IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.301
Volume 3, Issue 7, May 2023

Data Poison Detection using Machine Learning

K. Karthika[1], Sugashini T[2], Soundharya P[3]
Assistant Professor, CSE, Anjalai Ammal Mahalingam Engineering College, Thiruvarur, India[1]
Student, CSE, Anjalai Ammal Mahalingam Engineering College, Thiruvarur, India[2,3]

Abstract:*Fault-prediction techniques aim to predict the software modules that are faulty so that they can be beneficial in the upcoming phases of software development. Different performance criteria are being employed in order to boost the performance of the already existing ones. However, the main issue is the perspective of compiling their performances, which is ignored constantly. Classification is the most common technique used for the exclusion of faulty modules from non-faulty modules. Machine-learning techniques are used to find defects, faults, and ambiguities in software to achieve quality, maintainability, and reusability. Software fault prediction techniques are used to predict software faults by using statistical techniques. However, machine-learning techniques are also valuable in detecting software faults. Here, presents software fault prediction using machine-learning techniques to predict the occurrence of faults.*

Keywords:Logistic Regression; Decision Tree; SVM.

I. INTRODUCTION

In the past decade, humans have progressively focused on software-based systems in which software quality is regarded as the most critical element of user functionality. Because of the vast production of application software, software quality remains an unresolved problem that results in inadequate output for industrial and private applications. Designs for defect prediction are commonly utilized by industries, and such models help in predicting faults, estimating effort for testing software reliability, hazard analysis, etc. during the growth stage. A supervised machine learning predictive algorithm is consumed with a predefined collection of training data. The algorithm then gains expertise from the training dataset and produces rules for predicting the class label for a new data set.
Software quality can be enhanced by predicting defect in modules. Defect prediction is the method of designing models that are utilized in the initial stages of the process to detect defective systems, such as units or classes. This can be achieved by classifying the modules as defect prone or not. Different methods are used to identify the classification module, the most common of which is support vector classifier (SVC), random forest, naive bayes, decision trees (DT), and neural networks (NN).

II.RELATED WORK

Felix, E.A. and Lee, S.P., 2017 Software defect prediction provides actionable outputs to software teams while contributing to industrial success. Empirical studies have been conducted on software defect prediction for both cross project and within-project defect prediction. However, existing studies have yet to demonstrate a method of predicting the number of defects in an upcoming product release. This presents such a method using predictor variables derived from the defect acceleration, namely, the defect density, defect velocity, and defect introduction time, and determines the correlation of each predictor variable with the number of defects.
We report the application of an integrated machine learning approach based on regression models constructed from these predictor variables. An experiment was conducted on ten different data sets collected from the PROMISE repository, containing 22,838 instances. The regression model constructed as a function of the average defect velocity achieved an adjusted R-square of 98.6%, with a p-value of < 0.001. The average defect velocity is strongly positively correlated with the number of defects, with a correlation coefficient of 0.98. Thus, it is demonstrated that this technique can provide a blueprint for program testing to enhance the effectiveness of software development activities.

III.MACHINE LEARNING

Machine Learning is a system that can learn from example through self-improvement and without being explicitly coded by programmers. The breakthrough comes with the idea that a machine can singularly learn from the data (i.e., example) to produce accurate results.Machine learning combines data with statistical tools to predict an output. This output is then used by corporations to make actionable insights. Machine learning is closely related to data mining and Bayesian predictive modelling. The machine receives data as input, uses an algorithm to formulate answers.A typical machine learning tasks are to provide a recommendation. For those who have a Netflix account, all recommendations of movies or series are based on the user's historical data. Tech companies are using unsupervised learning to improve the user experience with personalizing recommendation.Machine learning is also used for a variety of tasks like fraud detection, predictive maintenance, portfolio optimization, automate tasks and so on.
Machine learning can be grouped into two broad learning tasks: Supervised and Unsupervised. There are many other algorithms

Supervised Learning

An algorithm uses training data and feedback from humans to learn the relationship of given inputs to a given output. For instance, a practitioner can use marketing expense and weather forecast as input data to predict the sales of cans.

Unsupervised Learning

Unsupervised learning is a type of machine learning where an algorithm is trained on a dataset without any explicit supervision or labeled output. In other words, the algorithm is left to discover patterns and relationships within the data on its own. It uses an unlabeled dataset and built –in libraries such as TextBlob to predict if a piece of text is positive, negative, or neutral.

IV.PROPOSED METHODOLOGY

1. DATA COLLECTION AND LOADING

Data used in this paper is a software defect prediction dataset JM1. This step is concerned with selecting the subset of all available data that you will be working with. ML problems start with data preferably, lots of data (examples or observations) for which you already know the target answer. Label the data to indicate which samples are clean and which are poisoned. Data for which you already know the target answer is called labeled data. Use various libraries such as PyTorch, TensorFlow, or Keras to load the datasets. Create data loaders to feed the data into the model in batches.

2. DATA PRE-PROCESSING

Organize your selected data by formatting, cleaning and sampling from it.Three common data pre-processing steps are:
Formatting: The data you have selected may not be in a format that is suitable for you to work with. The data may be in a relational database and you would like it in a flat file, or the data may be in a proprietary file format and you would like it in a relational database or a text file.

Cleaning: Cleaning data is the removal or fixing of missing data. There may be data instances that are incomplete and do not carry the data you believe you need to address the problem. These instances may need to be removed. Additionally, there may be sensitive information in some of the attributes and these attributes may need to be anonymized or removed from the data entirely.

Sampling: There may be far more selected data available than you need to work with. More data can result in much longer running times for algorithms and larger computational and memory requirements. You can take a smaller representative sample of the selected data that may be much faster for exploring and prototyping solutions before considering the whole dataset.

3. FEATURE EXTRACTION

Next thing is to do Feature extraction is an attribute reduction process. Unlike feature selection, which ranks the existing attributes according to their predictive significance, feature extraction actually transforms the attributes. The transformed attributes, or features, are linear combinations of the original attributes. Finally, our models are trained using the Classifier algorithm. We use the classify module on the Natural Language Toolkit library on Python. We use the labeled dataset gathered. The rest of our labeled data will be used to evaluate the models. Some machine learning algorithms were used to classify pre-processed data. The chosen classifiers were Random Forest. These algorithms are very popular in text classification tasks.

4. MODEL EVALUATION

Model Evaluation is an integral part of the model development process. It helps to find the best model that represents our data and how well the chosen model will work in the future. Evaluating model performance with the data used for training is not acceptable in data science because it can easily generate overoptimistic and over-fitted models.

Performance of each classification model is estimated based on its averaged. The result will be in the visualized form. Representation of classified data in the form of graphs.

Accuracy 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.

5. MACHINE LEARNING ALGORITHMS

Support Vector Machine

Support Vector Machine (SVM) is a supervised machine learning algorithm which can be used for both classification or regression challenges. It is generally utilized in characterization issues. In the SVM calculation, we plot every datum thing as a point in n-dimensional space (where n is the number of highlights you have) with the estimation of each element being the estimation of a specific arrangement. At that point,we perform order by finding the hyper-plane that separates the two classes quite well. Bolster Vectors are essentially the coordinates of individual perception. The SVM classifier is a wilderness which best isolates the two classes (hyper-plane/line)

Working with Support Vector Machine in Python:

```
#Import Library
# Import other necessary libraries like pandas, numpy…
from sklearn import tree
# Assumed you have, X (predictor) and Y (target) for training data set and
x_test (predictor) of test_dataset
# Create tree object
model = SVC (criterion='gini') # for classification
# model = SVR() for regression
# Train the model using the training sets and check score
model.fit(X, y)
model.score(X, y)
#Predict Output
predicted = model.predict(x_test)
```

Random Forest

Random forest is a type of supervised machine learning algorithm based on ensemble learning. Ensemble learning is a type of learning where you join different types of algorithms or the same algorithm multiple times to form a more powerful prediction model. The random forest algorithm combines multiple algorithms of the same type i.e. multiple decision trees, resulting in a forest of trees, hence the name "Random Forest". The random forest algorithm can be used for both regression and classification tasks.

Random Forest Working

The following are the basic steps involved in performing the random forest algorithm

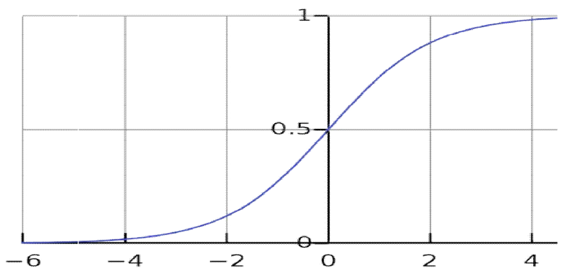
1. Pick N random records from the dataset.
2. Build a decision tree based on these N records.
3. Choose the number of trees you want in your algorithm and repeat steps 1 and 2.
4. For classification problems, each tree in the forest predicts the category to which the new record belongs. Finally, the new record is assigned to the category that wins the majority vote.

Logistic Regression

Logistic regression is a statistical technique used to predict probability of binary response based on one or more independent variables. It means that, given certain factors, logistic regression is used to predict an outcome which has two values such as 0 or 1, pass or fail, yes or no etc.

Working of Logistic Regression.

Probabilities are estimated using logistic/sigmoid function. The graph of the sigmoid function is an 'S' curve



The mathematical expression is given by

F(z)=1/1−e−z

where z=w0+w1·x1+w2·x2+…+wn·xn.

Here w0,w1,w2,...wn are the regression coefficients of the model and are calculated by Maximum Likelihood Estimation and x1,x2,x3,...,xn are the features or independent variables. F(z) calculates the probability of the binary outcome and using the probabilities we classify the given data point(x) into one of the two categories.

We split the data into sequential trains and test datasets for all experiments. The train set includes all labeled samples up to the 34th time-step (16670 transactions), and the test set includes all labeled samples from the 35th time-step, inclusive, onward (29894 transactions). We train each supervised model on the train set using all features and then evaluate them on the entire test set. To measure performance over time. We use the scikit-learn implementation of logistic regression (LR) and random forest (RF).

Decision Tree

Decision trees can be a useful algorithm for software defect prediction. In a decision tree, the data is split into smaller subsets based on the values of certain attributes, with the goal of maximizing the separation of the classes (i.e., defect vs. non-defect).

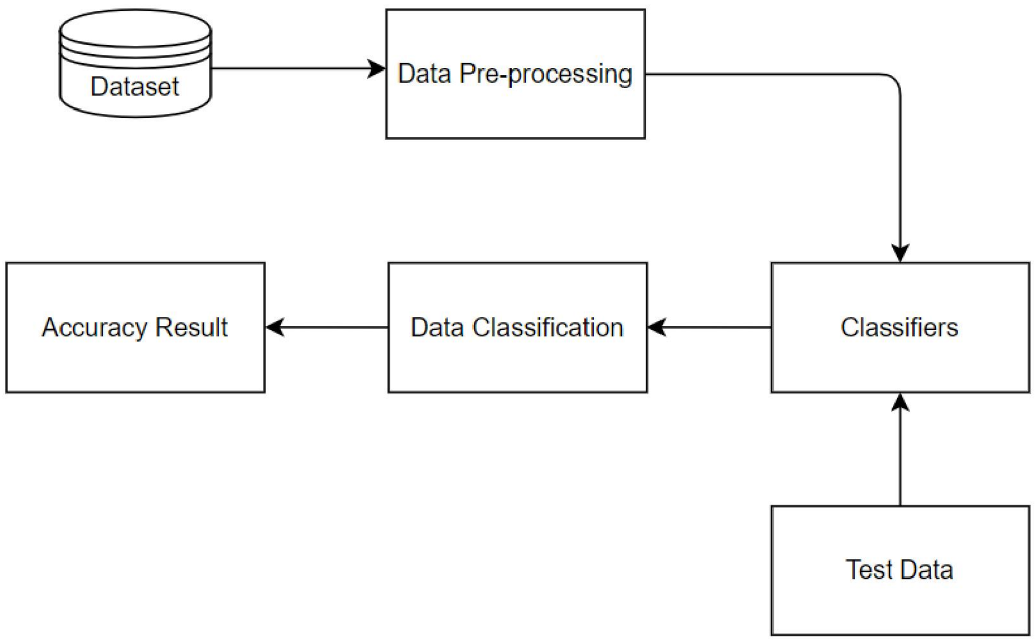
Decision Tree Working

Use the training data to construct a decision tree by recursively splitting the data into smaller subsets based on the most informative attributes.

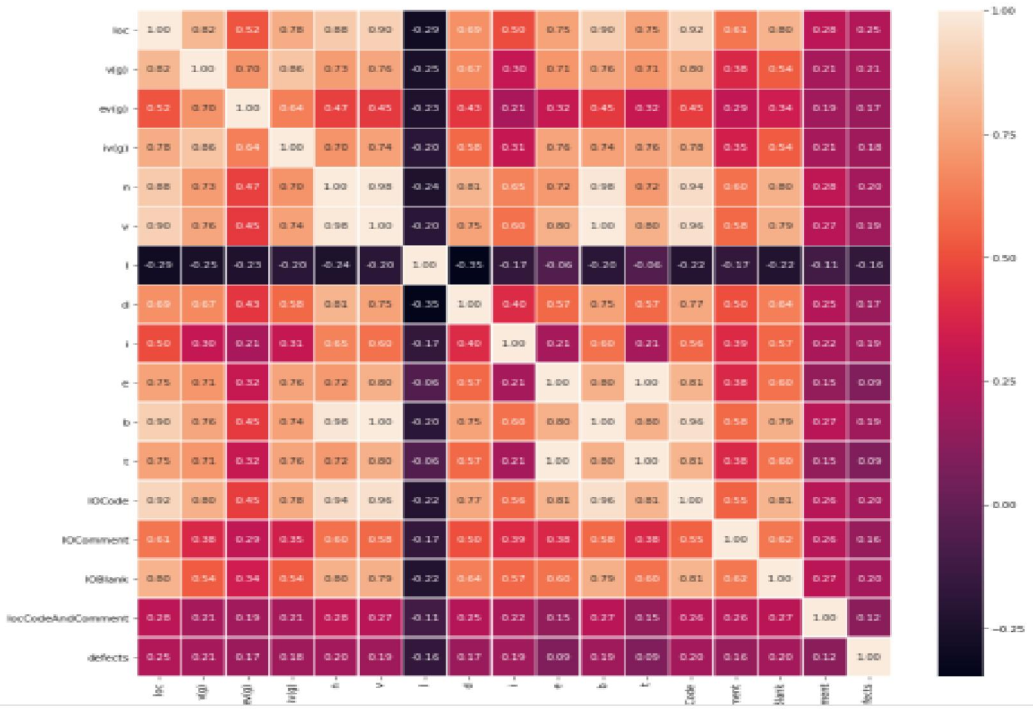
Use the testing data to evaluate the accuracy of the decision tree.

To avoid overfitting, prune the decision tree by removing branches that do not improve accuracy on the testing data. Once the decision tree is built and pruned, use it to predict defects in new software projects

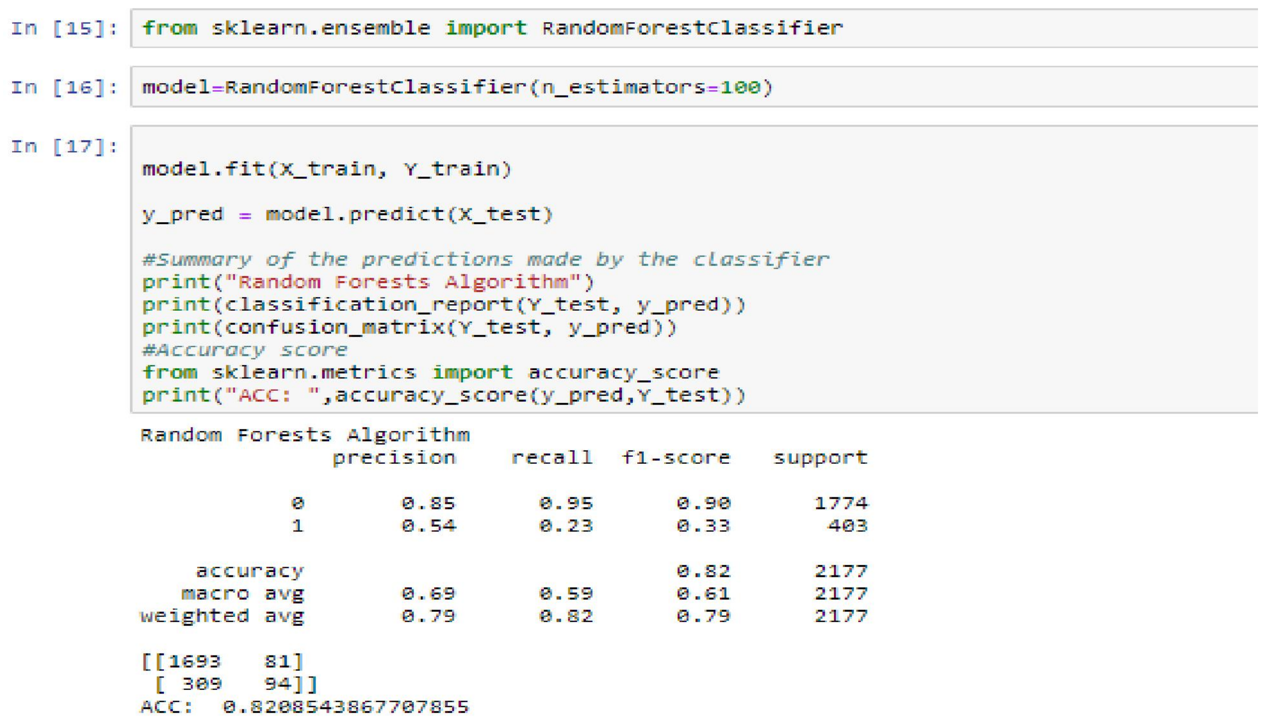
6. SYSTEM MODEL



Feature Extraction using Correlation

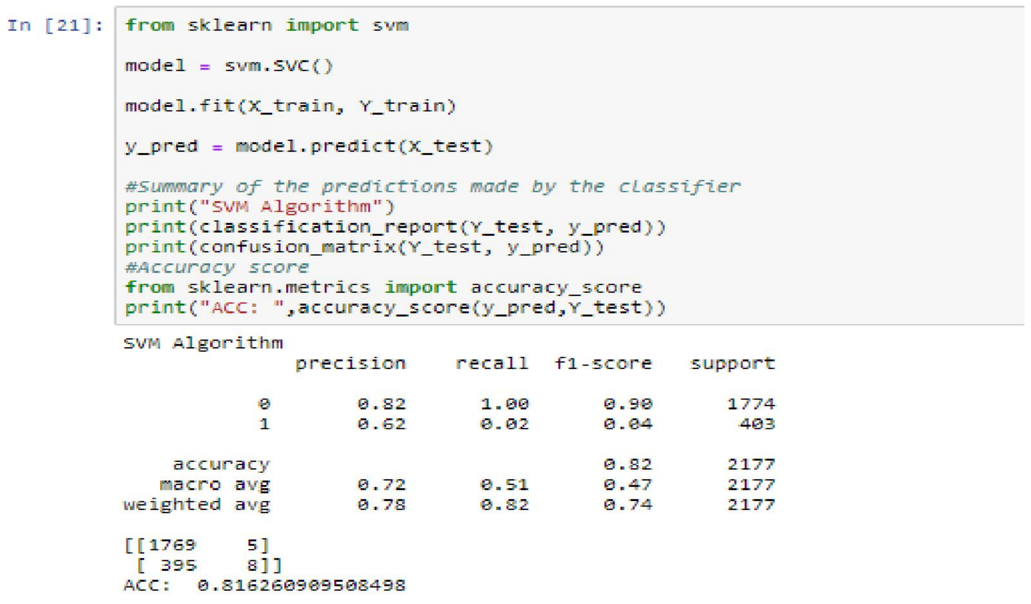


Accuracy based on Random Forest



```
In [15]: from sklearn.ensemble import RandomForestClassifier

In [16]: model=RandomForestClassifier(n_estimators=100)

In [17]:
         model.fit(X_train, Y_train)

         y_pred = model.predict(X_test)

         #Summary of the predictions made by the classifier
         print("Random Forests Algorithm")
         print(classification_report(Y_test, y_pred))
         print(confusion_matrix(Y_test, y_pred))
         #Accuracy score
         from sklearn.metrics import accuracy_score
         print("ACC: ",accuracy_score(y_pred,Y_test))

         Random Forests Algorithm
                       precision    recall  f1-score   support

                    0       0.85      0.95      0.90      1774
                    1       0.54      0.23      0.33       403

             accuracy                           0.82      2177
            macro avg       0.69      0.59      0.61      2177
         weighted avg       0.79      0.82      0.79      2177

         [[1693   81]
          [ 309   94]]
         ACC:  0.8208543867707855
```

Accuracy based on Support Vector Machine



```
In [21]: from sklearn import svm

         model = svm.SVC()

         model.fit(X_train, Y_train)

         y_pred = model.predict(X_test)

         #Summary of the predictions made by the classifier
         print("SVM Algorithm")
         print(classification_report(Y_test, y_pred))
         print(confusion_matrix(Y_test, y_pred))
         #Accuracy score
         from sklearn.metrics import accuracy_score
         print("ACC: ",accuracy_score(y_pred,Y_test))

         SVM Algorithm
                       precision    recall  f1-score   support

                    0       0.82      1.00      0.90      1774
                    1       0.62      0.02      0.04       403

             accuracy                           0.82      2177
            macro avg       0.72      0.51      0.47      2177
         weighted avg       0.78      0.82      0.74      2177

         [[1769    5]
          [ 395    8]]
         ACC:  0.8162609095084498
```

V. CONCLUSION

This research primarily seeks to use information-mining techniques to predict software-defects. Moreover, this domain is now a significant research field whereby numerous strategies have been explored to somehow enhance the efficiency of detecting software defects or predicting bugs. Throughout this study, by designing a new hybrid model using classification, dealt with the issue of classification accuracy for massive datasets.

These findings can be more improved with the use of several datasets. An increase in the number of datasets can enhance the findings. It is also possible to compare further techniques.

REFERENCES

[1] Jayanthi, R. and Florence, L., 2019. Software defect prediction techniques using metrics based on neural network classifiers. Cluster Computing, 22(1), pp.77-88.

[2] Felix, E.A. and Lee, S.P., 2017. Integrated approach to software defect prediction. IEEE Access, 5, pp.21524-21547.

[3] Wang, T., Zhang, Z., Jing, X., Zhang, L.: Multiple kernel ensemble learning for software defect prediction. Autom. Softw. Eng. 23, 569–590 (2015).

[4] Xu, Z., Xuan, J., Liu, J., Cui, X.: MICHAC: defect prediction via feature selection based on maximal information coefficient with hierarchical agglomerative clustering. In: 2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER), Suita, pp. 370–381 (2016).

[5] Ryu, D., Baik, J.: Effective multi-objective naïve Bayes learning for cross-project defect prediction. Appl. Soft Comput. 49, 1062 (2016).

[6] Shan C., Chen B., Hu C., Xue J., Li N.: Software defect prediction model based on LLE and SVM. In: Proceedings of the Communications Security Conference (CSC '14), pp. 1–5 (2014).

[7] Yang, Z.R.: A novel radial basis function neural network for discriminant analysis. IEEE Trans. Neural Netw. 17(3), 604–612(2006).

[8] K. Han, J.-H. Cao, S.-H. Chen, and W.-W. Liu, "A software reliability prediction method based on software development process," in Quality, Reliability, Risk, Maintenance, and Safety Engineering (QR2MSE), 2013 International Conference on. IEEE, 2013, pp. 280–283.

[9] S. Parthipan, S. Senthil Velan, and C. Babu, "Design level metrics to measure the complexity across versions of ao software," in Advanced Communication Control and Computing Technologies (ICACCCT), 2014 International Conference on. IEEE, 2014, pp. 1708–1714.

[10] A. Panichella, R. Oliveto, and A. De Lucia, "Cross-project defect prediction models: L'union fait la force," in Software Maintenance,

[11] Bautista, A.M., Feliu, T.S.: Defect prediction in software repositories with artificial neural networks. In: Mejia, J., Munoz,M., Rocha,Á., Calvo-Manzano, J. (eds.) Trends and Applications in Software Engineering.Advances in Intelligent Systems and Computing, vol.405. Springer, Cham (2016).

[12] H. Lu, B. Cukic, and M. Culp, "Software defect prediction using semi supervised learning with dimension reduction," in Automated Software Engineering (ASE), 2012 Proceedings of the 27th IEEE/ACM International Conference on. IEEE, 2012, pp. 314–317.