Predicting Network Attack Using UNSW-NB15 Dataset

Divya Khandelwal Nancy Saxena Chintan Shah Wen-Hao Tseng
San José State University

November 2021

Abstract

Cyber attacks are one of the biggest threads in this era of digital world. It is very important to combat the network attacks to establish a secure environment for all the users of a network. This project focuses on creating and testing Machine Learning Models over a large dataset of raw network packets to detect network attacks. The dataset used, is created by Cyber Range Lab of UNSW Canberra. The project analyses the performance of different ML models like XGBoost, Random Forest, etc. over the dataset that has been preprocessed using techniques like Dimension Reduction, MinMax Scaling etc. The performance, in terms if Acuracy and F1 score, is studied for each model and the inferences like best working model are derived.

1. Introduction

The occurrence of cyber security incidents have proliferated in recent years. Almost every year, one or two major information security incidents attract the attention of the world. Numerous studies have already been conducted in the field of cyber security utilizing data mining technologies. Using the UNSW-NB15 Dataset [1], we will predict the network attack that is happening over the network. This dataset has nine types of attacks, namely, Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode and Worms unlike other dataset like KDD-99 dataset which has only four attack types DOS, R2L, U2R, and PROBE. The attack distribution data of UNSW-NB15 is shown in Figure 1.

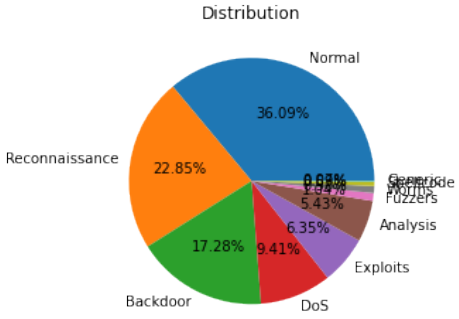

Figure 1

2. Literature Review

Many datasets have been used for Intrusion detection system like KDD'99. According to A. Divekar et al, "As with KDD-99, certain parameters were found unnecessary. A reduced set of 20 features found by Mean Decrease Impurity was used in this paper."[2]. Tavallaee et al. [3], while proposing an improved NSL-KDD, provided a comprehensive description of the KDD CUP 99's idiosyncrasies".Tavallaee et al. [3] developed NSL-KDD to rectify KDD-99 and overcome its drawbacks. However, it had some drawbacks like non representation of low footprint attacks[1]. A. Divekar et al, have compared and the datasets by applying preprocessing and feature Selection and found that uNSW-NB15 was found to be a modern substitute to NSL-KDD and KDD CUP 99 dataset.[2]
A svm based model was implemented by D. Jing and H. Chen.According to the authors"the performance

of our method is evaluated through accuracy, detection rate and false positive rate. Compared with other methods, the superiority of the proposed SVM method is shown by the experimental results."[4]. A multi-layer perceptron feed-forward artificial neural network with a single hidden layer was proposed by M. Al-Zewairi et al[5]. According to the authors, "The evaluation results demonstrate that the proposed classifier outperforms other models in the literature with 98.99% accuracy and 0.56% false alarm rate on unseen data."[5]The decision trees was one of the models that was compared with this Artificial neural network.

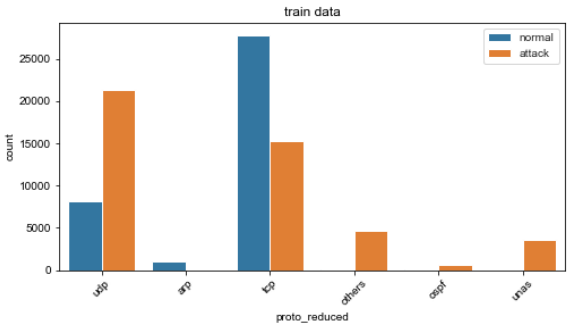

Figure 3.1

3. Exploratory Data Analysis

For our work, the UNSW-NB15 dataset contains 257,673 data instances with 49 features. The total classes of this dataset are 10 classes: one is for a *normal* network data (93 000 instances) and nine classes of anomalous network data (attacks classes). The attacks involved were *backdoor* (2,329 instances), *analysis* (2,677 instances), *fuzzers* (24,246 instances), *shellcode* (1 511), *reconnaissance* (13,987 instances), *exploits* (44,525 instances), *DoS* (16,353 instances), *worms* (174 instances), and *generic* (58,871 instances). The data distribution data of UNSW-NB15 as in Figure 1.

We perfomed 3 kinds of exploratory data analysis on the UNSW-NB15 dataset, namely countplot or barplot for all categorial or columns with small number of unique values, plot PDF for numerical features, and Correlation of the features and its heatmap.

In this data set, there are total 9 attack categories of attack and normal is non-attack. The data is highly imbalanced and have lots of non-attack than attacks. The most occured attack data categories are "*Reconnaissance*," "*Backdoor*," "*DoS*," "*Exploits*" and "*Analysis*." In the **protocol** category, most of the values are consists of udp and tcp. For attacks count of udp is lot higher. The bar plot is shown in Figure 3.1.

In **attack** data "dns" is present higher than any other values. There are few no of others and http also. In the **state** category we found the imbalce there are lots of int state for attacks.

For numerical features, we plot PDF. For better visualization, we also used log scale. There are some results worth pointing out. **dload**: destination bits per second, in Figure 3.2.1 and 3.2.2, for attack data all the values are very close to 0. For normal data they are distributed all over, has values close to 0 and also very large values. **sbytes**: source to destination bytes, most of normal category values are close to 0. Attack categories has most of its values around 5 in log1p graph. The spread of values is wider in attack compared to normal.

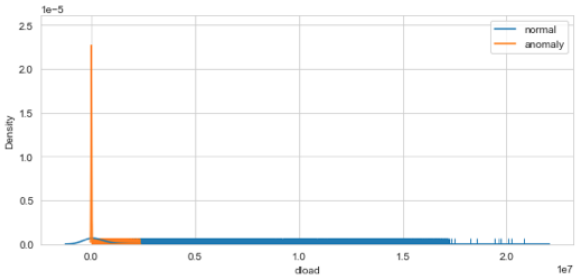

Figure 3.2.1

To get correlation values for all the features, we plot heatmap of correaltion shown in Figure 3.3 for better visualization. The most correlated features are: sbytes and sloss, sbytes and sloss, swin and dwin. These features are having very high correlation between them more than 95%. Although some features have high correlation between them, some of them is because they share same values, for instance, swin and dwin

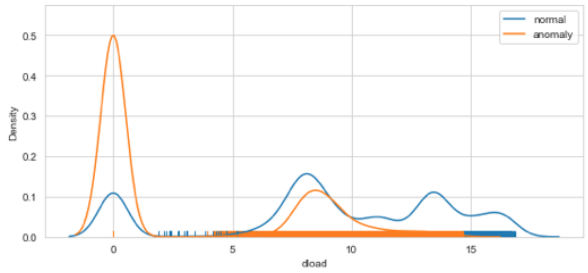

Figure 3.2.2

have correlation values is 99% between them. Even though these 2 columns are numerical but most of their values are only 0 and 255.

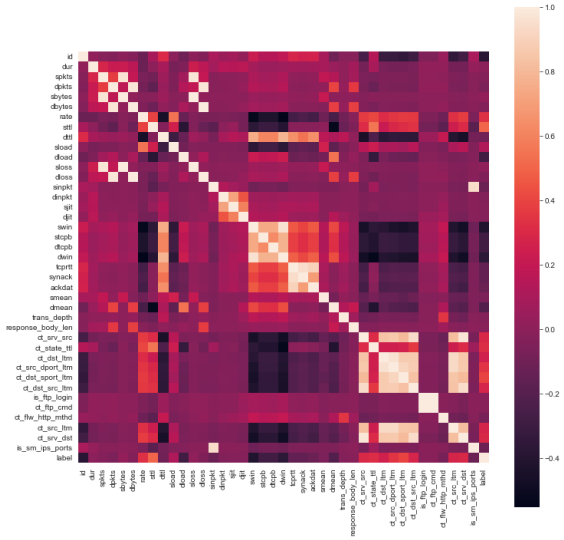

Figure 3.3

4. Data Preparation

Large data that is to be studied and worked upon is often raw and needs cleaning. Data is cleaned for errors like missing values, incorrect values, unnecessary and duplicate data etc. Therefore, data cleaning is the first step that is performed before working ahead with any dataset. In this project, following are the basic data preparation steps that have been performed:

1. Dropping unnecessary columns: The columns that add no information to the dataset are dropped so that number of features to work with are reduced.

2. Dealing with Missing Values: Generally, the dataset contains some missing values which need to be dealt with attentively. There are different ways to deal with a missing value:

   2.1 Drop the missing value record(row) or feature(column)

   2.2 Replace missing value with appropriate mean/mode/median value of the feature(column)

In this project, since input dataset did not contain any missing values, this step is not performed.

3. Incorrect values: There could some invalid entries into a feature that are not of the expected datatype of that feature. These values need to be corrected. In this project, few columns like "is_ftp_login" and "is_sm_ips_ports" that expected binary input contained non binary value. This is corrected to get non-erroneous results.

5. Data Pre-Processing

Data pre-preocessing is performed in order to generate a dataset that aids Machine Learning to predict more accurate results. Following data preprocessing steps have been performed in this project:

5.1 Encoding:

Dataset generally contains columns that hold categorical values. This is because categorical values are more decriptive that numerical values. But ML models cannot work with any non-numerical values. So, prior to feed data to Machine Learning model, encoding is performed. There are two types of encodings that are ususally performed:

1. Label Encoding: It is a simple technique to assign numbers to different categorical values. But it is generally misinterpreted by algorithms as having some sort of hierarchy/order.

2. One-Hot Encoding: It eliminates the hierarchy/order issues. But it adds more columns(features) to the data set which may contribute to overfitting.

In this project, both type of encoders were tested for. It is then infered from the results, that one-hot encoding is leading to increase in the feature numbers from 45 to above 200. So, label encoder is the best choice. It is used on the categorical columns like "dtype,""proto," "stype."

5.2 Data Normalization

Data normalization is done to make the data set cohesive and similar across all the fields and columns.In the UNSW-NB15 dataset, if the data normalization is not done it will lead to the suppression of the effectiveness of an important equally important attribute(on a lower scale because of other attributes having values on a larger scale.

5.2.1 MinMax normalization

The min max normalization is used to transform features to be on a similar scale. It reduced the values to the range of [0,1] The new point is calculated as :

X_new = (X - X_min)/(X_max - X_min)

Geometrically speaking, transformation squishes the n-dimensional data into an n-dimensional unit hypercube. The Description of the features after applying MinMax scaling:

	0	1	2	3	4	5	6	7	8	9	10
count	2.576730e+05	257673.000000	257673.000000	257673.000000	257673.000000	257673.000000	257673.000000	257673.000000	257673.000000	257673.000000	257673.000000
mean	2.077859e-02	0.834364	0.129659	0.434147	0.001764	0.001680	0.000596	0.000982	0.091254	0.705886	0.333681
std	9.957177e-02	0.161742	0.187162	0.088742	0.012771	0.010164	0.012105	0.009974	0.160345	0.401915	0.443945
min	0.000000e+00	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	1.333334e-07	0.856061	0.000000	0.400000	0.000094	0.000000	0.000006	0.000000	0.000031	0.243137	0.000000
50%	7.141668e-05	0.856061	0.000000	0.400000	0.000282	0.000182	0.000035	0.000012	0.002966	0.996078	0.114173
75%	1.142962e-02	0.901515	0.166667	0.500000	0.001033	0.000908	0.000093	0.000073	0.125000	0.996078	0.992126
max	1.000000e+00	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

Figure 5.2.1 shows the description of top 10 features after applying MinMax scaler.

5.2.2 Z-Score Normalization

Standardization or Z-Score Normalization is the transformation of features by subtracting from mean and dividing by standard deviation. This is often called as Z-score. The new data points are added as :

X_new = (X - mean)/Std

Geometrically speaking, it translates the data to the mean vector of original data to the origin and squishes or expands the points if std is 1 respectively.

Standardization does not get affected by outliers because there is no predefined range of transformed features.

	0	1	2	3	4	5	6	7	8	9	10
count	2.576730e+05	2.576730e+05	2.576730e+05	2.576730e+05	2.576730e+05	2.576730e+05	2.576730e+05	2.576730e+05	2.576730e+05	2.576730e+05	2.576730e+05
mean	-4.083106e-15	1.083822e-14	4.384457e-14	-4.342349e-13	-7.571561e-14	8.059113e-14	-1.773562e-15	3.449838e-14	-1.205741e-14	-2.244258e-15	1.185186e-13
std	1.000002e+00	1.000002e+00	1.000002e+00	1.000002e+00	1.000002e+00	1.000002e+00	1.000002e+00	1.000002e+00	1.000002e+00	1.000002e+00	1.000002e+00
min	-2.086799e-01	-5.158630e+00	-6.927619e-01	-4.892246e+00	-1.381212e-01	-1.653309e-01	-4.919593e-02	-9.840893e-02	-5.691122e-01	-1.756311e+00	-7.516275e-01
25%	-2.086786e-01	1.341449e-01	-6.927619e-01	-3.847878e-01	-1.307654e-01	-1.653309e-01	-4.867802e-02	-9.840893e-02	-5.689202e-01	-1.151363e+00	-7.516275e-01
50%	-2.079627e-01	1.341449e-01	-6.927619e-01	-3.847878e-01	-1.160537e-01	-1.474715e-01	-4.629561e-02	-9.719142e-02	-5.506790e-01	7.220262e-01	-4.944485e-01
75%	-9.389195e-02	4.151772e-01	1.977335e-01	7.420767e-01	-5.720722e-02	-7.603381e-02	-4.149626e-02	-9.113118e-02	2.104601e-01	7.220262e-01	1.483170e+00
max	9.834345e+00	1.024081e+00	4.650211e+00	6.376400e+00	7.816452e+01	9.822219e+01	8.256263e+01	1.001599e+02	5.667466e+00	7.317834e-01	1.500906e+00

Figure 5.2.2 shows the description of top 10 features after applying standard scaler.

In UNSW-NB15 dataset, the outliers play an important role. The outliers are the datapoints, where the algorithm can predict anomaly. In later sections, we will compare the effects of both kinds of Normalization on the tree based algorithms, with respect to accuracy and F1 score.

6. Data Analysis

6.1 Correlation Analysis

Correlation feature selection is used for eliminating or dropping columns which have high correlation variance. Figure 3.3 shows the complete visualization of correlation values.

sbytes sloss 0.995027191 dbytes dloss 0.99710885

After analysing the correlation matrix, sbytes is correlated with sloss by 0.995 and dbytes is correlated with dloss by 0.9971. Since selecting correlation variance as 0.95 will result in loss of most of the data. So to optimize that 0.99 factor is considered. Therefore keeping only one the correlated columns, 'sloss' and 'dloss' columns are dropped.

6.2 Principal Component Analysis

Principal Component Analysis, or PCA, is a dimensionality-reduction method that is often used to reduce the dimensionality of large data sets, by transforming a large set of variables into a smaller one that still contains most of the information in the large set. But this dimensionality reduction technique may reduce the accuracy of any model at quite a high rate. The principal component analysis algorithm was applied on the dataset considering the to capture the minimum variance of 99% else considering the less percent will lead to loss of most of the data. After applying PCA, the number of features that are responsible for the detection of 99% of variance has been reduced to 29 from 42 columns Figure shows the clustering of the PCA

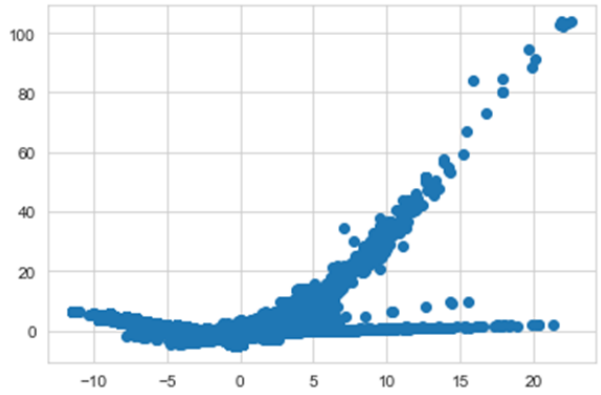

Figure 6.2.1

7. Data Modeling

Machine models needs to be trained on the network packets from the dataset to allow them to detect network attacks. There are different machine learning models available, but for this project, the following four are considered:

1. XGBoost.
2. GB Gradient.
3. Decision Tree.
4. Random Forest.

Datasets used for modelling:

1. Dataset without any preprocessing(X)
2. Dataset after applying Standard scaler and PCA(X_ss_pca)
3. Dataset after applying MinMax scaler(X_mm)
4. Dataset after applying Standard sclaer(X_ss)
5. Dataset after applying MinMax scaler and correlation analysis(X_mm_corr)
6. Dataset after applying Standard scaler and correaltion analysis(X_ss_corr)

7.1 XGBoost

XGBoost, which stands for Extreme Gradient Boosting, is a scalable, distributed gradient-boosted decision tree (GBDT) machine learning library. It provides parallel tree boosting and is the leading machine learning library for regression, classification. Extreme Gradient Boosting (xgboost) is similar to the gradient boosting framework but more efficient. It has both linear model solver and tree learning algorithms. So, what makes it fast is its capacity to do parallel computation on a single machine. When using boosting techniques all instances in the dataset are assigned a score that tells how difficult to classify they are. The XGboost model was applied to the different preprocessed dataset. Figure shows that the test accuracy is above 90% for the cases.

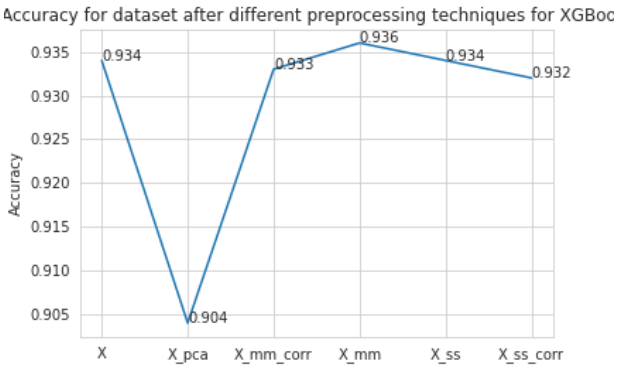

Figure 7.1.1 shows the accuracy plots for different preprocessed datasets.

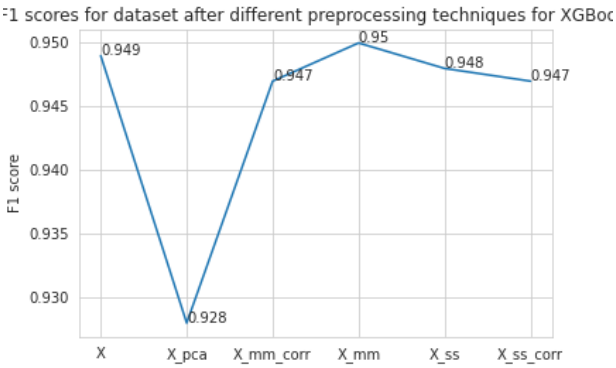

Figure 7.1.2 shows the F1 scores of the models build from different dataset.

7.2 Gradient Boost

Gradient boosting is a machine learning technique used in regression and classification tasks, among others. It gives a prediction model in the form of an ensemble of weak prediction models, which are typically decision trees. When a decision tree is the weak learner, the resulting algorithm is called gradient-boosted trees. A gradient-boosted trees model is built in a stage-wise fashion as in other boosting methods, but it generalizes the other methods by allowing optimization of an arbitrary differentiable loss function. All the trees are connected in series and each tree tries to minimise the error of the previous tree. Due to this sequential connection, the gradient boost algorithm is usually slow to learn, but also highly accurate.

Figure shows a good F1 score for the gradient descent algorithm. Also the model classifies in the test data above 90%. Although it takes time for the fitting due to its sequential connection.

7.3 Decision Tree

A tree can be "learned" by splitting the source set into subsets based on an attribute value test. This process is repeated on each derived subset in a recursive manner called recursive partitioning. The recursion is completed when the subset at a node all

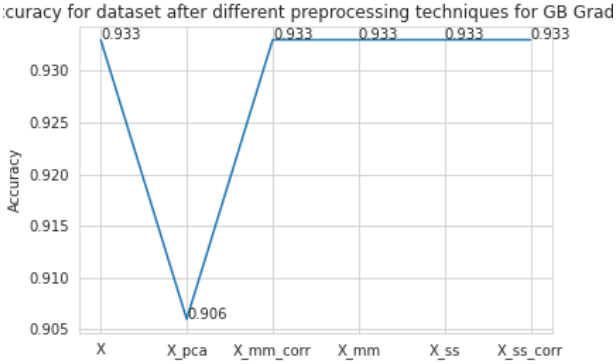

Figure 7.2.3

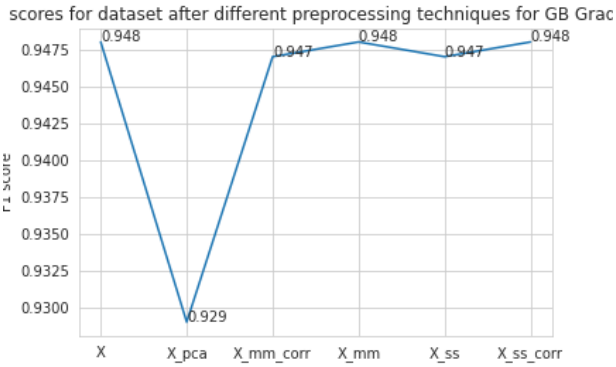

Figure 7.2.4

has the same value of the target variable, or when splitting no longer adds value to the predictions. The construction of decision tree classifier does not require any domain knowledge or parameter setting, and therefore is appropriate for exploratory knowledge discovery. The Decision Tree model was applied to the different preprocessed datasets and following results were observed.

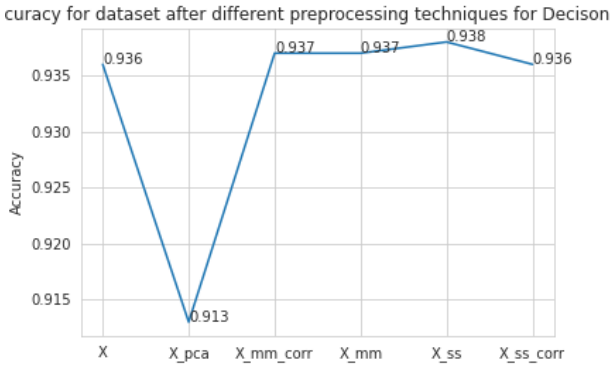

Figure 7.3.1 shows the accuracy of the decison trees on the differnt preprocessed dataset It can be observed that with standard scaling preprocessing technique, accuracy of DT is maximum and with PCA, it is lowest.

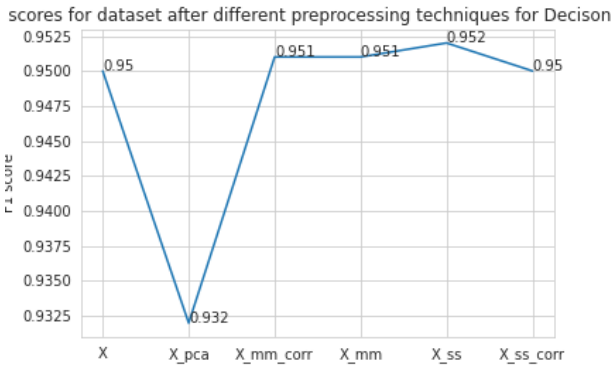

Figure 7.3.2 shows the F1-score of the decison trees on the different preprocessed dataset

7.4 Random Forest

Random Forest is a classification algorithm is combination of many decision trees. It is a better classifier than decision tree since it leverages the advantages of DT and overcomes its shortcomings. Therefore, the feature of Random forest model include simplicity and good accuracy.

One of the best ways to analysis the performance of a Machine Learning model is studying it accuracy and F1 score. The accuracy and F1 score of Random Model as a classifier is computed and plotted for different preprocessing techniques. It is observed that both accuracy (Figure 6.4.1) and F1 score (Figure 6.4-.) given by Random Forest is better that most of the other model that the testing is performed. This can be infered from this that Random Forest predicts more accurate results here.

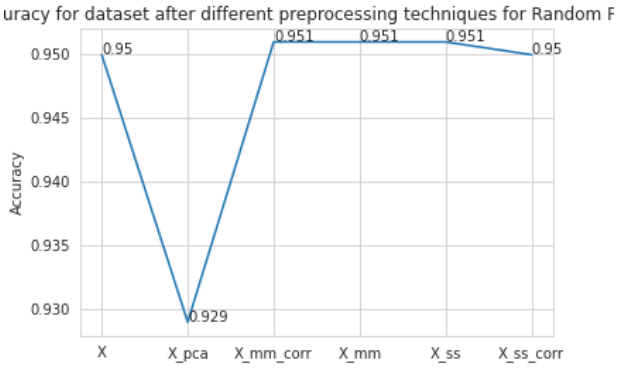

Figure 7.4.1 shows the F1-score of the decison trees on the different preprocessed dataset

It can be observed accuracy of RF is maximum for three preprocessing techniques i.e MinMax scaling with correlation, MinMax Scaling and Standard Scaling. It is lowest with PCA.

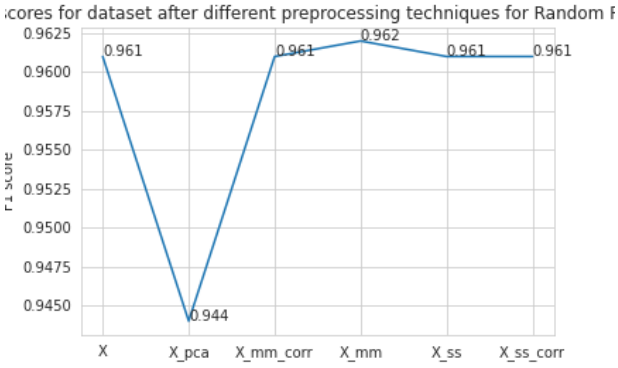

Figure 7.4.2 shows the F1-score of the decison trees on the different preprocessed dataset

It can be observed that with MinMax preprocessing technique, F1 of RF is maximum and with PCA, it is lowest.

8. Comparisons

8.1 Without Processing the dataset

The data modeling was done for the data without any processing which gave the following results shown in Figure 8.1.

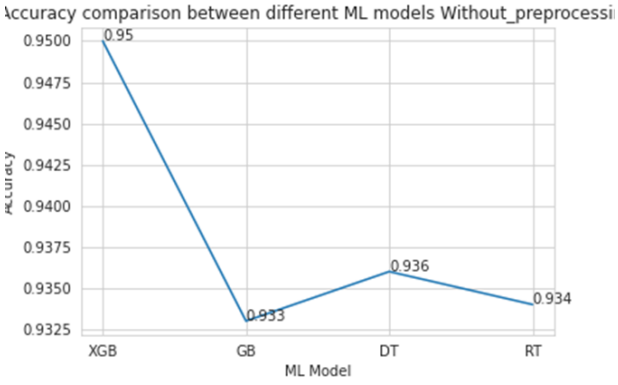

Figure 8.1

For XG Boost accuracy was 95%, for Gradient boost it dropped to 93.3% further for Decision tree it was around 93.6% and lastly for Random forest the accuracy was 93.4%.

8.2 After applying Min-Max Scaler algorithm

The data modeling was done for the dataset which gave the following results shown in Figure 8.2.

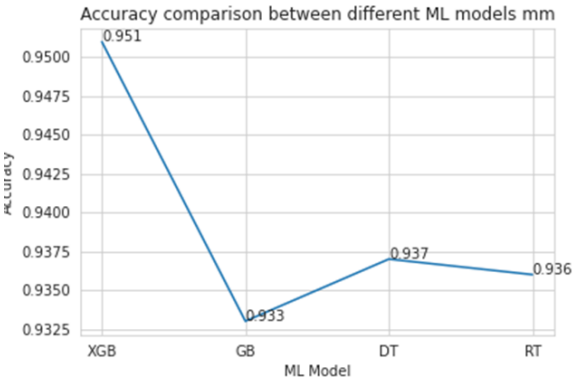

Figure 8.2

For XG Boost accuracy was 95.1%, for Gradient boost it dropped to 93.3% further for Decision tree it was around 93.7% and lastly for Random forest the accuracy was 93.6%.

8.3 After applying Min-Max Scaler with Correlation

The data modeling was done for the dataset which gave the following results shown in Figure 8.3.

For XG Boost accuracy was 95.1%, for Gradient boost it dropped to 93.3% further for Decision tree it was around 93.7% and lastly for Random forest the accuracy was 93.3%.

8.4 After applying Standard Scaler with PCA

The data modeling was done for the dataset which gave the following results shown in Figure 8.4. The accuracy dropped for this processing.

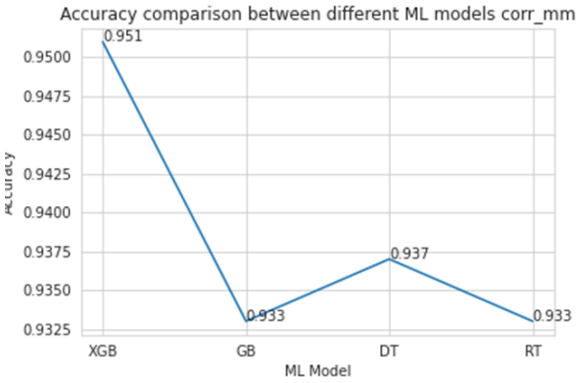

Figure 8.3

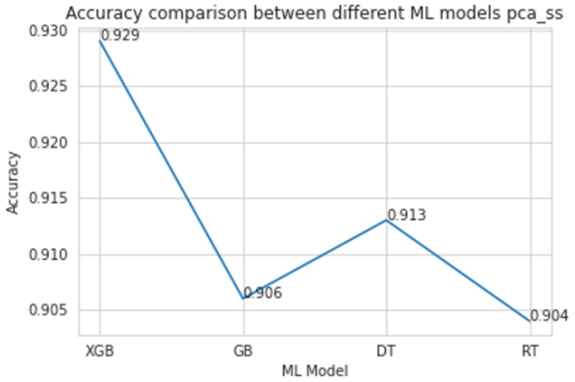

Figure 8.4

For XG Boost accuracy was 92.9%, for Gradient boost it dropped to 90.6% further for Decision tree it was around 91.3% and lastly for Random forest the accuracy was 90.4%.

8.5 After applying Standard Scaler

The data modeling was done for the dataset which gave the following results shown in Figure 8.5.

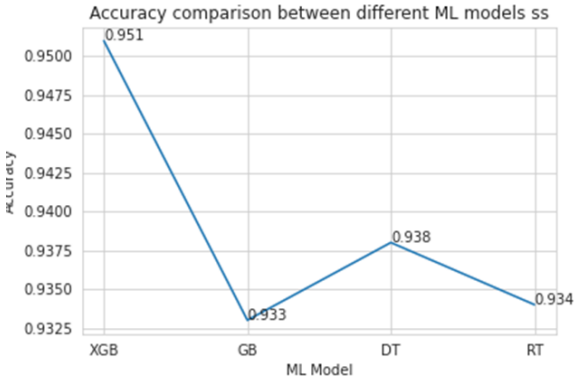

Figure 8.5

For XG Boost accuracy was 95.1%, for Gradient boost it dropped to 93.3% further for Decision tree it was around 93.8% and lastly for Random forest the accuracy was 93.4%.

8.6 After applying Standard Scaler with Correlation

The data modeling was done for the dataset which gave the following results shown in Figure 8.6.

For XG Boost accuracy was 95%, for Gradient boost it dropped to 93.3% further for Decision tree it was around 93.6% and lastly for Random forest the accuracy was 93.2%.
The standard scaler processing proved to be the most accurate for all the models with accuracy of 95.1%, 93.3%, 93.8% and 93.4% for XG Boost, Gradient Boost, Decision tree and Random Forest respectively. The main reason is Standard Scaler removes the mean

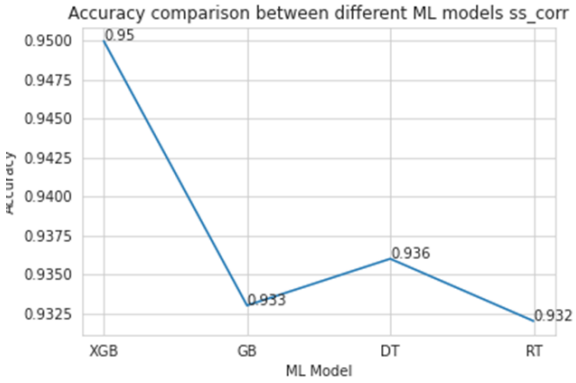

Figure 8.6

and scales the data to unit variance. It also shrinks the range of feature.

Example Analysis

Conclusions

References

[1] N. Moustafa and J. Slay, "UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," in *2015 military communications and information systems conference (MilCIS)*, 2015, pp. 1–6, doi: 10.1109/MilCIS.2015.7348942.

[2] A. Divekar, M. Parekh, V. Savla, R. Mishra, and M. Shirole, "Benchmarking datasets for anomaly-based network intrusion detection: KDD CUP 99 alternatives," *2018 IEEE 3rd International Conference on Computing, Communication and Security (ICCCS)*, pp. 1–8, 2018.

[3] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, pp. 1–6, 2009.

[4] D. Jing and H.-B. Chen, "SVM based network intrusion detection for the UNSW-NB15 dataset," *2019 IEEE 13th International Conference on ASIC (ASICON)*, pp. 1–4, 2019.

[5] M. Al-Zewairi, S. Almajali, and A. A. Awajan, "Experimental evaluation of a multi-layer feed-forward artificial neural network classifier for network intrusion detection system," *2017 International Conference on New Trends in Computing Sciences (ICTCS)*, pp. 167–172, 2017.