Software Defect Prediction using CNN

Krishna Teja Vemuri 17075057

Vempalli M G M Reddy 17075056

Sai Kiran Anumalla 17075051

Tutor - Sushant Kumar Pandey

June 5, 2020

1 Abstract

The aim of the project is to predict the defects in the software. Here in the project we have employed Convolutional Neural Networks CNN for software defect prediction and SMOTE for class imbalance. The results obtained are judged in terms of the mcc metric,f1 score values.

2 Introduction

The software defect prediction identifies the modules that are defective and it requires a wide range of testing. Early identification of an error leads to effective allocation of resources, reduces the time and cost of developing a software and high quality software. It can done using various machine learning techniques ranging from neural networks to support vector machines and decision trees. Here we employ the CNN based methods to deal with imbalanced data.

3 Problem Statement

The given NASA dataset with 8 csv files in it has highly polarized data in terms of positives and negatives for the data mentioned. Employ CNN to get results for the test data. Also deal with the class imbalance in the given data.

4 Methodologies

4.1 Software Defect Prediction

SDP identifies the modules that are defective and it requires a wide range of testing. Early identification of an error leads to effective allocation of resources, reduces the time and cost of developing a software and high quality software. Therefore, an SDP model plays a vital role in understanding, evaluating and improving the quality of a software system.[3]

4.2 Data Extraction

Data is read using readcsv of pandas and then converted to numpy arrays for calculation purpose.

4.3 Dealing The Class Imbalance

Changing the performance metric as there is a problem with accuracy measure.Resampling techniques like under-sampling, over-sampling and generating synthetic data using smote we polish our data-set.
SMOTE uses a nearest neighbors algorithm to generate new and synthetic data[2]
Changed performance metrics to F1-Score, Precision, Recall, MCC.

Precision: the number of true positives divided by all positive predictions. Precision is also called Positive Predictive Value. It is a measure of a classifier's exactness. Low precision indicates a high number of false positives.

Recall: the number of true positives divided by the number of positive values in the test data. Recall is also called Sensitivity or the True Positive Rate. It is a measure of a classifier's completeness. Low recall indicates a high number of false negatives.

F1 Score:the weighted average of precision and recall.

MCC:The Matthews Correlation Coefficient (MCC) has a range of -1 to 1 where -1 indicates a completely wrong binary classifier while 1 indicates a completely correct binary classifier. Using the MCC allows one to gauge how well their classification model/function is performing.

4.4 Convolutional Neural Network (CNN)

CNN typically consists of the following layers, we employ this process to build our CNN model.we perform these sequential steps of following layers while we take care of the hyperparameters

Input layer: here we input our data from a dataset after dealing with class imbalances and applying some data refining techniques.

Convolution layer: here we used 1D convolution layer i.e a 1D filter to get dot product with future vector by striding it from left to right.

Pooling layer: pooling layers reduce the dimensions of the data by combining the outputs of neuron clusters at one layer into a single neuron in the next layer Pooling can be of 3 types. They are:

- Max. Pooling: Taking Maximum for reducing the dimension.
- Min. pooing: Taking Minimum for reducing the dimension.
- Avg. pooling : Taking Average for reducing the dimension.

Activation function layer: This layer will apply element wise activation function to the output of the convolution layer. Some common activation functions are RELU, Sigmoid, Tanh, Leaky RELU, etc.

Fully connected layer: This layer is a regular neural network layer which takes input from the previous layer and computes the class scores and outputs the 1-D array of size equal to the number of classes.

Loss layer: The "loss layer" specifies how training penalizes the deviation between the predicted (output) and true labels and is normally the final layer of a neural network. Various loss functions appropriate for different tasks may be used.[4]

We used keras cnn for above in code

4.5 Tuning

One method to reduce over-fitting is dropout.we use dropout after pooling in our CNN model.To reduce over-fitting we also used L2 regularization.

- **L1 regularization**: Add a cost with regards to absolute value of the parameter.
- **L2 regularization**: Add a cost with regards to squared value of the parameters.[1]

5 Performance of methodologies used

The sample results over CM1 csv data file in give data-set:

- Loss-5.411756812201606
- Accuracy-0.7444444298744202
- F1 Score-0.7487008571624756
- Recall- 0.7552083134651184
- MCC-0.49311110377311707

The sample results over JM1 csv data file in give data-set:

- Loss-3.0224255912667797
- Accuracy- 0.5506833791732788
- F1 Score-0.5584981441497803
- Recall- 0.5685606002807617
- MCC-0.10196933895349503

The sample results over KC1 csv data file in give data-set:

- Loss-0.6322
- Accuracy-0.940170990485771
- F1 Score-0.7135854363441467
- Recall- 0.7529891729354858

The sample results over MC2 csv data file in give data-set:

- Loss- 0.6937344788639078
- Accuracy-0.47457626461982727
- F1 Score-0.4743865728378296
- Recall- 0.47438663244247437

The sample results over MC1 csv data file in give data-set:

- Loss-15.20323721199575
- Accuracy-0.5592020750045776
- F1 Score- 0.683159351348877
- Recall- 0.9537500143051147

The sample results over KC2 csv data file in give data-set:

- Loss-13.679805479853986
- Accuracy-0.599397599697113
- F1 Score-0.3702419698238373
- Recall- 0.2465277761220932

The sample results over KC3 csv data file in give data-set:

- Loss-0.6969784498214722
- Accuracy-0.40625
- F1 Score-0.4062499403953552
- Recall- 0.40625

Further tuning of the hyper-parameters is required to achieve better results.

5.1 Related Work

Defect predictors are expected to help to improve software quality and reduce the costs of delivering those software systems. There is a rapid growth of SDP research after the PROMISE repository was created in 2005. A variety of machine learning methods have been proposed and compared for SDP problems, such as decision trees, neural networks, Naive Bayes, support vector machines, and Artificial Immune Systems. It is discouraging but not surprising that no single method is found to be the best, due to different types of software projects, different algorithm settings, and different performance evaluation criteria for assessing the models. Among all, Random Forest appears to be a good choice for large data sets, and Naive Bayes performs well for small data sets. However, they didn't consider the data characteristic of class imbalance.
Numerous methods have been proposed to tackle class imbalance problems at data and algorithm levels. Data-level methods include a variety of re-sampling techniques, manipulating training data to rectify the skewed class distributions, such as random oversampling, random under-sampling, and SMOTE . They are simple and efficient, but their effectiveness depends greatly on the problem and training algorithms.[5]

References

[1] *Hyperparameter Optimization with Keras*, 2017 (accessed February 3, 2020). `https://towardsdatascience.com/hyperparameter-optimization-with-keras-b82e6364ca53`.

[2] Hui Han, Wen-Yuan Wang, and Bing-Huan Mao. Borderline-smote: a new over-sampling method in imbalanced data sets learning. In *International conference on intelligent computing*, pages 878–887. Springer, 2005.

[3] N Kalaivani and R Beena. Overview of software defect prediction using machine learning algorithms. *International Journal of Pure and Applied Mathematics*, 118(20):3863–3873, 2018.

[4] Jian Li, Pinjia He, Jieming Zhu, and Michael R Lyu. Software defect prediction via convolutional neural network. In *2017 IEEE International Conference on Software Quality, Reliability and Security (QRS)*, pages 318–328. IEEE, 2017.

[5] Shuo Wang and Xin Yao. Using class imbalance learning for software defect prediction. *IEEE Transactions on Reliability*, 62(2):434–443, 2013.