Empirical Analysis of Machine Learning Algorithms (supervised and unsupervised) on Software Defect Prediction Datasets

Report submitted in fulfillment of the requirements
for the Exploratory Project of

Second Year B.Tech.

by

Subhash and Yogesh Singh

Under the guidance of

Dr. Amrita Chaturvedi

Department of Computer Science and Engineering
INDIAN INSTITUTE OF TECHNOLOGY (BHU) VARANASI
Varanasi 221005, India
May 2017

Dedicated to

My parents, teachers,.....

Declaration

We certify that

1. The work contained in this report is original and has been done by ourselves and the general supervision of my supervisor.

2. The work has not been submitted for any project.

3. Whenever we have used materials (data, theoretical analysis, results) from other sources, we have given due credit to them by citing them in the text of the thesis and giving their details in the references.

4. Whenever we have quoted written materials from other sources, we have put them under quotation marks and given due credit to the sources by citing them and giving required details in the references.

Place: IIT (BHU) Varanasi
Date: May 8, 2022

Subhash and Yogesh Singh
B.Tech. Students
Department of Computer Science and Engineering,
Indian Institute of Technology (BHU) Varanasi,
Varanasi, INDIA 221005.

Certificate

*This is to certify that the work contained in this report entitled "**Empirical Analysis of Machine Learning Algorithms (supervised and unsupervised) on Software Defect Prediction Datasets**" being submitted by **Subhash (Roll No. 20075089)** and **Yogesh Singh (Roll No. 20075099)**, carried out in the Department of Computer Science and Engineering, Indian Institute of Technology (BHU) Varanasi, is a bona fide work of our supervision.*

Place: IIT (BHU) Varanasi
Date: May 8, 2022

Dr. Amrita Chaturvedi
Department of Computer Science and Engineering,
Indian Institute of Technology (BHU) Varanasi,
Varanasi, INDIA 221005.

Acknowledgments

We would like to express our sincere gratitude to our professor Dr. Amrita Chaturvedi, Assistant Professor, Department of Computer Science and Engineering, Indian Institute of Technology (B.H.U.) Varanasi, and her teaching assistant Mr. Rakesh Kumar for guiding us at each and every step of our exploratory project. Without their assistance, it would have been an uphill task to arrive at the conclusions reached in this report.

Place: IIT (BHU) Varanasi

Date: May 8, 2022 **Subhash and Yogesh Singh**

Abstract

Predicting problem-prone components before the testing stage of the software development process is the goal of software defect prediction studies. The most significant advantage of these prediction models is that additional testing resources may be properly devoted to fault-prone modules. While a few software defect prediction models have been created, there is still a need for a thorough review of these findings. As a result, we conducted a study to assess how machine learning has been used to predict software faults.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overview

Software quality is one of the essential aspects of a software. With increasing demand, software designs are becoming more complex, increasing the probability of software defects. The defects hidden in software modules threaten the security and decrease the reliability of the software product. Therefore, it is essential to fix the defective modules before delivering the product.

1.2 Motivation of the Research Work

Defect fixing is a complex and time-consuming task, and limited testing resources are usually unaffordable for supporting thorough code reviews. This requests a prioritization to better analyze the software product. In other words, developers and testers should reasonably allocate the limited resources to test the modules that have a high probability to contain defects. To seek for such prioritization, Software Defect Prediction (SDP) is proposed to identify the most defect-prone modules for priority inspection. The most active SDP methods are supervised models which first train a classifier on labeled modules and then use it to determine whether or not the unla-

beled modules contain defects. However, the supervised SDP models need the labeled modules of historical data of the current project or external projects which are not always available. In order to conduct defect prediction on unlabeled data, Unsupervised Defect Prediction (UDP) models are possible for this task, as UDP models do not need any labeled data.

1.3 Organisation of the Report

First we have given the breif introduction about Software Defect Prediction. We have used 17 datastes (12 publicly available NASA datasets which are collected from the PROMISE repository). Section 2.2.1 introduces the SMOTE technique for handeling the imbalanced data. We used 5 supervised and 11 unsupervised clustering based algorithms. Next there is description about feature selection technique in which we used Recursive Feature Selection (RFE) for choosing the required features. For unsupervised models labelling was done with the help pf SFM technique. Section 2.6 introduces the evaluation indicators used for our models. Next section reports our experimental results. In last we conclude our report.

Chapter 2

Project Work

2.1 Software Defect Prediction Using Machine Learning

Since the 1990s, software defect prediction studies have been using machine learning algorithms to identify fault-prone classes. The problem of software defect prediction is considered as a binary classification problem, where a class is either "defect prone"or "not defect prone". A module can either contain faults or can be defect free (not defect prone). Machine learning has been applied for both predicting the number of faults (i.e., a regression task) and categorization of modules into fault-prone and non-fault-prone classes (i.e., binary class classification). In machine learning, there are four learning types: supervised, unsupervised, semi-supervised, and reinforcement learning. In supervised learning, labeled data are needed to build the models. In unsupervised learning, hidden structures in data are discovered by detecting the feature correlations. Clustering and dimensionality reduction algorithms are considered under the unsupervised learning category.

2.2 Handeling the Imbalaced Data

For our analysis on software defect prediction we used 17 datsets out of which 12 are NASA datasets. The number of modules and number of features of each dataset has been listed in the table 2.1

Dataset	No. of Modules	No. of Features
ar1	121	30
ar3	63	30
ar4	107	30
ar5	36	30
ar6	101	30
cm1	327	38
JM1	7782	22
kc2	522	22
KC3	194	40
MC1	1988	39
MC2	125	40
MW1	253	38
PC1	705	38
PC2	745	37
PC3	1077	38
PC4	1287	38
PC5	1711	39

Table 2.1: Dataset Table

There are a number of previous studies which have used NASA data sets for defect prediction. They have demonstrated that 80 percentage of the defects occur in very few modules (20 perc.). This indicates that defective classes are present in minority as compared to non-defective classes, which results in imbalanced datasets. In such cases, the distribution of classes is one-sided, which may result in incorrect prediction of the minority class instances. Although, the minority class instances are low in number, but it is important to classify them correctly. Incorrect prediction of defective classes might result in escape of critical errors leading to bad quality software and higher testing costs. Therefore, it is important to address imbalanced

data problem for software defect prediction to improve software quality, to reduce prediction error and for successful deployment of the software.

There are many techniques to handle imbalanced datasets on various levels like data level, algorithm level, cost sensitive level, feature selection level and ensemble level.This study explores data level approach. In data level approach we specifically focus on oversampling methods.

This simplest approach involves duplicating examples in the minority class, although these examples don't add any new information to the model. Instead, new examples can be synthesized from the existing examples. This is a type of data augmentation for the minority class and is referred to as the Synthetic Minority Oversampling Technique, or SMOTE for short .[1]

2.2.1 SMOTE

SMOTE works by selecting examples that are close in the feature space, drawing a line between the examples in the feature space and drawing a new sample at a point along that line. Specifically, a random example from the minority class is first chosen. Then k of the nearest neighbors for that example are found (typically k=5). A randomly selected neighbor is chosen and a synthetic example is created at a randomly selected point between the two examples in feature space.

2.3 Algorithms

In our study we are using both supervised and unsupervised algorithms. The algorithms are -

Supervised Algorithms

- Naive Bayes (NB)

- Support Vector Machine (SVM)
- K Nearest Neighbours (KNN)
- Random Forest (RF)
- Logistic Regression (LR)

Unsupervised Algorithms

- K-means (KMS)
- Mini Batch K-means (MBM)
- Agglomerative Heirarchical Clustering (AHC)
- Balanced iterative reducing and clustering using hierarchies (BIRCH)
- Spectral Clustering (SC)
- Guassian
- Affinity Propagation (AP)
- Ordering Points To Identify Clustering Structure (OPTICS)
- Mean Shift (MS)
- Density-Based Spatial Clustering of Applications with Noise (DBSCAN)
- X -means

2.4 Feature Selection Technique

To train an optimal model, we need to make sure that we use only the essential features. If we have too many features, the model can capture the unimportant patterns and learn from noise. Apart from choosing the right model for our data, we need to choose the right data to put in our model. Wrapper feature selection methods create many models with different subsets of input features and select those features that result in the best performing model according to a performance metric. These methods are unconcerned with the variable types, although they can be computationally expensive. Recursive Feature Elimination (RFE) is a good example of a wrapper feature selection method.

2.4.1 Recursive Feature Elimination (RFE)

A different machine learning algorithm is given and used in the core of the method, is wrapped by RFE, and used to help select features. This is in contrast to filter-based feature selections that score each feature and select those features with the largest (or smallest) score. Technically, RFE is a wrapper-style feature selection algorithm that also uses filter-based feature selection internally. RFE works by searching for a subset of features by starting with all features in the training dataset and successfully removing features until the desired number remains. This is achieved by fitting the given machine learning algorithm used in the core of the model, ranking features by importance, discarding the least important features, and re-fitting the model. This process is repeated until a specified number of features remains.

2.5 Labelling Technique

In Unsupervised defect prediction data is divided into two clusters. But we dont know that which cluster is defective and which cluster is non-defective. So for labelling these

clusters we are using SFM technique.[2]

For the methods with predefined cluster number as 2 we first calculates the Sum of Feature values of each Module (SFM), then calculates the Average value of the SFMs (ASFM) for all modules in each cluster. The cluster with larger ASFM is labeled as defective, and another cluster is labeled as non-defective.

For the methods without predefined cluster numbers (multiple-cluster scenario), we first calculated the ASFMs for all clusters and the Mean values of these ASFMs (MASFM). Then, we labeled the clusters whose ASFMs are not less than MASFM as defective and labeled other clusters as non-defective. In other words, we used the average values on all features in each cluster to determine it class label.

2.6 Evaluation Indicators

To measure effectiveness of our models for SDP, we employed 3 indicators as our performance measurement inluding Accuracy, Matthew Correlation Coefficient (MCC) and F1-score.[3]

We first defined 4 basic terms as follows:

- True Positive (TP) - These are the number of modules that are defective and correctly identified by a model.
- False Negative (FN) - These are the number of modules that are defective but incorrectly identified by a model.
- True Negative (TN) - These are the number of modules that are non-defective and correctly identified by a model.
- False Postive (FP) - These are the number of modules that are non-defective but incorrectly identified by a model.

2.6. Evaluation Indicators

These terms were calculated with the help of confusion matrix. For our indicators we defined two more terms -

- Precision - It is defined as the fraction of relevant examples (true positives) among all of the examples which were predicted to belong in a certain class.

$$Precision = \frac{TP}{TP+FP}$$

- Recall - It is defined as the fraction of examples which were predicted to belong to a class with respect to all of the examples that truly belong in the class.

$$Recall = \frac{TP}{TP+FN}$$

1. Accuracy - It is defined as the percentage of correct predictions for the test data. It can be calculated easily by dividing the number of correct predictions by the number of total predictions.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$$

2. MCC - With the help of above four terms MCC can be calculated as follows :

$$MCC = \frac{TP*TN-FP*FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}$$

3. F1-score - The F1-score combines the precision and recall of a classifier into a single metric by taking their harmonic mean.

$$F1 - score = \frac{2*(precision*recall)}{precision+recall}$$

Chapter 3

Experimental Results

Since we needed to perform a total of 16 methods (11 unsupervised models and 5 supervised models) on 17 defect data, we obtained 272 records of the performance results.

Table 3.1 , 3.2 and 3.3 show the accuracy, f1-score and MCC of our models respectively.

			PERFORMANCE OF DIFFERENT FAULT PREDICTION TECHNIQUES IN TERMS OF ACCURACY														
Projects	Supervised Methods						Unsupervised Methods										
	NB	SVM	KNN	RF	LR	MAX	KMS	MBM	AHC	BIRCH	SC	GUASSIAN	Affinity Propagation	OPTICS	Mean Shfit	DBSCAN	X-means	MAX
AR1	0.79167	0.91731	0.90962	0.91731	0.88397	0.91731	0.9090909	0.91736	0.92562	0.89256	0.09091	0.92562	0.54018	0.64732	0.535714	0.5625	0.55804	0.92562
AR3	0.90714	0.85714	0.84048	0.89048	0.87381	0.90714	0.9047619	0.90476	0.90476	0.90476	0.15873	0.90476	0.54545	0.7	0.609091	0.5045	0.71818	0.90476
AR4	0.86	0.85091	0.84182	0.84091	0.81091	0.86	0.8411215	0.85981	0.86916	0.86916	0.8685	0.85981	0.63529	0.51149	0.66092	0.51149	0.57471	0.86916
AR5	0.91333	0.88	0.91667	0.89667	0.91667	0.91667	0.8611111	0.86111	0.86111	0.86111	0.80556	0.80556	0.50877	0.69643	0.535714	0.50877	0.67857	0.86111
AR6	0.82091	0.85091	0.83091	0.80182	0.86182	0.86182	0.7821782	0.78218	0.79208	0.79208	0.15842	0.79208	0.50289	0.48837	0.540698	0.53757	0.54651	0.79208
cm1	0.81648	0.87169	0.85019	0.85635	0.8626	0.87169	0.8287462	0.82263	0.78593	0.78593	0.82263	0.8685	0.86103	0.56575	0.865854	0.1311	0.85627	0.8685
JM1	0.55246	0.56874	0.70426	0.86768	0.27815	0.86768	0.7887433	0.76433	0.78861	0.78861	0.72113	0.82263	0.78615	0.77744	0.785944	0.21496	0.78617	0.82263
kc2	0.84822	0.80777	0.78099	0.77144	0.78879	0.84822	0.802682	0.83525	0.80268	0.80268	0.70704	0.80268	0.51928	0.59036	0.509639	0.63494	0.51928	0.83525
kc3	0.78368	0.81526	0.78974	0.81474	0.78447	0.81526	0.7783505	0.7732	0.73196	0.73196	0.73196	0.73196	0.50633	0.56962	0.506329	.	0.55063	0.77835
mc1	0.92959	0.97686	0.97636	0.97837	0.97133	0.97837	0.9763581	0.88732	0.97636	0.97636	0.648	0.97535	0.97289	0.49748	0.97637	0.02363	0.97636	0.97637
mc2	0.71218	0.71154	0.68013	0.72756	0.71859	0.72756	0.696	0.688	0.704	0.704	0.648	0.696	0.66667	0.496	0.674603	0.35714	0.696	0.704
mw1	0.81415	0.89323	0.88123	0.86539	0.89723	0.89723	0.743083	0.75099	0.73913	0.73913	0.80636	0.72332	0.88716	0.56126	0.88189	0.11024	0.86957	0.88716
pc1	0.87962	0.91348	0.90207	0.92201	0.91354	0.92201	0.8737589	0.87376	0.83404	0.86809	0.91489	0.77305	0.90268	0.50922	0.915014	0.08782	0.90638	0.91501
pc2	0.91409	0.97858	0.97858	0.97858	0.97451	0.97858	0.9691275	0.85235	0.97584	0.97584	0.91489	0.9745	0.95728	0.52081	0.97319	0.02279	0.96376	0.97584
pc3	0.206	0.87562	0.85614	0.8756	0.87651	0.87651	0.8727948	0.82266	0.87279	0.87279	.	0.87372	0.86957	0.49954	0.871985	0.12523	.	0.87372
pc4	0.86007	0.86162	0.83988	0.89661	0.87868	0.89661	0.8166278	0.79332	0.72883	0.81274	0.648	0.74592	0.85205	0.79876	0.863354	0.1382	0.85859	0.86335
pc5	0.7405	0.73816	0.68849	0.77439	0.73174	0.77439	0.7305669	0.72998	0.72297	0.73524	0.648	0.68323	0.73178	0.72881	0.730724	0.2757	0.73232	0.73524
Average	0.7853	0.84522	0.83927	0.86329	0.8249	0.86329	0.8338296	0.81876	0.82446	0.83018	0.64331	0.8211	0.72032	0.59757	0.73159	0.29666	0.73696	0.83383

Table 3.1: Performance of different algorithms in terms of accuracy

PERFORMANCE OF DIFFERENT FAULT PREDICTION TECHNIQUES IN TERMS OF F1-SCORE

Projects	Supervised Methods						Unsupervised Methods											
	NB	SVM	KNN	RF	LR	MAX	KMS	MBM	AHC	BIRCH	SC	GUASSIAN	AP	OPTICS	MS	DBSCAN	X-means	MAX
AR1	0.33952	0.33524	0.61477	0.68445	0.64285	0.68445	0.2666667	0.16667	0.18182	0.38095	0.14063	0.6087	0.176	0.66667	0.147541	0.69565	0.25564	0.69565
AR3	0.54278	0.53257	0.52678	0.60857	0.56524	0.60857	0.6666667	0.66667	0.66667	0.66667	0.23188	0.66667	0.16667	0.74809	0.358209	0.66667	0.63529	0.74809
AR4	0.56862	0.55565	0.63188	0.76918	0.64257	0.76918	0.32	0.48276	0.53333	0.53333	0.04444	0.44444	0.04494	0.528	0.044944	0.66667	0.27451	0.66667
AR5	0.72064	0.6873	0.69841	0.70952	0.7	0.72064	0.7368421	0.73684	0.73684	0.73684	0.22222	0.22222	0.66667	0.76056	0.1875	0.66667	0.57143	0.76056
AR6	0.33725	0.27	0.57212	0.81567	0.55591	0.81567	0.2666667	0.26667	0.22222	0.22222	0.26087	0.32258	0.66667	0.52688	0.275229	0.68254	0.30357	0.68254
cm1	0.40977	0.40048	0.57959	0.80561	0.60763	0.80561	0.2631579	0.29268	0.33962	0.33962	0.29268	0.04444	0.04167	0.25263	0.043478	0.22764	0.11321	0.33962
JM1	0.23169	0.3379	0.64094	0.71386	0.49105	0.71386	0.1189711	0.38124	0.15771	0.15771	0.81495	0.29268	0.02231	0.19815	0.009512	0.35371	0.0107	0.81495
kc2	0.35294	0.38241	0.4652	0.50607	0.42828	0.50607	0.0720721	0.54737	0.07207	0.07207	0.80102	0.07207	0.07425	0.57816	0.037825	0.72727	0.07425	0.80102
kc3	0.43343	0.34859	0.55896	0.74198	0.50076	0.74198	0.358209	0.37143	0.40909	0.40909	0.45833	0.45833	0.03704	0.56129	0.037037	.	0.25263	0.56129
mc1	0.33934	0.11692	0.56044	0.82981	0.47187	0.82981	0.0408163	0.125	0.04082	0.04082	0.04348	0.03922	0.06897	0.05666	0.078431	0.04523	0.04082	0.125
mc2	0.40967	0.43732	0.60878	0.7095	0.53524	0.7095	0.3448276	0.31579	0.37288	0.37288	0.04348	0.34483	0.15686	0.4878	0.163265	0.52071	0.26923	0.52071
mw1	0.51438	0.44651	0.51686	0.70685	0.54133	0.70685	0.3298969	0.33684	0.32653	0.32653	0.04082	0.31373	0.12121	0.2649	0.117647	0.19286	0.10811	0.33684
pc1	0.44961	0.39711	0.56947	0.81156	0.60222	0.81156	0.2521008	0.29921	0.27329	0.29008	0.03226	0.26606	0.08	0.18396	0.090909	0.15927	0.05714	0.29921
pc2	0.42742	0.40535	0.52425	0.77843	0.45074	0.77843	0.08	0.14063	0.1	0.1	0.03226	0.09524	0.11111	0.06299	0.166667	0.04205	0.06897	0.16667
pc3	0.55757	0.31692	0.65013	0.80976	0.60139	0.80976	0.2431012	0.28243	0.2431	0.2431	.	0.27898	0.01399	0.23546	0.014286	0.2213	.	0.28243
pc4	0.42736	0.33158	0.6197	0.87234	0.63889	0.87234	0.2337662	0.26923	0.31164	0.21498	0.04348	0.22695	0.07729	0.21752	0.053763	0.2418	0.05208	0.31164
pc5	0.25493	0.33645	0.64735	0.74346	0.51738	0.74346	0.1252372	0.13806	0.06324	0.21217	0.04348	.	0.11538	0.20275	0.053388	0.43171	0.1124	0.43171
AVERAGE	0.43041	0.39049	0.58739	0.74215	0.55843	0.74215	0.2775881	0.34232	0.29711	0.31289	0.22164	0.29357	0.15535	0.38426	0.110567	0.40886	0.2	0.40886

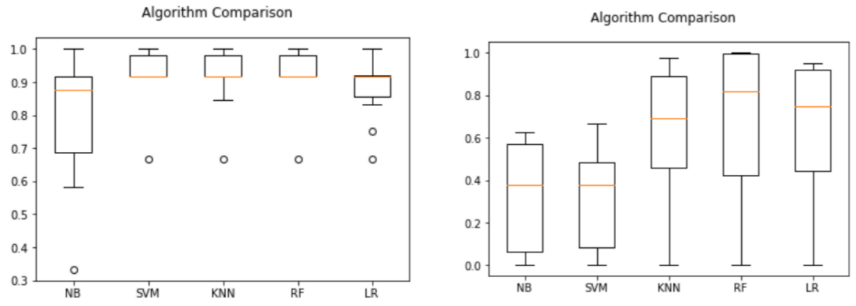
Table 3.2: Performance of different algorithms in terms of f1-score

PERFORMANCE OF DIFFERENT FAULT PREDICTION TECHNIQUES IN TERMS OF MCC

Projects	Supervised Methods						Unsupervised Methods											
	NB	SVM	KNN	RF	LR	MAX	KMS	MBM	AHC	BIRCH	SC	GUASSIAN	AP	OPTICS	MS	DBSCAN	X-means	MAX
AR1	0.1312	0.47761	0.79159	0.83918	0.68103	0.83918	0.2254255	0.15736	0.21029	0.32746	0.03675	0.58674	0.17184	0.29665	0.172935	0.2582	0.19911	0.58674
AR3	0.45136	0.31586	0.85742	0.89049	0.72904	0.89049	0.6170949	0.61709	0.61709	0.61709	0.06906	0.61709	0.21822	0.43279	0.349927	0.09449	0.4899	0.61709
AR4	0.36571	0.57775	0.59852	0.71026	0.66482	0.71026	0.3481883	0.45931	0.50494	0.50494	0.05893	0.45486	0.10783	0.38949	0.107833	0.07538	0.26619	0.50494
AR5	0.82137	0.68625	0.94135	0.93123	0.87368	0.94135	0.6607748	0.66077	0.66077	0.66077	0.31623	0.31623	0.13131	0.46525	0.138675	0.13131	0.41239	0.66077
AR6	0.46762	0.75097	0.63049	0.88624	0.69735	0.88624	0.1387597	0.13876	0.10479	0.10479	0.04176	0.20007	0.07581	0.02357	0.119575	0.20417	0.12985	0.20417
cm1	0.95138	0.70469	0.90111	0.97191	0.94021	0.97191	0.1686822	0.19143	0.2253	0.2253	0.19143	0.05893	0.03814	0.09165	0.05757	0.02119	0.08046	0.2253
JM1	0.36784	0.14571	0.71875	0.73803	0.57511	0.73803	0.1391025	0.24166	0.15701	0.15701	0.43675	0.19143	0.06551	0.14086	0.055821	0.00593	0.05984	0.43675
kc2	0.51599	0.5504	0.62639	0.71424	0.5097	0.71424	0.1730602	0.45403	0.17306	0.17306	0.24791	0.17306	0.1402	0.18103	0.098653	0.36674	0.1402	0.45403
kc3	0.24744	0.43398	0.71813	0.92509	0.713	0.92509	0.2260727	0.23334	0.24997	0.24997	0.31172	0.31172	0.05661	0.13934	0.056614	.	0.16784	0.31172
mc1	0.54579	0.41892	0.42305	0.8428	0.63	0.8428	0.0802098	0.12127	0.08021	0.08021	0.03952	0.05907	0.08148	0.04391	0.140785	0.00345	0.08021	0.14078
mc2	0.43801	0.70466	0.66957	0.71134	0.74982	0.74982	0.2693983	0.24276	0.29485	0.29485	0.03952	0.2694	0.18589	0.07614	0.242933	0.06552	0.28635	0.29485
mw1	0.74384	0.30445	0.45209	0.81158	0.67391	0.81158	0.2440848	0.25247	0.23998	0.23998	0.08021	0.22414	0.15386	0.17329	0.110801	0.02168	0.06129	0.25247
pc1	0.18174	0.64688	0.77777	0.75642	0.76316	0.77777	0.1832794	0.23018	0.19302	0.21837	0.12246	0.18694	0.07431	0.07664	0.176606	0.01158	0.05489	0.23018
pc2	0.51785	0.37075	0.6803	0.73024	0.61544	0.73024	0.0683739	0.17085	0.11582	0.11582	0.12246	0.10123	0.08922	0.07707	0.171479	0.00542	0.05097	0.17148
pc3	0.64038	0.76284	0.74309	0.87182	0.54432	0.87182	0.1842794	0.22139	0.18428	0.18428	.	0.18596	0.00415	0.06739	0.01542	0.01148	.	0.22139
pc4	0.8309	0.1732	0.53675	0.78772	0.68212	0.8309	0.1341821	0.14905	0.17368	0.11321	0.03952	0.08129	0.06336	0.10302	0.111516	0.01113	0.06174	0.17368
pc5	0.30418	0.11397	0.66428	0.75929	0.63044	0.75929	0.1293197	0.13078	0.05884	0.17728	0.03952	0.24734	0.13615	0.15111	0.124334	0.01489	0.13584	0.24734
AVERAGE	0.50133	0.47876	0.69004	0.81635	0.68665	0.81635	0.2347228	0.27485	0.24964	0.26144	0.13711	0.25091	0.10552	0.1723	0.13244	0.08141	0.16732	0.27485

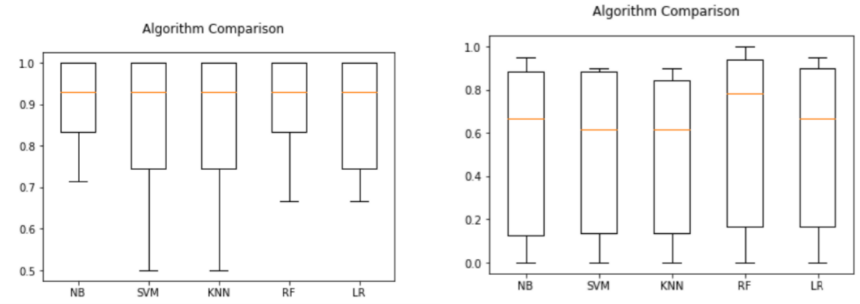
Table 3.3: Performance of different algorithms in terms of MCC

Fig 1 to Fig 17 depict the boxplots of 2 indicators of all supervised models for all 17 datasets.



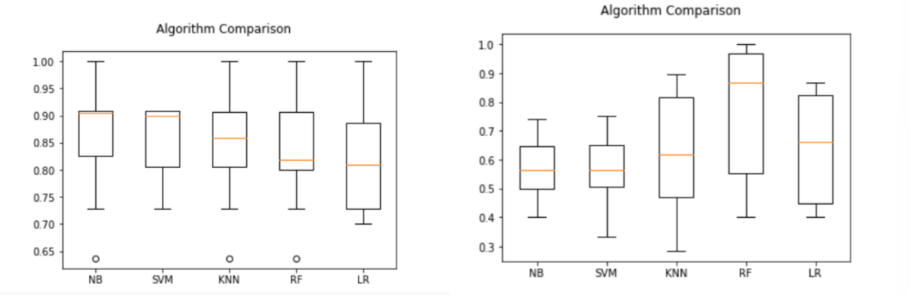
(a) accuracy (b) f1-score

Figure 3.1: Boxplots of supervised models on ar1 dataset



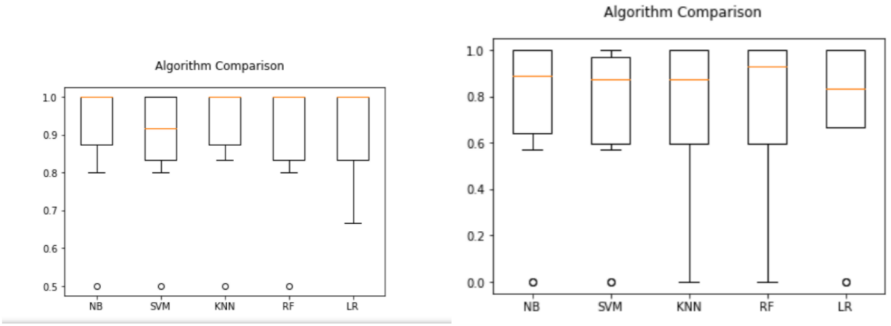
(a) accuracy (b) f1-score

Figure 3.2: Boxplots of supervised models on ar3 dataset



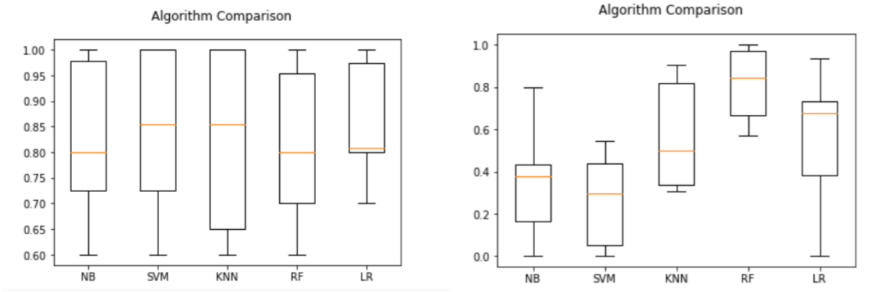
(a) accuracy (b) f1-score

Figure 3.3: Boxplots of supervised models on ar4 dataset



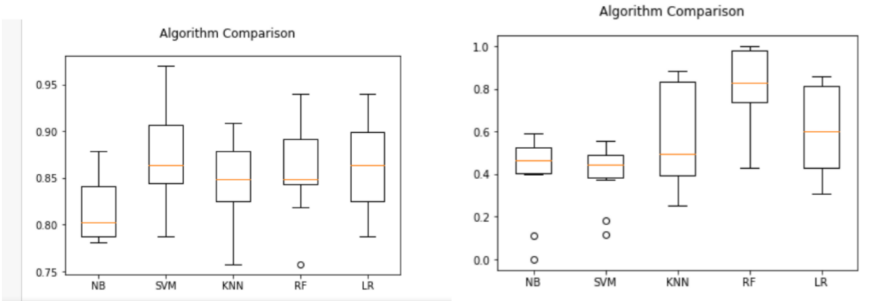
(a) accuracy (b) f1-score

Figure 3.4: Boxplots of supervised models on ar5 dataset



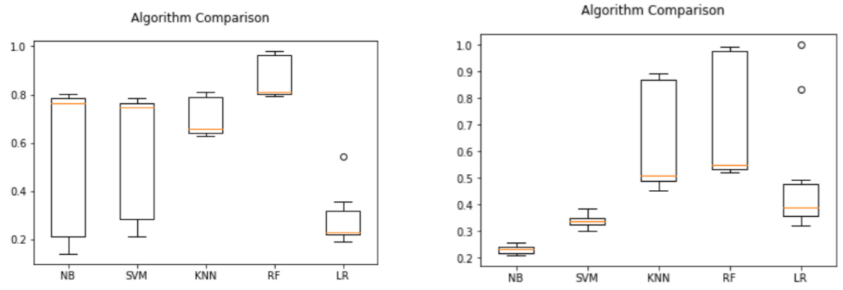
(a) accuracy (b) f1-score

Figure 3.5: Boxplots of supervised models on ar6 dataset



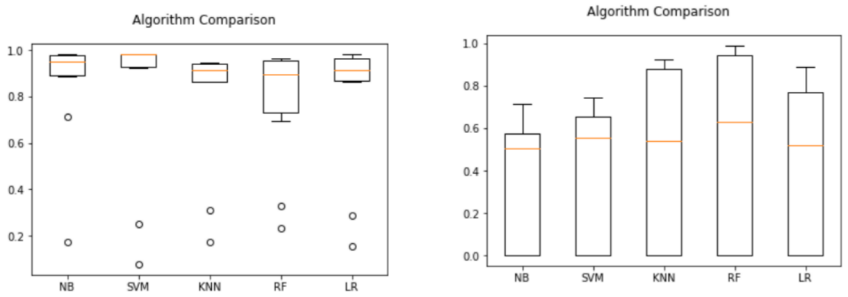
(a) accuracy (b) f1-score

Figure 3.6: Boxplots of supervised models on cm1 dataset



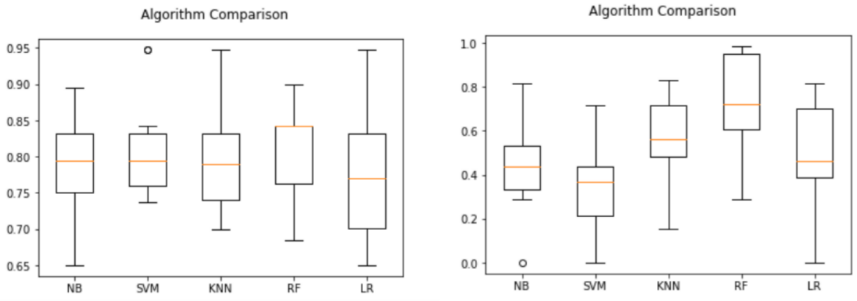
(a) accuracy (b) f1-score

Figure 3.7: Boxplots of supervised models on JM1 dataset



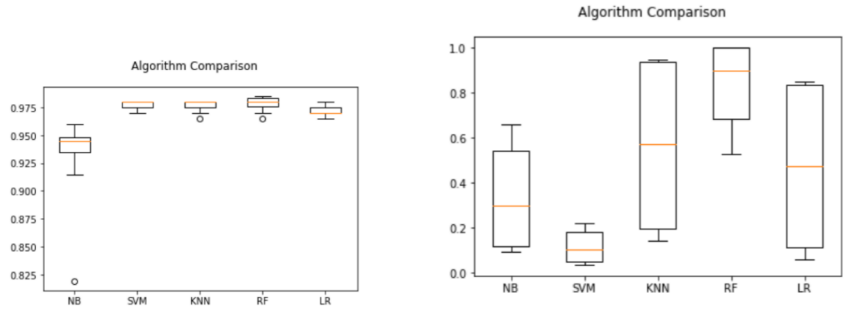
(a) accuracy (b) f1-score

Figure 3.8: Boxplots of supervised models on kc2 dataset



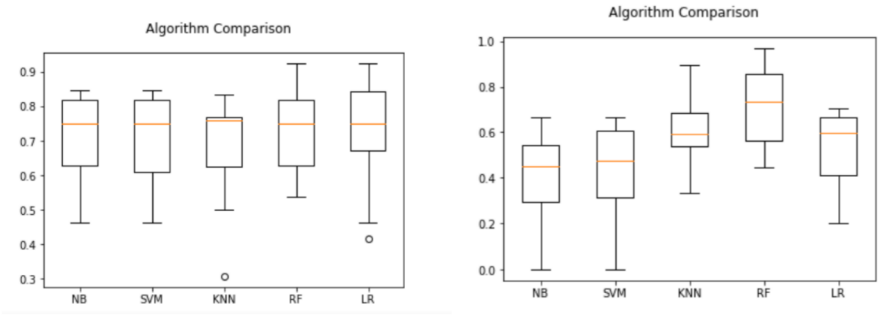
(a) accuracy (b) f1-score

Figure 3.9: Boxplots of supervised models on KC3 dataset



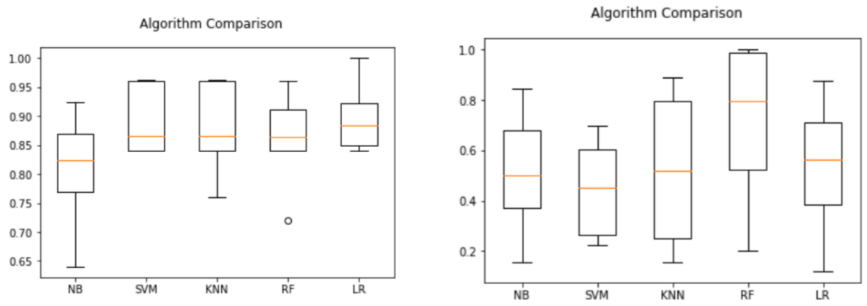
(a) accuracy (b) f1-score

Figure 3.10: Boxplots of supervised models on mc1 dataset



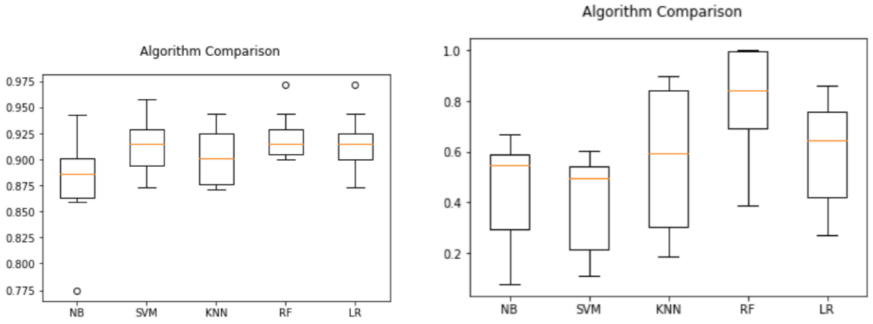
(a) accuracy (b) f1-score

Figure 3.11: Boxplots of supervised models on mc2 dataset



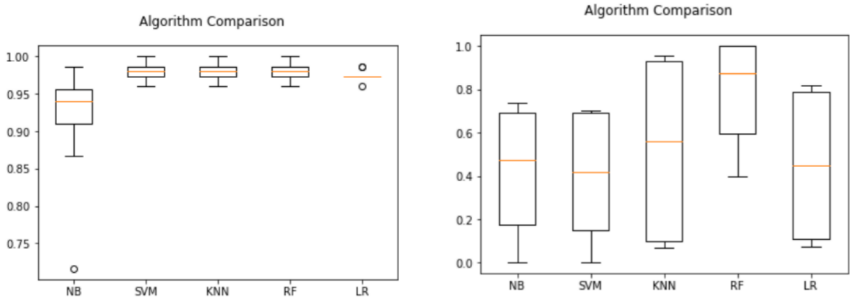
(a) accuracy (b) f1-score

Figure 3.12: Boxplots of supervised models on mw1 dataset



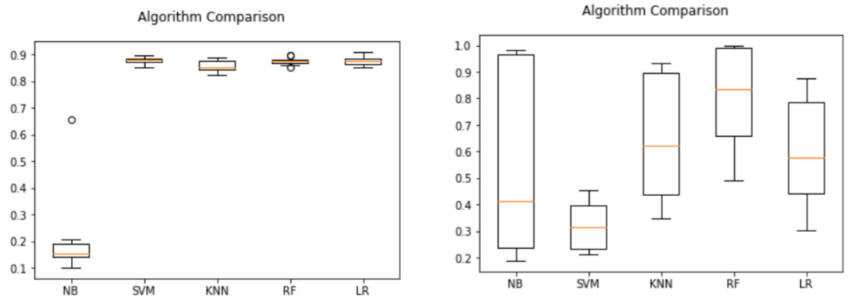
(a) accuracy (b) f1-score

Figure 3.13: Boxplots of supervised models on PC1 dataset



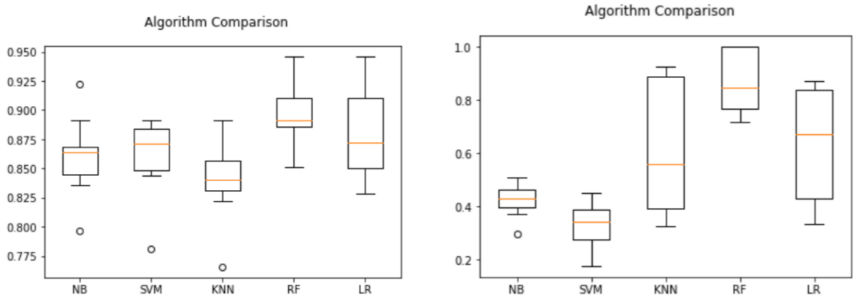
(a) accuracy (b) f1-score

Figure 3.14: Boxplots of supervised models on PC2 dataset



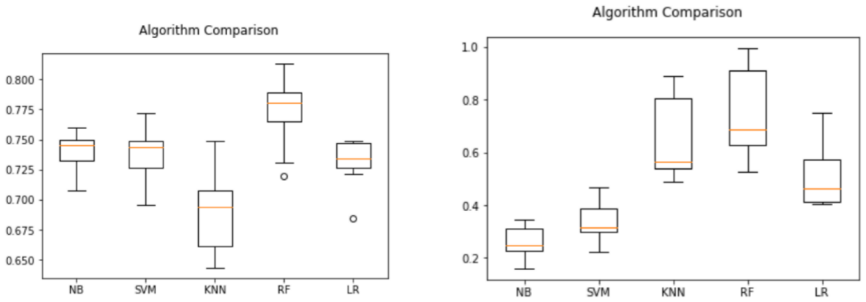
(a) accuracy (b) f1-score

Figure 3.15: Boxplots of supervised models on PC3 dataset



(a) accuracy (b) f1-score

Figure 3.16: Boxplots of supervised models on PC4 dataset



(a) accuracy (b) f1-score

Figure 3.17: Boxplots of supervised models on PC5 dataset

1. In terms of accuracy

From Table 3.1, we can observe that, for a particular dataset - In supervised model family except KNN all other models performed well.Also Random Forest (RF) performed best among all other models.In unsupervised models, KMS, BIRCH, Guassian and MS models performed better than other models.

If we talk about average performance of each model over all datasets, Rf is the best model among all supervised clustering models and KMS performed best among all unsupervised models.

2. In terms of f1-score

From Table 3.2, for a particular dataset, we can say that among all classifiers in the supervised model family RF outperformed other models. Also we can place KNN at second position. In unsupervised model family, MBM, OPTICS and DBSCAN performed better than other models.

If we see average performance of these models, RF is the best model in supervised model family and DBSCAN performed better in unsupervised model family.

2. In terms of MCC

From Table 3.3, we observe that, for a particular dataset among all classifiers in the supervised model family RF performed best. In unsupervised model family, MBM, BIRCH and Guassian performed better than other models.

If we see average performance of these models, RF is the best model in supervised model family and MBM performed better in unsupervised model family.

Chapter 4

Conclusions and Discussion

We conducted a small-scale comparison to analyze SDP performance differences among 11 clustering-based unsupervised models and 5 typical supervised models. We made the first step towards investigating the impacts of the feature types of defect data on the performance of these methods. Since the defect data was highly imbalanced so we also used oversampling methods to balance the data.

Our experimental results on 17 defect data indicate that not all clustering-based unsupervised models are worse than the supervised models.

Firstly we are using accuracy performance indicator-

If we compare supervised algorithms only, RF performed better over other algorithms.

In unsupervised models , KMS model performed better over other models.

If we see MCC performance of our models we find that RF in supervised model family and MBM in unsupervised model family performed better.

But when we have an imbalanced data ,accuracy is not a good metric. So we are making conclusion on the basis of f1-score.

To sum up, on defect data we can say that Random Forest in supervised model family and DBSCAN in unsupervised model family performed best on f1-score performance indicator.

Bibliography

[1] R. Malhotra and S. Kamal, "An empirical study to investigate oversampling methods for improving software defect prediction using imbalanced data," *Neurocomputing*, vol. 343, pp. 120–140, 2019.

[2] F. Zhang, Q. Zheng, Y. Zou, and A. E. Hassan, "Cross-project defect prediction using a connectivity-based unsupervised classifier," in *2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE)*. IEEE, 2016, pp. 309–320.

[3] Z. Xu, L. Li, M. Yan, J. Liu, X. Luo, J. Grundy, Y. Zhang, and X. Zhang, "A comprehensive comparative study of clustering-based unsupervised defect prediction models," *Journal of Systems and Software*, vol. 172, p. 110862, 2021.