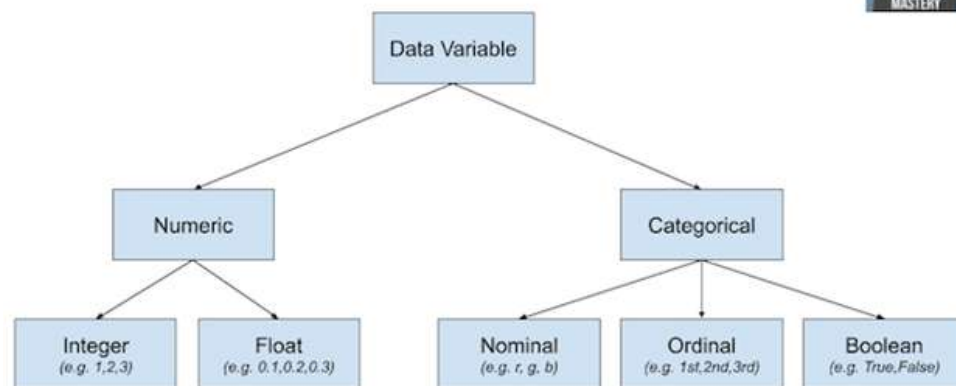


Fig. 3.1 Data Variable Types [5]

There are 2 types of features:

1. Categorical - values taken from a defined set. Example: Days of the week.
2. Numerical - values are continuous or integer-valued. Ex.: Speed of the car.

## 3.2 DATA PREPROCESSING

Pre-processing refers to the transformations applied to our data before feeding it to the algorithm. Data Pre-processing is a technique that is used to convert the raw data into a clean data set.

The pre-processing of data is the method for making the raw data ready for deployment in a ML approach. The pre-processing is the initial and most significant step in establishing any type of ML approach.

Many of us might not be provided with well-prepared and clean data while we work on a ML process. Furthermore, before we take up any kind of data-based project, it is necessary to clean up the data in order to format that data as per our project requirements and desired outcomes/ applications. Thus, the pre-processing of data has become indispensable now-a-days.

Practical data might sometimes contain missing values or noises and is in an unsuitable/ undesirable format that cannot be used directly in machine learning models. Thus, the data pre-processing is a necessary task for cleaning data and making it suitable for the ML projects, which improves the efficiency and accuracy of the model. The processes in a typical ML project are as follows:

- Getting the dataset
- Importing libraries
- Importing datasets
- Finding Missing Data
- Encoding Categorical Data
- Splitting dataset into training and test set
- Scaling of Feature

Practicing with various categorical datasets is the reason for the success of the ML as a field or it is also the reason for the success of a data scientist from the career perspective. However, identifying the appropriate dataset for our deployed ML project is always a daunting task. Thus, we will list the info of various sources from which you can get the appropriate datasets that suit your ML project.

### 3.2.1 Importing data into a dataset

- Choose your dataset when required from the dropdown list available on the web page of the shared Datasets in order to access the tab 'Import'.
- Select the source of import for the data from any of the following: Cloud Storage, BigQuery, or your local Personal Computer. Give all the data needed.
- You should define the Cloud Storage bucket when you open the CSV formatted files from your local Personal Computer. Those files get opened to the defined bucket before they get imported onto the AutoML Tables. Those files would be present there after the data import, until you pull out them.
- The bucket should be in a location, which should be same as that of the dataset location.
- Select 'Import' for initiating the import operation.
- Once the import operation gets completed, the tab 'Train' is shown in order to get ready for training your model.

There are 5 steps for ensuring missing data recognition to appropriately dealt with it, which are as follows:

- Ensure your data are coded correctly.
- Recognize missing values within each variable.



- Look out for patterns of missingness.
- Check for relations between observed and missing data.
- Decide on how to handle missing data.

Encoding categorical data is a process of converting categorical data into integer format so that the data with converted categorical values can be provided to the different models. In the field of data science, before going for the modeling, data preparation is a mandatory task

The fulfillment of ML projects lie not only on the hyperparameters and project itself, but also the way we operate and feed varied categorical variables to the project. As many ML projects could only acquire the numerical variables, pre-processing of the various kinds of variables has become an indispensable step. The data scientists have to convert these various kinds of variables to numbers format in a way that the project has the capability to comprehend and extricate the useful data.

As a matter of fact, a data scientist utilizes 70% to 80% of his/ her time in cleaning up and making the data ready to use. Also, the conversion of various kinds of data has become an indispensable operation, which not only raises the quality of the project, but also aids in proper feature handling.

The test-train split methodology is utilized to compute the fulfillment of ML algorithms for making forecasting on the data, which was not utilized for training the project.

The test-train split methodology is a quick and smoother methodology that yields various outcomes that would permit you to differentiate the fulfillment of ML algorithms towards the forecasting building issue. Though this methodology is easy to utilize and elucidate, there are certain circumstances in which you don't have to use this methodology. For instance, the circumstances where we deploy the small datasets and where the supplementary configuration is needed, we don't utilize

test-train split methodology. Note that the supplementary configuration might be needed for the classification purposes and for the cases of non-balanced datasets.

### 3.2.2 Evaluation of Test-Train Split

The test-train split methodology is done for examining the fulfillment of the ML methods.

- It is utilized for regression or classification issues.
- It is utilized for the supervised learning categorical methods.
- In this methodology, we take a dataset and then split the taken dataset into 2 segments. The initial segment is utilized to fit the project, which is known as the “training dataset”. The second and next segment is not utilized for the project training purpose, but the input part of it is given to the project, based on which the forecasting is done and further we infer the differentiations by comparing with the desired values. This type of dataset is known as the “testing dataset”.
- Training Dataset: It is utilized to fit our ML project.
- Testing Dataset: It is utilized to examine our fitted ML project.
- The goal is to compute the fulfillment of the ML project on the latest (new) data- the data which has not been utilized for training the project.
- This way we desire to utilize the project in real-time by fitting on data available with familiar inputs as well as the outputs. Then, we execute the forecasting on newer samples in future without having any desired target values or outputs in mind.
- The test-train split methodology is suitable where a sufficient enough huge dataset is available.

Feature scaling is a process that we deploy for normalizing the independent parameters range or features in the data. From the context of data processing, Feature

scaling is also known by the name “data normalization”, which is commonly executed at the time of data pre-processing operation.

- Absolute Maximum Scaling.
- Min-Max Scaling.
- Normalization.
- Standardization.
- Robust Scaling.

Feature scaling through standardization (or Z-score normalization) can be an important preprocessing step for many machine learning algorithms. Standardization involves rescaling the features such that they have the properties of a standard normal distribution with a mean of zero and a standard deviation of one.

### **3.3 FEATURE SELECTION**

While generating the forecasting project, the feature selection is the method by which we carry out the minimization of the count of input parameters.

The count of input parameters must be lowered to reduce the cost of the computation in the modeling and, in few circumstances, to raise the performance of the project.

The association between every input parameter and the target parameter is examined with the deployment of statistics, and then we select the type of input parameters that has the powerful link with the target parameter. Though the selection of statistical estimates relies on the input data type and output parameters, these methodologies are yet able to be quick and victorious.

Thus, choosing an allowable statistical estimate for the dataset during the execution of the filter-oriented feature selection could be challenging for a ML expert.

In this post, we'll learn how to use statistical measures to choose numerical and categorical data for filter-based feature selection.

- Unsupervised methodologies are divided into filter, wrapper, and intrinsic methodologies and the supervised methodologies also get divided into filter, wrapper, and intrinsic methodologies.
- Filter-oriented feature selection methodologies have its dependence or correlation existing between the input parameters, which might be subjected to filtering operations for selecting the most appropriate features with the deployment of the statistical estimates.
- On the basis of input data type parameters and the response or output parameter, statistical estimates for feature selection should be estimated carefully.

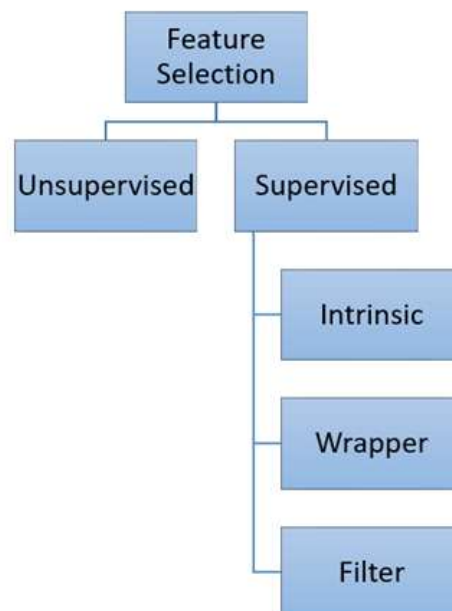


Fig. 3.2 Feature Selection

### 3.4 CLASSIFICATION

The Classification type of methodology is a Supervised Learning method that utilizes the training data for determining the categories of new observation. Classification is the method of acquiring the software knowledge from numerous observations or a dataset and then we execute the classification of the new observations into one among the several groupings or categories. Those groupings include 0 or 1, Yes or No, cat or dog, Spam or non-Spam, and many more. For describing the classes, the terms that we tend to use are: Targets/ categories or labels.

Dissimilar from the regression methodology, Classification generates a class instead of a value, such as “animal or fruit”, “Blue or Green”, and many more. Since the Classification methodology is a type of supervised learning approach, it utilizes input data that are labeled by comprising both input as well as the output.

- Binary Classifier: This type of classifier is used when there are only two possible outputs to a classification task. YES or NO, MALE or FEMALE, SPAM or NOT SPAM, CAT or DOG, and so on are some examples.
- Multi-class Classifier: A Multi-class Classifier is used when a classification task involves more than two outcomes. Classifications of different sorts of crops, for example, or classifications of different types of music.

### 3.5 ENSEMBLE TECHNIQUES

Ensemble learning is a common meta method in ML that merges forecasting made from a variety of projects in order to improvise the performance of the forecasting. Though there is no restriction in the number of ensembles, still the ensemble learning field is found dominated by the three methodologies from the perspective of forecasting modeling tasks. These Ensemble learning methodologies not only serve the purposes of the conventional algorithms, but also serve the various kinds of specialised processes. Boosting, Stacking, and Bagging are the three key groups of the ensemble learning methodologies. Note that everyone should need to develop a thorough comprehension of every aspect in your project for utilizing them towards successful forecasting.

- Boosting consists of orderly attaching ensemble associates, which get corrected before the project predictions in order to obtain a weighted mean of the forecasting.
- Stacking is the method of fitting several categorical projects to the identical data and then utilizing the other project for learning to achieve integration of the forecasting in the most suitable way.
- Bagging is the method of fitting several decision trees to varying samples of the identical dataset and then estimating the mean of the outcomes.

Boosting is an ensemble method that could learn from the flaws made by the earlier forecasters for making the forecasting far better in the time ahead. This method merges the numerous weak foundation learners for creating one powerful learner in order to improvise the forecasting abilities of the projects. Works are boosted by placing the weak learners in such an order to empower the weak learners to learn from the successive learner in order for achieving better forecasting projects.

Boosting is of different forms that include Adaptive Boosting (AdaBoost), XGBoost (Extreme Gradient Boosting), and gradient boosting. AdaBoost utilizes the weak learners in the structure of decision trees in order to indulge one split, which is well known by the name “decision stumps”. The major decision stump of the AdaBoost consists of observations having the same weights.

Likewise, XGBoost is another kind of ensemble method that utilizes decision trees along with the boosted gradient to yield improvised performance and speed. This boosting methodology depends majorly on the performance of the target project and speed of the computational operation. The training of the project with gradient boosted devices must adhere to a certain order, due to which the implementation of it is slower.

These gradient boosting methodologies ensure that the forecasters get added orderly onto the ensemble, wherein every preceded predictor corrects the flaws made by those that succeed to raise the accuracy of the model. Every newly added forecaster is ready for countering the adverse effects due to the flaws caused by the earlier forecasters. These gradient boosters detect the issues in the forecasting made by the learners and act against it appropriately with the help of the gradient of descent.

Stacking is a yet another ensemble methodology, which is mostly known as “stacked generalization”. This methodology operates by permitting the training method to ensemble the forecasting made by numerous other identical learning processes. This Stacking ensemble methodology is found to be applied in density computations, regression, classifications, and distance learning. This could also be utilized to compute the rate of errors that take place at the time of bagging.

Bagging, otherwise known as the bootstrap aggregating is majorly implemented in regression and classification applications. It raises the accuracy of the projects via the decision trees in order to reduce the variance suddenly. Reducing the variance will help in avoiding the overfitting and raising the accuracy, which were the problems existing in several forecasting projects.

This Bagging method could be further classified into two varieties such as aggregation and bootstrapping. Aggregation in the bagging is carried out to include every suitable output of the forecasting for randomizing the output. The forecasting can't be accurate enough since the consideration of all outputs would not be possible without aggregation. Hence, the aggregation is dependent on the possibility of bootstrapping strategies or it is based on all the outputs from the forecast project.

Bootstrapping is a type of sampling method, wherein the samples get extracted from the entire occupants (set) with the deployment of the substitution process. The sampling done via replacement methodology aids to randomize the selection process. The foundation learning methodology could operate based on the samples for completing the process.

Bagging is beneficial as the weakly foundation learners get merged to develop a sole powerful learner, which is more robust than the individual learners. Bagging also avoids the deviation, which in turns helps in lessening the overfitting of projects. Only challenge that the bagging method has to suffer is higher computational cost. Hence, bagging could result in higher bias in projects if the appropriate strategy of bagging is disregarded.

### **3.6 SOFTWARE DEFECT PREDICTION**

Software Defect Prediction (SDP) is one of the most assisting activities of the Testing Phase of SDLC. It identifies the modules that are defect prone and require extensive testing. This way, the testing resources can be used efficiently without violating the constraints.



Software Defect means a genuine error, malfunction, fault or failure within the Source Code of the Software, which prevents the Software from operating as intended model of defect prediction, reliability-based models use the operational profile of a system to predict failure rate that the project will face. Also in most projects, information collected in the testing and defect detection is analyzed too.

Usually, defect prediction models are investigated in the literature using a within-project context that assumes the existence of previous defect data for a given project [2]. This approach is called Within-Project Defect Prediction (WPDP).

Defect forecasting: Turn past mistakes into future gains

Gather data from previous releases. ...

Use prior defect reports to develop baseline predictions. ...

Organize projected defect identification instances according to sprint cycle and testing functions. ...

Re-evaluate predictions as velocity becomes more consistent.

Software defects originate in multiple origins. The approximate U.S. total for defects in requirements, design, code, documents, and bad fixes is 5.00 per function point. Best in class projects are below 2.00 per function point. Projects in litigation for poor quality can top 7.00 defects per function point.

Defect Software prediction is regarded as one of the most beneficial and cost-effective operations. It is regarded by software professionals as a critical phase that determines the quality of the product being built. It has played a significant role in refuting claims that the software industry is incapable of delivering requirements on time and on budget. Aside from that, client feedback on product quality has shifted dramatically from poor to satisfactory.

For defect prediction, various data miners have now supplanted earlier statistical approaches. The classification model, which places the component in one

of two classes: fault prone or non-fault prone, is the foundation of data mining. Initially, the classifier is given cases that are previously known and whose class we recognise. After the model has been trained, it is tested on unknown examples and the technique's prediction performance is evaluated. The requirement and design metrics were used for the majority of the research.

### 3.7 SDLC- SOFTWARE DEVELOPMENT LIFE CYCLE

In this project, we are going to make use of the waterfall model as our software development cycle. The waterfall model was selected due to the ease that the model offers via its step-by-step process during the implementation.

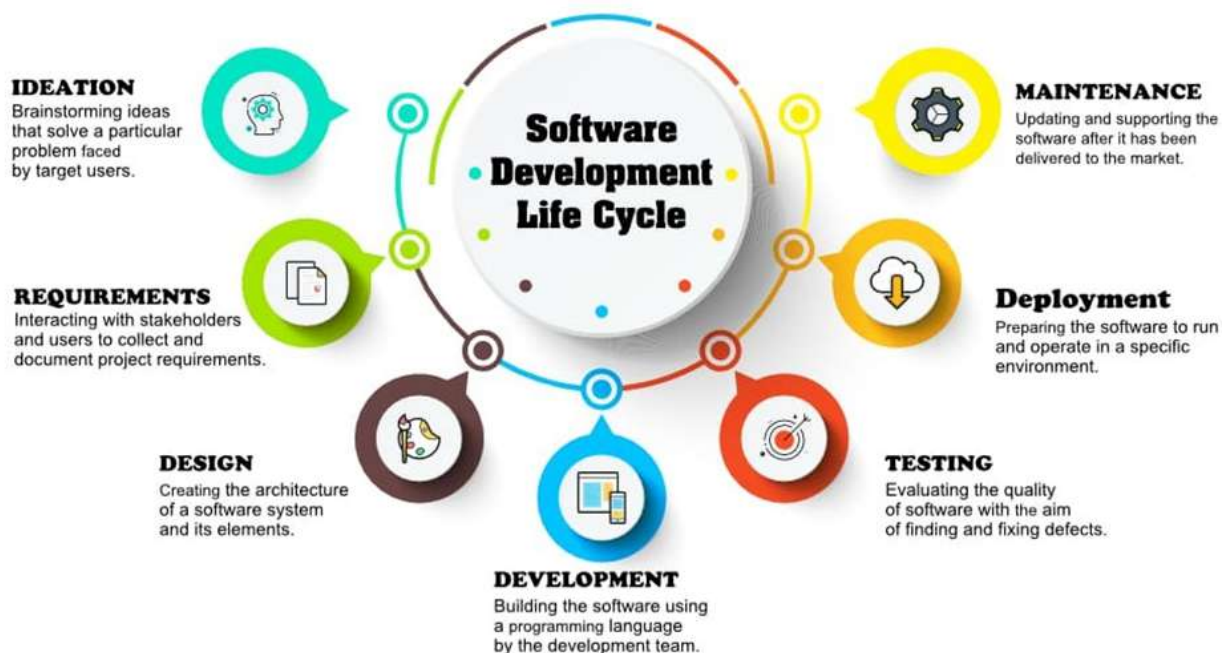


Fig 3.3 Pictorial Representation of the Waterfall Model

- **Necessity Gathering and investigation** – Every suitable necessity of the system that is to be devised are detained in this stage and then we document in a necessity specification record.
- **Design of System** – The necessity specifications from the initial stage are investigated in this stage for preparing the design of the system. This prepared design of the system aids in itemizing both the system and hardware necessities. This will in turn aid in specifying the architecture of the entire system.
- **Implementation** – The system is initially devised in compact programs known as units by using the inputs derived from the design of the system. This will be desegregated in the succeeding stage. Every unit is devised and investigated for their functionality. This is called Unit Testing.
- **Desegregation and Investigating** – All the units developed in the implementation phase are desegregated into a system once the testing of every unit gets completed. The whole system is investigated for any malfunctions and faults after the completion of desegregation.
- **Utilization of the system** – After the completion of both the non-functional and functional investigation, the product is applied in the environment of the customer or it is directly let out into the outer market.
- **Sustainment** – There will be few problems that might arise when utilized in the environment of the customer. For fixing these problems, patches are let out periodically. Furthermore, for enhancing the released product, far better versions of it are also let out in order to fulfill the customer's needs and thereby retaining them. Sustainment is carried out to implement the desired variations in the environment of the customer.

### **3.8 PRACTICABILITY INVESTIGATION**

The Practicability of the taken-up project is investigated in this stage and then we prepare the business proposal report consisting of a fundamental common strategy for the project along with few cost estimations. When we investigate the system, the practicability investigation of the devised system is done. This investigation is done to ensure that the devised system is not anyway a burden to the firm. To complete the Practicability investigation, the main necessities for the system must be understood well.

The three crucial considerations indulged in the Practicability investigation are as follows:

- .. Technical Practicability
- .. Social Practicability
- .. Economical Practicability

#### **3.8.1 Technical Practicability:**

This investigation is done to inspect the technical Practicability (i.e.) the technical necessities of the devised system. The devised system should not be having a higher demand on the accessible technical assets. Otherwise, higher demand would get developed for the system, which will raise the demand levels in the client side too. The devised system should have an unpretentious necessity, as negligible or nil variations are needed towards the implementation of the system.

### **3.8.2 Social Practicability:**

The reason for the investigation is inspecting the rate of system acceptance by the customer. This investigation includes the operation of training the customer to utilize the system much more efficiently. The customer should not be frightened by the system, rather the customer should feel its necessity. The rate of acceptance by the customer relies only on the methods, which are deployed to teach the customer regarding the system in order to make it familiar to them. Their rate of confidence should also be raised so as to enable them to make few constructive criticisms as the customer is the one who is the final utilizer of the system.

### **3.8.3 Economical Practicability:**

This investigation is done to inspect the economic effects that the devised system would be having in the firm. The quantum of money which the firm could invest into R&D- Research and Development activities of the devised system is restricted. The spend pattern is made justifiable. Hence, the devised system is able to be well inside the budget range. This economical practicability was possible since we had utilized the technologies that are easily available. If the cost is not the constraint, then the personalised products could be bought.

### **3.9 SYSTEM REQUIREMENTS SPECIFICATION**

#### **3.9.1 Non-functional and functional requirements:**

The investigation of the necessity is a much crucial operation that encourages the success of the devised system or the software project that is to be investigated. Necessities can be commonly divided as two kinds like non-functional and functional necessities.

#### **3.9.2 Non-functional necessities:**

The Non-functional necessities are normally the quality restrictions that the system should fulfill depending upon the contract of the project. The extent or priority up to which these elements are applied are not the same and it differs from one project to another. It is also known as a non-behavioral necessity.

These necessities normally deal with the following problems:

- Transportability
- Flexibility
- Supportability
- Security
- Dependability
- Performance
- Recyclability
- Scalability

Typical samples for non-functional necessities are listed below.

- 1) The Mails must be delivered by consisting of a latency of not in excess of 12 hours whatsoever the considered task.

- 2) Site must open in 3 seconds of time whenever the simultaneous users are more than 10,000
- 3) The processing of every request must be carried out inside the time of 10 seconds.

### **3.9.3 Functional necessities:**

These necessities are specifically demanded by the customer as the fundamental facilities that the system is desired to provide. All of the usefulness is required to be essentially established onto the system as per the contract parts. These necessities are stated or represented in the input form that is needed to be provided to the system, the processes executed, and the output desired. They are normally the necessities expressed by the customer, which one could notice straightly in the end product, which is the exact opposite of the non-functional necessities.

Typical samples for functional necessities are listed below.

- 1) Shutdown of the system is done if a cyber-attack takes place.
- 2) User authentication is executed every time the customer gets logged into the system.
- 3) The validation Mail is delivered for the customer unless he/she submit their details for the initial occasion registration in a few program systems.

## CHAPTER 4

### EXPERIMENTAL DESIGN

#### 4.1 INDEPENDENT AND DEPENDENT VARIABLE

In mathematical modeling, statistical modeling, and experimental sciences, independent and dependent variables are the variables. Independent variables are not considered as being dependent on any other variable in the experiment at hand. Time, mass, space, fluid flow rate, density, and prior values of any observed value of interest (e.g., human population count) are few frequent independent variables used to predict future values (the dependent variable). Dependent variables, on the other hand, get their name as their values are studied in an experiment under the assumption or demand that they are dependent on the values of other variables according to some law or rule (e.g., a mathematical function).

The dependent variable is always the one whose variation is being investigated by changing inputs, often known as regressors in a statistical setting. Any variable in an experiment that can be given a value without affecting the value of any other variable is referred to as an independent variable. Models and experiments are used to investigate the effects of independent variables on dependent variables. Independent variables are also included for various purposes, such as to account for their potential confounding effect, even though their influence is not directly of interest.



An independent variable is that type of variable which we could control or change in an experimental investigation to search for its effects. The name “independent” is coined from its ability of being not influenced by any other variables considered in the study.

Independent variables are known by the following other names:

- Predictor variables
- Right-hand-side variables
- Explanatory variables

These names are particularly utilized in the field of statistics, where we could compute the extent up to which the change in the independent variable change could be detailed or could forecast the variations taking place in the dependent variable.

The independent variable (IV) is a characteristic of any psychological experiment, which could be controlled or varied by the researchers instead of other kinds of variables. For example, the examination of the independent variable is possible when we examine the effects of investigating on the test scores.

If you're having problems identifying an experiment's independent variables, consider the following questions:

- Is the variable one that the researchers are tinkering with?
- Are researchers attempting to determine how one variable affects another?
- Is the variable something that can't be modified but isn't affected by the experiment's other variables?

The impacts of the independent variable on other variables, known as dependent variables, are of interest to researchers (DV). The independent variable is

one that the researchers either control or that already exists but is unaffected by other factors.

A dependent variable is a type of variable that varies because of the controlling of the independent variables. This variable is the outcome that we expect to compute and this variable will rely on the utilized independent variable.

In the field of statistics, these dependent variables are known by the following names:

- Responsive variables
- Left-hand-side variables
- Outcome variables

The dependent variable is what we report once the controlling of the independent variable is completed. We utilize this computational data to inspect up to what extent and whether the utilized independent variable impacts the dependent variable by the conduct of various statistical investigations.

We could infer from the discovery that we could compute the degree up to which the variation in an independent variable can influence the variation in the dependent variable. We could even forecast how much the dependent variable would vary because of the variation taking place in an independent variable.

The dependent variable will be tested or examined in any experimental set up. For instance, in an investigation aiming for tutoring of effect test scores, the dependent variable would be the test scores achieved by the participants, as we only compute them.

Likewise for a psychology experiment, the researchers investigate how the variations in one particular variable (the independent variable) is able to influence the dependent variable. Controlling the independent variables and examining its impact on the dependent variables would permit various researchers to infer various conclusions regarding the cause-and-effect associations.

## 4.2 EMPIRICAL DATA COLLECTION

The term empirical refers to data collecting that is based on evidence gathered through observation, experience, or the use of calibrated scientific instruments.

Empirical analysis is a method of studying and interpreting data that is based on evidence. Rather than ideas and conceptions, the empirical approach relies on real-world facts, measures, and outcomes. Empiricism is the belief that knowledge is gained largely via experience and the five senses.

During an experiment, empirical evidence is acquired. Analytical evidence is derived from previously stored data or historical data. In other words, rather than being obtained during a single experiment or observation, it was gathered through investigation of previous data.

In the world today, the word empirical refers to the gathering of data by having proof that is gathered through experience or observation or by deploying calibrated scientific devices. Thus, we could infer that various conclusion could be drawn only based on the examination or observation and through the experiments that we conduct to collect data.

## CHAPTER 5

### WORKING AND ANALYSIS

#### 5.1 DATASET

jm1.csv: Dataset collected from Kaggle .

- JM1 is a C-based real-time forecast ground technology. To make forecasts, it uses simulations.
- Halstead has source code extractors, and McCabe's data is being used. Those characteristics have been created in the 1970s in order to precisely characterise code features associated with software quality.
- Number of instances: 10,885.
- Each observation consists of 22 features.

cm1.csv: Dataset collected from Kaggle

- CM1 is a C-based NASA spacecraft instrument.
- Halstead contains source code extractors, and McCabe's data is used. These traits were created in the 1970s to scientifically characterise code attributes related to software quality. The definition of affiliation is a point of contention.
- Number of instances: 328.

- Each observation consists of 38 features.

mw1.csv: Dataset collected from Kaggle

- MW1 is written in C.
- Number of instances: 254.
- Each observation consists of 38 features.

## **5.2 DATASET PRE-PROCESSING**

- The preprocessing includes imputing null values, removing and cleaning unwanted data available in the dataset.
- If the categorization categories are not roughly equally represented, the dataset is unbalanced. One method for addressing this issue is to oversample the samples in the outvoted class.

## **5.3 FEATURE SELECTION**

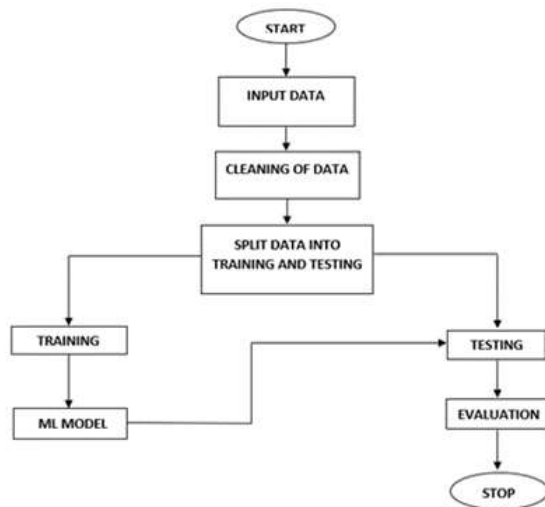
Feature Selection is performed with the help of the most extensively utilized technique known as Synthetic Minority Oversampling Technique (SMOTE).

SMOTE:

- Choose data from the minority class at random.
- To construct a synthetic sample for the outvoted class, compute the variance by a random range from 0 to 1.
- Repeat this until you reach the required outvoted class proportion.

## 5.4 CLASSIFICATION AND ENSEMBLE TECHNIQUES

- Once the dataset is prepared and pre-processed, we upload our dataset into the model.
- After uploading data, we perform necessary feature engineering techniques on the considered dataset. In this step, we remove noise from the data and divide the dataset into two different types of data such as train and test data.
- With the help of replace technique we will encode the dependent variable in order to move before building a model.
- In the train process, we build our proposed algorithm with the help powerful ML techniques with the help of Sklearn, CatBoost, LightGBM, extra trees classifier modules and the hybrid model is built using stacking classifier.
- The proposed architecture is based on machine learning data mining architecture. The proposed model is built using the methods involved in building a machine learning model.
- Here the proposed model mainly classifies whether software is defective or not.



not.

Fig. 5.1 Proposed Methodology

## 5.5 ML TECHNIQUES

An artificial intelligence (AI) technique which enables the computers to learn automatically without any obvious programming or any such interference is known as Machine Learning. It is a technique which is involved with creating computer programmes for adapting to new data. Let's look at the brief explanation about the method of utilizing Python for creating a basic form of machine learning algorithm and the basic concepts of machine learning.

Machine learning is one of the fast-emerging technologies which enables the computers to study and understand automatically based on the previous data entered. It utilizes different types of algorithms for creating the mathematical models that help to predict the upcoming data based on the previously entered data. It is applied to a wide range of applications like speech recognition, recommender systems, filtering emails, identifying images, auto-tagging in Facebook, etc.

Through using sample historical data which is also referred to as training data, a mathematical model is created by the machine learning algorithms. The necessary predictions or judgements are made using the generated mathematical models without any external programming being done. Since machine learning makes use of certain algorithms which learn automatically from the previously entered data, the more information given as input determines the performance that is received as output.

Machine learning is normally categorized into three different categories like supervised learning, unsupervised learning and reinforcement learning. Various concepts like clustering approaches, classification models, regression models, hidden Markov models, etc. comes under the hood of machine learning.

### 5.5.1 Cat Boost

CatBoost is an extremely high tree-based gradients boosting program. CatBoost is a technique for enhancing decision trees using gradients. Yandex researchers and developers built it, and this is utilized by Yandex and some other

companies like Careem taxi, CERN and Cloudflare for performing different jobs like self-driving the vehicles, personal assistance, weather forecasting, recommendation systems, search, etc. CatBoost is an open-source program and anybody can make use of it. Since it is open-source, anybody can use it. Putting Cat Boost & LightGBM to the test.

CatBoost comes out on top in the benchmark, which is fantastic. When it comes to datasets with a lot of categorical variables, however, this increase is considerable and clear.

CatBoost is an algorithm for gradient boosting in decision trees. It is developed by Yandex researchers and engineers, and is used for search, recommendation systems, personal assistant, self-driving cars, weather prediction and many other tasks at Yandex and other companies, including Careem taxi, Cloudflare, CERN.

The data from the machine learning model must be subjected to data preparation before subjecting the model to training. The important steps involved in the data preparation process are mentioned below:

- The survived column is removed initially which normally acts as the target variable.
- The data is then divided by forming two data frames such as x and y, among which one possess the target variable whereas other possess the valuable features for the model.
- Then the 'Pclass' column is converted to string data form. Then the null values existing within the features are filled to eliminate the null values.

Since CatBoost algorithm is a great machine learning algorithm especially for the categorical features, a list of column indices possessing the categorical data are generated by creating two helper functions. Then all the columns are converted to the category data forms.



While training time may be greater The Yandex benchmark reveals that prediction time is 13–16 seconds, which is faster than previous GBDT implementations. Times quicker than the other libraries. The default settings in CatBoost are better GBDT algorithms, which is fantastic news for newbies looking for a ready-to-use model to get starting with tree ensembles or Kaggle.



Fig. 5.2 Catboost Algorithm

Tournaments. CatBoost's other notable innovations include features interactivity, objects significance, and snapshots. Support CatBoost provides ranking right out of the box, in addition to classification and regression.

### 5.5.2 Extra Trees Classifier Algorithm

Extra Trees Classifier seems to be a collective approach that uses tree structure as its foundation and combines other decision tree algorithms like random forest and bootstrap aggregation (bagging). Extra Trees Classifier when acting as a Random Forest, randomizes specific decisions as well as portions of data can prevent overlearning as well as over-fitting from the data. When it comes to decision trees, it learns from only one trail of decisions and a single decision tree frequently fits the data it is learning from. Predictions based on a single decision tree are seldom reliable when applied to fresh data. Building many trees (n estimators) in a random forest model reduces the risk of over fitting and substitute for drawing notes (i.e., a bootstrapped sample) and nodes are split based on the best splitting or separation among a random group of characteristics chosen at each node. Extra Trees is similar to Random Forest because it generates several trees as well as divides nodes with

random feature subsets, but somehow it differs in two major ways: it does not bootstrap observations and nodes are divided on random splits rather than optimal splits. One is, as defaults, it constructs multiple decision trees with bootstrap is set to be False that implies it samples has no substitution and the second is, these nodes are divided randomly splits across a randomly selected subset of the characteristics chosen at each node. Randomness in Extra Trees is generated through random splits of all observations rather than by bootstrapping of data.

Huge quantity of unpruned decision trees is created from the training dataset and the functioning of Extra Trees algorithm takes place through this decision trees. In the regression phase, the predictions are done through taking mean for the prediction of decision trees and in the classification phase, the process of majority voting is utilized.

Extra Trees algorithm is capable of sampling the features in a random manner at each separating spot of a decision tree just as such in the case of random forest. But unlike the random forest classifier, the extra trees algorithm makes use of a greedy algorithm for gathering an ideal separating spot and thus gathers the separating spot in random manner.

Normally there are three different types of hyperparameters that helps in the tuning of Extra Trees algorithm. They are:

- The number of decision trees present in the collective group,
- The number of input features that are selected in a random manner and considered for each separating spot and
- The minimum number of samples in a node that are needed for creating a new separating spot.

There are few collective techniques responsible for the creating extra trees classifier. They are ordered from high variance to low variance in this section.

### 5.5.3 Gradient Boosting

Gradient boosting technique is one among the most effective methods in machine learning. Normally mistakes tend to occur even in the machine learning algorithm and thus occurred mistakes are categorized into two different types like variance errors and bias errors. It has been the most popular boosting procedure utilized for minimizing the bias error occurring in a model. The gradient boosting procedure, unlike the AdaBoosting approach, doesn't really allow us to choose any base estimator. The base estimator of the Gradient Boost method is constant, i.e., Decision Stump. We may use AdaBoost to modify the  $n$  estimator of the gradient boosting approach. Nevertheless, the computation default value becomes 100, if the number of  $n$  estimators are not supplied. The gradient boosting method can predict either continuous or categorical target variables. Their cost function is Mean Square Error (MSE) if it is utilized as a regressor, and Log loss if it is used as a classifier. So, look at an example to see how the Gradient Boosting Algorithm works. Age is the target variable in the following example, whereas LikesExercising, GotoGym, and DrivesCar are independent variables. Gradient Boosting Regressor is utilized in this case since the target variable is continuous. Let us now determine the estimator-2. Despite AdaBoost, the Gradient Boosting technique uses the first estimator's residues ( $age_i - \mu$ ) as root nodes, as seen below. Assume that another dependent variable is utilized for predictions using this estimate. As a result, the records with False GotoGym.

### 5.5.4 Light GBM

LightGBM is really a decision tree-based gradient boosting system which boosts effectiveness of the model while consuming less memory.

It uses two recent techniques: Gradient-based One Side Sampler exclusivity and Feature Bundling, that overcome the shortcomings of the histogram-based technique used in most GBDT frameworks. The LightGBM Algorithm's qualities are

generated by the two techniques of GOSS & EFB, which are discussed here. Researchers try hard to make the system function successfully and to offer it a competitive advantage over rival gradient boosting decision tree architectures.

#### Gradient-based LightGBM One-Side Sampling Technique:

Distinct data examples play different roles in the computation of information gain. For scenarios with larger gradients, the supervised learning will be higher. GOSS keeps examples with substantial gradients and removes cases with modest gradient at randomized to ensure the informativeness gain estimation. So, when quantity of mutual information varies widely, this approach can provide a more accurate gain estimation than uniform random sampling at same desired sampling frequency.

### **5.5.5 Hybrid Model**

Ensemble classifiers are commonly used to improve classification task accuracy. Throughout the current study, a hybrid model based on stack-based ensemble classifiers is used to determine whether or not software is faulty. To improve classification accuracy, the feature vector added with the resultant of a basic classifier for creating an enhanced feature set, along with the employment of collective model that is based on the hybrid stack for enhancing the set of features. CatBoost, gradient boosting, extra trees classifier, and LightGBM classifier are utilized to develop a stacking-based ensemble classifier using LGBM as Meta learners. The suggested model is implemented using our data set JM1. Python's Mlxtend package is used to build the stack-based ensemble of classifiers. The experimental findings exhibit that the proposed hybrid model and extra tree classifier with the feature set gives better performance and improved accuracy, on comparing with other existing machine learning techniques.

## 5.6 PERFORMANCE EVALUATION MEASURE

### 5.6.1 Accuracy:

The classification models are basically evaluated in terms of the accuracy parameter. Irrespective of definitions, let's say that accuracy refers to percentage of exact predictions made by the proposed model. The correct definition of accuracy is given as the number of exact predictions equal to that of total accuracy number of predictions.

The calculation of accuracy is done by dividing the total number of samples by the number of valid predictions (the appropriate diagonal in the matrix).

### 5.6.2 Precision:

The binary classification issues occurring in the proposed model does not limit or affect the precision parameter. The precision calculation for the imbalanced classification issue having more than two classes is done through the expression where the sum of true positives among all classes divided by the sum of true positives and false positives among all classes. The expression for precision calculation is given below:

$$\text{Precision} = \frac{\sum_{c \in C} \text{TruePositives}_c}{\sum_{c \in C} (\text{TruePositives}_c + \text{FalsePositives}_c)}$$

### 5.6.3 Recall:

Similar to precision, the binary classification issues occurring in the proposed model does not limit or affect the recall parameter. And the calculation of recall value for the imbalanced classification issue having more than two classes is also done through the similar expression as before where the sum of true positives among all classes divided by the sum of true positives and false negatives among all

classes. The only difference exists in the calculation of false negatives among all classes instead of the calculation of false positives among all classes as in the case of precision. The expression for recall calculation is given below:

$$\text{Recall} = \frac{\sum_{c \text{ in } C} \text{TruePositives}_c}{\sum_{c \text{ in } C} (\text{TruePositives}_c + \text{FalseNegatives}_c)}$$

Pd/recall/TP rate	$\frac{TP}{TP + FN}$
Pf/FP rate	$\frac{FP}{FP + TN}$
precision	$\frac{TP}{TP + FP}$
F-measure	$\frac{2 \times Pd \times \text{precision}}{Pd + \text{precision}} = \frac{2TP}{2TP + FP + FN}$
G-measure	$\frac{2 \times Pd \times (1 - Pf)}{Pd + (1 - Pf)}$
balance	$1 - \frac{\sqrt{(0 - Pf)^2 + (1 - Pd)^2}}{\sqrt{2}}$
accuracy	$\frac{TP + TN}{TP + FP + FN + TN}$
G-mean	$\sqrt{Pd \times (1 - Pf)}$
MCC	$\frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP) \times (TP + FN) \times (TN + FP) \times (TN + FN)}}$
AUC	the area under the ROC curve

Figure 5.3 Basic Indicators for defect Prediction

## 5.7 PROBABILITY SCORE

Statistical analysis consists of numerous advantages to both the organizations as well as the individuals and hence it is considered to be a great boon to the mankind. Some of the main reasons considered before investing in the statistical analysis are mentioned below:

- The decisions can be made easily through determining the sales profits either in yearly or quarterly or monthly basis.
- The wrong decisions can be corrected and helps the sales persons to be informed all time.
- The issue or the main reason of failure can be identified and thus helps to correct those failures. Let's take an instance where the wasteful expenses can be eliminated by identifying the reason for the increase in total costs.
- An efficient sales and marketing strategy can be made through conducting proper market analysis.
- The efficiency of various methods can be improved.

### **5.7.1 Probability Analysis Software**

Statistical analysis can't be made by everyone as it is a very complex statistical calculations and normally tend to be very costly and time-consuming process. Nowadays, numerous companies implement statistical software as it is considered to be a very significant tool for performing data analysis. Many complex calculations are made, new trends and patterns are identified, and various graphs, charts and tables are created accurately by the software within few minutes as it utilizes the concept of machine learning and artificial intelligence in it.

In Statistical we plot each algorithm got different accuracy value so show the plot in the build the bar plot in the shoes below figure

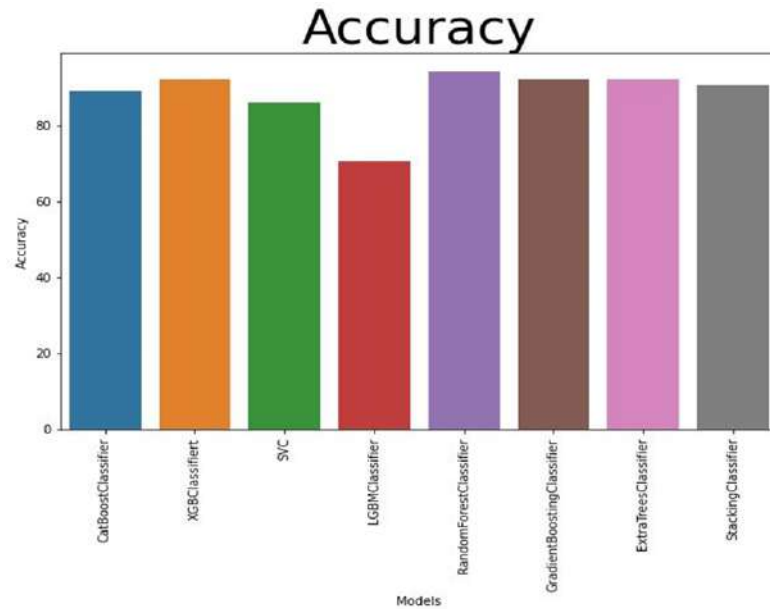


Figure 5.4 Probability Score

### 5.7.2 Probability Analysis Process

The statistical analysis process is performed by following the 5 significant steps mentioned below:

- Step 1: The nature of the data that is to be analysed are identified and described initially.
- Step 2: A relationship is established among the data analysed and the sample population to which it belongs to.
- Step 3: A model is created for clearly presenting and summarizing the relationship among the data and population.
- Step 4: The validity of the model is then verified.
- Step 5: The future events and trends likely to occur are predicted using the predictive analysis.



## CHAPTER 6

### RESULTS

In this section, we'll discuss the results obtained by employing the forementioned technique, as well as compare the accuracy, precision, and recall of each model. Compare accuracy, precision and recall for all machine learning technique we used.

**TABLE 6.1.** Results Obtained for jm1 Dataset

<b>Model</b>	<b>Accuracy (%)</b>	<b>Precision (%)</b>	<b>Recall (%)</b>
CatBoost Classifier	86.35	94.18	81.51
LGBM Classifier	86.31	94.03	81.54
Gradient Boosting Classifier	80.77	89.64	76.24
Stacking Classifier	87.85	86.99	88.60
Extra Trees Classifier	87.40	86.20	88.40

---

**TABLE 6.2.** Results Obtained for cm1 Dataset

---

Model	Accuracy (%)	Precision (%)	Recall (%)
CatBoost Classifier	87.04	97.92	88.68
LGBM Classifier	87.96	96.88	90.29
Gradient Boosting Classifier	87.04	95.83	90.20
Stacking Classifier	86.11	95.83	89.32
Extra Trees Classifier	85.19	95.83	88.46

---

**TABLE 6.3.** Results Obtained for mw1 Dataset

Model	Accur acy (%)	Precisio n (%)	Recall (%)
CatBoost Classifier	85.71	100.00	85.71
LGBM Classifier	85.71	98.61	86.59
Gradient Boosting Classifier	86.90	98.61	87.65
Stacking Classifier	85.71	98.61	86.59
Extra Trees Classifier	85.71	100.00	85.71

Table 6.4. Comparison between other's work.

Author	Algorithms Used
Guisheng Fan, Xuyang Diao, and Huiqun Yu [1]	<ul style="list-style-type: none"> <li>· Random Forest</li> <li>· Random Forest method with hidden features learned by Restricted Boltzmann Machine</li> <li>· Random Forest method with hidden features generated by Deep Belief Network</li> <li>· Convolutional Neural Network</li> <li>· Recurrent Neural Network</li> <li>· Attention-Based Recurrent Neural Network</li> </ul>
Ebubeogu Amarachukwu Felix, and Sai Peck Lee [2]	<ul style="list-style-type: none"> <li>· Naïve Bayes</li> <li>· Logistic Regression</li> <li>· Neural Network</li> <li>· K-nearest neighbor (KNN)</li> <li>· Support Vector Machine</li> <li>· Random Forest</li> </ul>
Zhou Xu, Jin Liu, Xiapu Luo, Zijiang Yang, Yifeng Zhang, Peipei Yuan, Yutian Tang, and Tao Zhang [3]	<ul style="list-style-type: none"> <li>· Kernel Principal Component Analysis (KPCA)</li> <li>· Weighted <a href="#">Extreme Learning Machine</a> (WELM)</li> </ul>

Zhou Xu, Shuai Pang, Tao Zhang, Xia-Pu Luo, Jin Liu, Yu-Tian Tang, Xiao Yu, and Lei Xue [4]	· Logistic Regression
Ruchika Malhotra, Shine Kamal [5]	· Decision Tree · Random Forest · Naïve Bayes · AdaBoost · Bagging
Ruchika Malhotra, Shine Kamal [6]	· Random Forest · J48 · Naïve Bayes
This Study	· CatBoost Classifier · XGB Classifier · SVM · LGBM Classifier · Random Forest Classifier

---

Table 6.4 compares the models/algorithms utilized in this study with those utilized in previously existing research works. We employed the Synthetic Minority Oversampling Technique, which is the most widely used technique for imbalanced datasets, because all of the datasets are imbalanced. The Stacking Classifier algorithm is employed in the work, which is a collaborative learning technique that uses a meta-classifier to merge numerous classification models. Furthermore, the Extra Trees Classifier algorithm is also employed, which is a form of collaborative learning method that gathers the outcomes of several de-correlated decision trees

collected in a "forest" to get a classification result. In comparison to all other algorithms, the Extra Trees Classifier algorithm considerably improves performance.

Also Flask was used to create the web application. Flask is a Python-based microweb toolkit. It is known as a microframework because it does not require the usage of any specific tools or libraries. It includes a database layer of abstraction, data validation, as well as other common-task-accomplishing components that rely on third-party libraries.



Figure 6.1 Upload Dataset.

The above images represent the loading procedure of required dataset in order to implement the project.

ID	File	Size	Type	Status	Created At	Updated At	Deleted At	ID Code	ID Comment	ID Bank	ID Category	ID Comment	Unit	Op	Unit	Op	Unit	Op	Unit	Op
11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11
12	12	12	12	12	12	12	12	12	12	12	12	12	12	12	12	12	12	12	12	12
13	13	13	13	13	13	13	13	13	13	13	13	13	13	13	13	13	13	13	13	13
14	14	14	14	14	14	14	14	14	14	14	14	14	14	14	14	14	14	14	14	14
15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15
16	16	16	16	16	16	16	16	16	16	16	16	16	16	16	16	16	16	16	16	16
17	17	17	17	17	17	17	17	17	17	17	17	17	17	17	17	17	17	17	17	17
18	18	18	18	18	18	18	18	18	18	18	18	18	18	18	18	18	18	18	18	18
19	19	19	19	19	19	19	19	19	19	19	19	19	19	19	19	19	19	19	19	19
20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20
21	21	21	21	21	21	21	21	21	21	21	21	21	21	21	21	21	21	21	21	21
22	22	22	22	22	22	22	22	22	22	22	22	22	22	22	22	22	22	22	22	22
23	23	23	23	23	23	23	23	23	23	23	23	23	23	23	23	23	23	23	23	23
24	24	24	24	24	24	24	24	24	24	24	24	24	24	24	24	24	24	24	24	24
25	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25
26	26	26	26	26	26	26	26	26	26	26	26	26	26	26	26	26	26	26	26	26
27	27	27	27	27	27	27	27	27	27	27	27	27	27	27	27	27	27	27	27	27
28	28	28	28	28	28	28	28	28	28	28	28	28	28	28	28	28	28	28	28	28
29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29
30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30
31	31	31	31	31	31	31	31	31	31	31	31	31	31	31	31	31	31	31	31	31
32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32
33	33	33	33	33	33	33	33	33	33	33	33	33	33	33	33	33	33	33	33	33
34	34	34	34	34	34	34	34	34	34	34	34	34	34	34	34	34	34	34	34	34
35	35	35	35	35	35	35	35	35	35	35	35	35	35	35	35	35	35	35	35	35
36	36	36	36	36	36	36	36	36	36	36	36	36	36	36	36	36	36	36	36	36
37	37	37	37	37	37	37	37	37	37	37	37	37	37	37	37	37	37	37	37	37
38	38	38	38	38	38	38	38	38	38	38	38	38	38	38	38	38	38	38	38	38
39	39	39	39	39	39	39	39	39	39	39	39	39	39	39	39	39	39	39	39	39
40	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40

Figure 6.2. View Dataset.

The above shown figure is showing us a sample data which was uploaded to the system.



Figure 6.3. Preprocessing

The above shown figure is the place where preprocessing was implemented.

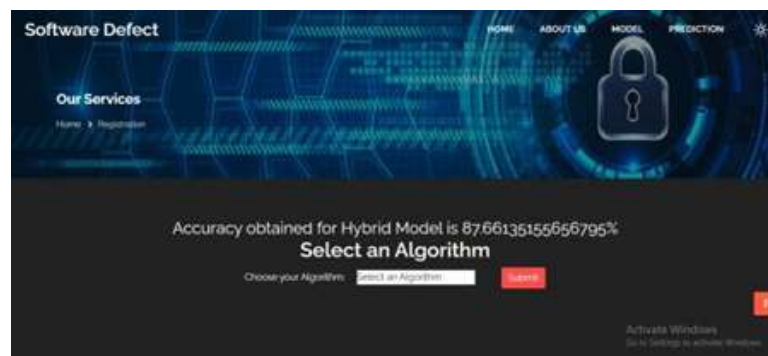


Figure 6.4 Model trained with Hybrid Model.

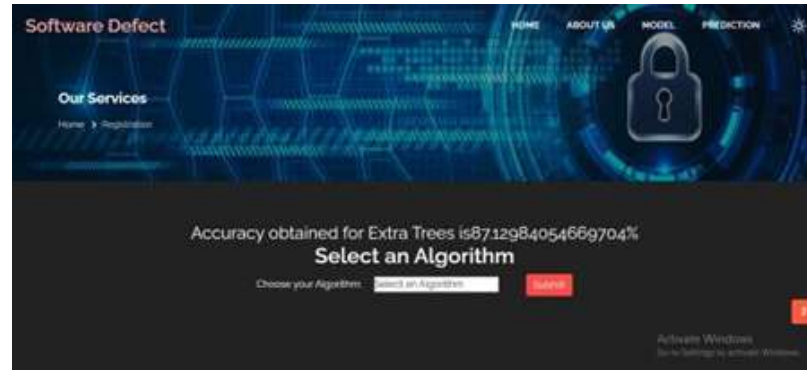


Figure 6.5 Model Trained with Extra Trees Classifier



Figure 6.6 Predicting Defected Software

The above shown figure is taking input fields provided by the user to detect whether a software is defected or not depending on the attributes.



## CHAPTER 7

### CONCLUSIONS AND FUTURE SCOPE

Software faults can degrade the quality of software, causing problems for both customers and developers. As software designs and technology have become more intricate, manual programme identification has become a challenging and time-consuming task. As a result, autonomous software detection has been a hotspot for industrial research in recent years. The purpose of this research is to use data-mining techniques to predict software flaws. Furthermore, this problem has grown in importance as a research area, with numerous approaches being explored to increase the effectiveness of detecting software flaws or anticipating defects in some way.

In this research study, machine learning is utilized to resolve the issues caused. Using three datasets from the NASA Promise dataset repository, we examine the performance of state-of-the-art machine learning approaches. We develop predictions using a variety of algorithms and can detect poor software. This is accomplished in a user-friendly environment using Python programming and machine learning techniques such as CatBoost, Gradient Boosting, LightGBM, Random Forest, and MIXtend to construct a hybrid model and extra trees classifier, both of which outperform with higher accuracy.

In Hybrid Model, all three Accuracy, Precision, and Recall values had attained maximum equal values. Values of these three performance measures indicate how well the proposed model performs.

If same values are attained for all, it suggests that the proposed prediction method is correct and accurate.

The use of many datasets aids in the detection of software flaws. If the number of datasets is increased, the results may be improved. Other strategies can be compared as well. The most popular and broadly utilized techniques were taken into consideration in this study. In the future, new methods are likely to be proven and used in deeper investigation. This is an area where there is still a lot of opportunity for improvement. We can think about a few different ways that advanced deep learning algorithms are used, as well as the necessity for more data collection by academics.

Additional experimental tests using other datasets would be one topic of future investigation. These datasets would be obtained from open repositories or software companies. The second topic of future research would be to undertake an experiment combining machine learning and deep learning approaches. Another possible study direction is to combine already existing features to generate new qualities. Finally, doing a case study using a variety of software quality datasets acquired from real-world projects of various sizes of software companies would be beneficial. Another method for enhancing the accuracy of the prediction model to include more software measures in the learning process.

**References:**

- [1]. G. Fan, X. Diao, H. Yu, K. Yang, and L. Chen, Software Defect Prediction via Attention-Based Recurrent Neural Network, Hindawi, 2019.
- [2]. E. A. Felix, and S. P. Lee, Predicting the number of defects in a new software version, PLOS ONE (2020).
- [3]. Z. Xu, J. Liu, X. Luo, Z. Yang, Y. Zhang, P. Yuan, Y. Tang, and T. Zhang, Inform. Softw. Technol. **106**, 182–200 (2019).
- [4]. Z. Xu, J. Xuan, J. Liu, and X. Cui, Cross project defect prediction via balanced distribution adaptation based transfer learning. Journal of Computer Science and Technology 34(5), 1039–1062 (2019).
- [5]. R. Malhotra, and S. Kamal, Neurocomputing 343, 120–140 (2019).
- [6]. R. Malhotra, and S. Kamal, Tool to handle imbalancing problem in software defect prediction using oversampling methods. In: 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI), IEEE, 2017, pp. 906–912.
- [7]. R. Malhotra, and J. Jain, Int. J. Softw. Eng. Knowl. Eng. 31, 193–215 (2021).
- [8]. D. Ryu, J. Baik, Effective multi-objective naïve Bayes learning for cross-project defect prediction. Appl. Soft Comput. 49, 1062 (2016).
- [9]. C. Shan, B. Chen, C. Hu, J. Xue, and N. Li, Software defect prediction model based on LLE and SVM. In: Proceedings of the Communications Security Conference (CSC '14), 2014, pp. 1–5.
- [10]. Z. R. Yang, A novel radial basis function neural network for discriminant analysis. IEEE Trans. Neural Netw. 17(3), 604–612 (2006).
- [11]. K. Han, J. -H. Cao, S. -H. Chen, and W. -W. Liu, A software reliability prediction method based on software development process. In: 2013 International Conference on Quality, Reliability, Risk, Maintenance, and Safety Engineering (QR2MSE). IEEE, 2013, pp. 280–283.
- [12]. K. Han, J. -H. Cao, S. -H. Chen and W. -W. Liu, “A software reliability prediction method based on software development process,” in *2013 International Conference on Quality, Reliability, Risk, Maintenance, and Safety Engineering (QR2MSE)* (IEEE, 2013), pp. 280–283.

- [13]. Y. Zhou, Y. Yang, H. Lu, et al., *ACM Trans. Softw. Eng. Methodol.* **27**, 1–51 (2018).
- [14]. M. H. Halstead, “Elements of software science,” in *Operating and Programming Systems Series*, vol. 2 (Elsevier, Amsterdam, Netherlands, 1977).
- [15]. T. J. McCabe, *IEEE Trans. Softw. Eng.* **SE-2**, 308–320 (1976).
- [16]. M. Jureczko and D. Spinellis, “Using object-oriented design metrics to predict software defects,” in *Models and Methods of System Dependability* (Oficyna Wydawnicza Politechniki Wrocławskiej, Wrocław, Poland, 2010).
- [17]. F. Yamaguchi, M. Lottmann and K. Rieck, “Generalized vulnerability extrapolation using abstract syntax trees,” in *Proceedings of the 28th Annual Computer Security Applications Conference* (ACM, Orlando, FL, USA, 2012), pp. 359–368.
- [18]. T. Mikolov, M. Karafiát, L. Burget, J. Černocký and S. Khudanpur, “Recurrent neural network based language model,” in *Proceedings of the Eleventh Annual Conference of the International Speech Communication Association, Makuhari, Japan* (September 2010).
- [19]. D. M. Powers, *J. Mach. Learn. Technol.* **2**, pp. 37–63 (2011).
- [20]. J. M. Lobo, A. Jiménez-Valverde and R. Real, *Glob. Ecol. Biogeogr.* **17**, pp. 145–151 (2008).
- [21]. N. Nagappan and T. Ball, “Using software dependencies and churn metrics to predict field failures: an empirical case study,” in *Proceedings of the First International Symposium on Empirical Software Engineering and Measurement (ESEM 2007)* (IEEE, Madrid, Spain, September 2007), pp. 364–373.
- [22]. R. Moser, W. Pedrycz and G. Succi, “A comparative analysis of the efficiency of change metrics and static code attributes for defect prediction,” in *Proceedings of the 30th International Conference on Software Engineering* (ACM, Leipzig, Germany, May 2008), pp. 181–190.
- [23]. Lessmann, S., Baesens, B., Mues, C., & Pietsch, S. (2008). Benchmarking classification models for software defect prediction: A proposed framework and novel findings. *IEEE Transactions on Software Engineering*, *34*(4), 485-496.
- [24]. Shepperd, M., Song, Q., Sun, Z., & Mair, C. (2013). Data quality: Some comments on the nasa software defect datasets. *IEEE Transactions on Software Engineering*, *39* (9), 1208-1215.

## LIST OF PUBLICATIONS

[1]Nishthaa,Ruchika Malhotra , “Identification Of Defects in a Software using Machine Learning” accepted at **Parul University International Conference on Engineering and Technology**

Indexed by Scopus

Paper Id- 177

**Abstract-** Now a days, the usage of software among the individuals have increased a lot when compared with earlier days. As the technology is increasing rapidly, the evolution of artificial intelligence has taken place. In the underlying time of the Software Development Life Cycle (SDLC), Software Defect Prediction (SDP) remains a basic and important duty. There have been a lot of trials going on in the previous few days to detect the quality of the program, which leads to giving the software a guaranteed quality. SDP indicates the probability of software shortcoming at a beginning phase of software development process and henceforth it will be more straightforward to distinguish and address them and furthermore diminish issues that would happen at later stages. This will work on the general nature of the software item. In the recent years, a few machine learning(ML) algorithms have utilized instances of defective and non-defective modules to construct prediction models. Software metric have been utilized as input to these ML algorithms to address the software modules. Here in this project for the detection of defects in software we are using ML algorithms namely CatBoost, XGBoost (Extreme Gradient Boosting), Support Vector Machine (SVM), Light Gradient Boosting Machine (LightGBM) and Random Forest (RF).We perform these on Promise repository dataset used for SDP. Most of the software defect datasets have imbalanced output results (count of defect and not defect too much vary), so we perform SMOTE technique to overcome this problem. From our research, CatBoost and RF Classifier are best algorithms for detecting software defects. We will discuss more about our dataset, pre-processing techniques, modeling, evaluation

and model comparison in this paper, and we also compare our work with previous other author works.

[2]Nishthaa,Ruchika Malhotra , “Software Defect Identification with Hybrid and Extra Tree Models”, accepted at **International Conference on Emerging Trends in IoT and Computing Technologies-2022**

Indexed by Scopus

Paper id-93

**Abstract-** The end users who are using the software and its products is vastly increased when compared to the earlier days. As we are seeing that the technology has evolved a lot and it has delivered an extraordinary technology named artificial intelligence. Identifying defects in a software in the current time can be held with Software Development Life Cycle (SDLC) and it stays a fundamental and crucial task. In the present days, a few instances of defective and non-defective modules are used to construct prediction models which utilize machine learning(ML) algorithms. To address the software modules, software metrics were used as input to these ML algorithms. In order to detect the defects in a software, few powerful ML algorithms are implemented and in existing system the algorithm named CatBoost & Random Forest (RF) gives an adequate accuracy. But we need to identify the defects in a software using ML algorithms with better model which must give some improved performance when compared with RF and CatBoost. So here in this paper we are using extra trees classifier and hybrid model to identify the defects in a software.