ANDROID BOTNET DETECTION USING SUPERVISED MACHINE LEARNING METHODS

SHALINI .S

(20PCA016)

Project Report Submitted

In partial fulfillment of the requirements for the Award of

Master of Computer Applications

DEPARTMENT OF COMPUTER SCIENCE

AVINASHILINGAM INSTITUTE FOR HOME SCIENCE AND HIGHER EDUCATION FOR WOMEN

COIMBATORE – 641043

MAY – 2022

CHAPTER - 1

INTRODUCTION

1.1 ABOUT THE PROJECT

A botnet is a collection of devices connected that each control two or more bots. DDoS attacks, data theft, spam, and giving the attacker access to a device and its connections are all possible with botnets. The botnet's operator can administer it with command - and - control (C&C) software. The term "botnet" is a combination of "robot" and "network." Typically, the phrase has a nasty or hostile connotation. A botnet attack is a sort of cyberattack in which malware infects a huge number of internet-connected computers that are then controlled by a malicious hacker. Botnet assaults include spamming, data theft, stealing sensitive information, and launching unpleasant DDoS attacks.

An Android botnet is one of the most dangerous types of malware attack even though it can be controlled remotelyby a botmaster and used to carry out damaging attacks. Android Botnet is similar to SPYWARE in that it allows attackers to gain complete control of a device after it has been installed. Matryosh is a botnet that targets Operating system that have exposed the Android Debug Bridge (ADB) debug interface to the internet. Matryosh, according to Connect different, is an ADB-targeting malware that hides it command and control nodes behind the Tor network. Several researchers have employed a variety of well-known Machine Learning (ML) techniques to distinguish Android botnet from benign application.

In Quartely3, Spamhaus Malware Labs research team discovered an increase of 82% of the total of new botnet commands and controllers (C&Cs). And it witnessed a rise in the deployment of backdoor viruses, with crooks using FastFlux as a cover. As a result, we've added several countries and internet providers to the Top 20 lists.

The Quartely3 2021 Spamhaus Botnet Attack Report, In the third quarter of 2021, Section contains Software Labs found 2,656 botnet C&Cs (command and controllers), up from 1,462 in the previous quarter. It represents an 82% gain over the preceding quarter. The current average improved from 487 malware C&Cs in Quarter2 to 885 malware C&Cs in Quartely3.

1.2 MOTIVATION AND JUSTIFICATION

A botnet can be remotely controlled by attackers though malicious or illegal purposes, which can have a direct and indirect impact on users. Classifying the Android Botnet helps in detecting the attacks that are trying to access the user control.

1.3 PROBLEM STATEMENT

To identify fraudulent Android applications or malware attacks that are attempting to acquire sensitive data or get access to a user's device that is connected through a network.

1.4 OBJECTIVE

To develop a Supervised Machine learning models to detect and classify the Android Botnet attack that are trying to gain the access user control.

CHAPTER - 2

ABSTRACT

In today's world, A cyberattack is any offensive manoeuvre that attacks computer information systems or computer networks in the modern world, mobile applications and network architecture. Cyber threats are increasing day by day, botnet is one of the major cyber attack that affect mobile devices. A botnet is a malware-infected network of computers controlled by a single attacker, known as the "bot-herder". A bot is an autonomous computer under the command of the bot-herder. According to the recent statistics report, in Quartely3 Spamhaus Malware Labs discovered 2,656 botnet C&Cs in 2021, up from 1,462 throughout Quartely2 in 2021. This was an increase of 82 percent from the previous quarter. In Quartely3, the monthly average grew from 487 botnet C&Cs per month to 885 botnet C&Cs per month. The number of new botnet C&Cs discovered by Quartely3 has increased by a staggering 82 percent.

This project, investigates the detection of Android Botnet using Supervised Machine learning algoithms. Machine learning is an artificial intelligence branch (AI) to provide automatic detection of botnet attacks without any human intervention. The process of Android Botnet Detection using Machine Learning methods consists of six phases. The Phase 1 is the data acquisition. In Phase 2, deals with data pre-processsing to remove unrelevant data. In Phase 3, the appropriate wrapper based feature selection methods are used to select the significant features. In Phase 4, deals with model building using supervised machine learning methods includes Support Vector Machine, Naive Bayes, Decision Tree, Multi- layer Perceptron and Random Forest. In Phase 5, the comparative analysis is made between the supervised machine learning models to suggest the suitable model for Android Botnet detection using top 85 features. While training the model, Random forest detect 99.92% of android botnet in top 85 feature. The Evaluation of the models based on the performance metrics such as accuracy, precision, recall, f1 score in a significant way.

Keywords: Android Botnet, Evaluation Metrics, Malware, Regression Error Metrics, Supervised Machine Learning.

CHAPTER - 3

METHODOLOGY

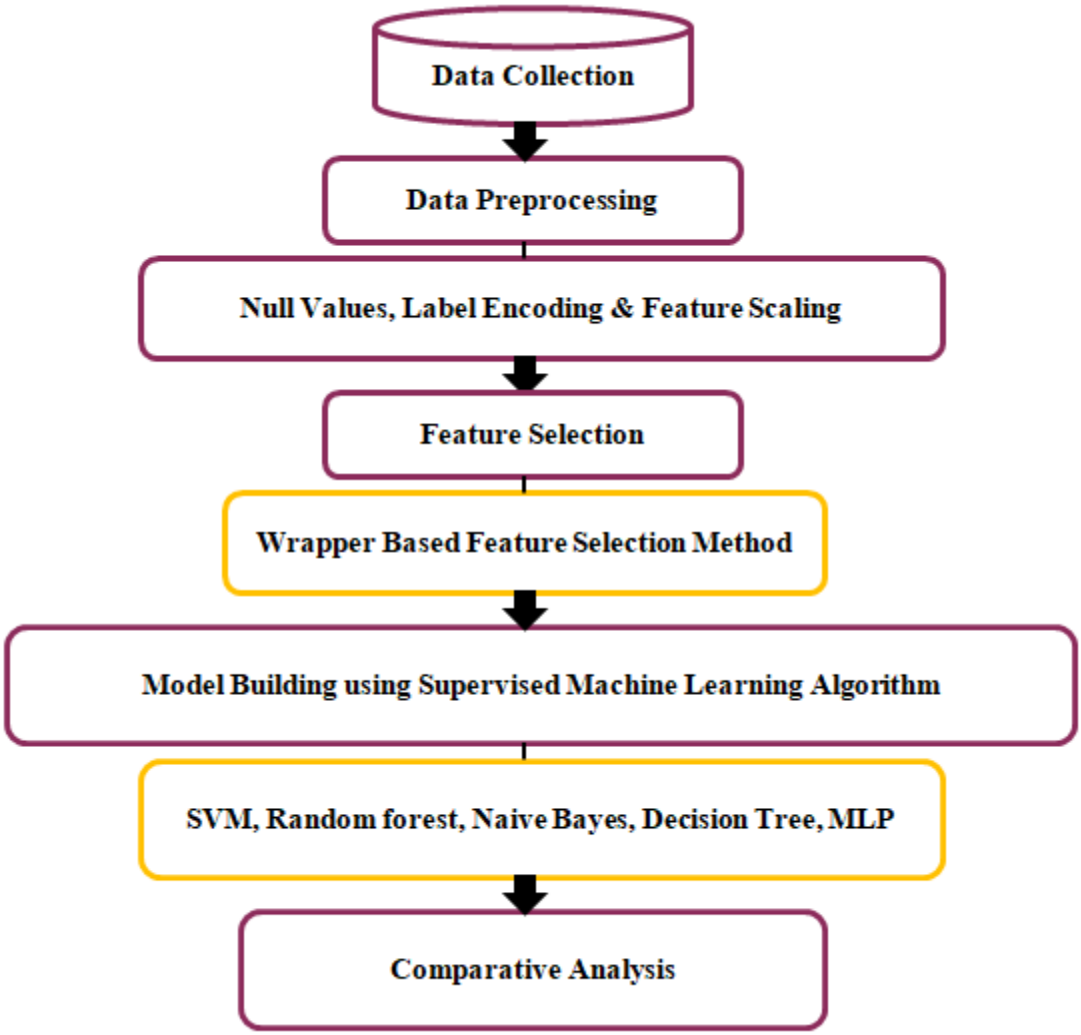


Figure 3.1 Methodology Overview

The following figure represents, Android Botnet entire methodology is divded into five phases namely, Data collection, Data Pre-processing, Wrapper based Feature Selection method, Model Building using Supervised Machine learning algorithms and Comparative analysis based on Performance Evaluation Metrics.

CHAPTER - 4

RESULTS AND DISCUSSION

4.1 PHASE 1: DATA COLLECTION

The following figure represents relationship between the Android Botnet Application and Android Botnet Application from ISCX Dataset.

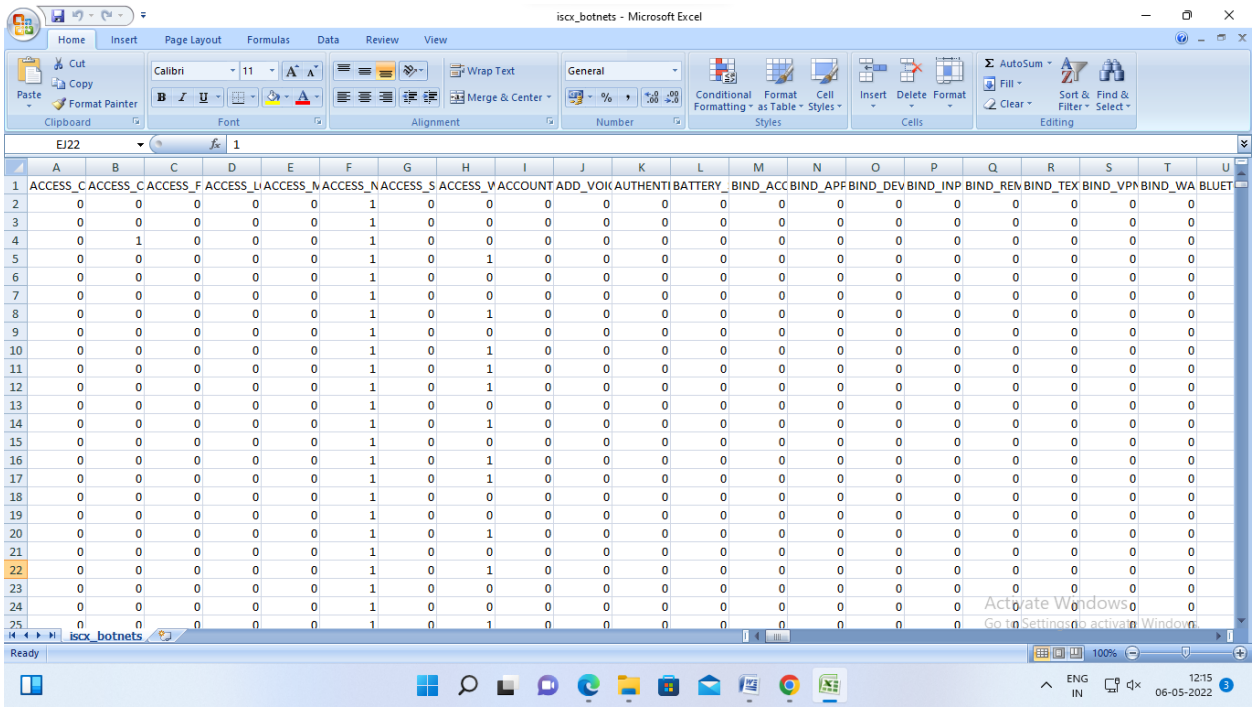


Figure 4.1 Data Collection

Description

The above figure 4.1 represents, ISCX Android Botnet dataset contains a huge number of Android botnets from 14 different botnet families. This dataset was created using 6802 Android applications, which included 4873 benign clean apps and 1929 botnet application from the ISCX botnet dataset. This dataset contains 342 static features extracted from application files.

4.2 PHASE 2: DATA PRE-PROCESSING

4.2.1 EXPLORATORY DATA ANALYSIS

The following figure represents relationship between the Botnet as 1 and Benign as 0 from ISCX Dataset.

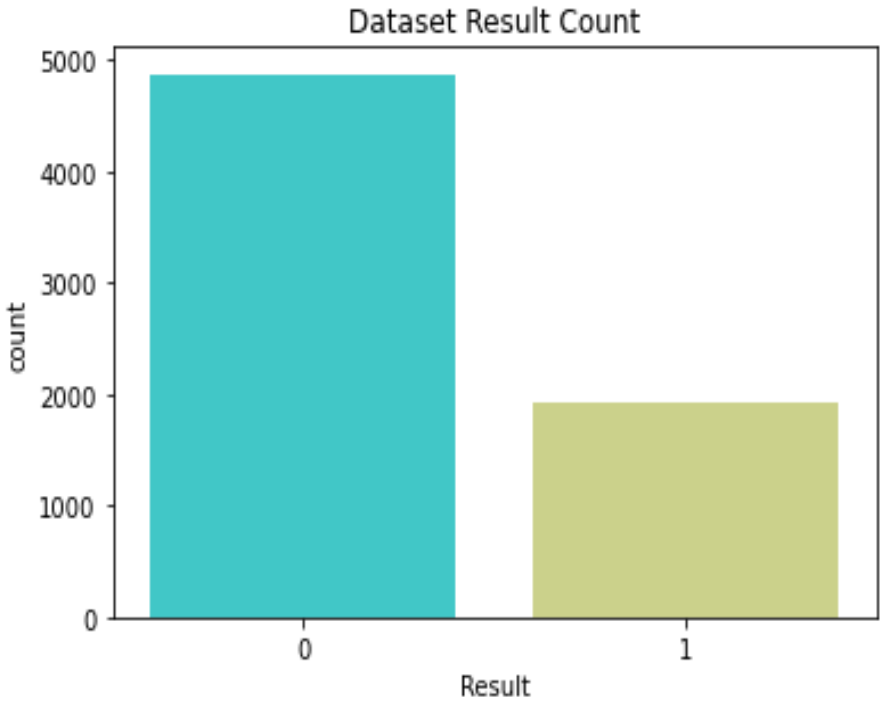


Figure 4.2.1: Exploratory Data Analysis after Label Encoding

Description

The above figure. 4.2.1 represents the Android Benign Application is used from the ISCX Dataset consisting of a large number of related variables such as 4873 Applications and the Android Botnet Application is consisting 1929 Applications. The observe relationships between Botnet and Benign.

4.2.2 Feature scaling using Standardization

```
array([[-0.08777075, -0.38799648, -0.42001137, ..., -0.42122171,
        -0.08691626,  1.58939591],
       [-0.08777075, -0.38799648, -0.42001137, ..., -0.42122171,
        -0.08691626,  1.58939591],
       [-0.08777075,  2.57734294, -0.42001137, ..., -0.42122171,
        -0.08691626,  1.58939591],
       ...,
       [-0.08777075, -0.38799648, -0.42001137, ..., -0.42122171,
        -0.08691626, -0.62916986],
       [-0.08777075, -0.38799648, -0.42001137, ..., -0.42122171,
        -0.08691626, -0.62916986],
       [-0.08777075,  2.57734294, -0.42001137, ..., -0.42122171,
        -0.08691626, -0.62916986]])
```

Figure 4.2.2: Feature scaling using Standardization

Description

The above figure 4.2.2 represents, Feature scaling using Standardization method.

4.2.3 Train and Test Split

```
from sklearn.model_selection import train_test_split
X = pd.DataFrame(dataset.iloc[:,:-1])
y = pd.DataFrame(dataset.iloc[:,-1])

# Splitting the dataset into train and test sets: 80-20 split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 42)
X_train.shape, y_train.shape, X_test.shape, y_test.shape
```

```
((5441, 342), (5441, 1), (1361, 342), (1361, 1))
```

Figure 4.2.3: Training and Testing Splitting

Description

The above figure 4.2.3 represents Training and Testing Splitting, comprising for 80% train data and 20% for test data out off 342 features and 6802 records from ISCX Dataset.

4.3 PHASE 3: WRAPPER BASED FATURE SELECTION METHOD

4.3.1 Forward Feature Selection

	feature_idx	cv_scores	avg_score	feature_names
1	(255,)	[0.5223200314277512]	0.52232	(TelephonyManager.*getDeviceId,)
2	(92, 255)	[0.6756324606714466]	0.675632	(SEND_SMS, TelephonyManager.*getDeviceId)
3	(92, 255, 280)	[0.7405149056465065]	0.740515	(SEND_SMS, TelephonyManager.*getDeviceId, Ljav...
4	(92, 229, 255, 280)	[0.7525297180932282]	0.75253	(SEND_SMS, Binder, TelephonyManager.*getDevice...
5	(92, 228, 229, 255, 280)	[0.7639235231961226]	0.763924	(SEND_SMS, IBinder, Binder, TelephonyManager.*...
...	...	...	...	...
81	(0, 6, 10, 14, 24, 29, 30, 39, 49, 57, 58, 62,...	[0.8854849084681463]	0.885485	(ACCESS_CHECKIN_PROPERTIES, ACCESS_SURFACE_FLI...
82	(0, 6, 10, 14, 24, 29, 30, 39, 49, 57, 58, 62,...	[0.8857053731278064]	0.885705	(ACCESS_CHECKIN_PROPERTIES, ACCESS_SURFACE_FLI...
83	(0, 6, 10, 14, 24, 29, 30, 39, 49, 57, 58, 62,...	[0.8859366134825668]	0.885937	(ACCESS_CHECKIN_PROPERTIES, ACCESS_SURFACE_FLI...
84	(0, 6, 10, 14, 24, 29, 30, 39, 49, 57, 58, 62,...	[0.8861259435872642]	0.886126	(ACCESS_CHECKIN_PROPERTIES, ACCESS_SURFACE_FLI...
85	(0, 6, 10, 14, 24, 29, 30, 34, 39, 49, 57, 58,...	[0.886344257219875]	0.886344	(ACCESS_CHECKIN_PROPERTIES, ACCESS_SURFACE_FLI...

85 rows × 4 columns

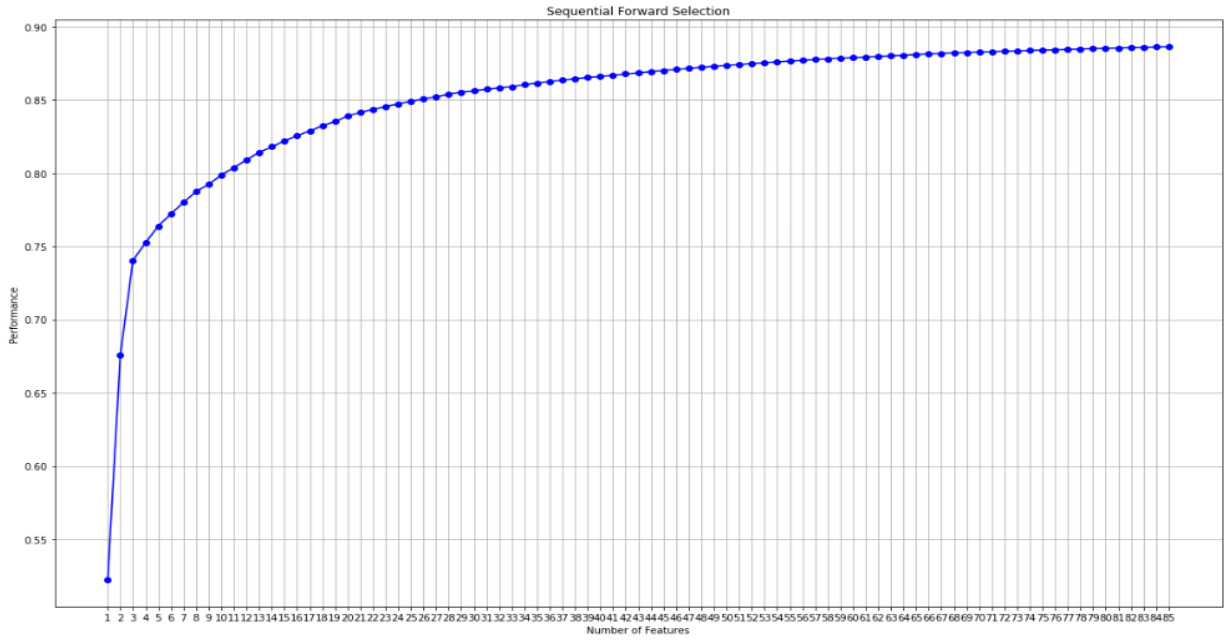


Figure 4.3.1: Top 85 features using Forward Feature Selection

Description

The above figure 4.3.1 represents, top 85 features selected using forward feature selection out off 342 features from ISCX Dataset.

4.3.2 Backward Feature Selection

	feature_idx	cv_scores	avg_score	feature_names
342	(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13,...	[0.9004281241641401]	0.900428	(ACCESS_CHECKIN_PROPERTIES, ACCESS_COARSE_LOCA...
341	(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13,...	[0.9004281327548341]	0.900428	(ACCESS_CHECKIN_PROPERTIES, ACCESS_COARSE_LOCA...
340	(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13,...	[0.9004281785472775]	0.900428	(ACCESS_CHECKIN_PROPERTIES, ACCESS_COARSE_LOCA...
339	(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13,...	[0.9004282098797457]	0.900428	(ACCESS_CHECKIN_PROPERTIES, ACCESS_COARSE_LOCA...
338	(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13,...	[0.9004281371212516]	0.900428	(ACCESS_CHECKIN_PROPERTIES, ACCESS_COARSE_LOCA...
...	...	...	...	...
89	(0, 6, 10, 14, 24, 28, 29, 30, 32, 34, 57, 58,...	[0.8877294389355183]	0.887729	(ACCESS_CHECKIN_PROPERTIES, ACCESS_SURFACE_FLI...
88	(0, 6, 10, 14, 24, 28, 29, 30, 32, 34, 57, 58,...	[0.8874978849360228]	0.887498	(ACCESS_CHECKIN_PROPERTIES, ACCESS_SURFACE_FLI...
87	(0, 6, 10, 14, 24, 28, 29, 30, 32, 34, 57, 58,...	[0.8873328941909271]	0.887333	(ACCESS_CHECKIN_PROPERTIES, ACCESS_SURFACE_FLI...
86	(0, 6, 10, 14, 24, 28, 29, 30, 32, 34, 57, 58,...	[0.8870558916113889]	0.887056	(ACCESS_CHECKIN_PROPERTIES, ACCESS_SURFACE_FLI...
85	(0, 6, 10, 14, 24, 28, 29, 30, 32, 34, 57, 58,...	[0.8867991692041735]	0.886799	(ACCESS_CHECKIN_PROPERTIES, ACCESS_SURFACE_FLI...

258 rows × 4 columns

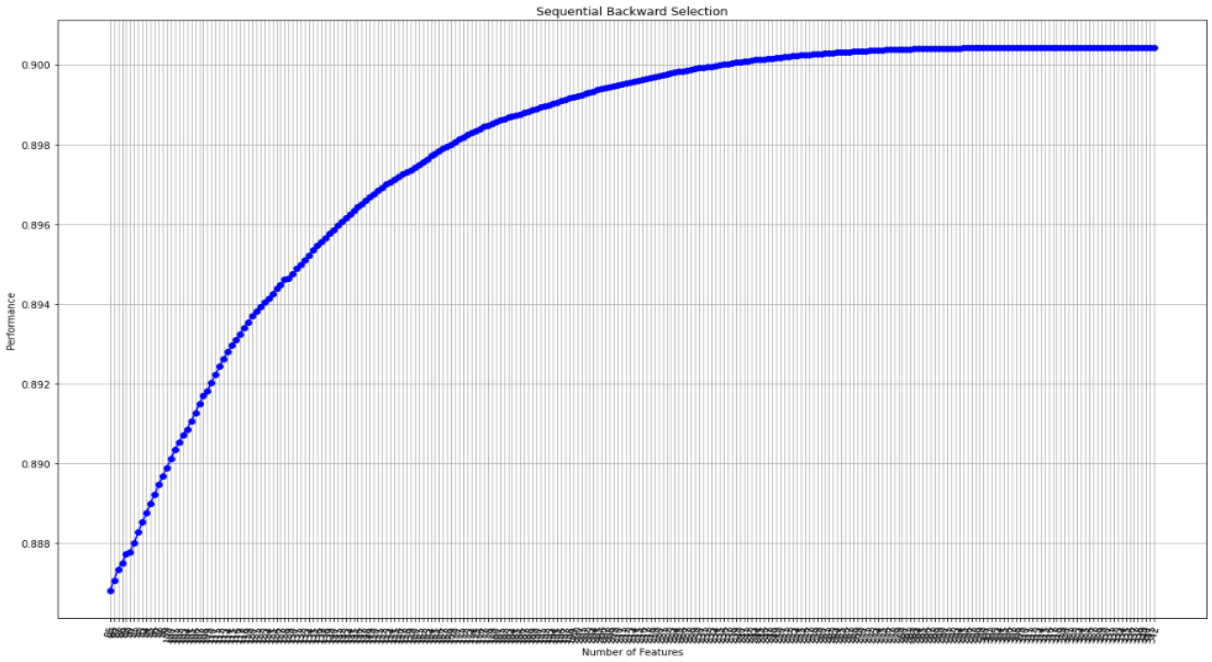


Figure 4.3.2: Top 85 features using Backward Feature Selection

Description

The above figure 4.3.2 represents, top 85 features selected using Backward feature selection out off 342 features from ISCX Dataset.

4.3.3 Stepwise Feature Selection

	feature_idx	cv_scores	avg_score	feature_names
1	(255,)	[0.5223200314277512]	0.52232	(TelephonyManager.*getDeviceId,)
2	(92, 255)	[0.6756324606714466]	0.675632	(SEND_SMS, TelephonyManager.*getDeviceId)
3	(92, 255, 280)	[0.7405149056465065]	0.740515	(SEND_SMS, TelephonyManager.*getDeviceId, Ljav...
4	(92, 229, 255, 280)	[0.7525297180932282]	0.75253	(SEND_SMS, Binder, TelephonyManager.*getDevice...
5	(92, 228, 229, 255, 280)	[0.7639235231961226]	0.763924	(SEND_SMS, IBinder, Binder, TelephonyManager.*...
...	...	...	...	...
81	(0, 6, 10, 14, 24, 29, 30, 39, 49, 57, 58, 62,...	[0.8854849084681463]	0.885485	(ACCESS_CHECKIN_PROPERTIES, ACCESS_SURFACE_FLI...
82	(0, 6, 10, 14, 24, 29, 30, 39, 49, 57, 58, 62,...	[0.8857053731278064]	0.885705	(ACCESS_CHECKIN_PROPERTIES, ACCESS_SURFACE_FLI...
83	(0, 6, 10, 14, 24, 29, 30, 39, 49, 57, 58, 62,...	[0.8859366134825668]	0.885937	(ACCESS_CHECKIN_PROPERTIES, ACCESS_SURFACE_FLI...
84	(0, 6, 10, 14, 24, 29, 30, 39, 49, 57, 58, 62,...	[0.8861259435872642]	0.886126	(ACCESS_CHECKIN_PROPERTIES, ACCESS_SURFACE_FLI...
85	(0, 6, 10, 14, 24, 29, 30, 34, 39, 49, 57, 58,...	[0.886344257219875]	0.886344	(ACCESS_CHECKIN_PROPERTIES, ACCESS_SURFACE_FLI...

85 rows × 4 columns

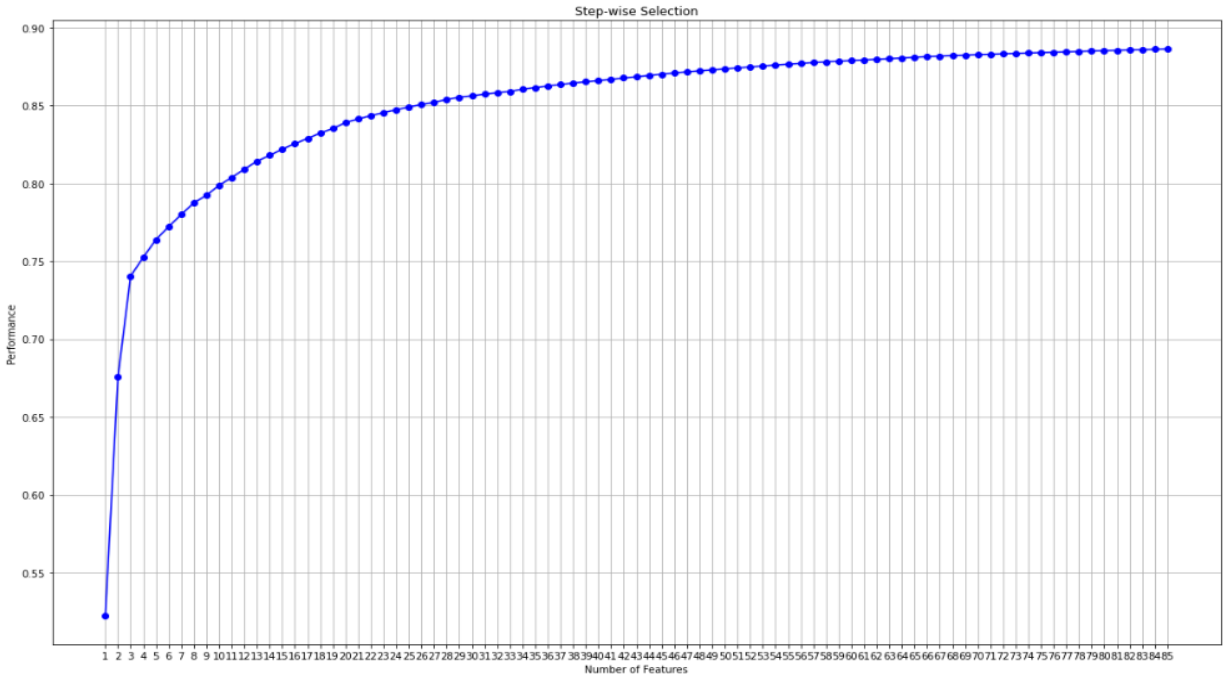


Figure 4.3.3: Top 85 features using Stepwise Feature Selection

Description

The above figure 4.3.3 represents, top 85 features selected using Stepwise feature selection out off 342 features from ISCX Dataset.

4.3.4 Train and Test Split after feature selection

The following figure represent training and testing splitting after complete Wrapper based Feature Selection Method.

```
from sklearn.model_selection import train_test_split
X = pd.DataFrame(df.iloc[:,:-1])
y = pd.DataFrame(df.iloc[:,-1])

# Splitting the dataset into train and test sets: 80-20 split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 42)
X_train.shape, y_train.shape, X_test.shape, y_test.shape
```

```
((5441, 84), (5441, 1), (1361, 84), (1361, 1))
```

Figure 4.3.4: Training and Testing Splitting

Description

The above figure 4.3.4 represents Training and Testing Splitting. Top 85 features selected from Wrapper based Feature Selection Method and create another dataframe to split comprising for 80% train data and 20% for test data out off top 85 features and 6802 records from forward feature selection.

4.4 PHASE 4: MODEL BUILDING

4.4.1: SUPPORT VECTOR MACHINE

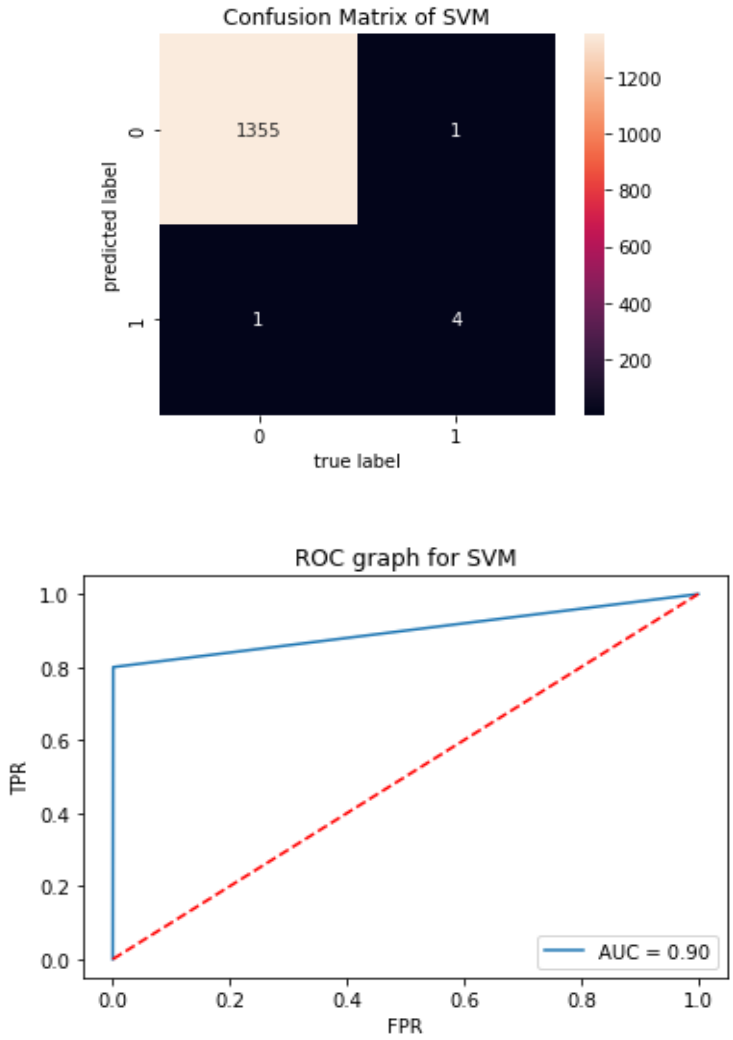


Figure 4.4.1: Confusion Matrix & ROC Curve using Support Vector Machine

Description

The above figure 4.4.1 represents Model Building SVM. This Matrix and ROC curve represents the performance model of the SVM classifier relationship between True positive and False positive to predicate model accuracy level.

4.4.2: NAVIE BAYES CLASSIFIER

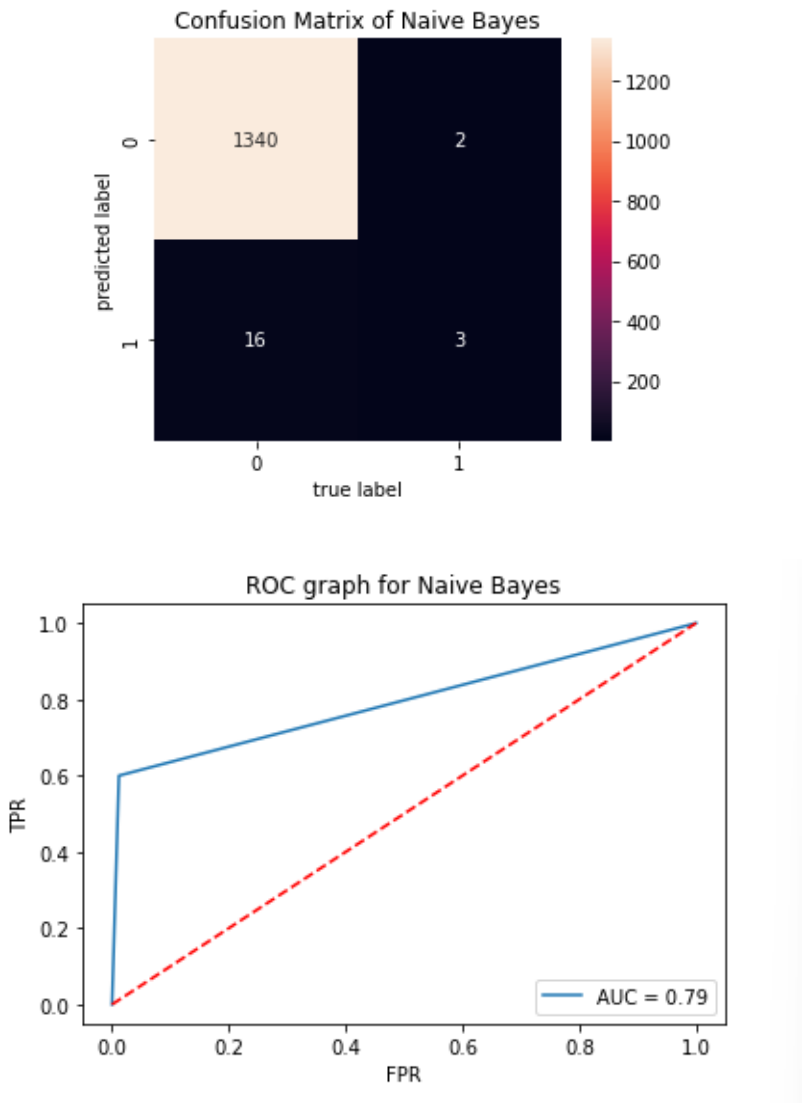


Figure 4.4.2: Confusion Matrix and ROC curve using Naive Bayes classifier.

Description

The above figure 4.4.2 represents Model Building Naive Bayes. This Matrix and ROC curve represents the performance model of the NB classifier relationship between True positive and False positive to predicate model accuracy level.

4.4.3: RANDOM FOREST CLASSIFIER

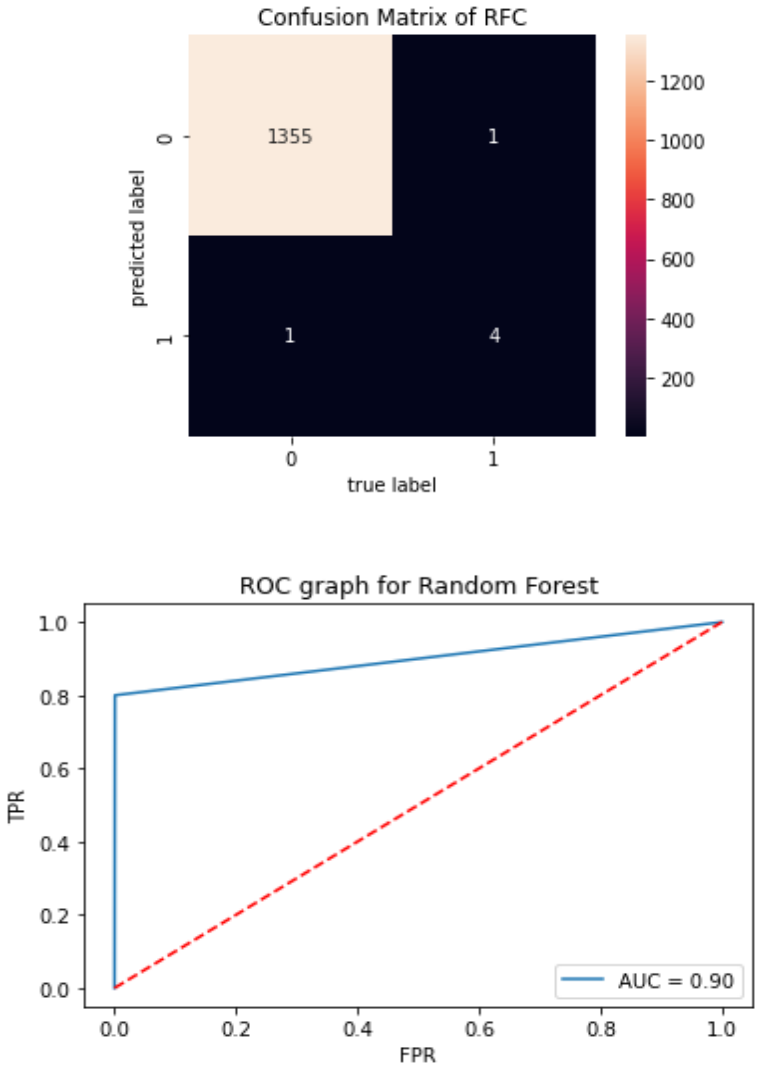


Figure 4.4.3: Confusion Matrix and ROC curve using Random Forest Classifier.

Description

The above figure 4.4.3 represents Model Building Random Forest. This Matrix and ROC curve represents the performance model of the Random forest classifier relationship between True positive and False positive to predicate model accuracy level.

4.4.4: DECISION TREE CLASSIFIER

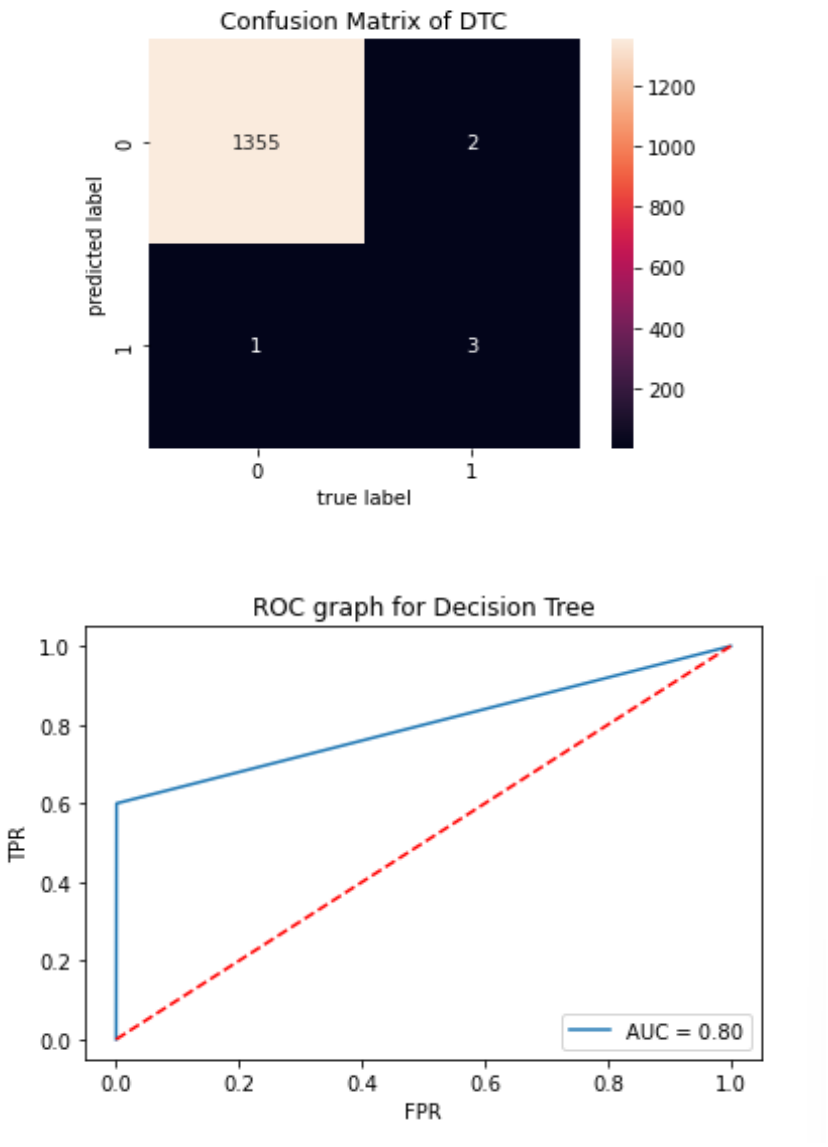
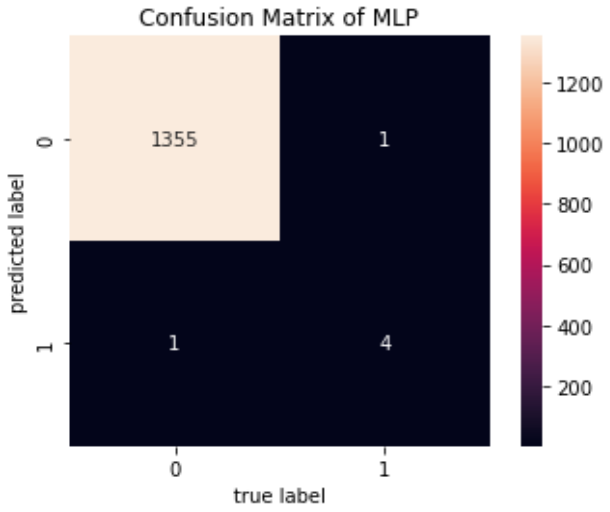


Figure 4.4.4: Confusion Matrix and ROC curve using Decision Tree Classifier.

Description

The above figure 4.4.4 represents Model Building Decision Tree. This Matrix and ROC curve represents the performance model of the Decision Tree classifier relationship between True positive and False positive to predicate model accuracy level.

4.4.5: MULTILAYER PERCEPTRON



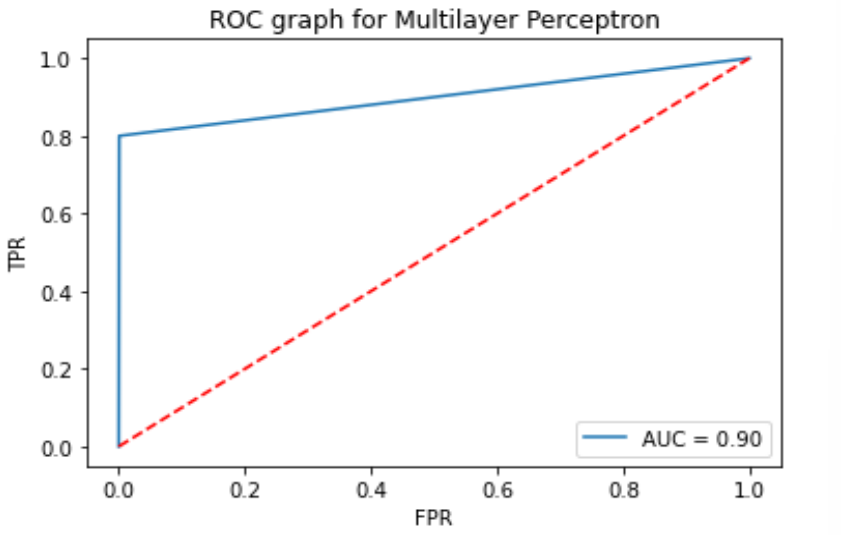


Figure 4.4.5: Confusion Matrix and ROC curve using Multilayer Perceptron (MLP).

Description

The above figure 4.4.5 represents Model Building MLP. This Matrix and ROC curve represents the performance model of the MLP classifier relationship between True positive and False positive to predicate model accuracy level.

3.5 PHASE 5: COMPARATIVE ANALYSIS

4.5.1 Model Train and Test Accuracy Score Comparison:

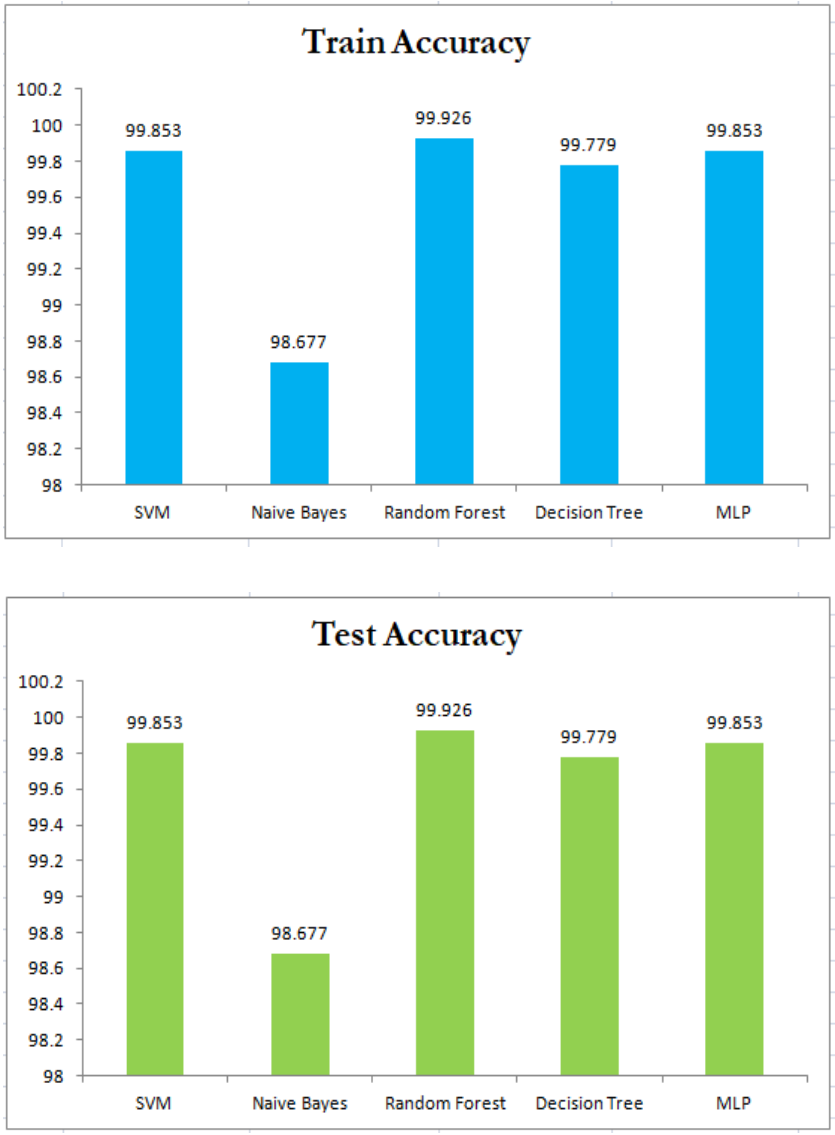


Figure 4.5.1: Model Train and Test Accuracy Comparison.

Description

The above figure 4.5.1 represents all the five Supervised Machine Learning classifier Model Train and Test accuracy Comparisons.

4.5.2 Model Precision and Recall Score Comparison

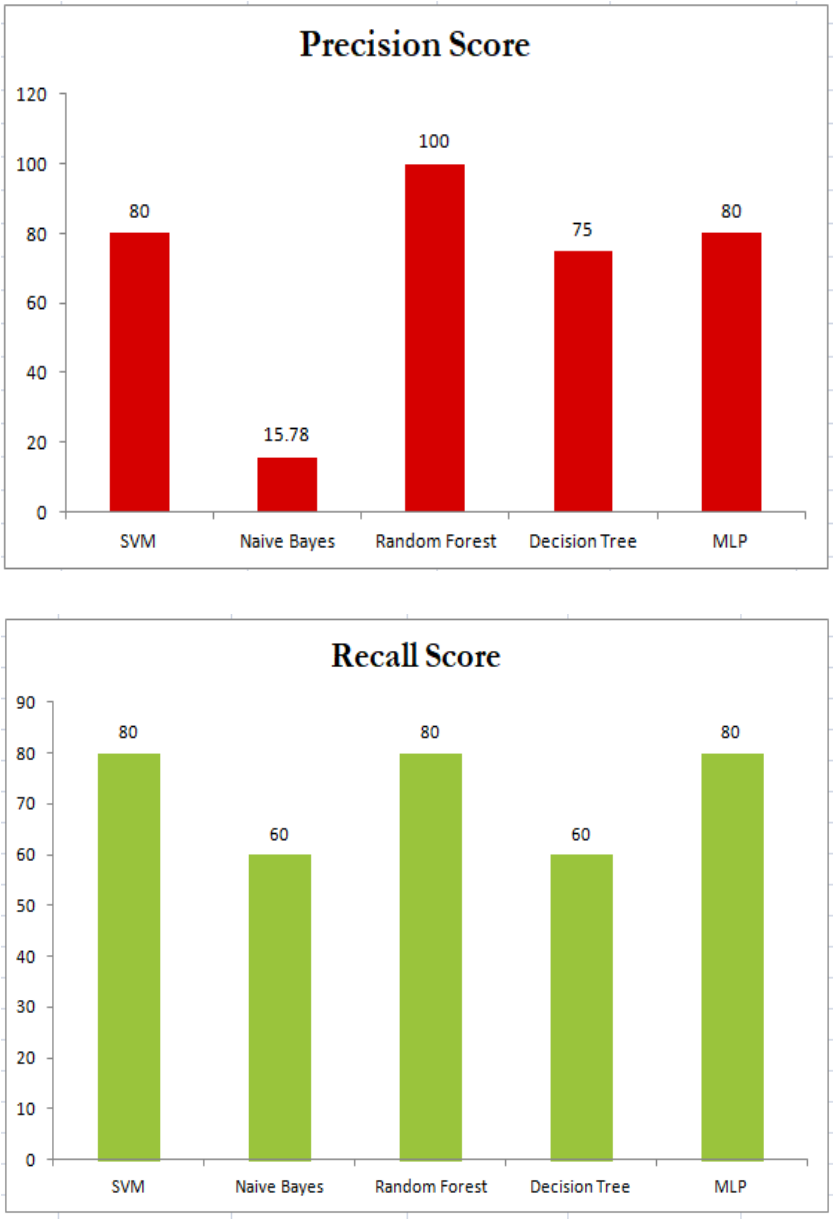
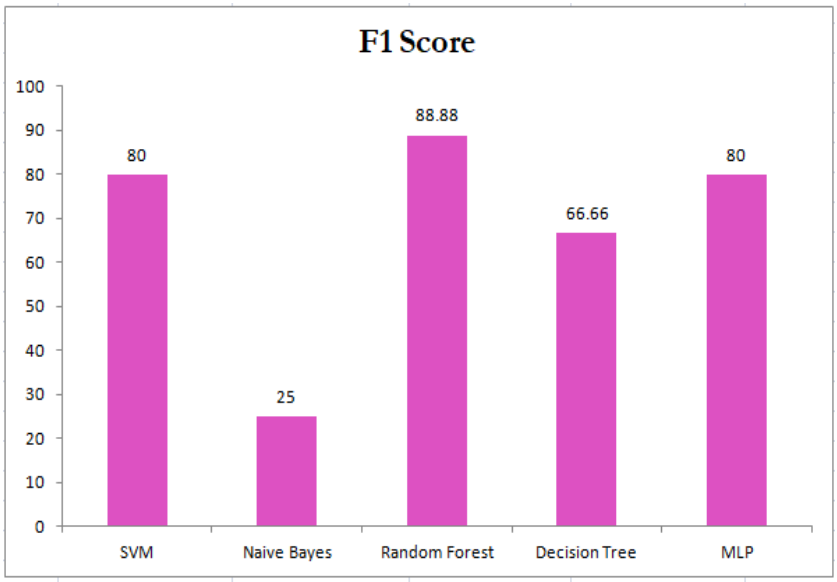


Figure 4.5.2: Model Precision and Recall Score Comparative Analysis.

Description

The above figure 4.5.2 represents all the five Supervised Machine Learning classifier Model Precision and Recall accuracy Comparisons.

4.5.3 Model F1 and AUC Score Comparison



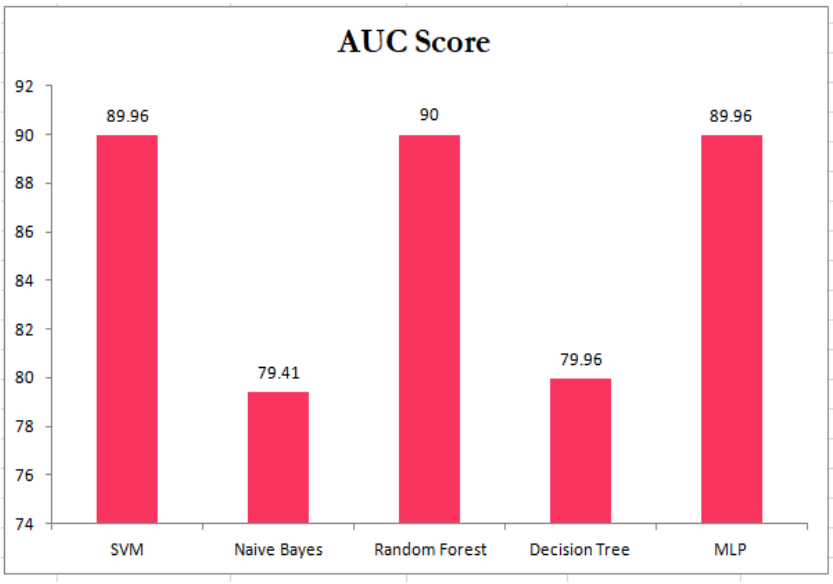


Figure 4.5.3: Model F1 and AUC Score Comparative Analysis.

Description

The above figure 4.5.3 represents all the five Supervised Machine Learning classifier Model F1 Score and AUC Score Comparisons.

4.5.4 Model ROC Curve and Mean Absolute Error Comparison

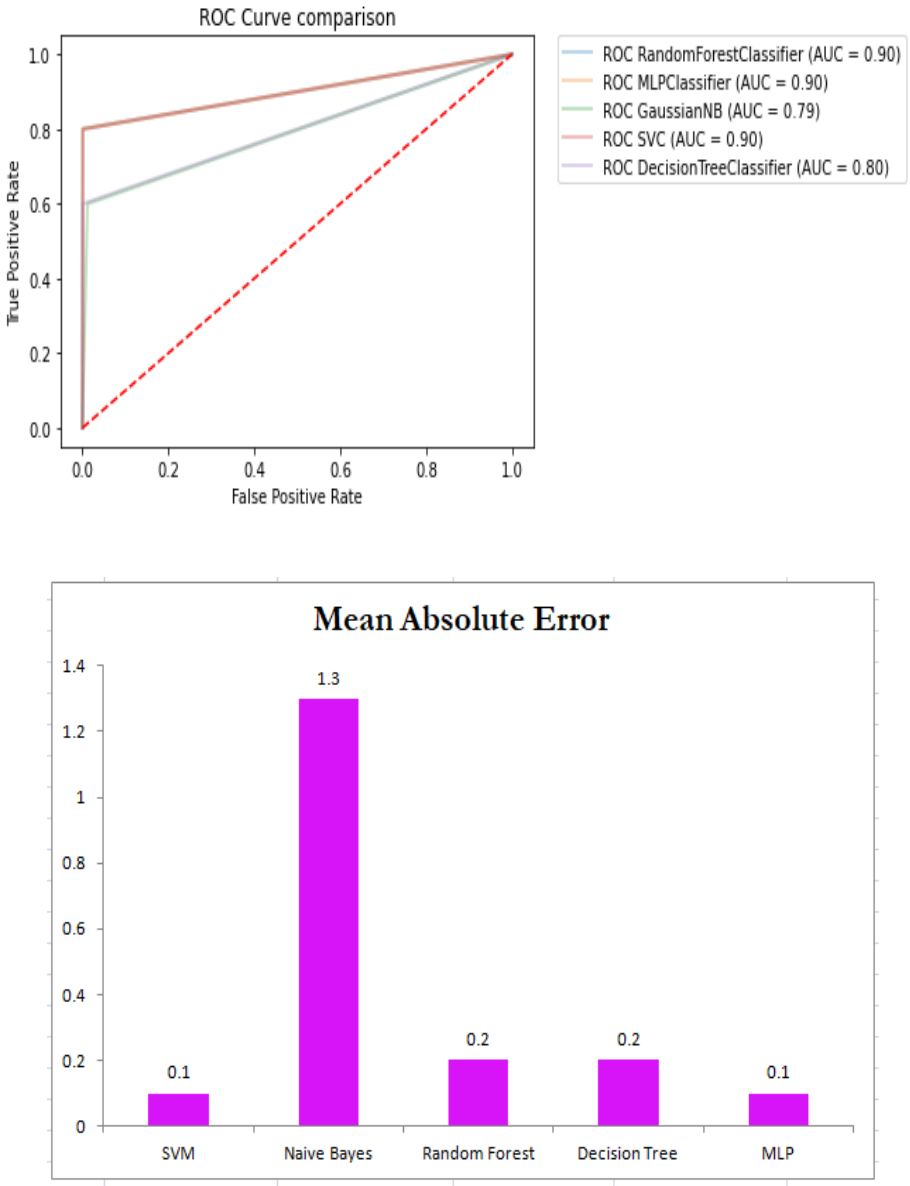


Figure 4.5.4: ROC Curve and Model Mean Absolute Error Comparison.

Description

The above figure 4.5.4 represents all the five Supervised Machine Learning classifier ROC Curve and Mean Absolute Error Comparisons.

4.5.5 Model Mean Squared Error and Root Mean Squared Error Comparison

Figure 4.5.5: Model Root Mean Square Error Comparison

Description

The above figure 4.5.5 represents all the five Supervised Machine Learning classifier Model Root Mean Squared Error Comparisons.

CHAPTER - 5

CONCLUSION

With the expanding number of Internet users and its commercial aspect, a corresponding number of crimes enter the market, posing a threat to authorized customers, Network infrastructure, and the timeliness of services offered by it. The point of this study worked and detected the android botnet attack, based on its classes. Here the Random Forest, SVM and MLP worked better than Decision tree and Naïve Bayes classifiers using top 85 features. While training the model, Random forest detect 99.92% of android botnet in top 85 feature.

The other algorithm fails to detect most of the android botnet. Thus it shows that random forest model is most sufficient than other algorithm in detecting the malware attack at top X feature. Thus, the Android Botnets are detected and classified, which helps in detecting the attacks that are trying to access the user control.

CHAPTER - 6

REFERENCES

1. Ahmad Karim, Syed Adeel Ali Shah, Rosli Bin Salleh, Muhammad Arif, Rafidah Md Noor, Shahaboddin Shamshirband, (2015), *Mobile Botnet Attacks – an Emerging Threat: Classification,* Review and Open Issues. KSII Transactions on Internet and Information Systems, 9(4). DOI:10.3837/tiis.2015.04.012

2. Alharbi, A., & Alsubhi, K. (2021*). Botnet Detection Approach Using Graph-Based Machine Learning.* IEEE Access, 9, 99166–99180. DOI:10.1109/access.2021.3094183

3. FarhanTariq, Shamim Baig (2020), *Comparative Analysis of Network flow-based Botnet Detection Methods Using Supervised Machine Learning Algorithms*. International Journal of Advanced Trends in Computer Science and Engineering, 9(5), 8498–8503. DOI:10.30534/ijatcse/2020/229952020

4. Hashim, H. A. B., Saudi, M. M., & Basir, N. (2017a). *Android Botnet Features for Detection Mechanism*. Advanced Science Letters, *23*(6), 5314–5317. DOI.10.1166/asl.2017.7366

5. Hijawi, W., Alqatawna, J., Al-Zoubi, A. M., Hassonah,M. A., & Faris, H. (2021). *Android botnet detection using machine learning models based on a comprehensive static analysis approach.* Journal of Information Security and Applications, 58, 102735. DOI:10.1016/j.jisa.2020.102735

6. Kumar, K. (2021). *Comprehensive Method of Botnet Detection Using Machine Learning*. International Journal of Open Source Software and Processes, 12(4), 37–61. DOI:10.4018/ijossp.287613

7. Rasheed, M. M., Faieq, A. K., & Hashim, A. A. (2020). *Android Botnet Detection Using Machine Learning.* Ingénierie Des Systèmes d Information, 25(1), 127–130. DOI:10.18280/isi.250117

8. Shahid Anwar, Mohamad Fadli Zolkipli, Vitaliy Mezhuyev, Zakira Inayat, (2020), *A Smart Framework for Mobile Botnet Detection Using Static Analysis*. KSII Transactions on Internet and Information Systems, 14(6). DOI:10.3837/tiis.2020.06.015

9. Shatnawi, Ahmed S., et al. "*An Android Malware Detection Approach Based on Static Feature Analysis Using Machine Learning Algorithms*." Procedia Computer Science, vol. 201, 2022, pp. 653–58. *Crossref*, DOI.10.1016/j.procs.2022.03.086.

10. Shatnawi, A. S., Yassen, Q., & Yateem, A. (2022). *An Android Malware Detection Approach Based on Static Feature Analysis Using Machine Learning Algorithms.* Procedia Computer Science, 201, 653–658. DOI:10.1016/j.procs.2022.03.086

11. Yerima, S. Y., Alzaylaee, M. K., Shajan, A., & P, V. (2021). *Deep Learning Techniques for Android Botnet Detection.* Electronics, 10(4), 519. DOI:10.3390/electronics10040519