Anomaly Detection Challenge

Challenge 3: Network Anomaly Detection

Ishmeet Kaur (03677735)

Mustika Rizki Fitriyanti (03667399)

December 2016

1 Introduction

The aim of this challenge is to detect network attacks using binary classification. The dataset consisted of training data, featureset and the test data. The goal of the challenge was to classify the network traces in the test set as normal or anomalous with the help of a highly imbalanced training dataset.

2 Data Preprocessing and Analysis

The training dataset consisted of 56041 entries and 45 features . Out of all these entries in the training dataset, only 41 entries were of anomalous (attack type) whose attack categories were provided in the feature attack cat.

2.1 Data Analysis

As the dataset consisted of only 41 entries labelled as attack, the dataset was highly imbalanced in the ratio of 41: 56000.The conventional algorithms are often biased towards the majority class , the minority class are often treated as outliers thereby ignoring the minority class and misclassifying the dataset.So, this can be classified as an imbalanced dataset classification problem.Also, the range of all the features were different.

2.2 Handling Imbalanced Data

In order to handle the imbalanced dataset, there exists several approaches, out of which we tried the following:

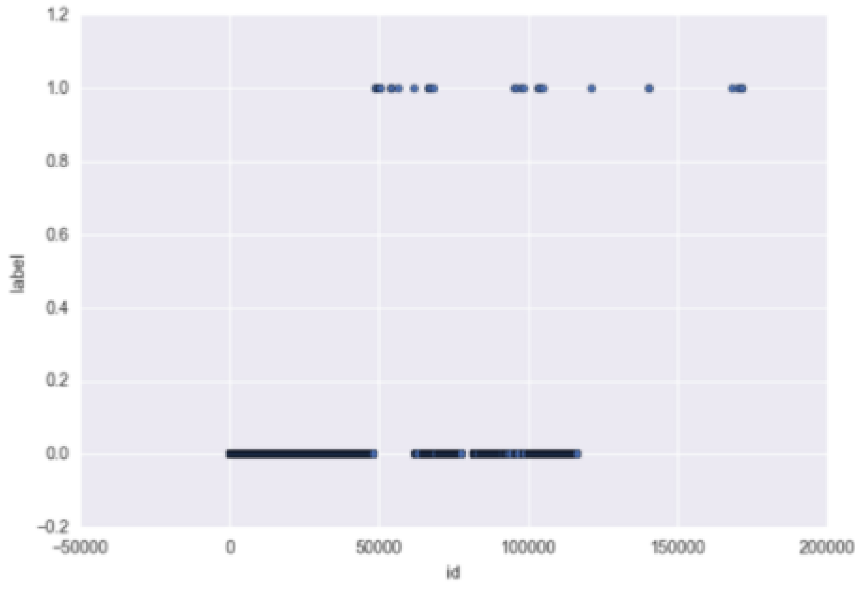

Figure 1. The imbalance proportion between label 0 and label 1 in the data

2.2.1 Entire Dataset(No modification)

First , the entire dataset was put into the models as it is to to evaluate the effect of the imbalanced data.

2.2.2 Manual Random UnderSampling

Random 41 samples were generated from the training dataset of label 0 and used for classification with the 41 samples labelled as 1.

2.2.3 Sampling Techniques using imbalanced-learn package

The data was balanced using a package called imbalanced-learn , a python package offering a number of re-sampling techniques commonly used in datasets showing strong between-class imbalance.The algorithms tried from the package are mentioned below:

- Undersampling Under-sampling refers to the process of reducing the number of samples in the majority classes.
  - Random majority under-sampling.
  - Extraction of majority-minority Tomek links[1]: Two Modifications of CNN which uses the samples in the boundary.
  - Under-sampling with Cluster Centroids
  - Near Miss-(1,2,3) (The classification of a test point based on the classes of its nearest neighbours)

    - Condensed Nearest Neighbour
    - One-Sided Selection
    - Neighborhood Cleaning Rule
    - Edited Nearest Neighbours
    - Repeated Edited Nearest Neighbours

- Oversampling It refers to the process of increasing the number of samples in the minority classes.
    - Random minority over-sampling with replacement
    - SMOTE - Synthetic Minority Over-sampling Technique
    - SMOTE(1 and 2) - Borderline SMOTE of types 1 and 2
    - SVM SMOTE - Support Vectors SMOTE
    - ADASYN - Adaptive synthetic sampling approach for imbalanced learning
- Over-sampling followed by under sampling Using SMOTE + Tomek links ,SMOTE + ENN
- Ensemble sampling Using Balance Cascade and Easy Ensembling

2.2.4 Anomaly Detection

As anomaly detection is the detection of rare events. Class imbalance can also be an anomaly detection problem .We used One Class SVM to train the classifier and analysed the anomalies.

2.2.5 Class Weights

Classification algorithms like Linear SVC,SGD, Logistic Regression and Random Forest was used to put additional cost on the model for making classification mistakes on the minority class during training.

3 Feature Engineering

The features importances were analysed using simple brute force analysis and Gradient Boosting Classifier.Some features which were notably found important are illustrated by the graph :

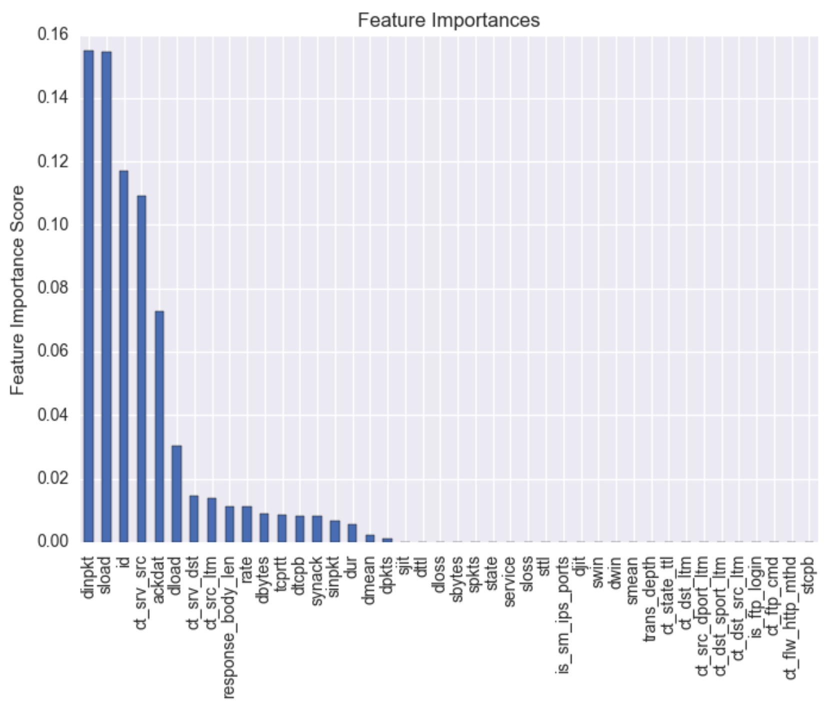

Figure 2. Feature Importance Score on the training data

3.1 Dimensionality Reduction by PCA

The best n components for the PCA of all the features providing the maximum accuracy (that is providing features with high variance) was found(which was 10).

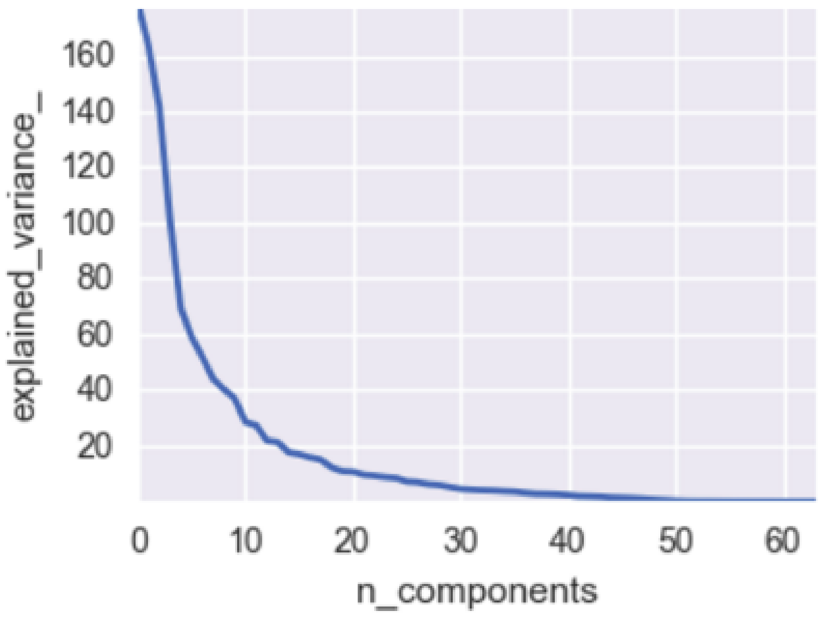

Figure 3. The number of components for PCA

3.2 Normalization

3.3 Feature Selection

All the features were tried initially with and without sampling along with normalization . Also, the reduced features by dimensionality reduction of PCA were also tried but did not give good results.

4 Anomaly Detection

OneClass SVM was used for anomaly detection with all the features to detect the 41 outliers out which many entries in the training data with label 0 were misclassified.

5 Models and Class Weighting

We tried fitting our dataset to the following models:

- Naive Bayes Classifier
- Decision Tree Classifier
- K- nearest Neighbors
- Random Forest

- Logistic Regression
- AdaboostClassifier
- Support Vector Machine
- SGD
- Adagrad
- One Class SVM
- Gradient Boosting Classifier
- Extra Trees Classifier
- OneVs RestClassifier

All these models were used for fitting the training data with a stratified cross validation of 5 and 10.

6 Observations and Results

- Original Features gave an accuracy till 0.99 in the training set but the accuracy decreased to 0.52 in the test set as the data was highly imbalanced.
- Manual Undersampling of the dataset gave a lower accuracy as compared to the accuracy of the random undersampling of the imbalanced learn package.
- Increasing the metrics for choosing the best model proved to be useful for selecting the classification algorithm like precision,recall, f1weight.
- Undersampling using condensed nearest neighbours and near miss methods highly overfitted the dataset giving an accuracy of above 0.95 but lowering the test data accuracy .
- Ensemble Sampling using Balance Cascade with PCA (10 components) and easy ensemble increased the accuracy to 0.6799 .
- Easy Ensemble increased the accuracy to 0.75912 with random forest(81 number of estimators)
- Oversampling methods like Smote which tried to reduce the imbalance by "synthetic" oversampling (creation of additional tuples with the "minority" class) resulted in overfitting .
- Anomaly Detection methods like one class SVM , used to fit the dataset showed a slight overfitting to the majority class but detected 2801 anomalies in the training data .

- Class weighting methods when used with Random forest gave a higher cross validation accuracy but were highly overfitting(45 percent accuracy)
- SVC(linear) when used for classification for the class weights had a very long runtime because of which SGD (Stochastic Gradient Descend) was used which treats the data in batches and performs a gradient descent aiming to minimize expected loss.
- SGD when used with default parameters gave an accuracy of 0.72491 with increased number of iterations(1000)
- SGD was highly sensitive to alpha(the learning rate), increasing alpha increased the overfitting and reduced the accuracy but decreasing alpha reduced the accuracy. The best alpha so far founded by brute force was around 0.00074 which is the ration of samples with label 1(attack) to the whole dataset.
- SGD with the alpha (0.00074) and 1000 iterations gave an accuracy of 73.35(public score) on the test data.
- To keep the learning rate constant, Adagrad was tried using the lighting package available in the scikit learn contribution projects, but it highly overfitted the data giving a cross validation accuracy of 0.99
- Balance Cascade, which explored the majority class in a supervised manner, whereas Easy Ensemble, learns different aspects of the original majority class in an unsupervised manner performed in an equivalent manner as random undersampling and increased the accuracy to 0.79
- Bagging Algorithms performed better(like random forest and Extra trees Classifier) the data had high variance(when trained with the original undersampled-random data) and increased the accuracy by around 0.5(to 0.84612)

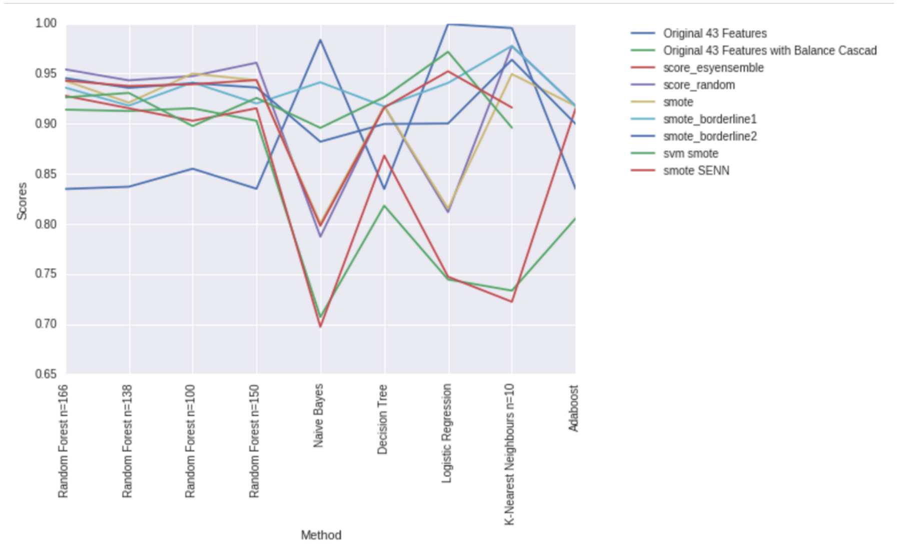

Figure 4. The score using Oversampling

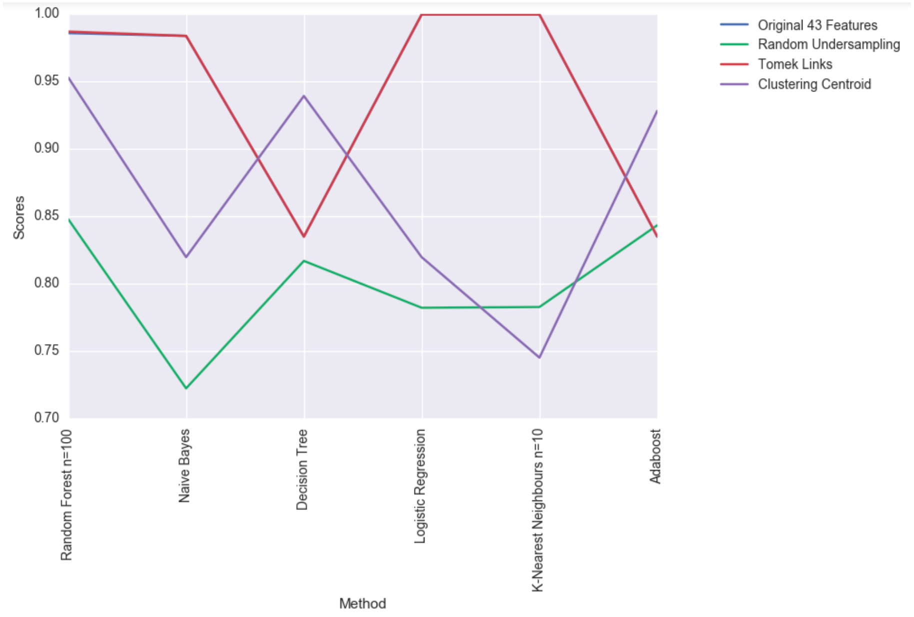

Figure 5. The score using undersampling(I)

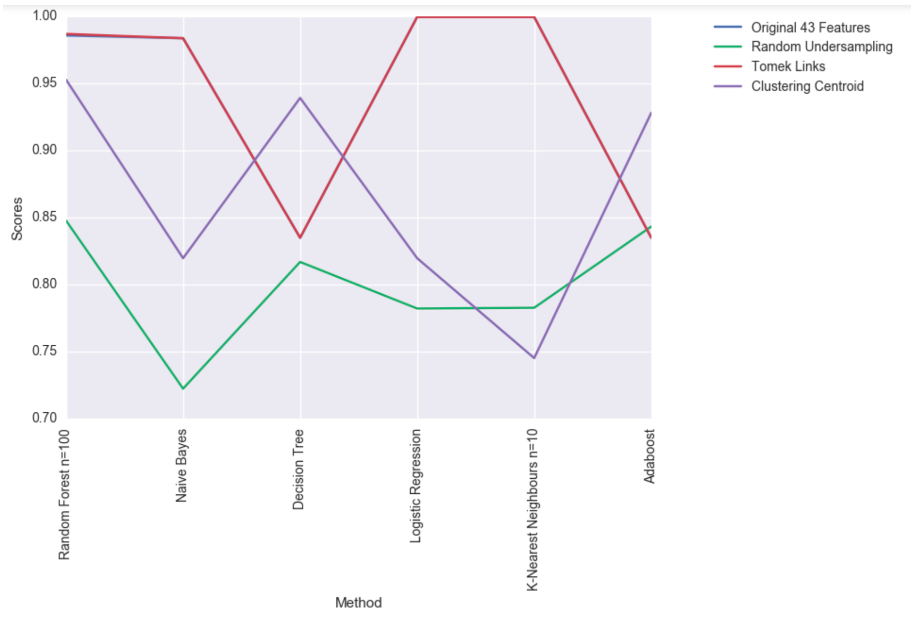

Figure 6. The score using undersampling(II)

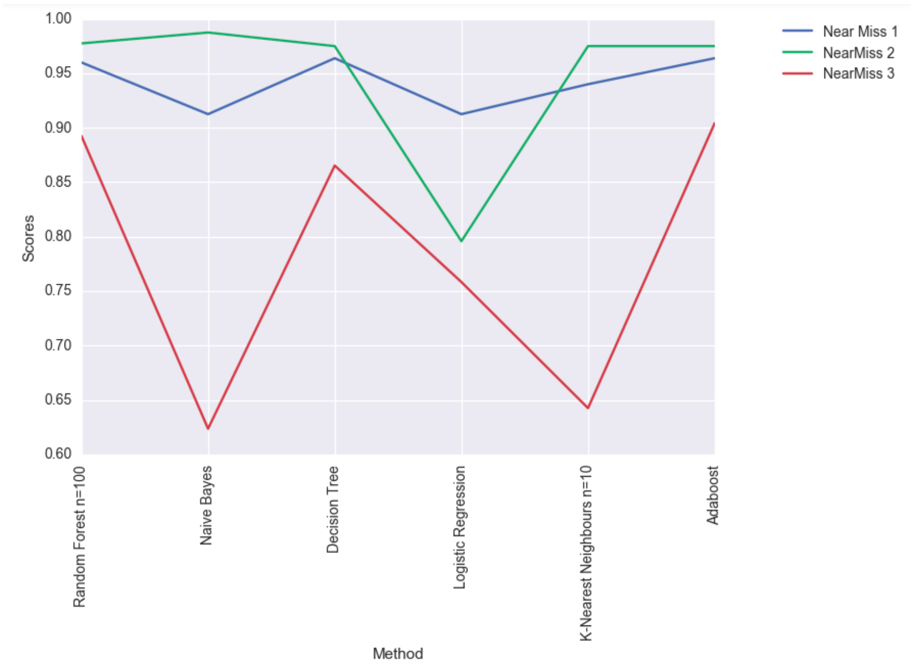

Figure 7. The score using Near Miss Methods Undersampling

7 Summary

The best algorithm found was extra trees classifier with 186 number of estimators with a public score of 0.84612.

References

[1] $https://github.com/scikit-learn-contrib/imbalanced-learn$

[2] $http://contrib.scikit-learn.org/lightning/$