TAIPEI TECH Since1912 國立臺北科技大學 National Taipei University of Technology

Project Title: Anomaly Detection in Big Data Using Apache Spark

Big Data Mining and Application Term project final report

Prepared By:

1. Direselign Addis (ID: 106999405)
2. Zewdie Mossie (ID: 106999404)

Presentation date: June 29/2018

Submitted to:

Prof. J. H. Wang

1. Introduction

The constant increase in the size of networks and the data massively produced by them has made the data analysis very challenging principally the data attaining the boundaries of big data and it becomes even more difficult to detect intrusions in the case of big data. In this era, the experts find very limited tools and methods to analyze big data for security reasons. Either need to device new tools or we can use existing tools in a novel manner to achieve the purpose of big data security analysis. In this project, we use apache spark a big data tool for analyzing the big dataset for anomaly detection. The anomaly detection performed after some preprocessing and then using different machine learning algorithms Random forest and Kmeans clustering. The main objective of this project is to analyze the dataset using MAP and REDUCE and check the most efficient algorithm in the context of anomaly detection. In this regard, we set to compare their training time, prediction time, and the rate of accuracy. The analysis implemented on Kddcup99 version NLS-KDD dataset available for research. Although this dataset is of size in megabytes but it meets our purpose here for big data security analytics.

2. Apache spark

Apache spark is an open source big data tool. It is an efficient big data analysis tool came with the concept of in-memory data storage and processing which makes it faster than its counterparts. It is a fast processing engine for parallel processing. Basically, spark was designed to improve the concept of Map-reduce model. The main part of spark libraries include MLlib which is machine learning library and it is also used in our project. It also contains other useful like Spark streaming, Spark sql and GraphX. Spark support different languages like scala, python, java, and R but we use python due to its ease of use and widely usage in the data science. Spark also supports integration of other big data tools.

In this project, first we need to train the classifier with dataset and then test it with the testing version of the dataset. Furthermore, we also calculate the training time, prediction time and accuracy of the algorithms. The algorithms as a part of our analysis are Random forest and KMeans clustering.

3. Intrusion Detection

Network intrusions detection protects a computer network from unauthorized users, including perhaps insiders. The intrusion detector learning task is to build a predictive model (i.e. a classifier) capable of distinguishing between bad connections, called intrusions or attacks, and good or

normal connections. A connection is a sequence of TCP packets starting and ending at some well-defined times, between which data flows to and from a source IP address to a target IP address under some well-defined protocol. Each connection is labeled as either normal, or as an attack, with exactly one specific attack type. Attacks fall into four main categories:

DOS: denial-of-service, e.g. syn flood, R2L: unauthorized access from a remote machine, e.g. guessing password, U2R: unauthorized access to local superuser (root) privileges, e.g., various "buffer overflow" attacks and Probing: surveillance and other probing, e.g., port scanning.

4. Dataset

In our analysis, we will use NSL-KDD dataset for the comparison of different algorithms using apache spark. It is a widely used dataset by the research community in network security. The dataset has **125973** instances in training and **22544** instances in test dataset and these instances contain 41 features. It contains different types of attacks which are about 38 in training and 28 in testing dataset. The dataset was generated in MIT Lab. It was produced by processing the tcpdump portions of DARPA IDS evaluation dataset.

5. Proposed Method

In the proposed model, the dataset analyzed for security analytics which is stored in the local computer. The dataset is loaded by analyzing tool in our case is the apache spark. After loading the data, it undergoes the different data standardization process. In the project, normalization is done by using the common techniques of one hot encoding, and others. After the normalization process the data analyzed for anomaly detection using different machine learning algorithms. The whole process is depicted by model expressed as in figure 1.

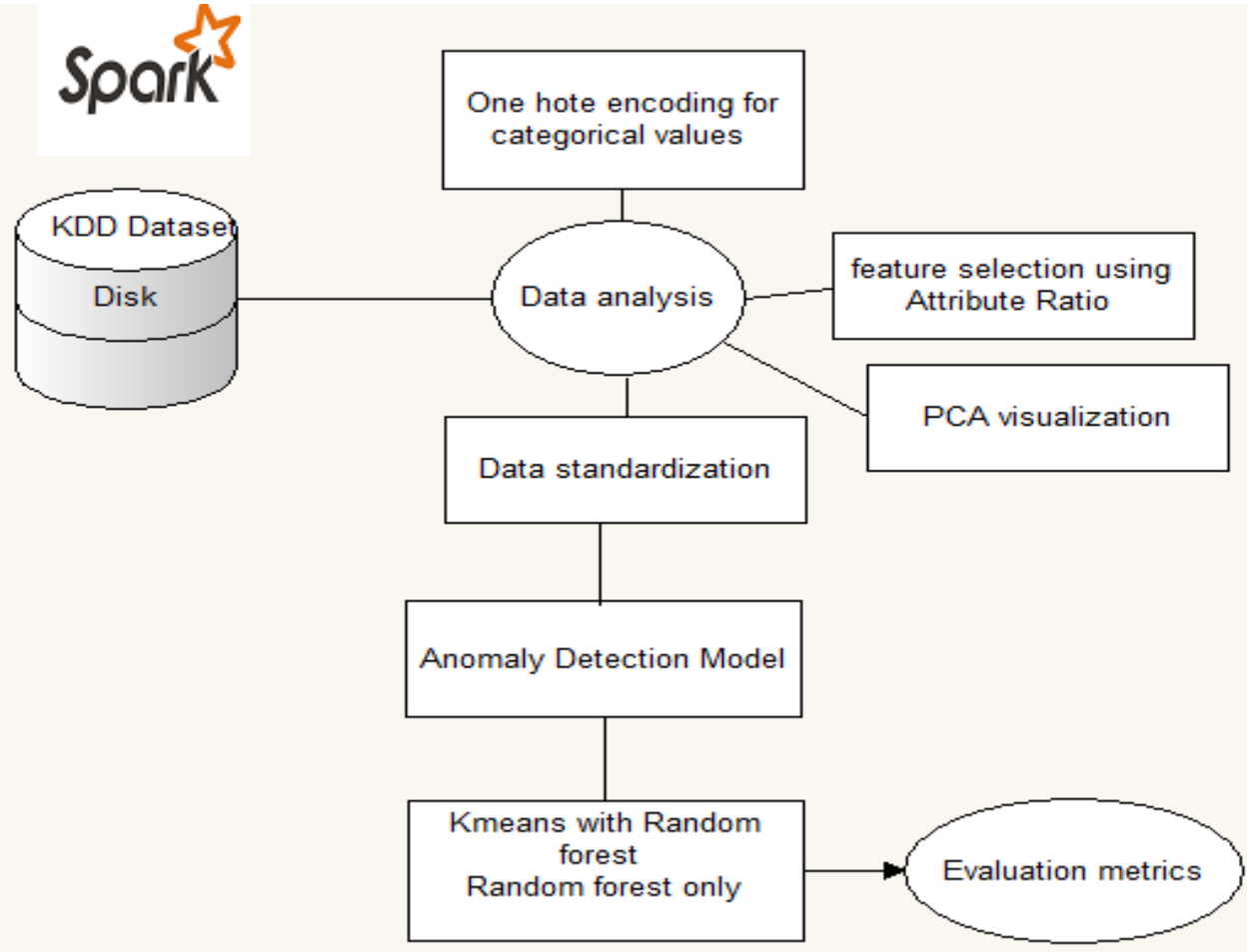


Figure 1: Spark based anomaly detection model

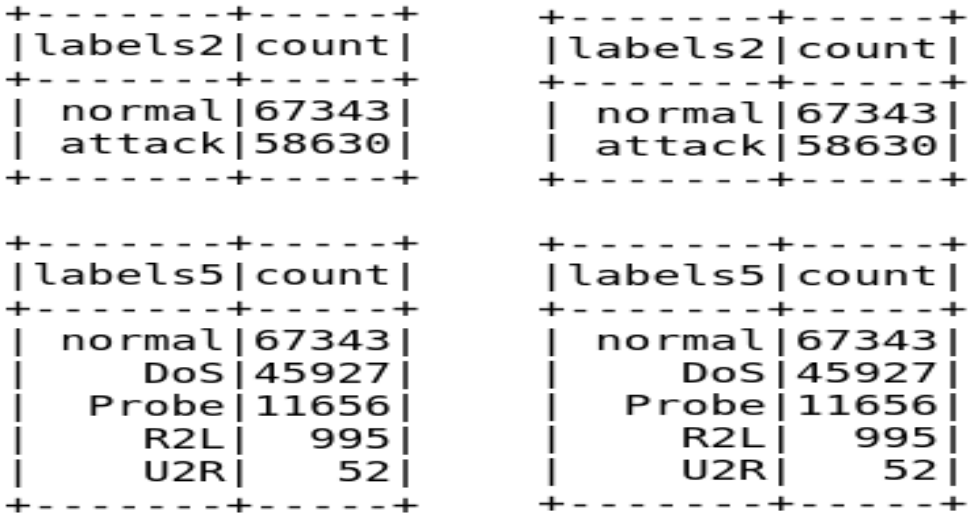
6. Experiment and Results Analysis

6.1. Experiment setup

We used Apache spark 2.3.0 Open source Distributed computing platform to prepare data, compute the feature selection subsets for AR, train the machine learning and evaluate the classification performance on each of these feature sets. The machine we used is HP desktop with Ubuntu 18.04 OS which has 24 GB RAM. The algorithm selected and their parameters setup is Kmeans with k=8, intialstep=25, maxiter=100 and Random forest with numberTree=500, MaxDepth=20.

6.2. Data Preprocessing

The first part of data preparation is dividing connections into normal and attack classes based on 'labels' column. Then attacks are splited to four main categories: DoS, Probe, R2L and U2R. The following figure shows the result of the labels figure1 for training and figure 2 for testing.



labels2	count
normal	67343
attack	58630

labels5	count
normal	67343
DoS	45927
Probe	11656
R2L	995
U2R	52

Figure 1: Training

labels2	count
normal	67343
attack	58630

labels5	count
normal	67343
DoS	45927
Probe	11656
R2L	995
U2R	52

Figure 2: Testing

We have also done identification of the type of data and ready for machine learning. One Hot Encoding (OHE) is used for treating categorical variables. Custom function created for demonstration purposes. However, it could be easily replaced by Spark OneHotEncoder. We also use the standardization both in training and testing dataset since we use the distance based algorithms to increase the performance as shown in figure 3.

id	indexed_features	labels2_index	labels2	labels5_index	labels5
0	[-0.0186098224759...	0.0	normal	0.0	normal
1	[-0.0186098224759...	0.0	normal	0.0	normal
2	[-0.0186098224759...	1.0	attack	1.0	DoS
3	[-0.0186098224759...	0.0	normal	0.0	normal
4	[-0.0186098224759...	0.0	normal	0.0	normal
5	[-0.0186098224759...	1.0	attack	1.0	DoS
6	[-0.0186098224759...	1.0	attack	1.0	DoS
7	[-0.0186098224759...	1.0	attack	1.0	DoS
8	[-0.0186098224759...	1.0	attack	1.0	DoS
10	[-0.0186098224759...	1.0	attack	1.0	DoS

Figure 3: Sample Standardization result

6.3. Feature selection using Attribute Ratio

Attribute Ratio approach is used for feature selection purposes. This approach was described by Hee-su Chae and Sang Hyun Choi in [Feature Selection for efficient Intrusion Detection using Attribute Ratio]. So we also followed their approach to do this project.

Attribute ratio (AR) is calculated by average or frequency of features. To calculate AR, we have to calculate class ratio (CR). Class Ratio (CR) is attribute ratio of each class for Attribute j. CR is calculated by two methods according to the type of attributes.

CR can be calculated as for numeric:

$$\text{CR (j)} = \frac{Avg(C(j)}{Avg(total)} \qquad (1)$$

CR can be calculated as for binary:

$$\text{CR (j)} = \frac{freqency(1)}{frequency(0)} \qquad (2)$$

AR is maximum value of CR. AR can be calculated as:

$$\text{AR (j)} = \text{Max (CR (j))} \qquad (3)$$

The figure 4 shows sample result of attribute ration based on the above formula.

```
          31
          6.017663955688477

Out[26]: OrderedDict([('num_shells', 326.11353550295854),
                      ('urgent', 173.03983516483518),
                      ('num_file_creations', 62.23362492770388),
                      ('num_failed_logins', 46.03855641845592),
                      ('hot', 40.77451681709518),
                      ('dst_bytes', 9.154854355343401),
                      ('src_bytes', 8.464064204948945),
                      ('duration', 7.225829157212557),
                      ('dst_host_srv_diff_host_rate', 5.756880682756574),
                      ('dst_host_diff_srv_rate', 4.83734184897426),
                      ('num_access_files', 4.694879248658319),
                      ('dst_host_same_src_port_rate', 4.393080378884017),
                      ('num_compromised', 4.338539274983927),
                      ('diff_srv_rate', 4.069085485070395),
                      ('dst_host_srv_rerror_rate', 3.667920527965924),
                      ('srv_rerror_rate', 3.667741802325429),
```

Figure 4: Attribute Ration sample result

6.4. Data visualization using PCA

PCA algorithm is used for visualization purposes.

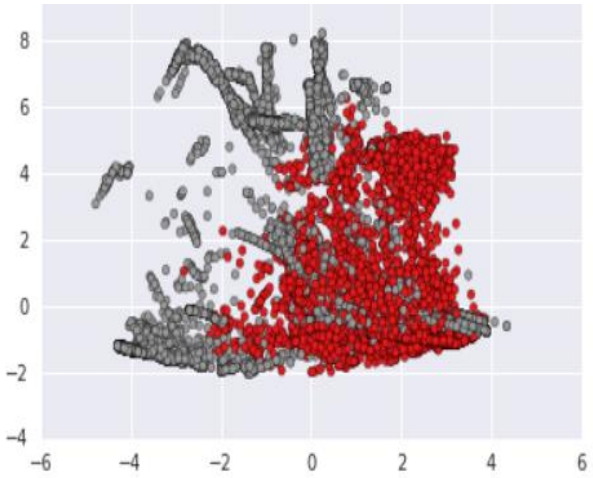


Figure 5: Attack vs Normal

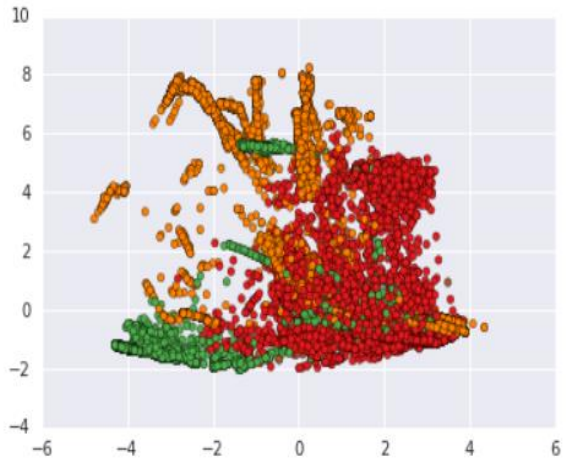


Figure 6: 4 Attack vs Normal

6.5. Evaluation Metrics

Different metrics from sklearn are used for evaluating results. The most important from them for this task are False positive Rate, Detection Rate and F1 score. But we also include Accuracy, AUC, precision and recall. As evaluating via sklearn requires to collect predicted and label columns to the driver, we again replaced with PySpark metrics later.

6.6. KMeans Clustering

The idea of the first approach is to clustered data into clusters and then train different Random Forest classifiers for each of the clusters. As KMeans cannot truly handle binary/categorical features only numeric features are used for clustarization. Clusters are splited into two categories. Frist category contains clusters that have both 'attack' and 'normal' connections and have more than 25 connections. For the first category Random Forest classifiers are applied. Second category contains all other clusters and maps cluster to 'attack' or 'normal' based on majority. All clusters that contains less or equal than 25 connections are treated as outliers and are mapped to 'attack' type. The following is the output of the clustering.

cluster_labels2	attack	normal	count
0	6659	46448	53107
1	9125	2266	11391
2	626	61	687
3	27742	101	27843
4	2670	5073	7743
5	1	0	1
6	0	24	24
7	2	42	44

Figure 7: KMeans clustering

6.7. Results

Based on the selected standardization, feature selection and clustering our proposed method for the project achieved the following results. The feature selection was performed on 41 features and we used attribute ration to select the best features and compare the result using evaluation

metrics. The following shows sample prediction using KMeans clustering and random forest algorithms.

id	labels2_index	labels2	labels5	kmeans_rf_prob	kmeans_rf_pred	dos_prob
0	1.0	attack	DoS	0.9999906726341035	1.0	0.9999669406023199
1	1.0	attack	DoS	0.999950223923172	1.0	0.9999233042386835
2	0.0	normal	normal	0.001824135578193935	0.0	0.0
3	1.0	attack	Probe	0.9130690507248003	1.0	0.057441474540098375
4	1.0	attack	Probe	0.27753057889822597	1.0	0.032
5	0.0	normal	normal	0.0	0.0	0.0
6	0.0	normal	normal	0.1624893085106383	1.0	0.006
7	1.0	attack	R2L	0.0	0.0	0.003220413436692506
8	0.0	normal	normal	0.0	0.0	0.0
9	1.0	attack	R2L	3.448275862068966E-4	0.0	0.0

only showing top 10 rows

Figure 8: Sample prediction for DOS

6.8. Detecting each type of attack

Supervised approach for detecting each type of attacks separately both after and before feature selection is done for comparison purpose.

	normal	attack
normal	13248	80
attack	427	11378

Accuracy = 0.979827
AUC = 0.978913

False Alarm Rate = 0.0060024
Detection Rate = 0.963829
F1 score = 0.978206

	precision	recall	f1-score
0.0	0.97	0.99	0.98
1.0	0.99	0.96	0.98
avg / total	0.98	0.98	0.98

Figure 9: Using the AR DOS

	normal	DoS
normal	9645	66
DoS	1656	5802

Accuracy = 0.899703
AUC = 0.88558

False Alarm Rate = 0.00679642
Detection Rate = 0.777957
F1 score = 0.870779

	precision	recall	f1-score
0.0	0.85	0.99	0.92
1.0	0.99	0.78	0.87
avg / total	0.91	0.90	0.90

Figure 10: Using the whole DOS

```
        normal   Probe
normal   13322       6
 Probe      12    2373

Accuracy = 0.998854
AUC = 0.997259

False Alarm Rate = 0.00045018
Detection Rate = 0.994969
F1 score = 0.996222

             precision    recall  f1-score   support

        0.0       1.00      1.00      1.00     13328
        1.0       1.00      0.99      1.00      2385

avg / total       1.00      1.00      1.00     15713
```

Figure 11: Using the AR Prob

```
        normal   Probe
normal    9493     218
 Probe     946    1475

Accuracy = 0.904055
AUC = 0.793402

False Alarm Rate = 0.0224488
Detection Rate = 0.609252
F1 score = 0.717064

             precision    recall  f1-score

        0.0       0.91      0.98      0.94
        1.0       0.87      0.61      0.72

avg / total       0.90      0.90      0.90
```

Figure 12: Using the whole Prob

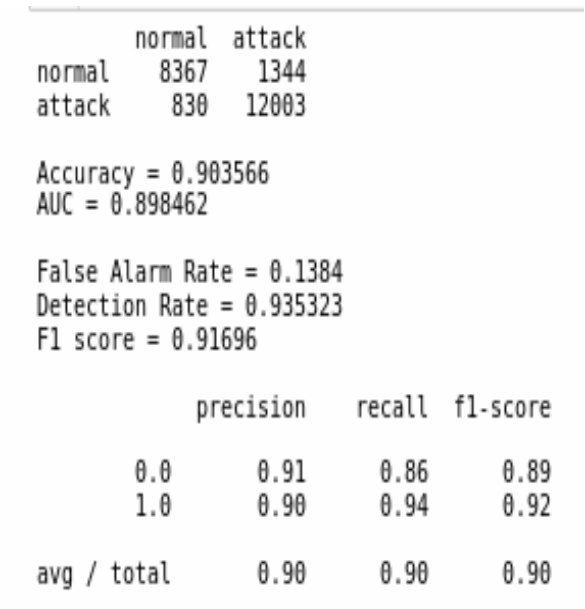
```
         normal R2L&U2R
 normal   13324       4
R2L&U2R      14     214

Accuracy = 0.998672
AUC = 0.969148

False Alarm Rate = 0.00030012
Detection Rate = 0.938596
F1 score = 0.959641

             precision    recall  f1-score

        0.0       1.00      1.00      1.00
        1.0       0.98      0.94      0.96

avg / total       1.00      1.00      1.00
```

Figure 13: Using the AR U2R&R2L

```
         normal R2L&U2R
 normal    9709       2
R2L&U2R    2727     227

Accuracy = 0.784524
AUC = 0.53832

False Alarm Rate = 0.000205952
Detection Rate = 0.076845
F1 score = 0.142633

             precision    recall  f1-score

        0.0       0.78      1.00      0.88
        1.0       0.99      0.08      0.14

avg / total       0.83      0.78      0.71
```

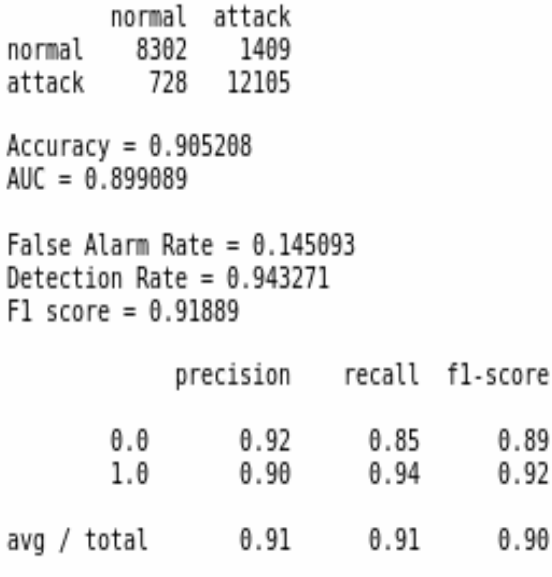
Figure 14: Using the whole U2R&R2L

Combined attack Detection



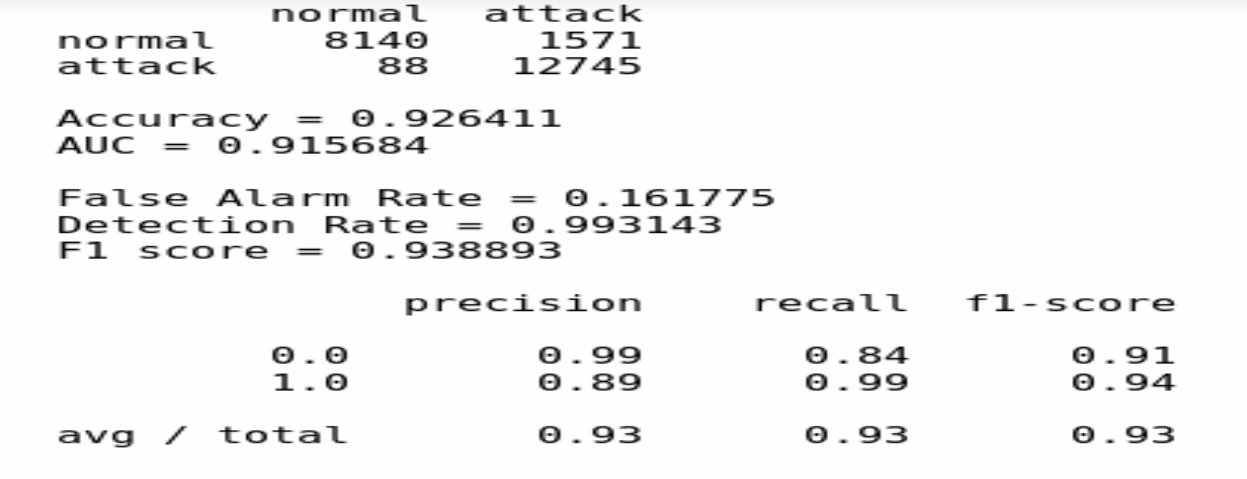
```
        normal  attack
normal    8367    1344
attack     830   12003

Accuracy = 0.903566
AUC = 0.898462

False Alarm Rate = 0.1384
Detection Rate = 0.935323
F1 score = 0.91696

              precision    recall  f1-score

       0.0       0.91      0.86      0.89
       1.0       0.90      0.94      0.92

avg / total      0.90      0.90      0.90
```

Figure 14: Using the AR



```
        normal  attack
normal    8302    1409
attack     728   12105

Accuracy = 0.905208
AUC = 0.899089

False Alarm Rate = 0.145093
Detection Rate = 0.943271
F1 score = 0.91889

              precision    recall  f1-score

       0.0       0.92      0.85      0.89
       1.0       0.90      0.94      0.92

avg / total      0.91      0.91      0.90
```

Figure 15: Using the whole



```
          normal  attack
normal      8140    1571
attack        88   12745

Accuracy = 0.926411
AUC = 0.915684

False Alarm Rate = 0.161775
Detection Rate = 0.993143
F1 score = 0.938893

                precision    recall  f1-score

          0.0        0.99      0.84      0.91
          1.0        0.89      0.99      0.94

avg / total          0.93      0.93      0.93
```

Only Random forest using AR

Summary

The best result was achieved by KMeans Clustering with Random Forest Classifiers. When the AR feature selection is used it gives around ~98-99% of detection rate with reasonable ~14-15% of false alarm rate and F1 score of 0.94.The computing time is also efficient in all aspects. So Spark based anomaly detection is effective for intrusion detection system.

Reference

1. http://Spark.appache.org/
2. [NSL-KDD](http://www.unb.ca/research/iscx/dataset/iscx-NSL-KDD-dataset.html) http://www.stepbystepcoder.com/using-spark-for-anomaly-frauddetection-k-means-clustering/
3. M. Zaharia, H. Karau, A. Konwinski, P. Wendell, "Learning spark lightning-fast analysis," Oreilly media 2015
4. Sang-Hyun Choi and Hee-Su Chae, Feature Selection using Attribute Ratio in NSL-KDD data, International Conference Data Mining, Civil and Mechanical Engineering (ICDMCME'2014), Feb 4-5, 2014 Bali (Indonesia)