Software Defect Prediction Analysis using Machine Learning Algorithms

Data Mining Course, Bachelor in Technology
Department of Information Technology, IIIT Allahabad

Authors :
Abhishek Kumar Gupta IIT2018187
Puja kumari IIT2018191
Prabha Kumari IIT2018195

Under Supervision of:
Prof. OP Vyas
DMW Course Instructor
IIIT Allahabad

***ABSTRACT* :In this paper, we propose to extract a set of specifics from the first set of basic transformation measures using the Artificial Neural Network (ANN), and then train to differentiate according to the extracted elements using the decision tree and compare it to the other three between Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA) and Kernel PCA. We use a database of five open source sources from the NASA Promise Data Repository to conduct this comparative study. To test, three widely used metrics are used: Accuracy, F1 scores and areas below the Receiver Operating Curve feature.**

Keywords- *Machine Learning, Artificial Neural Network, Principal Component Analysis, Linear Discriminant Analysis, Kernel PCA, Decision Tree, Area under ROC curve*

INTRODUCTION

In order to build high-quality softwares, feature prediction has become a priority as a lot of time and effort is put into software testing and its use of errors in some other way. False prediction methods are suggested to help prioritize software testing and debugging; can recommend software components that may be problematic for developers. **[1]** Many parameters are considered when predicting whether the software is an organization or not including the number of lines of code, its complexity, the number of operators and operators used in the code and other factors. We looked at a set of the first 22 features to predict whether the module is a disruptive entity.

Artificial Neural Network is a machine learning algorithm based on the functioning of neural biological networks. It has many areas connected by heavy edges. We propose to extract a set of explicit elements in the first set of basic transformation measures using the

Artificial Neural Network (ANN) and then train to differentiate according to the extracts from the decision tree and compare it to the three alternatives in which Analysis (PCA), Linear Discriminant Analysis (LDA) and Kernel PCA. Decisions Trees fall under a supervised learning approach to planning and reversal that can be easily identified. It works by splitting or reversing according to a specific task (here Entropy) in a training set with a label. It divides the population or sample on the basis of the most important separator by identifying the most important variant from the database. They are working on the goal of Selfishness.

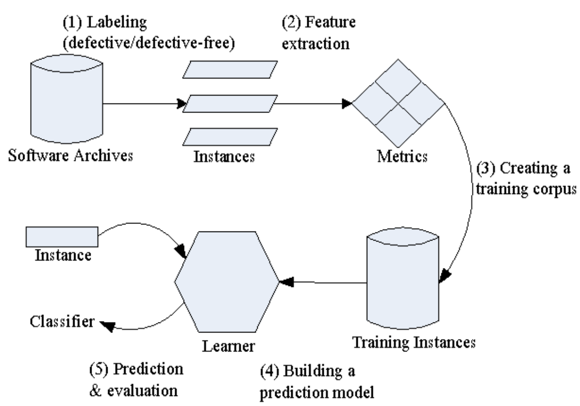


Figure1 : Defect Prediction Process

PCA, LDA and Kernel PCA methods are used to reduce size. LDA and PCA are straightforward conversion strategies, the difference being that the LDA should be monitored and the PCA not monitored. PCA is a more effective way to reduce size while LDA is often very specific. The PCA manages the entire database while the LDA tries to discriminate between classes within the data. On the other hand, KPCA is not a straightforward form of PCA but it is like an extension of PCA

To test, three widely used metrics are used: Accuracy, F1 scores and areas below the Receiver Operating Curve feature. **[12]** The accuracy of the phase alone can be misleading at times so the other two metrics are also considered. F1 scores are rated with Precision rating and Recall scores False Positives and False Negatives. Basically, it is a harmonic definition of the two. Generally ROC curves can be said as a complete report of sensitivity and specificity. It has been found that the Artificial Neural Network surpasses all other ways to reduce size. The Kernel PCA has done very well among other ways to reduce size.

BACKGROUND

Here,we introduce the background of defect prediction technique.

Defect Prediction: The process of predicting code areas that contain defects is called Software defect prediction. It helps developers allocate their testing efforts by first checking buggy code. It ensures the reliability of large -scale software. Below given figure represents file-level defect prediction process.

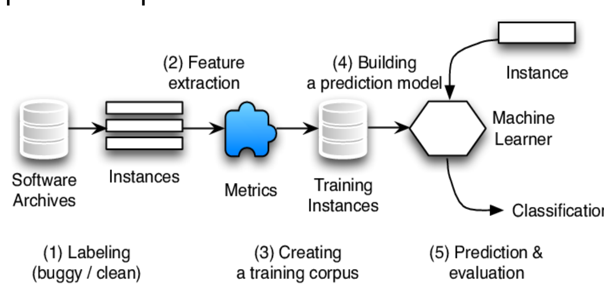


Fig2 : File-level defect prediction process

Algorithm of file level defect prediction process is following:-

1.Collect source code files (instances) from archives and level them as buggy or clean.

- File containing at least one post-release bug is labeled as buggy .
- Otherwise the file is labeled as clean.

2.Extract features (code metrics and CK features) from each file.

- The instances with the corresponding features and labels to train classifiers using algorithms like ANN,LDA,PCA,KPCA.

3.New instances are fed into trained classifier to predict whether the files are buggy or clean.

LITERATURE SURVEY

S.No.	**Paper Title**	**Name of the Conference/journal (Year)**	**Methodology**	**Results**	**Paper Link**
1.	Deep learning for just-in-time defect prediction	in QRS'15: Proc. of the International Conference on Software Quality, Reliability and Security, 2015	used learning algorithms to predict defects at change level. They made use of a deep learning algorithm to predict the same. They first created a Deep Belief Network to extract a set of expressive features from the initial set of linear features and then used Logistic Regression as a classifier to predict buggy and non-buggy changes.	Compare the result using accuracy f1 score and area under Roc	https://ieeexplore.ieee.org/document/7272910
2.	A large-scale empirical study of just-in-time quality assurance	IEEE Transactions on Software Engineering (Volume: 39, Issue: 6, June 2013)	In this paper we consider defect prediction models that focus on identifying defect-prone ("risky") software changes instead of files or packages.To build a change risk model, we use a wide range of factors based on the characteristics of a software change, such as the number of added lines, and developer experience	Predictor achieves an average precision of 37 percent and recall of 67 percent for open source projects, which translates to an average improvement of 90 percent over the random predictor.	https://ieeexplore.ieee.org/document/6341763
3.	Software defect prediction analysis using machine	2017 7th International Conference on Cloud Computing, Data Science &	In this paper we have analyzed the most popular and widely used Machine Learning algorithms - ANN (Artificial Neural Network), PSO(Particle Swarm Optimization), DT (Decision	The results demonstrated the dominance of Linear Classifier over other algorithms in terms of defect prediction accuracy.	https://ieeexplore.ieee.org/document/7943255

	learning algorithms	Engineering - Confluence	Trees), NB(Naive Bayes) and LC (Linear classifier)		
4.	Software Defect Prediction via Convolutional Neural Network	2017 IEEE International Conference on Software Quality, Reliability and Security (QRS)	In this paper, we propose a framework called Defect Prediction via Convolutional Neural Network (DP-CNN), which leverages deep learning for effective feature generation	The experimental results show that in average, DP-CNN improves the state-of-the-art method by 12%	https://ieeexplore.ieee.org/document/8009936
5.	Personalized defect prediction	2013 28th IEEE/ACM International Conference on Automated Software Engineering (ASE)	This paper proposes personalized defect prediction-building a separate prediction model for each developer to predict software defects. As a proof of concept, we apply our personalized defect prediction to classify defects at the file change level	In this experiment result improves the F1-score by 0.01-0.06 compared to the traditional change classification	https://ieeexplore.ieee.org/document/6693087
6.	Automatically Learning Semantic Features for Defect Prediction	2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE)	In this paper, we leverage Deep Belief Network (DBN) to automatically learn semantic features from token vectors extracted from programs' Abstract Syntax Trees (ASTs).	In this paper semantic features improve WPDP on average by 14.7% in precision, 11.5% in recall, and 14.2% in F1	https://ieeexplore.ieee.org/abstract/document/7886912
7.	Revisiting common bug prediction findings using effort-aware models	2010 IEEE International Conference on Software Maintenance	In this paper, we revisit two common findings in the bug prediction literature: 1) Process metrics (e.g., change history) outperform product metrics (e.g., LOC), 2) Package-level predictions outperform file-level predictions.	we find that if we test 20% of all modules based on the predicted fault density, we would detect 74% of faults using file-level models and 62% of faults using package-level models.	https://ieeexplore.ieee.org/document/5609530
8.	Recent advances in deep learning for speech research at Microsoft	In Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on,	This paper provides selected samples of our experiments on applying deep learning methods to advancing speech technology and related applications, including feature extraction, acoustic modeling, language modeling, speech understanding, and dialogue state estimation.	we presented experimental evidence that spectrogram features of speech are superior to MFCC with DNN, in contrast to the earlier long-standing practice with GMM-HMMs	https://www.researchgate.net/publication/261153438_Recent_advances_in_deep_learning_for_speech_research_at_Microsoft

		2013, pp. 8604–8608			
9.	How, and why, process metrics are better	2013 35th International Conference on Software Engineering (ICSE)	In this paper we analyze the applicability and efficacy of process and code metrics from several different perspectives.	Our results suggest that code metrics, despite widespread use in the defect prediction literature, are generally less useful than process metrics for prediction. Second, we find that code metrics have high stasis; they don't change very much from release to release	https://ieeexplore.ieee.org/document/6606589
10.	Random Forests and Decision Trees	In International Journal of Computer Science Issues, Vol. 9, Issue 5, No 3, September 2012	We compared the classification results obtained from methods i.e. Random Forest and Decision Tree (J48). The classification parameters consist of correctly classified instances, incorrectly classified instances, F-Measure, Precision, Accuracy and Recall.	It can be concluded that the Random Forest achieves increased classification performance and yields results that are accurate and precise in the cases of large number of instances	https://www.researchgate.net/publication/259235118_Random_Forests_and_Decision_Trees
11.	Software defect prediction using neural networks	Proceedings of 3rd International Conference on Reliability, Infocom Technologies and Optimization	we have developed a model based on text mining techniques that will be used to assign the severity level to each defect report based on the classification of existing reports done using the machine learning method namely, Radial Basis Function of neural network	The values of sensitivity for medium, low and very low severity defects are in the range of 56% to 75%, in contrast to its values for high severity defects which is in the range of 70% to 100% for most of the runs	https://ieeexplore.ieee.org/document/7014673
12.	A survey of dimensionality reduction techniques	in International Journal of Emerging Trends & Technology in Computer Science, Volume 3, Issue 6, November-December 2014.	A different family of algorithms poses the dimensionality reduction problem as one of projecting the original data onto a subspace with some interesting properties	It can be concluded that the Random Forest achieves increased classification performance and yields results that are accurate and precise in the cases of large number of instance	https://www.researchgate.net/publication/260755521_A_survey_of_dimensionality_reduction_techniques
13.	Dimensionality reduction and generalization	in the 24th International Conference on Machine	Using probabilistic estimates for inte- gral operators we can prove error estimates for KPCR and propose a parameter choice	We show that perform-ing KPCA and then ordinary least squares on the projected data, a	https://www.researchgate.net/publication/221345784_Dimensionality_r

		Learning, Corvallis, OR, 2007.	procedure allowing us to prove consistency of the algorithm.	procedure known as ker- nel principal component regression (KPCR), is equivalent to spectral cut-on regulariza- tion, the regularization parameter being ex- actly the number of principal components to keep.	eduction_and_generalization
14.	A Survey Of Dimensionality Reduction And Classification Methods	In International Journal of Computer Science & Engineering Survey, Vol.3, No.3, June 2012	A different family of algorithms poses the dimensionality reduction problem as one of projecting the original data onto a subspace with some interesting properties	We have analyzed the number of citations that the most relevant papers in each section have received in the last decade (2003-2012). In Table I we show the number of citations summarized by large areas as well as their share (%) for the different years	https://www.researchgate.net/publication/276197488_A_Survey_Of_Dimensionality_Reduction_And_Classification_Methods
15.	Kernel Principal Component Analysis and its Applications in Face Recognition and Active Shape Models	Rpi, Troy, Ny, Usa, 2011. Copyright 2011	We show some experiment results to compare the performance of kernel PCA and tradi-tional PCA for pattern classification. We also implement the kernel PCA-based ASMs, and use it to construct human face models.	We found that Gaussian kernel PCA-based ASMs are promis-ing in providing more deformation patterns than tradi-tional ASMs.	https://www.researchgate.net/publication/229158312_Kernel_Principal_Component_Analysis_and_its_Applications_in_FaceRecognition_and_Active_Shape_Models
16.	The Meaning and Use of the Area Under a Receiver Operating Characteristic (ROC) Curve	Radiology, 143, 1982, pp. 29-36	A large number of theoretically based measures has been proposed to reduce an entire ROC curve to a single quantitative inclex of diagnostic accuracy; all of these measures have been rooted in the assumption that the functional form of the ROC curve is the same as that implied by supposing that the underlying distributions for normal and abnormal groups are Caussian	To amplify the three-way equivaLence between the area under an ROC curve,	https://www.researchgate.net/publication/16134792_The_Meaning_and_Use_of_the_Area_Under_a_Receiver_Operating_Characteristic_ROC_Curve

DATASET DESCRIPTION

The five datasets used in this project were taken from NASA Promise Dataset Repository (http://promise.site.uottawa.ca/SERepository/datasets-page.html) namely pc1, cm1, jm1, kc1, kc2 each have no miss values and 22 attributes that get from McCabe and Halstead features extractors.

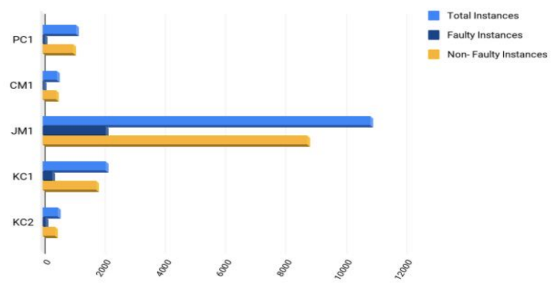


Figure 2 : Dataset characteristics

dataset	Language Used	Total instance	Defective instances	NonDefective instances
PC1	C	1,109	77	1032
CM1	C	498	49	449
JM1	C	10,885	2106	8,779
KC1	C++	2,109	326	1783
KC2	C++	522	105	415

Table 1: Characteristics of Datasets

Since the data had more instances of non buggy modules than buggy, then to prevent biasing we did under sampling.

RESEARCH METHODOLOGY

The Tree Decision Separator or classifier is used to make the model learn from the test set and after that the model is tested in the training set and the action steps are calculated. However, having too many traits and circumstances can lead to a model overreacting. Therefore, we first reduced the size of the data into a set of 6 integrated features using 4 different techniques and then trained the model using the Tree Truth. Detailed comparisons are made based on performance metrics including accuracy, F1-Scores and Area Under the Receiver Operating Characteristics (ROC).

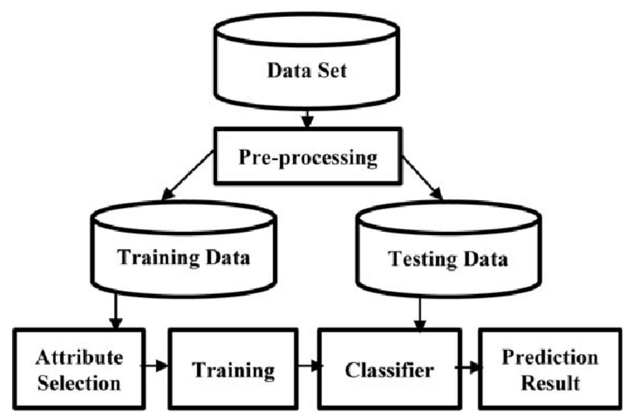


Figure3: Overview of SDP Process

The following are the algorithms used to reduce the size, as well as a brief description.

A. Artificial Neural Network (ANN)

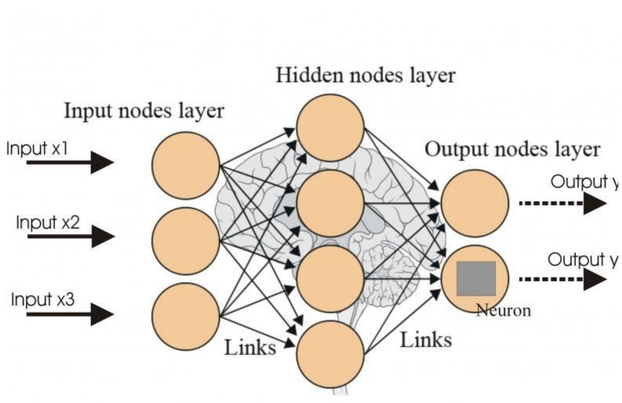


Figure 4 : ANN

This algorithm is somehow derived from the human brain or the human nervous system and uses a collection of hidden layers with different numbers of locations called neurons. Each neuron picks up inputs from a few or all of the neurons of the previous layer and processes the input using activated instruments and activation function. It then sends the output to multiple neurons of the next layer. Depending on the output and the cost function, the instruments are updated over and over again until the parameters fit exactly the model.

To train our model, we use 3 hidden layers with different numbers of neurons. Integrated features removed from the network and used for model training using the Decision Tree Separator or classifier.

B. Principal Component Analysis (PCA)

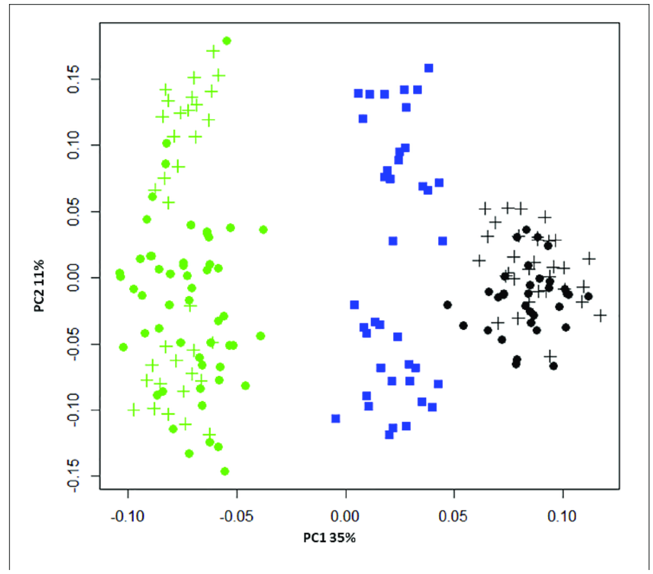


Figure5 : PCA

PCA is a mathematical data analysis method that converts the first set of variables into a set of specific combinations, known as key components (PCs), with specific characteristics regarding variability. This maintains the size of the system while storing information on a flexible connection [17].

The PCA algorithm is used in such a way that it produces 6 new independent features that most accurately define database variability, without dependent variability. Since the final stage of each case can be considered when converting data into a subset, that is why it is an Unsupervised Model.

C. Linear Discriminant Analysis (LDA)

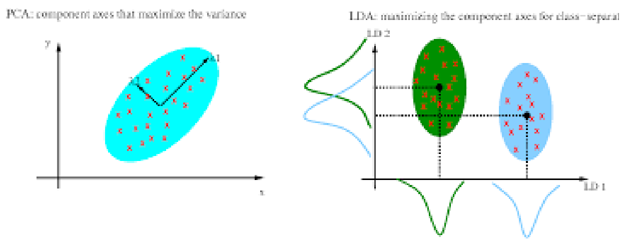


Figure6 : LDA

In high-quality data, it is difficult to find similarities between different data points so the model is difficult to analyze. The LDA algorithm specifies down the data with the maximum size to the minimum size space provided by the divider to train the model. The LDA aims to increase the distance between the phase and reduce the distance within the phase by the size of the reduced area [15].

The LDA algorithm is used in such a way that it produces 6 new independent features that greatly differentiate the data classes, which are buggy and non-buggy. Since the extracted features are derived from the dependent variable dependence, this is why it is a supervised model.

D. Kernel Principal Component Analysis (KPCA)

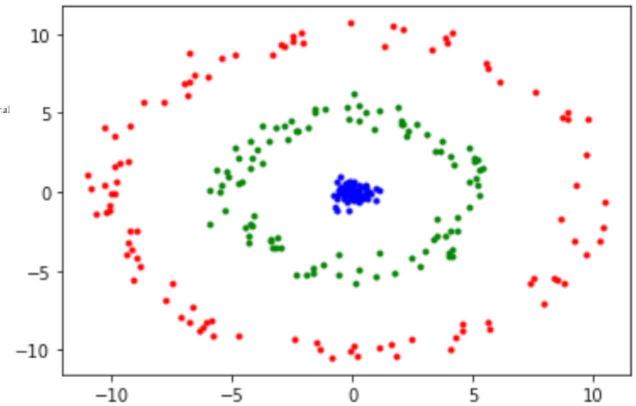


Figure7: KPCA

Standard PCA only allows for a decrease in line size. However, if the data has complex structures that cannot be properly represented in a specific location, a standard PCA will not be very helpful. The Kernel PCA thus extends the standard key analysis (PCA) to the top feature space using the kernel algorithm. An accurate understanding of the Gaussian kernel PCA is that it uses distances between different training data points, such as the nearest neighbor k or meeting methods [19]. The Gaussian kernel PCA reveals more complex data structures than conventional PCA.

Gaussian RBF, Polynomial, Hyperbolic Tangent are some of Kernel's most popular works. We have used the Gaussian RBF kernel function to reduce the size of our data set

Decision Tree Algorithm

The Decision Tree algorithm, which is a supervised learning algorithm and works on the principles of entropy and information acquisition, has been used to classify. Data entry measures data pollution i.e., how the data set is disrupted.

The most critical aspect of the Decision Tree algorithm is the method of selecting the attribute used for each tree node, because there are certain features that separate the data completely from other attributes. The algorithm works on the goal of greed that is, it looks at a solution that seems to be the best at the moment without looking at the overall picture.

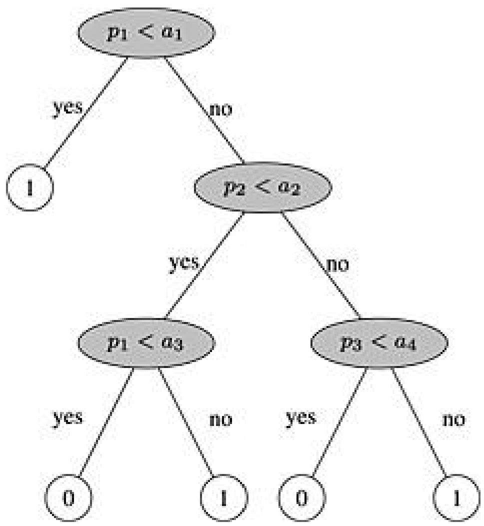


Figure8 : Decision Tree for SDP

The Decision Tree algorithm uses Information Gain, which calculates the reduction of entropy or gain , to classify a set of data using a specific attribute.

The algorithm is advantageous as it requires minimal data purification and is not influenced by vendors and lost prices at a reasonable rate.

PERFORMANCE MEASURES

	Predicted buggy	Predicted clean
True Buggy	TP	FN
True Clean	FP	TN

Table 3: Confusion Matrix

A. Accuracy

This is the ratio between correctly predicted instances of the test set to the total number of instances of the test set.

Accuracy = (TP + TN) / (TP + FN + FP + TN)

B. F1 scores

In some cases, the confusion of accuracy can lead to misinterpretation of the results, which is why we take some performance metrics called F1 points. The F1 score is a harmonic definition of Precision and Recall, also calculated from the confusion matrix.

Precision is the ratio of actual correctly predicted positive (buggy) instances to the total number of predicted positive instances (Precision = TP /(TP + FP).

Recall is also known as Sensitivity. Recall is the ratio of actual correctly predicted positive (buggy) instances to the total number of actual positive instances (Recall = TP /(TP + FN).)
Taking the harmonic mean,
we get F1 score = (2*Recall*Precision)/(Recall + Precision)

C. Area Under the ROC Curve (AUC)

The performance of the predicted models was assessed by setting the Receiver Operating Characteristics (ROC) curve and the area below the curve. The ROC curve, defined as a sensitivity strategy in y-coordinate compared to its 1st specification (defined as estimating of classified error classes in the number of actually classified classes) in x link, operates a method to assess the quality or performance of predicted models [19].

VALIDATION METHOD

We used to hold the cross verification method to verify the data set. If all data sets used have too many scenarios, the training set and the test set are divided into a 3: 1 ratio.

The training set was used to differentiate training and the model was validated in the test set.

RESULT

We got the confusion matrix by applying various techniques . Accuracy and F1 score obtained are given below in the table :

DATA SET	ANN	PCA	LDA	KPCA
PC1	0.90	0.920	0.910	0.92
CM1	0.86	0.864	0.864	0.872
KC1	0.78	0.818	0.812	0.81
KC2	0.84	0.778	0.793	0.81
JM1	0.78	0.754	0.748	0.75

Table 1 : Accuracy of each technique

DATA SET	ANN	PCA	LDA	KPCA
PC1	0.95	0.95	0.95	0.95
CM1	0.92	0.92	0.92	0.93
KC1	0.86	0.89	0.88	0.88
KC2	0.90	0.86	0.87	0.89
JM1	0.86	0.85	0.84	0.84

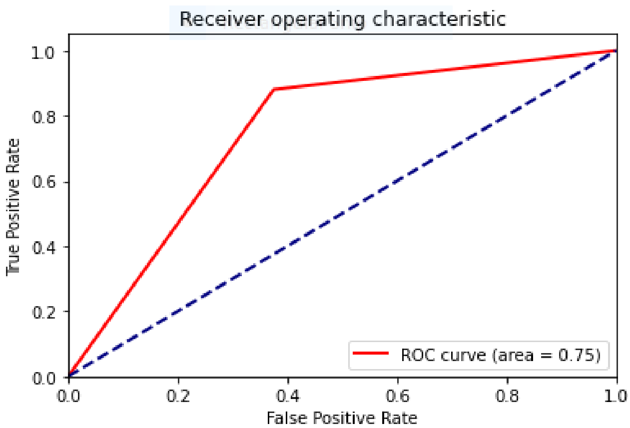
Table 2 : F1 Scores of each technique

DATA SET	ANN	PCA	LDA	KPCA
PC1	0.50	0.70	0.62	0.70
CM1	0.70	0.69	0.75	0.58
KC1	0.62	0.63	0.63	0.62
KC2	0.73	0.61	0.65	0.62
JM1	0.65	0.59	0.57	0.59

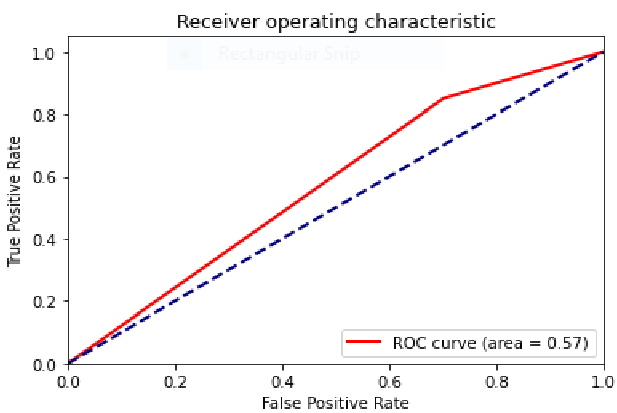
Table 3 : ROC Curve Area..

LDA : ROC of Different Datasets

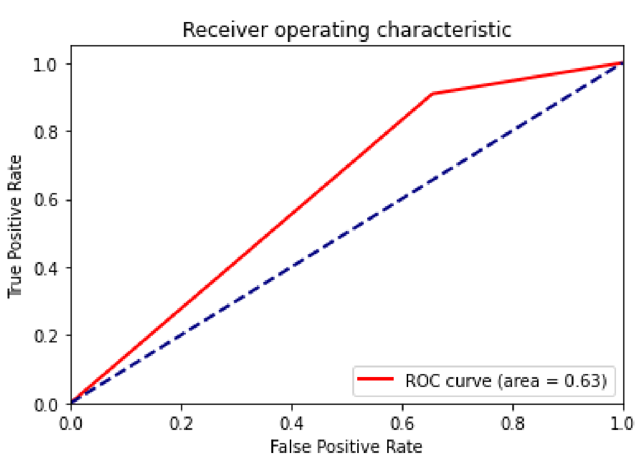
Dataset 1 : cm1



Dataset 2: jm1

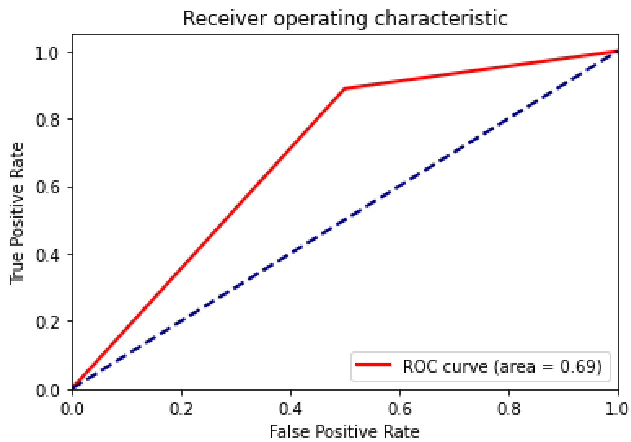


Dataset 3 : kc1

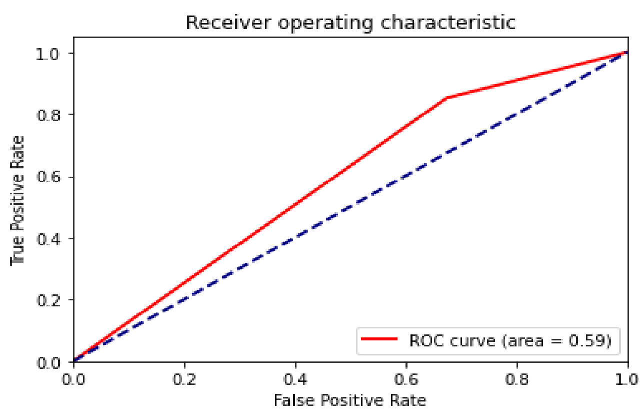


<u>PCA :</u> ROC of different datasets

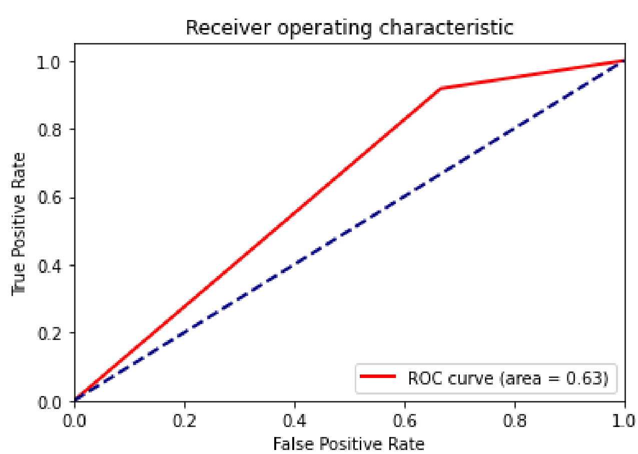
Dataset 1 : cm1



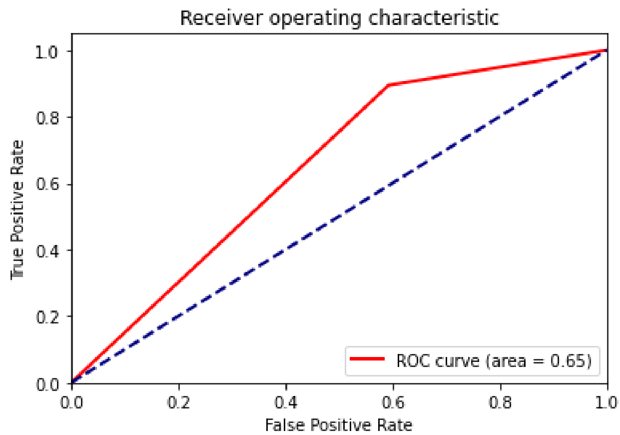
Dataset 2 : jm1



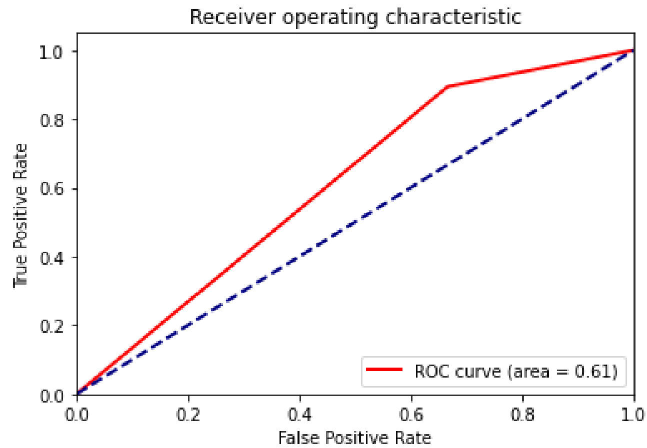
Dataset 3 : kc1



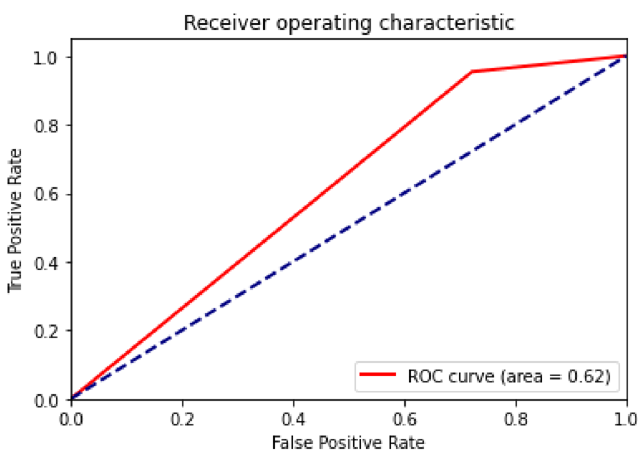
Dataset 4 : kc2



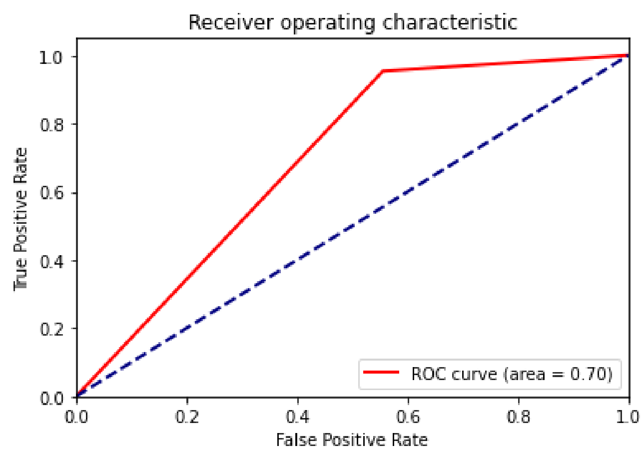
Dataset 4 : kc2



Dataset 5 : pc1

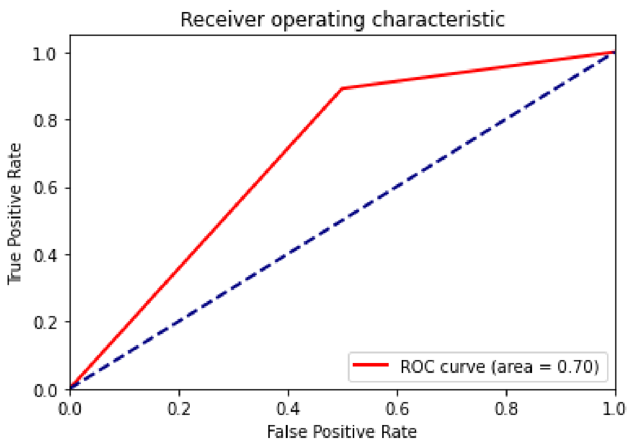


Dataset 5 : pc1

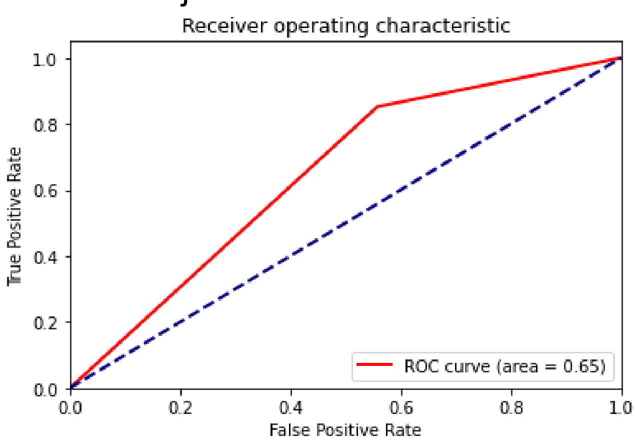


ANN : ROC of different dataset

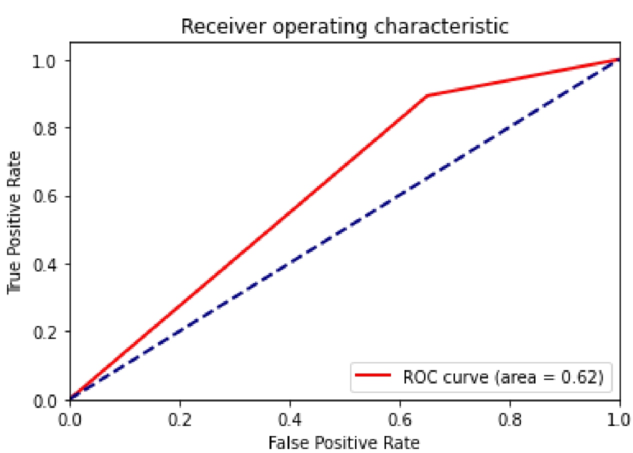
Dataset 1 : cm1



Dataset 2 : jm1

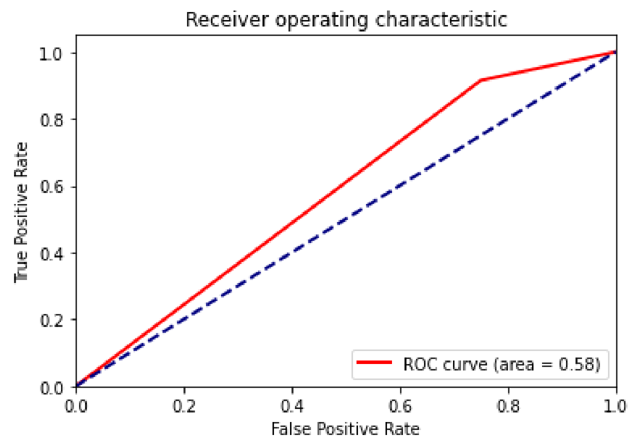


Dataset 3 : kc1

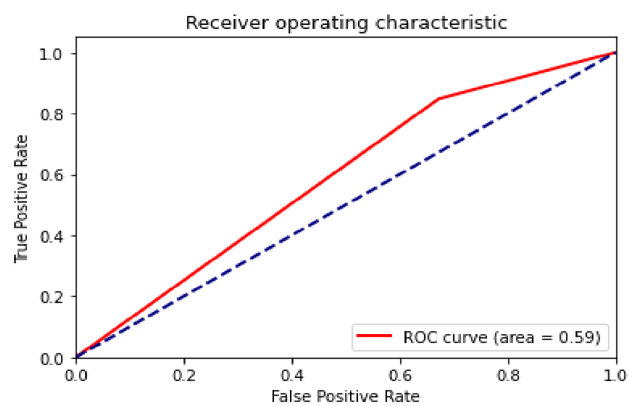


KPCA : ROC of different dataset

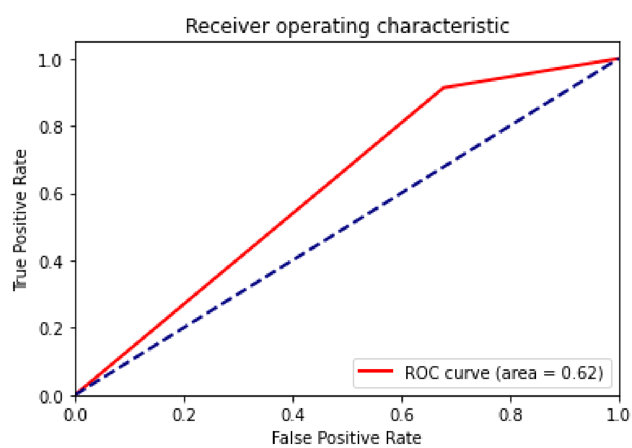
Dataset 1 : cm1



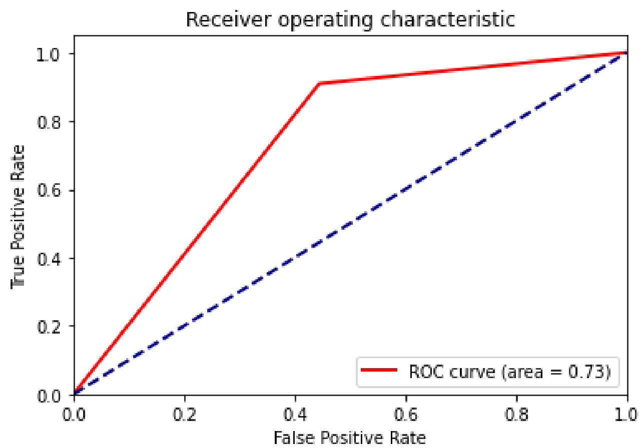
Dataset 2 : jm1



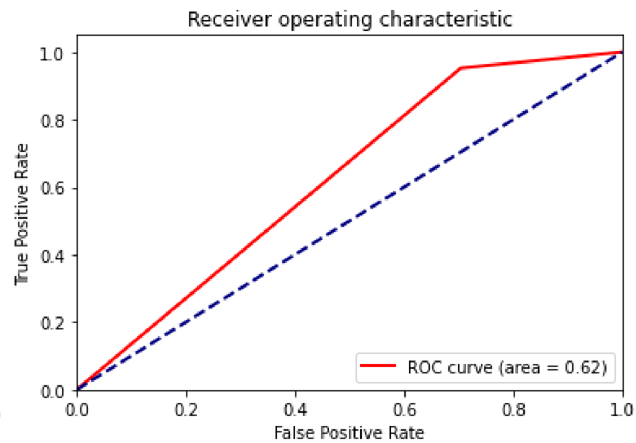
Dataset 3 : kc1



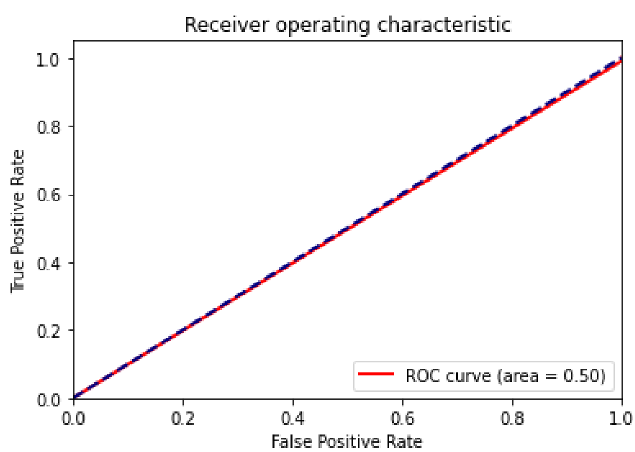
Dataset 4 : kc2



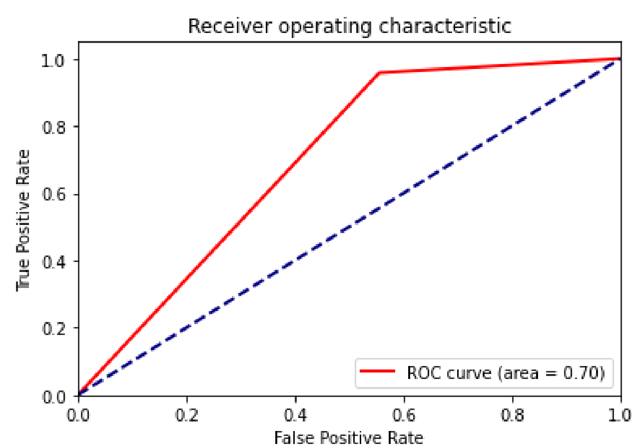
Dataset 4 : kc2



Dataset 5 : pc1



Dataset 5 : pc1



THREATS OF VALIDITY

In this section, we discuss the various threats to the validity of our comparative study.

Construct Validity

We estimate the performance of the model using the hold-out cross-validation method. Training and test sets are constructed randomly so they may overfit the data. Using other performance estimation techniques might give different results. Apart from the considered attributes, there might be other factors affecting the presence of defects.

Internal validity

The data set used contains information regarding features determined by McCabe and Halstead feature extractors, which are known to have certain limitations.

External Validity

We used only 5 open-source data sets taken from NASA Promise Repository and so our results may not generalize to all software. Replication of this a comparative study taking into account other datasets may produce more generalized results.

Conclusion Validity

The datasets being used have a class imbalance problem. So, we used AUC to evaluate the performance of our model but it can still be partial for non-buggy instances

CONCLUSION

In this paper, we proposed various size reduction strategies and compared the results obtained on the basis of the accuracy of the forecast, the F1 points and the area below the curve. Artificial Neural Networks (ANN) and KPCA performed well in comparison to other models in terms of accuracy .

FUTURE SCOPE

In future,we would like to improve the neural network model . We can improve it by changing its various parameters like number of hidden layers , neurons in each layer , optimizers and the cost function. We will also try other models like Naive Bayes , Kernel Support Vector Machine, Random Forest and compare its results with these models.

APPENDICES

Appendix 1

S.No.	Metric	Description
1.	Process Metrics	It is used for improving software development and maintenance.
2.	Line of code(LOC)	It is a software metric used to measure the size of a computer program by counting the number of lines in the text of the program's source code.
3.	Number of changes	Number of builds in which a specific component has changed
4.	Number of Instances	Number of features which are extracted from the software archive
5.	Number of principal component	Number of those variables which are constructed as linear combinations or explain maximal amount of variance that is to say the lines that capture most information of the data.
6.	Number of bugs	Number of defects or we can say defect prone-modules in software system

Appendix 2

S.No.	ALGORITHM	Description
1.	ANN (Artificial Neural Network)	It is a computational and mathematical model that is inspired by the biological nervous system. It uses the processing of the brain as a basis to develop algorithms that can be used to model complex patterns and prediction problems.
2.	PCA (Principal Component Analysis)	PCA is a mathematical data analysis method that converts the first set of variables into a set of specific combinations, known as key components (PCs)with specific characteristics regarding variability. This maintains the size of the system while storing information on a flexible connection
3.	LDA (Linear Discriminant Analysis)	The LDA algorithm is used in such a way that it produces 6 new independent features that greatly differentiate the data classes, which are buggy and non-buggy.
4.	KPCA (Kernel Principal Component Analysis)	It is a non-linear dimensionality reduction technique and also extension of PCA Algorithm - which is a linear dimensionality reduction technique -using kernel methods.

REFERENCES

1. X. Yang, D. Lo, X. Xia, Y. Zhang, and J. Sun, "Deep learning for just-in-time defect prediction," in QRS'15: Proc. of the International Conference on Software Quality, Reliability and Security, 2015. [**Deep learning for just-in-time defect prediction**]

2. Y. Kamei, E. Shihab, B. Adams, A. E. Hassan, A. Mockus, A. Sinha, and N. Ubayashi, "A large-scale empirical study of just-in-time quality assurance," TSE, vol. 39, no. 6, pp. 757–773, 2013. [**A large-scale empirical study of just-in-time quality assurance**]

3. T. Jiang, L. Tan, and S. Kim, "Personalized defect prediction," in ASE, 2013, pp. 279–28. 4. M. Tan, L. Tan, S. Dara, and C. Mayeux, "Online defect

prediction for imbalanced data," in ICSE'15, pages 99–108. Personalized defect prediction

4. Praman Deep Singh, Anuradha Chug, " Software defect prediction analysis using machine learning algorithms" in 2017 7th International Conference on Cloud Computing, Data Science & Engineering . Confluence. Software defect prediction analysis using machine learning algorithms

5. S. Wang, T. Liu, and L. Tan, "Automatically learning semantic features for defect prediction," in ICSE'16: Proc. of the International Conference on Software Engineering, 2016.

Automatically Learning Semantic Features for Defect Prediction

6. Y. Kamei, S. Matsumoto, A. Monden, K.-i. Matsumoto, B. Adams, and A. E. Hassan, "Revisiting common bug prediction findings using effort-aware models," in ICSM, 2010, pp. 1–10. Revisiting common bug prediction findings using effort-aware models

7. Jian Li, Pinjia He, Jieming Zhu, and Michael R. Lyu, "Software Defect Prediction via Convolutional Neural Network". Software Defect Prediction via Convolutional Neural Network

8. P. D. Singh and A. Chugh, "Software Defect Prediction Analysis Using Machine Learning Algorithms," in International Conference on Cloud Computing, Data Science & Engineering – Confluence, 2017. Software Defect Prediction Analysis Using Machine Learning Algorithms

9. G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," Science, vol. 313, no. 5786, pp. 504–507, 2006. Reducing the dimensionality of data with neural networks

10. G. E. Hinton, "Learning multiple layers of representation," Trends in cognitive sciences, vol. 11, no. 10, pp. 428–434, 2007. Learning multiple layers of representation

11. L. Deng, J. Li, J.-T. Huang, K. Yao, D. Yu, F. Seide, M. Seltzer, G. Zweig, X. He, J. Williams et al., "Recent advances in deep learning for speech research at microsoft," in Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on, 2013, pp. 8604–8608.

Recent advances in deep learning for speech research at microsoft,

12. F. Rahman and P. Devanbu, "How, and why, process metrics are better," in ICSE, 2013, pp. 432–441. How, and why, process metrics are better

13. J. Ali, R. Khan, N. Ahmad, I. Maqsood, "Random Forests and Decision Trees," in International Journal of Computer Science Issues, Vol. 9, Issue 5, No 3, September 2012 . Random Forests and Decision Trees,

14. R. Jindal, R. Malhotra, A. Jain, "Software Defect Prediction Using Neural Networks," IEEE Conference 2014.

Software Defect Prediction Using Neural Networks

15. V. A. Kumar, N. Elavarasan, " A Survey on Dimensionality Reduction Technique," in International Journal of Emerging Trends & Technology in Computer Science, Volume 3, Issue 6, November-December 2014. A Survey on Dimensionality Reduction Technique

16. S. Mosci, L. Rosasco, A. Verri, "Dimensionality Reduction and Generalization," in the 24th International Conference on Machine Learning, Corvallis, OR, 2007. Dimensionality Reduction and Generalization

17. N. Varghese, V. Verghese, Gayathri. P and Dr. N. Jaisankar, "A Survey Of Dimensionality Reduction And Classification Methods," in International Journal of Computer Science & Engineering Survey, Vol.3, No.3, June 2012. A Survey Of Dimensionality Reduction And Classification Methods

18. Q.Wang, "Kernel Principal Component Analysis and its Applications in Face Recognition and Active Shape Models," Rpi, Troy, Ny, Usa, 2011. Copyright 2011. Kernel Principal Component Analysis and its Applications in Face Recognition and Active Shape Models

19. J. Hanley, BJ. McNeil, "The meaning and use of the area under a Receiver Operating Characteristic ROC curve", Radiology, 143, 1982, pp. 29-36. The meaning and use of the area under a Receiver Operating Characteristic ROC curve