# SOFTWARE DEFECT PREDICTION USING MACHINE LEARNING TECHNIQUES

A DISSERTATION

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE AWARD OF DEGREE
OF

MASTER OF TECHNOLOGY
IN
**SOFTWARE ENGINEERING**

Submitted by:

**KISHWAR KHAN**
**(2K16/SWE/08)**

Under the supervision of:

**DR. RUCHIKA MALHOTRA**
**(ASSOCIATE PROFESSOR, CSE)**
**Department of CSE, DTU**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi – 110042

JULY 2018

DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi – 110042

## CANDIDATE'S DECLARATION

I, KISHWAR KHAN, 2K16/SWE/08 a student of M.TECH (Software Engineering) declare that the project Dissertation titled "Software Defect Prediction Using Machine Learning Techniques" which is submitted by me to Department of Computer Science and 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 any Degree, Diploma, Fellowship or other similar title or recognition.

Place: DTU, Delhi KISHWAR KHAN

Date: (2K16/SWE/08)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi – 110042

# **CERTIFICATE**

I, hereby certify that the Project titled "Software Defect Prediction Using Machine Learning Techniques" submitted By KISHWAR KHAN, Roll number: 2K16/SWE/08, Department of Computer Science and 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 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: DTU, Delhi
Date:

**Dr. RUCHIKA MALHOTRA**
(Associate Professor, CSE, DTU)
Supervisor

# ACKNOWLEDGEMENT

First of all I would like to thank the Almighty, who has always guided me to work on the right path of the life. My greatest thanks are to my parents who bestowed ability and strength in me to complete this work.

I owe a profound gratitude to my project guide Dr. Ruchika Malhotra Ma’am who has been a constant source of inspiration to me throughout the period of this project. It was her competent guidance, constant encouragement and critical evaluation that helped me to develop a new insight into my project. Her calm, collected and professionally exemplary style of handling situations not only steered me through every problem, but also helped me to grow as a matured person.

I am also thankful to her for trusting my capabilities to develop this project under her guidance.

**KISHWAR KHAN**
**M.TECH (SWE)**
**2K16/SWE/08**

# ABSTRACT

Software defect prediction is a process of classification which determines whether a software module is defective or not. A defect prediction model is a method where a set of independent variables (the predictors) are used to predict the value of a dependent variable (the defect-proneness of a class) using a machine learning classifier.

Innumerable studies are present in literature that studies the effect of dimensionality reduction on performance of models developed for Software Defect Prediction. It is said to improve certain models. Also, Software defect prediction is a costly activity and the problem relies in the fact that many feature-extraction methods based on traditional as well as novel like deep learning are there for dimensionality reduction. So, it becomes very difficult to choose any method based on its working, its pros and cons and its performance for dimensionality reduction. Thus, there arises a need of comparison study for feature extraction technique which exists earlier in literature.

This study aims to provide literature review on the previously existing feature reduction techniques in software defect prediction. The study helps software developers in identifying the commonly prevalent as well as novel feature extraction techniques, their characteristics and their performance in area of software defect prediction and guides the researchers in conducting future research. The comparison is performed on nine open-source software-systems written in Java using four mostly used feature extraction technique and a machine learning classifier. The model validation is performed by 10 fold cross validation method and the performance measure used is accuracy and ROC-AUC. Results of the study indicate that autoencoders is an effective method to reduce the dimensions of a dataset successfully.

# CONTENTS

| Titles | | Page No. |
|---|---|---|

# LIST OF TABLES

| Table No. | Title | Page No. |
|---|---|---|

# LIST OF FIGURES

| Figure No. | Title | Page No. |
|---|---|---|

# LIST OF ABBREVIATIONS

| | |
|---|---|
| QA | Quality Assurance |
| SDP | Software Defect Prediction |
| PCA | Principal Component Analysis |
| LDA | Linear Discriminant Analysis |
| AE | Autoencoders |
| ROC-AUC | Receiver Operating Characteristics-Area Under Curve |
| SVM | Support Vector Machine |
| KPCA | Kernel-Principal Component Analysis |
| CV | Cross validation |
| PC | Principal Component |

# CHAPTER 1

# INTRODUCTION

## 1.1 Overview

Quality assurance is an archetypal resource-impelled activity when requirements based on the time-to-market metrics of software delivery need to be met. In some software industry, it is analyzed that if there is some week based delay in software delivery it may cause some grave approximately (22-23%) loss for software having lifetime around 52 weeks [38]. Because of the expanded requirement for the quick arrival of programming to the market is a concerned issue for organizations in many divisions of programming markets. Despite the fact that it is basic to meet such a squeezing need, indiscreet quality confirmation can request specialized record. The negative impact of a faulty programming stature is regularly deadly in the association. Hence, quality confirmation ends up plainly hypercritical quickly before the product uncovers; in any case, at that stage, time and HR are normally inadequate for disposing of each torpid deformity by the due date. Designers or quality confirmation supervisors in this manner critically request procedure that successfully predicts absconds and empowers the use of best undertaking in settling them. Therefore, defect prediction in programming's and their discharges has been an energetic research territory in programming designing zone.

Software defect prediction is a process of classification which determines whether a software module is defective or not. It helps Quality Assurance (QA) teams to concentrate on their finite resources on the most defect susceptible software modules. In this, every software component is categorized by a class tag and numbers of metrics. The class tag indicates if this module is faulty [38].

We know that data mining and machine learning are prevailing days by day, a number of classification models have been introduced during the past decade. In spite of that, a problem that scares the modeling process is high-dimensionality of defect data, i.e., datasets with extreme features having irrelevant and redundant ones. Previous studies have exposed that the high-dimensionality issue can lead to high computational charge and deprivation of the accuracy of definite models [9], [10], [13]. Due to these causes, a range of feature reduction techniques was projected to improve this problem of high-dimensionality by removing extraneous and repeated features.

According to [39], with an increase in the dimensionality of data, there is an exponential increase in the amount of data needed to offer a reliable performance, this fact is termed as the 'curse of dimensionality' by Bellman when taking into account issues related to dynamic optimization. A well-liked undertaking to this issue of high-dimension datasets is to look for a projection of the data against a lesser amount of features, which conserve the inofrmation so far as possible.

To conquer this issue, it is essential to discover a manner to reduce the number of variables in consideration [1], [8], [39]. Two widely used methods are:

**(a) Feature selection**

This method [8] chooses a subset of actual features from available features, without whichever failure of valuable information**.** It directs the distinct job of exploring a subset of specified features that are helpful to resolve the domain issue.

Feature selection methods are classified into three classes [8] i.e. filter, wrapper, embedded / hybrid.

**(b) Feature extraction**

Feature extraction [24] is a technique to find novel features from specified features by employing some conversions to decrease complication and to provide an easy demonstration of each variable in feature space as a linear arrangement of input variables. It is more general technique than feature selection technique. PCA, ICA, LDA, and K-PCA are some of the approaches of feature extraction.

Feature extraction is also called second-order features [32]. If the second-order features are united linearly throughout feature extraction, it's linear feature extraction. Otherwise, it is called non-linear feature extraction. In this study, we are comparing both linear as well as non-linear technique.

## 1.2 Research Objective

In software defect prediction, when we think about dimension reduction methods specifically, PCA, K-PCA, and LDA are two widely used approaches of conventional methods while Auto-encoders is a novel technique of deep learning which be capable of effortlessly handling non-linear & big data [31]. Dimensionality reduction is a major issue in numerous real-world studies as data used for software defect prediction sometimes has high dimensions and transforming these data from high dimensions to low dimensions is crucial to boosting the effectiveness of prediction process [38]. A large number of machine learning methods in the past used linear transformation which is based on factorization projections or orthogonal projections for dimensionality reduction. These types of methods generally are not valuable for non-linear feature reduction other than that it can effectively solve many simple problems related to limited constraints data. Deep learning in contrast to that has effective visualization capacity and address data complex in nature.

## 1.3 Proposed Work

The main aim of this study is to carry out a comprehensive comparison of various types of feature extraction techniques like LDA PCA, Stacked Auto Encoders and K-PCA;  detailed study of existing methodologies for dimension reduction; Model development to evaluate the techniques with the open source datasets.

## 1.4 Thesis Organization

This thesis work is bifurcated into six different chapters. Starting with the *Chapter 1* gives the brief introduction about the issues discussed in this study. The chapter explains the need for and use of defect prediction models. It defines the defect-related terminologies explaining how they affect the software systems and human life. It also addresses the problem of ‘curse of dimensionality’. It also describes the motivation behind this study along with the research objectives.

*Chapter 2* sums up the related studies with respect to software defect prediction and dimensionality reduction. A lot of research has been carried out in defect prediction area in the context of feature reduction. This chapter summarizes the major contributions and findings of the previous studies. The literature survey conducted by the author in defect prediction finds out that high dimensional data is becoming a serious problem. Many studies [5], [9], [15], [13], [11] have been investigating in this field by using feature reduction techniques. Most of the studies use PCA and LDA for feature reduction while another method such as autoencoders is still unexplored in the area of defect prediction Furthermore, the related work describes the previous studies which have used procedural metrics and have applied various ML techniques for building models. *Chapter 3* describes the research methodology used in the experiment. It briefly discusses the data collection method, different datasets and the data preprocessing technique. Various feature extraction methods together with the detailed explanation of the method are described. The chapter further defines the performance metrics used to evaluate the prediction models and also briefly discusses the Friedman test which is used for finding the statistical significance of the techniques. The 10-fold cross validation method used for model evaluation is explained in this section. *Chapter 4* provides the details regarding the experimental design of the study. It describes the dependent, independent variables used to carry out the research. The chapter further defines the hypothesis formation on which the study is based and describes various tools used for implementation of the experiments. In *Chapter 5* the obtained results are stated and analyzed using statistical tests. We have performed an extensive comparison between various feature extraction methods using non-parametric test i.e. Friedman test. *Chapter 6* concludes the final outcome of the study. It states which method performed the best and guides the researchers to make use of novel feature extraction techniques to further improve the performance of defect prediction models. The chapter also provides the future scope of the research.

# CHAPTER 2

# LITERATURE REVIEW

Some of the previous studies about SDP are concisely summarized to depict the drift and trends in literature focusing on feature selection and extraction in software defect prediction. Liu et al. [1] examined the effect of some 32 feature selection methods such as filter-based, wrapper-based, clustering-based and extraction-based on the NASA dataset. Gayatri et al. [3] proposed a novel technique for feature selection build on Decision_Tree_Induction and compared it with RELIEF method and found that proposed method performed better than the others.

Ceylan et al. [4] conducted experiments Principal Component Analysis is used for dimensionality-reduction, and for classification Decision Tree, Multi-Layer Perceptron and Radial Basis Functions are used. Khoshgoftaar et al. [6] carried his research on the influence of data sampling followed by wrapper-based feature selection method and found out that the proposed method appreciably better than the classification method based on unsampled fit data.

In [9] authors evaluated a technique based on the Bat-based search Algorithm (BA) for the purpose of feature selection process, and the Random Forest algorithm (RF) for purpose of classification to perform software defect prediction.In this authors has also compared various classification technique with the proposed method and reported that proposed technique is providing better results. Lu et al. [10] experimentally investigated their proposed approach based on active learning on successive versions of eclipse dataset and results indicated that proposed technique along with uncertainity_sampling performs better than other classification methods. They improved their result by using feature selection before active learning.

A technique based on semi-supervised learning SDP was proposed by Cukic et al. [11], they analyzed that pre-processing technique along with multidimensional scaling is implanted decreasing the dimensional complexity of software metrics. Gao et al.[12] investigated an approach focused on feature compression using geometric mean (GM) along with conditional random field technique (CRF) for SDP which outperforms than other techniques. By utilizing the technique of Mutual information (MI) for feature selection Jin et al. [16] addressed that it is showing very good results along with many classifiers for NASA datasets.

In [13] Mahama et al. reported their study in scenarios by including various techniques such as ranker for dimensionality reduction, data-sampling for imbalanced data and iterative_partition_filter for reducing noise in data to enhance the results of Software Defect Prediction. Bisi et al. [14] showed that PCA for feature reduction along with ANN shows the better result in terms of accuracy as compared to SA i.e. Sensitivity Analysis for scale features with ANN.

Yang et al. [15] observed that their proposed work based on the ReliefF algorithm for dimension reduction and liner-correlation analysis can enhance the SDP on NASA dataset using Naive-Bayes, multilayer-perceptron and SVM classifiers. Qin et al. [17] explored that by using multilevel data preprocessing consisting of double feature selection and tripartite instances_filtering can enhance the process of SDP. In addition to that, they also reported that unrelated features by feature selection and data imbalance can be handled by resampling method.

Ji et al. [18] showed that their proposed work NASM performs better than other traditional technique, as it is based on maximal information-coefficient-matrix to select the features by clustering. The traditional techniques are based on PCA and other sampling techniques.

Kakkar and Jain [19] performed a comparative study on feature selection. Algorithms such as Best-Fit search and Greedy-Stepwise method and some ranker method were used with 3 different lazy-predictors .i.e. IBK, K-star & LWL and validated their results using 10-fold cross-Validation. It was observed that LWL outperforms among all other classifiers. FECAR a novel feature-clustering and feature-Ranking structure were developed by Liu et al. [20]. It was based on chi-square, Information-gain and relief measure. The experiments were performed on open source datasets of NASA and Eclipse.

Frieyadie and Putri [21] analyzed that by combining sampling technique with feature compression the prediction of software defects can be enhanced. Smote and random-sampling were used along with Chi Square, information Gain, and relief methods and it was shown that SMOTE+relief+Naive-Bayes outperformed among all other combinations.

Han et al. [22] observed that results of 61 features are comparable to that of only 2 features by using features selection techniques such as principal components analysis as well as variable importance and classifiers used were Random-Forest, Neural-Networks and Support Vector Machine.

Khoshgoftaar et al. [23] empirically showed that ensembles of feature selection techniques can perform better than the single feature selection techniques by employing 17 ranker-based and 11 threshold-based FS techniques and results showed that ensemble of a number of rankers is better than the ensemble of many rankers.

Rana et al. [24] investigated that to decrease the dimensions of input-space by dropping unrelated metrics InformationGain can be used as compared to PCA which selects the features other than keeps the illustration of all feature-variable undamaged.
Miao et al. [25] explored some different feature selection method that is the cost-sensitive feature and embedded a cost-matrix into FS methods and showed that their proposed work outperforms than other traditional techniques in terms of cost.

Verma and Gupta focused on exploring a way to reduce the features after feature selection and estimated the result of this on software defect prediction. They analyzed that FalsePositiveRate is diminished by means of the proposed scheme of feature selection [26].
In a study conducted by Malhotra et al. [27], it was analyzed that correlation-based feature selection is used to preprocess the data because this technique is fast simple and can handle both redundant and unrelated data.

Postma et al. [28] performed a systematic comparative analysis of various types of feature selection and extraction techniques and provide their expert review on these techniques. It also explained the weakness of non-linear techniques and provides the measure to improve their performance.

Usharani et al. [30] explored different and traditional as well as novel techniques of feature selection and feature extraction for dimensionality reduction.

Pooja Chenna in [31] performed a comparative study on various traditional as well as deep learning techniques of feature extraction such as PCA, RBM, and autoencoder and implemented RBM on ECL and result showed that RBM outperform among all for Visualization in 3-Dimensional Space

# CHAPTER 3

# RESEARCH METHODOLOGY

This section provides insight about the procedures and methods used in empirical study .here we have briefly discussed each method required to perform the empirical study such as process involved in data collection, preprocessing, classification, model validation, measuring performance and selection of statistical test.

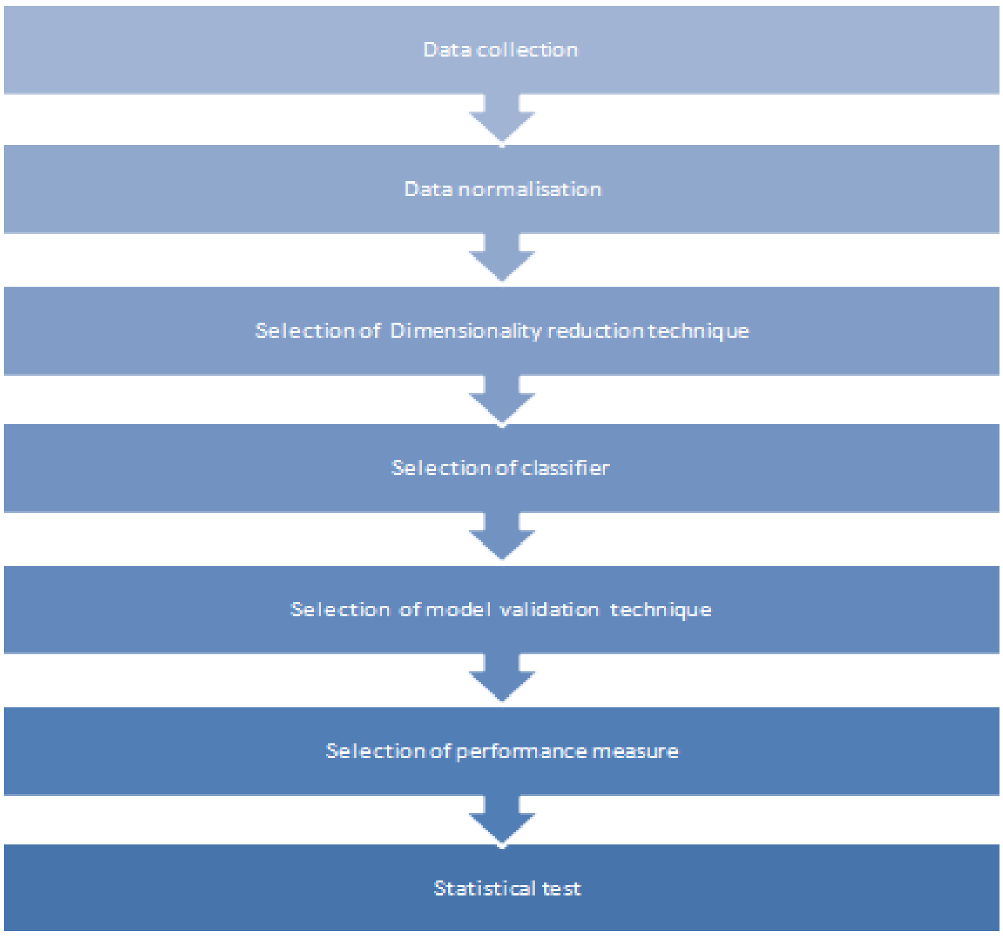

Data collection
Data normalisation
Selection of Dimensionality reduction technique
Selection of classifier
Selection of model validation technique
Selection of performance measure
Statistical test

**Fig. 3.1 Research methodology for SDP**

## 3.1 Dataset Collection

In this study, 9 open source projects from Apache software Foundation Systems which are publicly available in the open source dataset repository called PROMISE Repository are used as datasets. This is the mostly used repository in Software fault Prediction. For this study, different projects of varied size as well as with different defect rate is considered and these projects are having 20 Object Oriented metrics and also there is defect label for each class which shows whether a module is defective or not. Table 3.1 represents the release, total number of classes, size in terms of KLOC of each project, number of defective_classes and defective_percentage of each project.

**Table 3.1 Dataset Details**

| S.No | Projects | Release | Classes | KLOC | Defective Classes | Defective percentage |
|---|---|---|---|---|---|---|
| 1 | Ant | 1.7 | 745 | 208 | 166 | 22 |
| 2 | ArcPlatform | 1 | 234 | 31 | 27 | 12 |
| 3 | Camel | 1.6 | 965 | 113 | 188 | 19 |
| 4 | jEdit | 4.3 | 492 | 202 | 11 | 2 |
| 5 | Log4j | 1.2 | 205 | 38 | 180 | 92 |
| 6 | Poi | 3.0 | 442 | 129 | 281 | 64 |
| 7 | Prop | 6.0 | 660 | 97 | 66 | 10 |
| 8 | Tomcat | 6 | 858 | 300 | 77 | 9 |
| 9 | Xalan | 2.7 | 909 | 428 | 898 | 99 |

## 3.2 Data Normalisation

It is unlikely to have real-time statistics within a definite range. Hence pre-processing of the data performs to be a crucial move when there is not any compulsion to give weight to some definite feature while using classification or clustering techniques. Preprocessing can be done in two ways i.e. normalization or standardization. In comparative study it is essential to preprocess the data as maintaining it in a definite range is very imperative.

Selection of normalization technique for a specific data totally depends upon the user. For this comparative study, 20 input variables are not in same range of magnitude.

Hence, we perform data normalization on these attributes using min-max normalization to transform data within the range of [0, 1]. The equation for min-max normalization is given below

$$z = \frac{x - \min(x)}{\max(x) - \min(x)} \tag{1}$$

‘x’ denotes the data_point.
min(x) denotes the minimum_value of x of a feature.
max(x) denotes the maximum_value of x of a feature.
z denotes the normalized value.
During preprocessing step it must be ensured that no noise is injected in the original data.
Since software defect prediction is a classification process, the dependent variable (defect proneness) should represent only two classes i.e. defective or non_defective. The bug proneness is labeled with defect severity which ranges from 0 to 10 for all the dataset. In order to preprocess this variable, classes with bug severity less than 1 are labeled as non_defective other than that are labeled as defective.

## 3.3 Dimensionality Reduction Techniques

Many diverse techniques have been used for dimensi0nality reducti0n. These approaches can be supervised or unsupervised methods [33] When any discriminate analysis utilize class label called as supervised technique while some do not use class label mentioned to as unsupervised approaches. Some supervised approach methods are like LDA (Linear Discriminent Analysis), NN (Neural Network) used for dimensionality reduction.

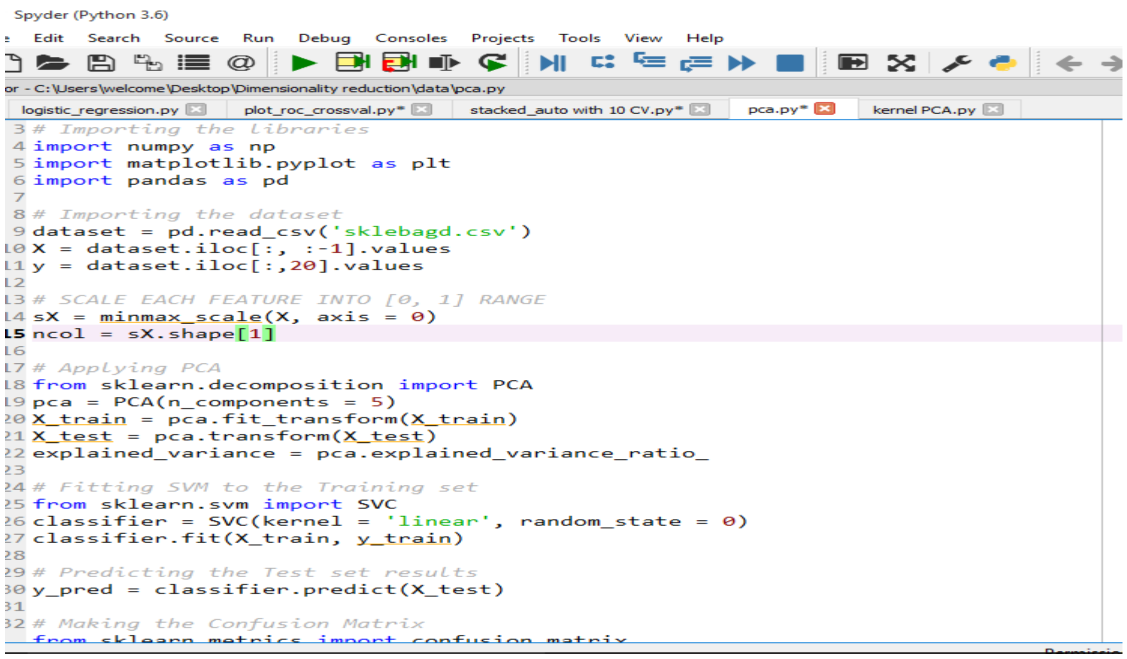
Unsupervised methods like PCA (Principal_Component_Analysis), ICA (Independent Component Analysis), KPCA (Kernel Principal Component Analysis), Restricted Boltzmann machines (RBM) and autoencoders etc. are used for decreasing dimensions of the dataset. For this comparative study, some of the above-mentioned techniques which are frequently used [14], [18], [22], [24], [28], [31] are considered for comparison for software defect prediction

**Table 3.2 Comparison of various Feature extraction techniques**

| Techniques | PCA | KPCA | LDA | AUTOENCODERS(AE) |
|---|---|---|---|---|
| Linear /Nonlinear | Linear | Nonlinear | Linear | Nonlinear |
| Supervised/ Unsupervised | Unsupervised | Unsupervised | Supervised | Unsupervised |
| Traditional/Novel | Traditional | Traditional | Traditional | Novel |
| Advantages | Association between modules of the data vector be undoubtedly seen and it capture the second order Correlation value. | Calculates the Kernel matrix to decrease the Features. | Linear mapping Dimensionality of the subspace is restricted by number of classes of data. | When finding distributed & compact representations, the encoder often ends up capturing the salient features of the input, yielding a much richer representation. |
| Disadvantages | Recognize the linear integration of variables and disregard the high order correlation value. | Dimension of kernel matrix is proportional to the square of the number of instances in the dataset. | Suffers from Small Sample Size problem | Autoencoders usually have a high number of connections. Therefore, backpropagation approaches converge slowly and are likely to get stuck in local minima. |

### 3.3.1 Principal Component Analysis (PCA)

It is the majority employed data transformation technique for preprocessing and feature extraction that lessens the feature space by seizing linear reliance amid diverse features. PCA looks for principal components (PC) that are the linear amalgamation of actual features such as so as to they are orthogonal to one another and seize the greatest sum of the variance in the data. Usually, it is probably to seize high variance by means of merely a very small number of PC’s [37].


Spyder (Python 3.6)
Edit Search Source Run Debug Consoles Projects Tools View Help
or - C:\Users\welcome\Desktop\Dimensionality reduction\data\pca.py
logistic_regression.py
plot_roc_crossval.py*
stacked_auto with 10 CV.py*
pca.py*
kernel PCA.py

```
# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Importing the dataset
dataset = pd.read_csv('sklebagd.csv')
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:,20].values

# SCALE EACH FEATURE INTO [0, 1] RANGE
sX = minmax_scale(X, axis = 0)
ncol = sX.shape[1]

# Applying PCA
from sklearn.decomposition import PCA
pca = PCA(n_components = 5)
X_train = pca.fit_transform(X_train)
X_test = pca.transform(X_test)
explained_variance = pca.explained_variance_ratio_

# Fitting SVM to the Training set
from sklearn.svm import SVC
classifier = SVC(kernel = 'linear', random_state = 0)
classifier.fit(X_train, y_train)

# Predicting the Test set results
y_pred = classifier.predict(X_test)

# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
```

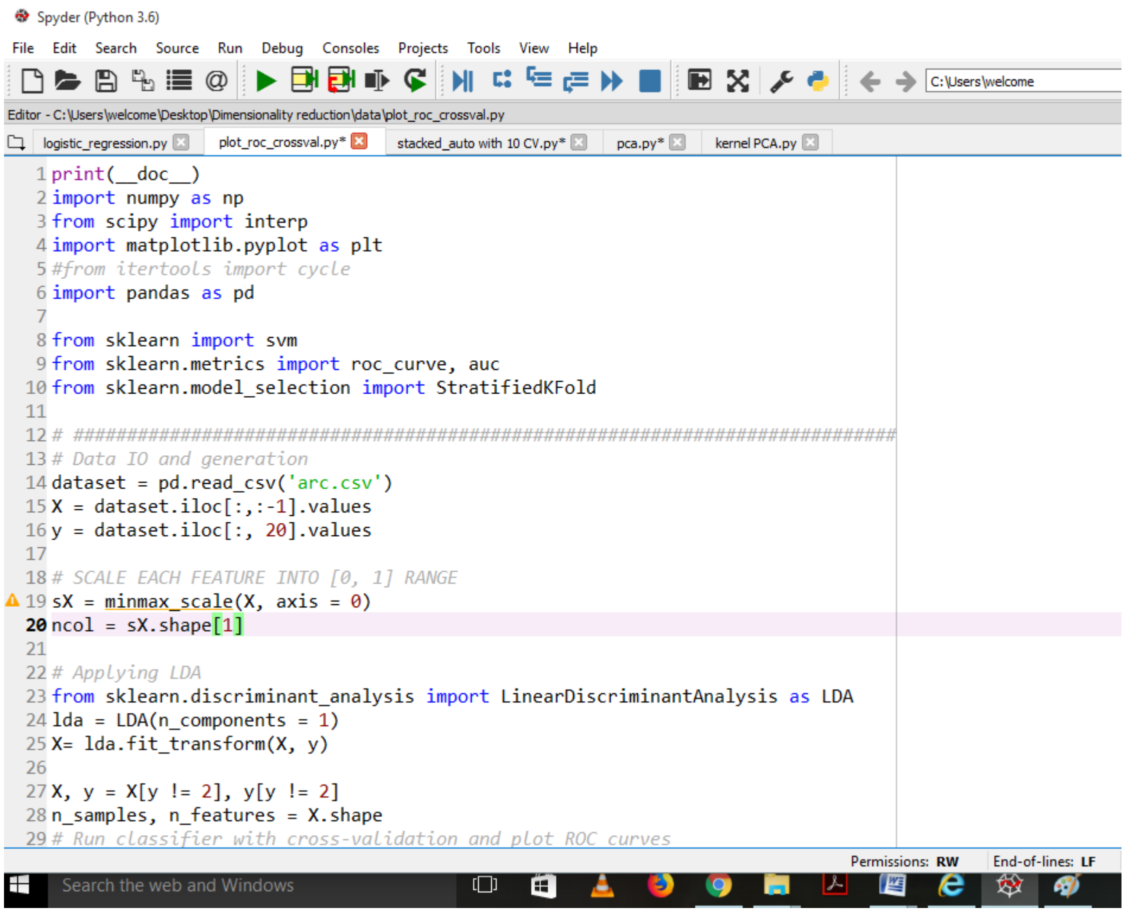
**Fig 3.2 Implementation of PCA technique in SPYDER.**

In turn, to find PC’s, from original data covariance matrix, is calculated and then every eigenvalue are calculated. PCs are those eigenvectors that correspond to the largest eigenvalues.

### 3.3.2 Linear Discriminant Analysis (LDA)

This is another widely used feature extraction technique used in software defect prediction. The major objective of the LDA resides in calculating a base of vectors providing the greatest difference amid the classes, attempting to get the most out of the between-class-differences, diminishing the within-class-differences one time by means of scatter matrices [34].

It also goes through small-sample-size problem which is present in high dimension data where a number of present samples is less than dimensions of the samples. D-LDA, R-LDA, and K-DDA are some types of LDA [34].


Spyder (Python 3.6)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\welcome
Editor - C:\Users\welcome\Desktop\Dimensionality reduction\data\plot_roc_crossval.py
logistic_regression.py
plot_roc_crossval.py*
stacked_auto with 10 CV.py*
pca.py*
kernel PCA.py
Permissions: RW
End-of-lines: LF
Search the web and Windows

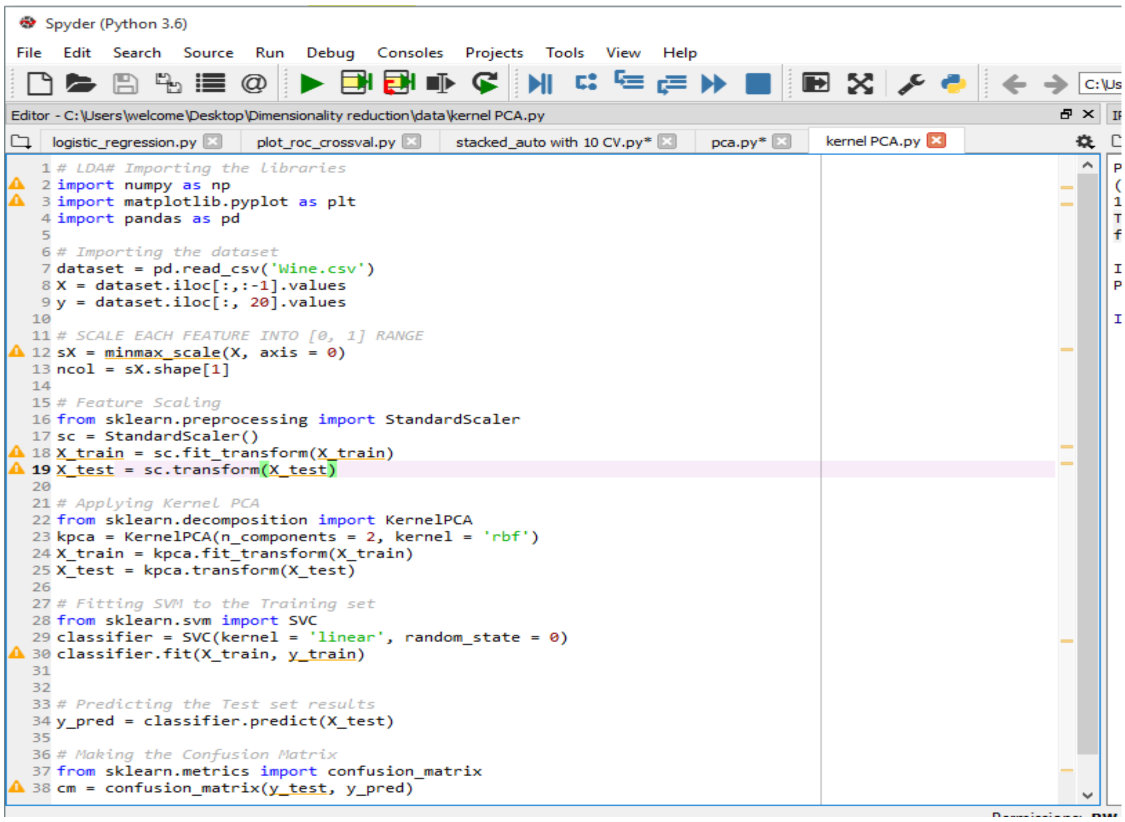
```
print(__doc__)
import numpy as np
from scipy import interp
import matplotlib.pyplot as plt
#from itertools import cycle
import pandas as pd

from sklearn import svm
from sklearn.metrics import roc_curve, auc
from sklearn.model_selection import StratifiedKFold

# ###########################################################################
# Data IO and generation
dataset = pd.read_csv('arc.csv')
X = dataset.iloc[:,:-1].values
y = dataset.iloc[:, 20].values

# SCALE EACH FEATURE INTO [0, 1] RANGE
sX = minmax_scale(X, axis = 0)
ncol = sX.shape[1]

# Applying LDA
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as LDA
lda = LDA(n_components = 1)
X= lda.fit_transform(X, y)

X, y = X[y != 2], y[y != 2]
n_samples, n_features = X.shape
# Run classifier with cross-validation and plot ROC curves
```

**Fig. 3.3 Implementation of LDA technique in SPYDER**

### 3.3.3 Kernel-PCA (K-PCA)

It is the extension of conventional linear-PCA in a high dimensional scope which is builds using a kernel function [28]. During previous years, the redevelopment of linear methods based on the ‘kernel’ has directed towards the scheme of booming methods for instance Support Vector Machines. In this technique instead of the covariance matrix, principal eigenvectors of the kernel matrix are calculated.

The redevelopment of PCA in kernel space is clear-cut since a kernel matrix is like the in_product of the data points in the high dimensional space that is build using the kernel function.


Spyder (Python 3.6)
File Edit Search Source Run Debug Consoles Projects Tools View Help
logistic_regression.py
plot_roc_crossval.py
stacked_auto with 10 CV.py*
pca.py*
kernel PCA.py

Editor - C:\Users\welcome\Desktop\Dimensionality reduction\data\kernel PCA.py

```python
# LDA# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Importing the dataset
dataset = pd.read_csv('Wine.csv')
X = dataset.iloc[:,:-1].values
y = dataset.iloc[:, 20].values

# SCALE EACH FEATURE INTO [0, 1] RANGE
sX = minmax_scale(X, axis = 0)
ncol = sX.shape[1]

# Feature Scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)

# Applying Kernel PCA
from sklearn.decomposition import KernelPCA
kpca = KernelPCA(n_components = 2, kernel = 'rbf')
X_train = kpca.fit_transform(X_train)
X_test = kpca.transform(X_test)

# Fitting SVM to the Training set
from sklearn.svm import SVC
classifier = SVC(kernel = 'linear', random_state = 0)
classifier.fit(X_train, y_train)


# Predicting the Test set results
y_pred = classifier.predict(X_test)

# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
```

**Fig. 3.4 Implementation of K-PCA technique in SPYDER.**

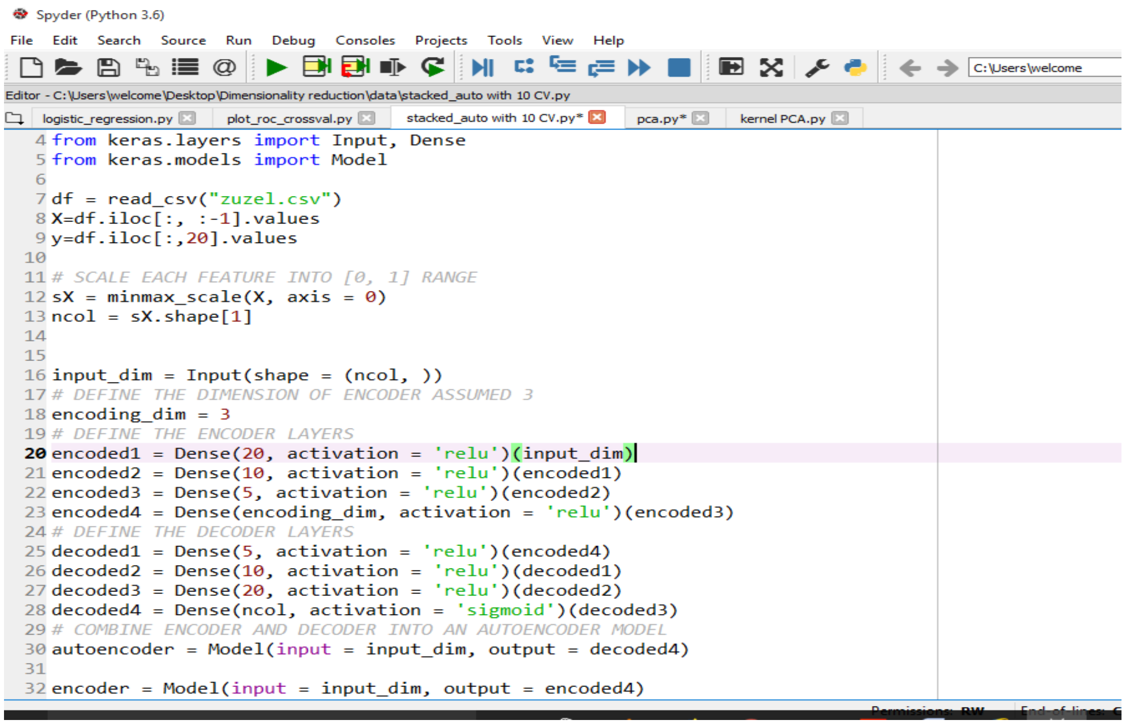
### 3.3.4. Autoencoders (AE)

Autoencoder was at first presented soon after the 1980s [36] as a linear feature extraction technique. A noteworthy preferred benefit of Autoencoder is that it is simple to stack for producing diverse levels of new features to show actual ones by adding hidden layers.

In order to allow the autoencoder to learn a nonlinear mapping between the high-dimensional and low-dimensional data representation, sigmoid activation functions are generally used [28]. If linear activation function is used instead of sigmoid activation function, an autoencoder is identical to PCA.

Stacked auto-encoders are best utilized for unsupervised learning as it is great in catching hierarchical groups,  that is essential layers of the system learns more elevated amount features and as we go deeper in the network, it attempts to learn to bring down level features in deep learning that supplanted the learning procedures utilized in traditional neural networks.

Stacked Auto-encoders according to its name is a stack of auto-encoders. They are too known as Stacked Auto-Associators as these attempt to relate the output with the input and seek to locate intermediate representations [31]. Conventionally, in this output from one AE is taken as input for the next AE and this procedure are iterated until every lone AE in the network is pretrained. The resultant present at the output layer of the network is with reduced dimensions. There are diverse types of AE like Sparse AE, Contractive AE, and Denoising AE.


Spyder (Python 3.6)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\welcome
Editor - C:\Users\welcome\Desktop\Dimensionality reduction\data\stacked_auto with 10 CV.py
logistic_regression.py
plot_roc_crossval.py
stacked_auto with 10 CV.py*
pca.py*
kernel PCA.py

```
from keras.layers import Input, Dense
from keras.models import Model

df = read_csv("zuzel.csv")
X=df.iloc[:, :-1].values
y=df.iloc[:,20].values

# SCALE EACH FEATURE INTO [0, 1] RANGE
sX = minmax_scale(X, axis = 0)
ncol = sX.shape[1]


input_dim = Input(shape = (ncol, ))
# DEFINE THE DIMENSION OF ENCODER ASSUMED 3
encoding_dim = 3
# DEFINE THE ENCODER LAYERS
encoded1 = Dense(20, activation = 'relu')(input_dim)
encoded2 = Dense(10, activation = 'relu')(encoded1)
encoded3 = Dense(5, activation = 'relu')(encoded2)
encoded4 = Dense(encoding_dim, activation = 'relu')(encoded3)
# DEFINE THE DECODER LAYERS
decoded1 = Dense(5, activation = 'relu')(encoded4)
decoded2 = Dense(10, activation = 'relu')(decoded1)
decoded3 = Dense(20, activation = 'relu')(decoded2)
decoded4 = Dense(ncol, activation = 'sigmoid')(decoded3)
# COMBINE ENCODER AND DECODER INTO AN AUTOENCODER MODEL
autoencoder = Model(input = input_dim, output = decoded4)

encoder = Model(input = input_dim, output = encoded4)
```

**Fig. 3.5 Implementation of Autoencoder technique in SPYDER**.

## 3.4 Classification Techniques

In this study, a comparative performance analysis of different feature extraction technique along with machine learning techniques is explored for software defect prediction on publicly available datasets. Machine learning techniques are proven to be useful in terms of software defect prediction. The data from software repository contains lots of information in assessing software quality, and machine learning techniques can be applied to them in order to extract software defects information.

The machine learning techniques are classified into two broad categories in order to compare their performance; such as supervised learning versus unsupervised learning. In supervised learning algorithms such as classifier like Multilayer perceptron, Naive Bayes classifier, Support vect0r machine, Random Forest and Decision Trees are compared. In case of unsupervised learning methods like Radial base network function, clustering techniques such as K-means algorithm, K nearest neighbor are compared against each other. But in this study, we are investing only Support Vector Machine (SVM) because in most of the literature based on feature extraction SVM is used [3], [6], [15], [18], [22], [25], [29].

### 3.4.1 Support Vector Machine (SVM)

It assumes (SVM) [5], [12] utilizes non-linear mapping for original training data to transform it into the higher dimension. Then it searches for an optimal linear hyperplane for separation. The hyperplane can be found using margins and support vectors. SVM is used for classification purpose and is based on supervised learning. Support Vector Machine (SVM) is a powerful and flexible type of supervised learning model, used for classification and regression analysis. It has been applied to learning algorithms that analyze data and recognize patterns. It has the advantage of reducing problems of overfitting or local minima. In addition, it is based on structural risk minimization as opposed to the empirical risk minimization of neural networks.
Given a set of samples for training purpose [7], an SVM training algorithm builds a model that assigns new examples into one category or the other, making it a non-probabilistic binary linear classifier.

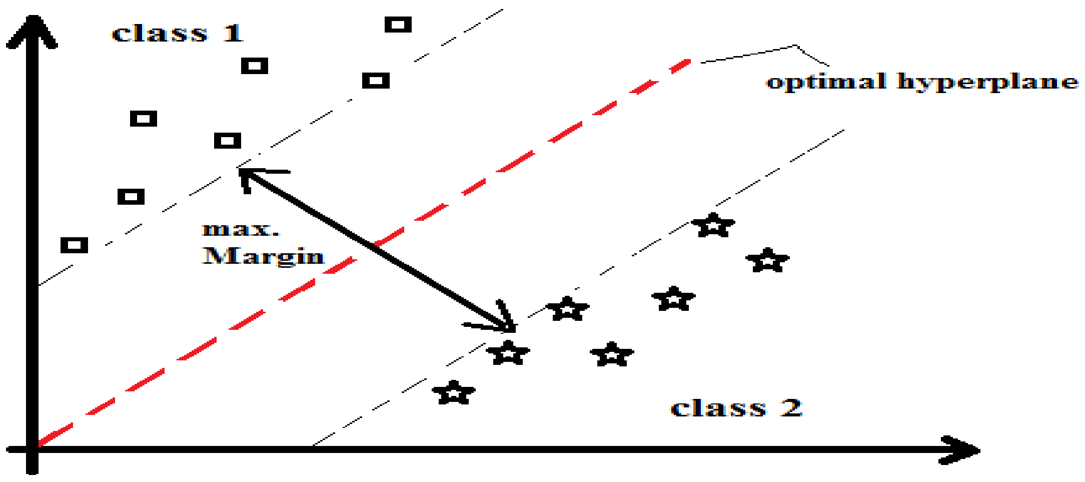


**Fig. 3.6 Optimal hyperplane in Support vector machine**

## 3.5 Model validation Technique

There are many types of model validation techniques such as hold-out method, Leave-one-out, k-fold Cross-validation and bootstrapping etc for validating the model against the training data. From the literature study, it can be inferred that widely used model validation technique is 10 fold Cross-validation [1], [3], [6], [7], [8]. Cross-validation is a method to estimate predictive m0dels by dividing the original dataset into a training_set for training the model developed, and a test set to assess the performance of the model developed.

In k-fold cross-validation, as shown in fig. 3.7, the original_sample is arbitrarily subdivided into k-equal_size subsampIes[9][11][13][14]. Out of the k-subsamples, a lone subsample is reserved as the test data for validating the model, and the leftover ‘k-1’ subsamples are taken for the training of the model. Then the CV process is iterated k times, with a piece of the k subsamples used accurately once as the test data. A mean/average value of the ‘k’ numbers of results obtained from the foIds, can be taken to generate the final inference.

The benefit of this technique over all other methods such as hold-out method, Leave-one-out, and bootstrapping is that all instances are used for both training and testing, and every instance is used for testing just once. There are some subtype of K-fold cross-validation technique such as Stratified K-fold cross-validation and repeated K-fold cross-validation.

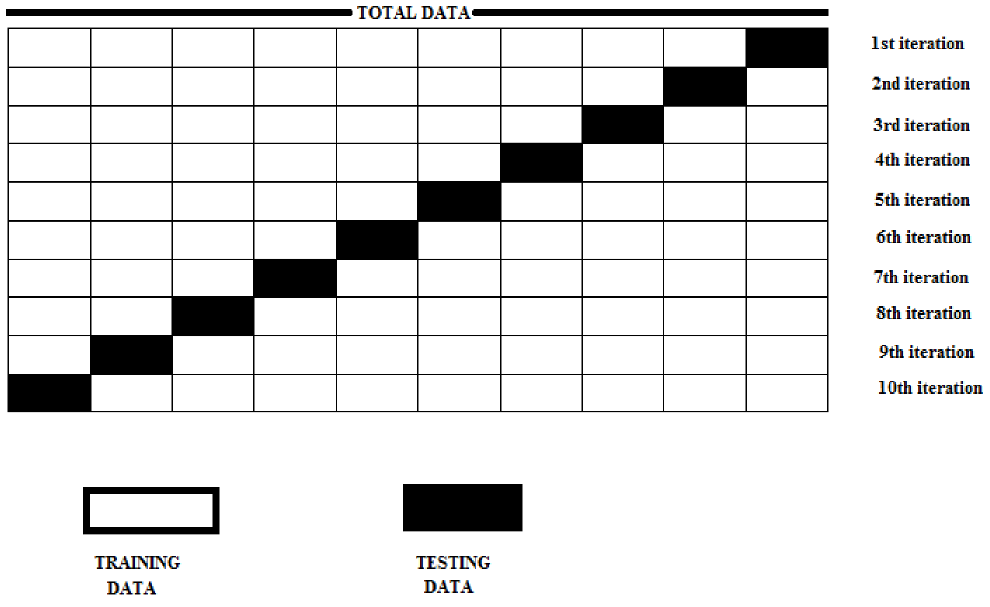


**Fig. 3.7 10-fold cross-validation**

## 3.6 Performance Measures

There is an extensive number of performance measures for classification [1], [21], [3], [7], [10], [12]. These measures have been used for different applications and to evaluate different things. Commonly used performance measures for Software defect prediction are Accuracy, Precision, Recall, F1-measure, PR-AUC & ROC-AUC etc. Out of these techniques Accuracy and ROC- AUC are considered for this comparative study.

### 3.6.1 Accuracy

Traditionally, accuracy is the widely used technique for performance estimation. Accuracy shows the ratio of all correctly classified instances [11], [16], [17]. However, accuracy is not proper to measure particularly in defect prediction because of class imbalance of defect prediction [7].

The value of accuracy can be represented by a single value that lies between 0 to 1.

Accuarcy=(T_P+T_N )/ N

where,

T_P= #observations which are ‘defective’ and predicted to be ‘defective’

T_N= #observations which are ‘non_defective’ and predicted to be ‘non_defective’

N= total number of samples in a dataset.

### 3.6.2 ROC - AUC

Nowadays ROC-AUC is the most widely used performance measure [1], [21], [3], [6], [9], [10], [11], [15], [17]. ROC-AUC Curve is most commonly utilized to visualize the performance of a binary value-based classifier, and AUC is the optimal method, to sum up, its results in a lone number which ranges from 0 to 1. The ROC-AUC curve is utilized to differentiate the trade-off between true_positive_rate and false_positive _rate as shown in fig.3.8. A predictor that results in a large area under the curve of ROC is superior over a classifier with a minor area under the curve. An ideal classifier produces an AUC that equals 1.

The benefits of using ROC-AUC are its vigor towards imbalanced class distributions [24]. Therefore, it is particularly well suited for software defect prediction tasks

.

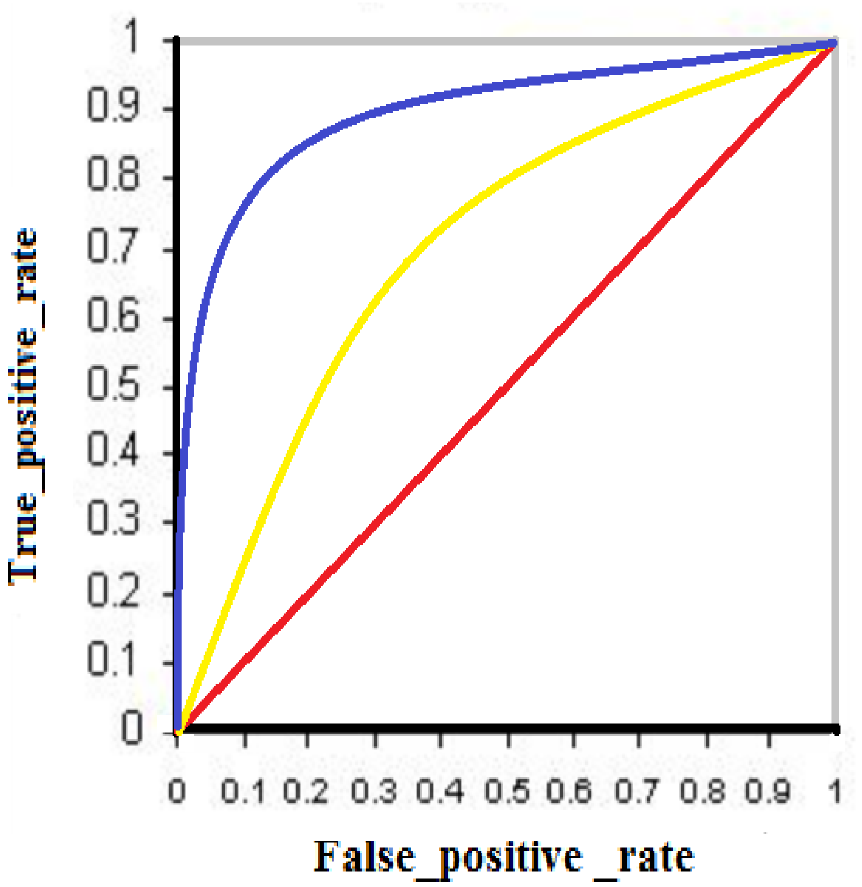


**Fig. 3.8 ROC-AUC curve representation**

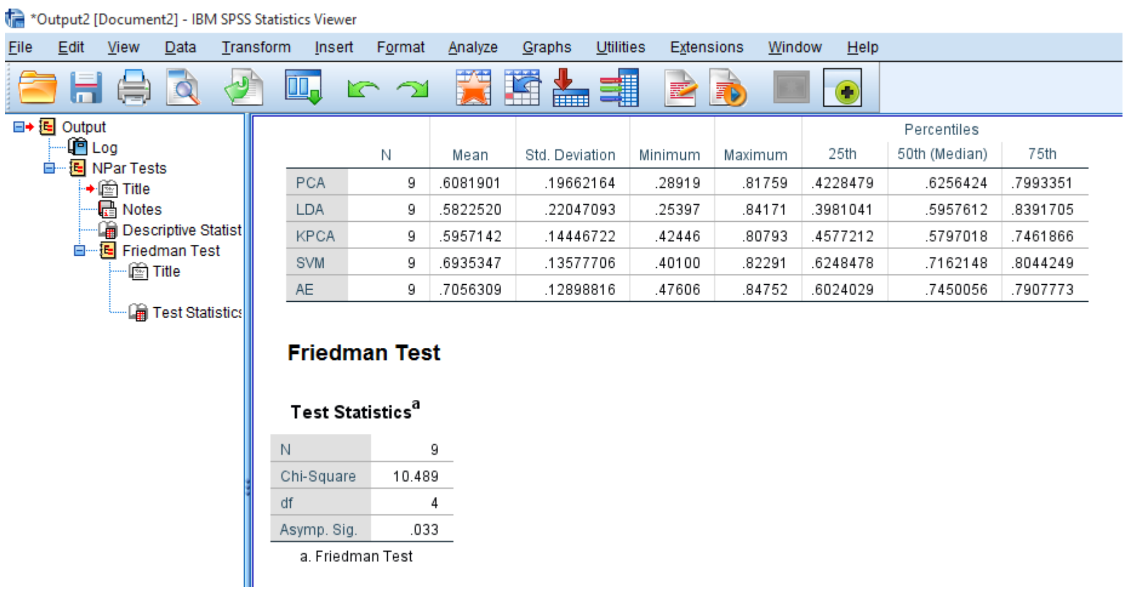
## 3.7 Statistical Test

With a specific end goal to statistically assess the execution of feature extraction technique we utilize a statistical test called Friedman test. Not at all like parametric tests, assumptions made in the non-parametric tests are not rigorous and one may overlook the nearness of anomalies in the datasets, variance_homogeneity, and normality distributions so on [40].

Lessmann et al. learn that lone couple of past examinations have utilized statistical tests for performance validation inferring conclusions only by manual investigation of exact outcomes may deceive and can make irregularity crosswise over more than one explore performed on a similar subject. To evade this situation, we utilize a statistical test to create verified conclusions.

### 3.7.1 Friedman Test

Friedman test is a non-parametric statistical test based on the rankings of performance values rather than the actual values [1], [2]. In this work, we use this test to detect whether the performance differences among the 4 feature selection methods and a classifier SVM are random. It is practically equivalent to the repeated-measures ANOVA in non-parametrical statistical methodology; in this manner, it is a multicomparison test that intends to recognize noteworthy contrasts between the characteristics of at least two algorithms or techniques.



| | N | Mean | Std. Deviation | Minimum | Maximum | Percentiles | | |
|---|---|---|---|---|---|---|---|---|
| | | | | | | 25th | 50th (Median) | 75th |
| PCA | 9 | .6081901 | .19662164 | .28919 | .81759 | .4228479 | .6256424 | .7993351 |
| LDA | 9 | .5822520 | .22047093 | .25397 | .84171 | .3981041 | .5957612 | .8391705 |
| KPCA | 9 | .5957142 | .14446722 | .42446 | .80793 | .4577212 | .5797018 | .7461866 |
| SVM | 9 | .6935347 | .13577706 | .40100 | .82291 | .6248478 | .7162148 | .8044249 |
| AE | 9 | .7056309 | .12898816 | .47606 | .84752 | .6024029 | .7450056 | .7907773 |

**Friedman Test**

**Test Statistics**[a]

| | |
|---|---|
| N | 9 |
| Chi-Square | 10.489 |
| df | 4 |
| Asymp. Sig. | .033 |

a. Friedman Test

**Fig. 3.9 Friedman test statistics**

It processes the positioning of the experimental outcomes for the algorithm, for each function, providing rank 1 to the top of them, and rank ‘k’ to the worst. Assuming that according to the null hypothesis laid, the performance of the techniques used are equivalent and doesn’t show any significant difference. Consequently, their rankings should be akin. The Friedman’s statistic χ2.

$$\chi^2 = \frac{12}{nk(k+1)}\sum R^2 - 3n(k+1) \tag{2}$$

is distributed according to χ2 with k − 1 degrees of freedom, where ‘k’ given in above equation number (2) is a number of techniques being compared, for this study it is set to be 5 at the alpha value of α=0.05. And n represents the number of instances for which the techniques are compared whereas R2 represent the sum of squares of these instances.

# CHAPTER 4

# EXPERIMENTAL DESIGN

In this section, we have discussed the experimental setup required for empirical comparison of feature extraction techniques. This segment provides information about variables selection (independent variable and dependent variable), hypothesis formulation and tools used for the experiments are also discussed.

## 4.1 Variable Selection

The two main variables in an experiment are the independent and dependent variable. There is cause-effect relationship between independent and dependent variable.

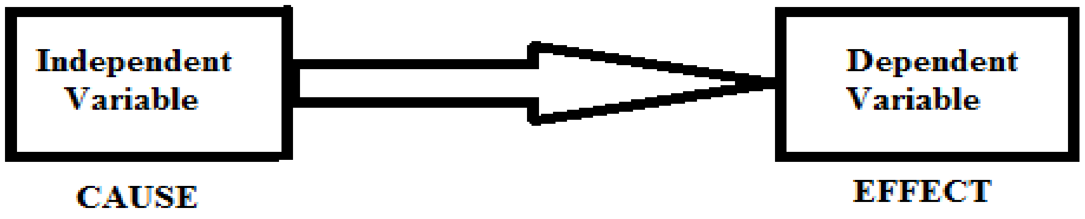


**Fig. 4.1 Relationship between independent and dependent variable**

In this study of Software defect prediction the variable used is:

An **independent variable** is the variable that is changed or controlled in a scientific experiment to test the effects on the dependent variable e.g. 20 metrics (LOC and CK metrics) used in this study are Resp0nse for the Class (RFC), Weighted Meth0ds per Class (WMC), Depth of Inheritance (DIT), Number of Children (NOC), Coupling between Objects (CBO) Lack of Cohesion in methods (LCOM) ,Afferent Coupling(CA), Efferent Coupling(CE), number of public methods (NPM) another

Lack of Cohesion in methods (LCOM3), lines of code (LOC) and Data access method (DAM) etc.

A **dependent variable** is a variable being tested and measured in a scientific experiment eg. Value showing whether a class is defect free or not. The dependent variable is 'dependent' on the independent variable. As the experimenter changes the independent variable, the effect on the dependent variable is observed and recorded. The dependent variable for this study is ‘defective’ is a binary variable which indicates the defectiveness of the class. A class is said to be defective, if there is a chance of observing a defect in the class in future versions of a project otherwise, a class is termed as non-defective.

## 4.2 Hypothesis Formulation

The following set of Hypothesis has been developed to evaluate the defect prediction model using machine learning techniques based on feature extraction.

***Null Hypothesis (H0):*** The ROC-AUC results of the defect prediction models developed using SVM doesn’t show any significant difference when no feature extraction method or four different feature extraction methods (PCA, LDA, K-PCA, Autoencoders) are used for the given datasets.

***Alternate Hypothesis (H1):*** The ROC-AUC results of the defect prediction models developed using SVM shows significant difference when no feature extraction method or four different feature extraction methods (PCA, LDA, K-PCA, Autoencoders) are used for the given datasets.
Friedman test is a statistical test performed for hypothesis testing as well as for the comparison of the results. The alpha level is set to be 0.05.

## 4.3 Tools Used

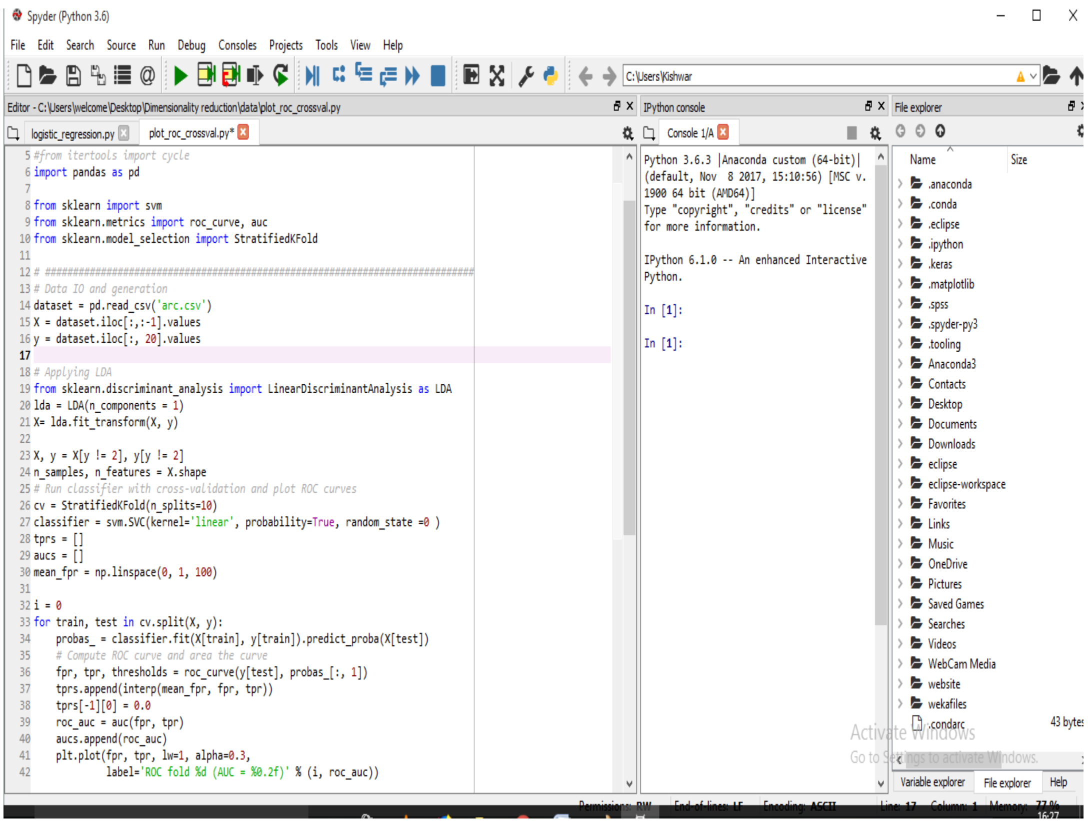
To perform the empirical research various types of open-source and proprietary tools are required throughout the process. In this study, two tools are employed; Spyder is used to perform the feature selection, classification and model validation etc. whereas SPSS a proprietary software developed by IBM is used for hypothesis testing using Friedman test.

## 4.3.1 SPYDER

All the experiments are implemented in python language. There are various tools available for machine learning techniques such as Weka, KEEL SPSS, Orange, RStudio, and Matlab etc. Spyder is a cross-platform and open-source IDE for performing machine learning tasks in Python Language. It consists of a huge number of machine learning libraries such as numpy, scipy, pandas, matplotlib, and scikit-learn, as well as other open source software and other libraries such as tensorflow, theano, keras and pytorch can also be imported to perform deep learning related experiments.

Advantages of using SPYDER are that it supports multi Python consoles and the ability to explore and amend variables from GUI.

In this study SPYDER is used to perform all the tasks such as preprocessing, feature extraction, classification, validating the model and for calculating performance measures.



```python
#from itertools import cycle
import pandas as pd

from sklearn import svm
from sklearn.metrics import roc_curve, auc
from sklearn.model_selection import StratifiedKFold

# ###############################################################################
# Data IO and generation
dataset = pd.read_csv('arc.csv')
X = dataset.iloc[:,:-1].values
y = dataset.iloc[:, 20].values

# Applying LDA
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as LDA
lda = LDA(n_components = 1)
X= lda.fit_transform(X, y)

X, y = X[y != 2], y[y != 2]
n_samples, n_features = X.shape
# Run classifier with cross-validation and plot ROC curves
cv = StratifiedKFold(n_splits=10)
classifier = svm.SVC(kernel='linear', probability=True, random_state =0 )
tprs = []
aucs = []
mean_fpr = np.linspace(0, 1, 100)

i = 0
for train, test in cv.split(X, y):
    probas_ = classifier.fit(X[train], y[train]).predict_proba(X[test])
    # Compute ROC curve and area the curve
    fpr, tpr, thresholds = roc_curve(y[test], probas_[:, 1])
    tprs.append(interp(mean_fpr, fpr, tpr))
    tprs[-1][0] = 0.0
    roc_auc = auc(fpr, tpr)
    aucs.append(roc_auc)
    plt.plot(fpr, tpr, lw=1, alpha=0.3,
             label='ROC fold %d (AUC = %0.2f)' % (i, roc_auc))
```

```
Python 3.6.3 |Anaconda custom (64-bit)|
(default, Nov  8 2017, 15:10:56) [MSC v.
1900 64 bit (AMD64)]
Type "copyright", "credits" or "license"
for more information.

IPython 6.1.0 -- An enhanced Interactive
Python.

In [1]:

In [1]:
```

**Fig. 4.2 Model development in SPYDER**

## 4.3.2 SPSS

It stands for **Statistical Package for the Social Sciences**. This software is developed by IBM and provide sophisticated statistical analysis, an enormous collection for machine learning algorithms, text-analysis, incorporation with big-data and flawless use into applications. Its user-friendlinessexibility and scalability make it handy to users with all proficiency levels.

In this study, SPSS is used for performing the Friedman test for hypothesis testing as well as for statistical validation of all the techniques used for software defect prediction.

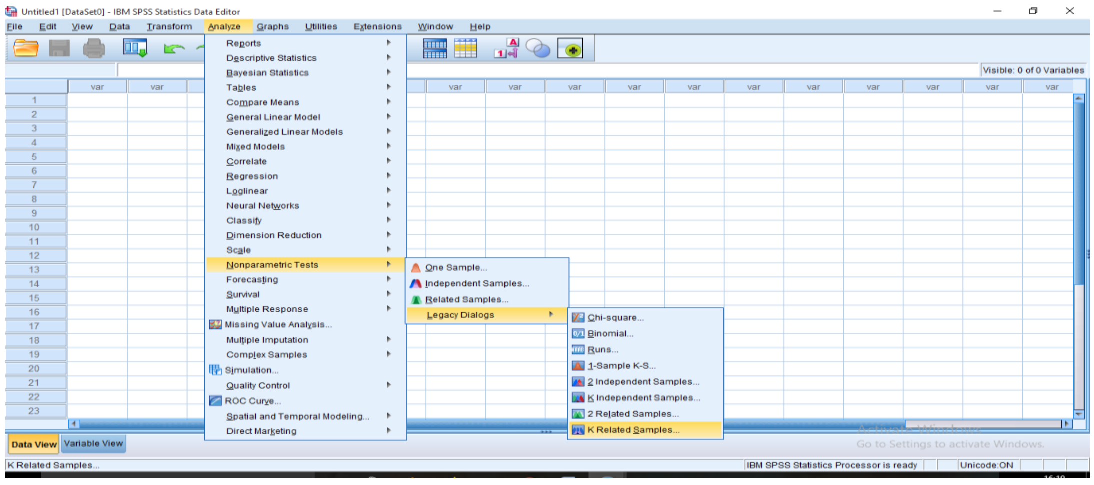


**Fig. 4.3 Friedman test performed in SPSS**

# CHAPTER-5

# RESULTS AND DISCUSSIONS

In this section, computed results of various features extraction techniques and classification techniques for software defect prediction using earlier conventional as well as deep learning techniques are presented and discussed in the form of Tables 5.1 and 5.2 as well as in the form of line graph (in fig.5.1 and 5.2) for better visualization and understanding. The model is validated using 10 cross-validation technique and performance measures used are accuracy and ROC-AUC.

From the results, it can be inferred that the model built by using feature extraction and classification techniques mainly have accuracy greater than or equal to 70% corresponding to the data set.

**Table 5.1 Accuracy calculated for each technique**

| S.No. | Projects | PCA –SVM | LDA-SVM | KPCA-SVM | AE- SVM | SVM |
|---|---|---|---|---|---|---|
| 1 | Ant | 0.808 | **0.828** | 0.776 | 0.776 | 0.827 |
| 2 | Arc | 0.880 | **0.897** | 0.880 | 0.881 | 0.898 |
| 3 | Camel | 0.800 | 0.801 | 0.801 | **0.811** | 0.796 |
| 4 | Jedit | 0.975 | 0.957 | 0.877 | **0.977** | 0.970 |
| 5 | Log4j | 0.925 | 0.902 | 0.892 | **0.922** | 0.922 |
| 6 | Prop | 0.853 | 0.902 | 0.893 | **0.903** | 0.902 |
| 7 | Poi | 0.765 | **0.774** | 0.632 | 0.706 | 0.765 |
| 8 | Tomcat | 0.907 | 0.908 | 0.906 | **0.909** | 0.906 |
| 9 | Xalan | 0.985 | 0.986 | 0.878 | **0.987** | 0.978 |

The potential of distinct methods varies on distinct datasets during improvement of software defect prediction models. Table III displays that different approaches work distinctly for every dataset as a top presentation by means of a definite performance measure is specified with a distinct method for every dataset. For example–for jedit, all the techniques performed well in terms of accuracy. Similarly, for Arc dataset, all the techniques performed very well whereas for Poi every method performed on average. These methods can be affected by some attribute of a specific dataset. On the other hand, there is the requirement to execute additional work to really assess which sort of method experience by the features of a data set.

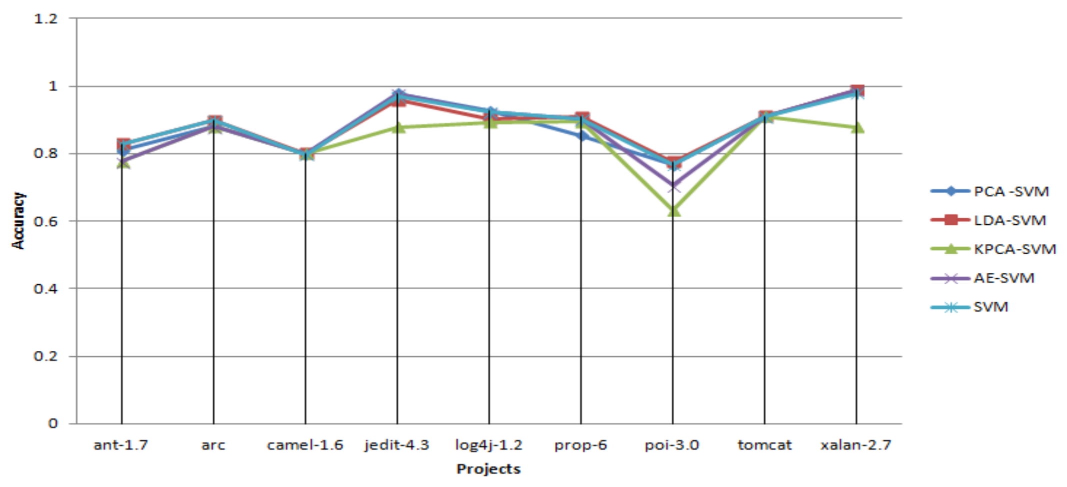


**Fig. 5.1 Line graph representing the accuracy of each technique**

But we have used another performance measure i.e. ROC-AUC whose values are compiled in the form of tables IV and line diagram fig. The advantage of using ROC-AUC is that it acts as the primary indicator of the comparative performance of the prediction model as it can cope with imbalanced and noisy data and is insensate to the alterations in the class division [27].

**Table 5.2 ROC-AUC value calculated for each technique**

| S.No. | Projects | **PCA** –SVM | LDA-SVM | KPCA-SVM | AE- SVM | SVM |
|---|---|---|---|---|---|---|
| 1 | Ant | 0.817 | **0.840** | 0.773 | 0.818 | 0.792 |
| 2 | Arc | 0.499 | **0.838** | 0.467 | 0.738 | 0.735 |
| 3 | Camel | 0.625 | 0.435 | 0.515 | **0.696** | 0.689 |
| 4 | Jedit | 0.289 | 0.595 | 0.447 | **0.762** | 0.560 |
| 5 | Log4j | 0.704 | 0.253 | 0.424 | **0.745** | 0.716 |
| 6 | Prop | 0.345 | 0.360 | 0.579 | **0.508** | 0.401 |
| 7 | Poi | 0.810 | 0.841 | 0.807 | **0.847** | 0.816 |
| 8 | Tomcat | 0.592 | 0.595 | 0.625 | **0.757** | 0.707 |
| 9 | Xalan | 0.788 | 0.478 | 0.719 | 0.476 | **0.822** |

To analyze the comparative performance of different methods using the ROC-AUC, initially, we have developed hypothesis which is briefly discussed in previous sections. To perform the hypothesis testing a non-parametric Friedman test is used, which resulted in the rejection of null hypothesis for ROC-AUC and it also shows that results of a model constructed for software defect prediction using distinct methods show a significant difference from each other when assessed using ROC-AUC. According to this test Autoencoders performance the best on the basis ROC-AUC and got rank one among all other techniques. Hence, it can be concluded that **Autoencoders** is an effective method for performing feature selection in software.

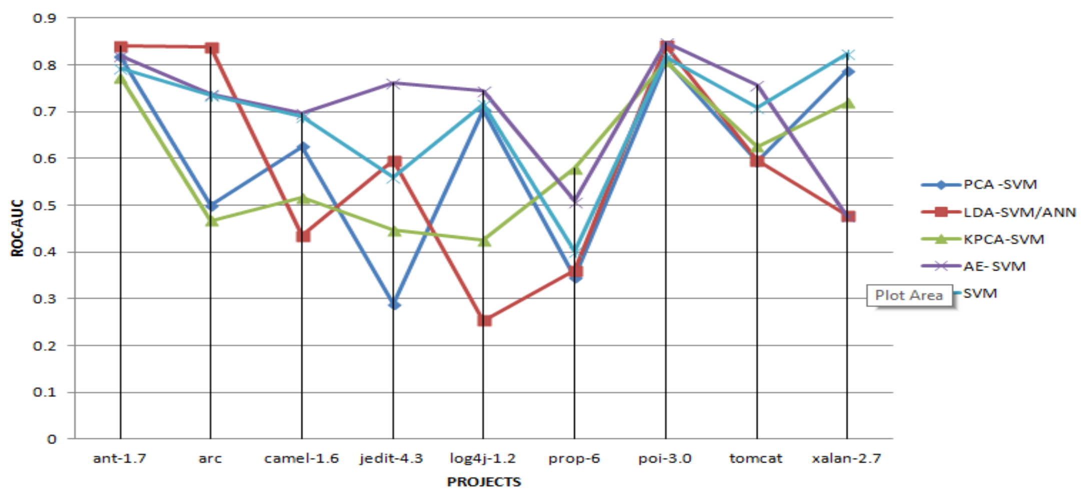


**Fig. 5.2 Line graph representing ROC-AUC value of each technique**

# CHAPTER 6

# CONCLUSIONS AND FUTURE WORK

In this thesis, we have examined and studied the performance of 4 feature extraction techniques along with a classification technique based on support vector machine. The dataset is taken from Promise data repository which contains different metrics as independent variables whereas defect_proneness is taken as the dependent variable having binary class label as defective or non_defective. In this study for data preprocessing data normalization using ‘min-max normalization’ is used which keeps the value in the range [0, 1]. The performance of the techniques is evaluated using various performance measures such as accuracy and ROC-AUC.

The objective of this work is to gain insights and compare the abilities of distinct feature extraction techniques and to rank them according to ROC-AUC measure. The statistical comparison is also performed by using non-parametric Friedman test to evaluate the difference between the methods and to rank them

The results demonstrate that different techniques perform differently on different datasets. From the table III. and IV. it can be concluded that autoencoders performed best among all other methods and got rank one. However, LDA also performed well for many datasets.

In future, more studies should be done to assess different feature extraction methods of deep learning based on RBM and SOM techniques as these are the novel methods for feature extraction as well as classification. Different data sets can also be used for future studies along with inter cross-validation method. Different performance measure such as G-measure and H-measure can also be used for evaluating the performance of different techniques

# CHAPTER 7

# REFERENCES

[1] Z. Xu, J. Liu, Z. Yang, G. An and X Jia, "The Impact of Feature Selection on Defect Prediction Performance: An Empirical Comparison", *IEEE 27th International Symposium on Software Reliability Engineering (ISSRE)***,** pp. 309 – 320, 2016.

[2] S. García, D. Molina, M. Lozano and F. Herrera "A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: a case study on the CEC'2005 Special Session on Real Parameter Optimization", *J Heuristics*, pp. 617–644, 2009.

[3] N.Gayatri, S.Nickolas and A.V.Reddy, "Feature Selection Using Decision Tree Induction in Class level Metrics Dataset for Software Defect Predictions", Proceedings *of the World Congress on Engineering and Computer Science*, vol. 1, 2010.

[4] E. Ceylan, F. Kutlubay and A. B. Bener, "Software Defect Identification Using Machine Learning Techniques", *Proceedings of the 32nd EUROMICRO Conference on Software Engineering and Advanced Applications*, pp. 240 – 247, 2006.

[5] S. Joseph and G. P. Margaret, "Software Defect Prediction Using Enhanced Machine Learning Technique", *International Journal of Innovative Research in Computer and Communication Engineering*, vol. 4, 2016.

[6] T. M. Khoshgoftaar and K. Gao, "Feature Selection with Imbalanced Data for Software Defect Prediction", *International Conference on Machine Learning and Applications*, pp. 235 - 240, 2009.

[7] R. Chang, X. Mu and L. Zhang, “Software Defect Prediction Using Non-Negative Matrix Factorization”, *Journal Of Software*, vol. 6, 2011.

[8] P. Jindal and D. Kumar, “A Review on Dimensionality Reduction Techniques”, *International Journal of Computer Applications,* vol. 173, 2017.

[9] D. R. Ibrahim, R. Ghnemat and A. Hudaib “Software Defect Prediction using Feature Selection and Random Forest Algorithm”, *International Conference on New Trends in Computing Sciences*, pp. 252 – 257, 2017.

[10] H. Lu, E. Kocaguneli and B. Cukic, “Defect Prediction between Software Versions with Active Learning and Dimensionality Reduction”, *IEEE 25th International Symposium on Software Reliability Engineering*, pp. 312 – 322, 2014

[11] H. Lu, B. Cukic and M. Culp, “Software Defect Prediction Using Semi-supervised Learning with Dimension Reduction” *Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering*, pp. 314 – 317, 2012.

[12] Y. Gao, C. Yang and L. Liang, “Software Defect Prediction based on Geometric Mean for Subspace Learning”, *IEEE 2nd Advanced Information Technology, Electronic and Automation Control Conference*, pp. 225 – 229, 2017.

[13] K. Bashir, T. Li, C. Wondaferaw and Y. Mahama, “Enhancing Software Defect Prediction Using Supervised-Learning Based Framework” *12th International Conference on Intelligent Systems and Knowledge Engineering*, pp. 1 – 6, 2017.

[14] M. Bisi and N. K. Goyal, “ Early Prediction of Software Fault-Prone Module using Artificial Neural Network”, *International Journal of Performability Engineering,* pp. 43-52, vol. 11, 2015.

[15] Y. Xia, G. Yan and X. Jiang and J. Yang, “A New Metrics Selection Method for Software Defect Prediction”, *IEEE International Conference on Progress in Informatics and Computing*, pp. 433 – 436, 2014.

[16] P. Wang, C. Jin and S. Jin, “Software Defect Prediction Scheme Based on Feature Selection”, *Fourth International Symposium on Information Science and Engineering,* pp. 477 – 480, 2012.

[17] G. K. Armah, G. Luo and K. Qin, “Multi_Level Data Pre_Processing for Software Defect Prediction ”, *6th International Conference on Information Management, Innovation Management and Industrial Engineering,* pp. 170 – 174, 2013.

[18] H. Ji, S. Huang, Y. Wu, Z. Hui and X. Lv, “A New Attribute Selection Method Based on Maximal Information Coefficient and Automatic Clustering” *Fourth International Conference on Dependable Systems and Their Applications,* pp. 22 – 28, 2017.

[19] M. Kakkar and S. Jain,”Feature Selection in Software Defect Prediction: A Comparative Study’, *6th International Conference - Cloud System and Big Data Engineering (Confluence),* pp. 658 – 663, 2016.

[20] S. Liu, X. Chen, W. Liu, J.Chen, Q.Gu and D. Chen*, “*FECAR: A Feature Selection Framework for Software Defect Prediction”, *IEEE 38th Annual Computer Software and Applications Conference,* pp. 426 – 435, 2014.

[21] S. A. Putri and Frieyadie, “Combining Integreted Sampling Technique with Feature Selection for Software Defect Prediction”, *5th International Conference on Cyber and IT Service Management,* pp. 1 – 6, 2016

[22] W. Han, C. Lung and S. Ajila, “Using Source Code and Process Metrics for Defect Prediction - A Case Study of Three Algorithms and Dimensionality Reduction”, *Journal of Software*, pp. 883-902, vol. 11, 2016.

[23] H. Wang, T. M. Khoshgoftaar and Amri Napolitano,“A Comparative Study of Ensemble Feature Selection Techniques for Software Defect Prediction”, *Ninth International Conference on Machine Learning and Applications,* pp. 135 – 140, 2010.

[24] Z. A. Rana, M. M. Awais and S. Shamail, “Impact of Using Information Gain in Software Defect Prediction Models”, *Intelligent Computing Theory* 2014, pp. 637–648, 2014.

[25] L. Miao, M. Liu and D. Zhang, “Cost-Sensitive Feature Selection with Application in Software Defect Prediction”, *21st International Conference on Pattern Recognition*, pp. 967 – 970, 2012.

[26] R. Verma and A. Gupta , “An approach of Attribute Selection For Reducing False Alarms”, *International Conference on Software Engineering*, pp. 1 – 7, 2012.

[27] R. Malhotra, L. Bahl ,S. Sehgal, P. Priya, “Empirical comparison of machine learning algorithms for bug prediction in open source software”, *2017 International Conference on Big Data Analytics and Computational Intelligence*, pp. 40-45, 2017.

[28] V. Maaten, Laurens & Postma and Eric & Herik, “Dimensionality Reduction: A Comparative Review”, *Journal of Machine Learning Research*, 2007.

[29] O.Moein, S.Yasser, R. Mohammad and T. Akbarzadeh, “Pre-Training of an Artificial Neural Network for Software Fault Prediction”, *7th International Conference on Computer and Knowledge Engineering*, pp. 223 – 228, 2017.

[30] G.N.Ramadevi and K.Usharani, “Study on Dimensionality Reduction Techniques and Applications”, *International Journal Publications of Problems and Applications in Engineering Research*, pp. 134-140, vol. 04, 2013.

[31] P. Chenna, “Comparative Study of Dimension Reduction Approaches With Respect to Visualization in 3-Dimensional Space", *Master of Science in Computer Science Theses*, Kennesaw State University 2016, Accessed on: Feb 16, 2018. Available: https://pdfs.semanticscholar.org/c457/9b0368027c216e397aea1b29e0eaa4b08fd0.pdf.

[32] H. Yan and H. Tianyu, “Unsupervised Dimensionality Reduction for High-Dimensional Data Classification”, *Machine Learning Research*, pp. 125-132, vol. 2, 2017.

[33] O. Saini and S. Sharma, “A Review on Dimension Reduction Techniques in Data Mining”, *Computer Engineering and Intelligent Systems,* vol.9, pp. 7-14, ,2018.

[34] S.Chitra, 2dr.G.Balakrishnan, “A Survey Of Face Recognition On Feature Extraction Process Of Dimensionality Reduction Techniques”, *Journal of Theoretical and Applied Information Technology,* vol. 36, 2012.

[35] N.Varghese, V.Verghese, P. Gayathri and N. Jaisankar, “A Survey of Dimensionality Reduction and Classification Methods”, *International Journal of Computer Science & Engineering Survey*, vol.3, 2012.

[36] Q. Meng, D. Catchpooley, D. Skillicornz and P. J. Kennedy, “Relational Autoencoder for Feature Extraction”, *International Joint Conference on Neural Networks*, pp. 364 – 371, 2017.

[37] N. Sharma and K. Saroha, “Study of Dimension Reduction Methodologies in Data Mining”, International Conference on Computing, Communication & Automation, pp. 133 – 137, 2015.

[38] F. Vahid and T. D. Givargis, *Embedded System Design: A Unified Hardware/ Software Introduction*. Wiley 2001, Available at- http://as.wiley.com/WileyCDA/WileyTitle/productCd0471.html.

[39] Z. M. Hira and D. F. Gillies, “A Review of Feature Selection and Feature Extraction Methods Applied on Microarray Data”, *Advances in Bioinformatics*, vol.4, 2015.

[40] J. Demšar, “Statistical comparisons of classifiers over multiple data sets”, *J Machine Learning*, vol. 7, pp. 1–30, 2006.

[41] A. K. Noulas and B. J. A. Krose, “Deep Belief Networks for dimension reduction”, *Belgian-Dutch Conference on Artificial Intelligence*, pp. 185-191, 2008.