Configuration Manual

MSc Research Project
MSc Cybersecurity

Sumit Kumar
Student ID: X20258526

School of Computing
National College of Ireland

Supervisor: Mr. Jawad Salahuddin

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Sumit Kumar		
Student ID:	X20258526		
Programme:	MSc in Cybersecurity	**Year:**	2022-2023
Module:	MSc Research Project		
Lecturer:	Mr. Jawad Salahuddin		
Submission Due Date:	01-02-2023		
Project Title:	Network Intrusion Detection of Android Smartphones using Machine Learning and Ensemble Learning Techniques		
Word Count:	909	**Page Count:**	7

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Sumit Kumar

Date: 01-02-2023

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Sumit Kumar
Student ID: X20258526

1 Introduction

This configuration handbook contains the fundamental setup and equipment required to do this project work. This thesis intends to create a model that uses machine learning and ensemble learning algorithms to identify an intrusion specifically a remote access trojan (RAT) occurring on Android devices. The configuration handbook is crucial and will include all the necessary software, hardware and implementation techniques to develop this project.

2 Requirement of Hardware

Operating System: Windows 11
RAM: 16.0 GB
Processor: 11th Gen Intel Core i5-11320H @ 3.20GHz 2.50 GHz
Storage: 512 GB SSD
System Type: 64-bit operating system, x64-based processor

3 Requirement of Software

Anaconda Navigator

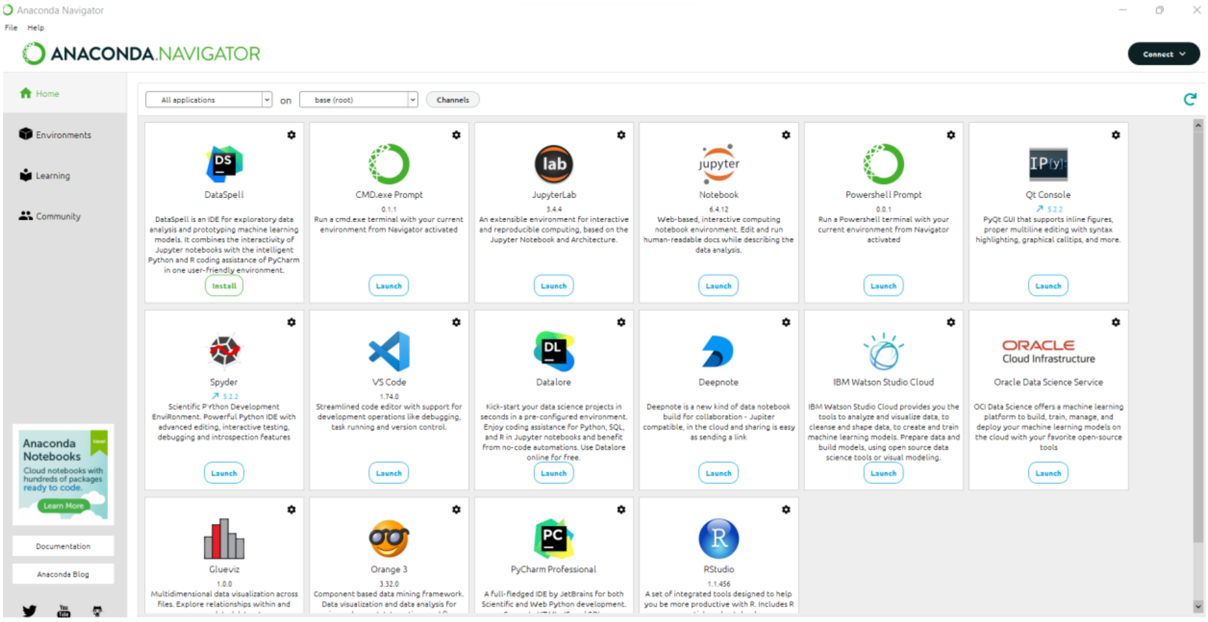
Jupyter Notebook

Python 3.11.1

Python was the programming language used in this project, while Jupyter Notebook was the IDE chosen for the research. Several Python libraries were utilised for analysis and visualisation that will be explained below.

I used a comprehensive package called Anaconda Navigator that comes with Jupyter Notebook, and the necessary Python setup to run the code. I have installed Anaconda 64 Bit version on Windows 11 machine. After a successful installation, Jupyter Notebook needs to be launched from the navigator. Anaconda can be downloaded and installed from below link.(*Anaconda | Anaconda Distribution*, no date)

https://www.anaconda.com/products/distribution



DataSpell

DataSpell is an IDE for exploratory data analysis and prototyping machine learning models. It combines the interactivity of Jupyter notebooks with the intelligent Python and R coding assistance of PyCharm in one user-friendly environment.

CMD.exe Prompt

0.1.1

Run a cmd.exe terminal with your current environment from Navigator activated

JupyterLab

3.4.4

An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.

Notebook

6.4.12

Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.

Powershell Prompt

0.0.1

Run a Powershell terminal with your current environment from Navigator activated

Qt Console

↗ 5.2.2

PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.

Spyder

↗ 5.2.2

Scientific PYthon Development EnviRonment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features

VS Code

1.74.0

Streamlined code editor with support for development operations like debugging, task running and version control.

Datalore

Kick-start your data science projects in seconds in a pre-configured environment. Enjoy coding assistance for Python, SQL, and R in Jupyter notebooks and benefit from no-code automations. Use Datalore online for free.

Deepnote

Deepnote is a new kind of data notebook build for collaboration - Jupiter compatible, in the cloud and sharing is easy as sending a link

IBM Watson Studio Cloud

IBM Watson Studio Cloud provides you the tools to analyze and visualize data, to cleanse and shape data, to create and train machine learning models. Prepare data and build models, using open source data science tools or visual modeling.

ORACLE Cloud Infrastructure

Oracle Data Science Service

OCI Data Science offers a machine learning platform to build, train, manage, and deploy your machine learning models on the cloud with your favorite open-source tools

Glueviz

1.0.0

Multidimensional data visualization across files. Explore relationships within and

Orange 3

3.32.0

Component based data mining framework. Data visualization and data analysis for

PyCharm Professional

A full-fledged IDE by JetBrains for both Scientific and Web Python development.

RStudio

1.1.456

A set of integrated tools designed to help you be more productive with R. Includes R

4 Python Libraries Installed

The following is a list of the Python libraries needed for the research that were installed in the environment using the import command:

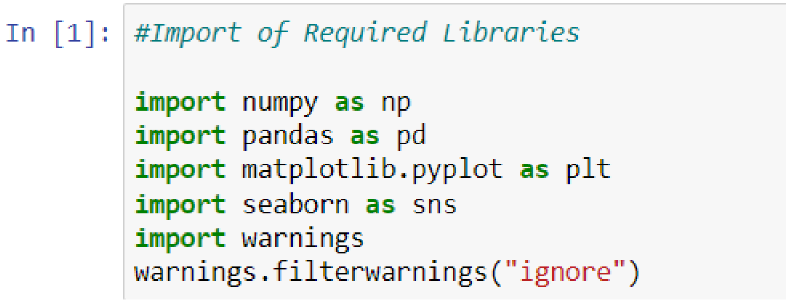
NumPy: Array manipulation is done using the Python library NumPy, which stands for numerical Python.

Pandas: Data analysis is possible with the Python package Pandas. For working with time series and numerical data, it provides a wide variety of data structures and techniques.

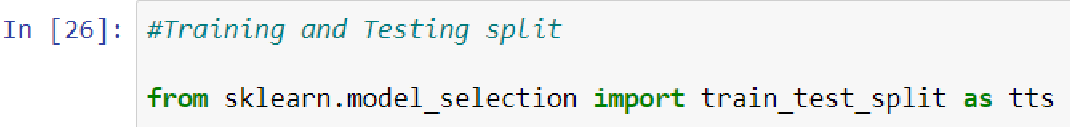
Matplotlib: A comprehensive tool for creating static, animated, and interactive visualisations is offered by the Matplotlib toolbox for Python. Matplotlib makes both tough problems and simple ones doable.

Seaborn: Matplotlib serves as the basis for the graph-plotting package Seaborn. With its aid, random distributions can be seen.

Scikit-learn or sklearn: Scikit-learn is probably the most useful Python machine learning library. The sklearn toolkit includes a number of efficient machine learning and statistical modelling methods, including classification, regression, clustering, and dimensionality reduction.(*Libraries in Python - GeeksforGeeks*, no date)



```
In [1]:  #Import of Required Libraries

         import numpy as np
         import pandas as pd
         import matplotlib.pyplot as plt
         import seaborn as sns
         import warnings
         warnings.filterwarnings("ignore")
```



```
In [26]: #Training and Testing split

         from sklearn.model_selection import train_test_split as tts
```

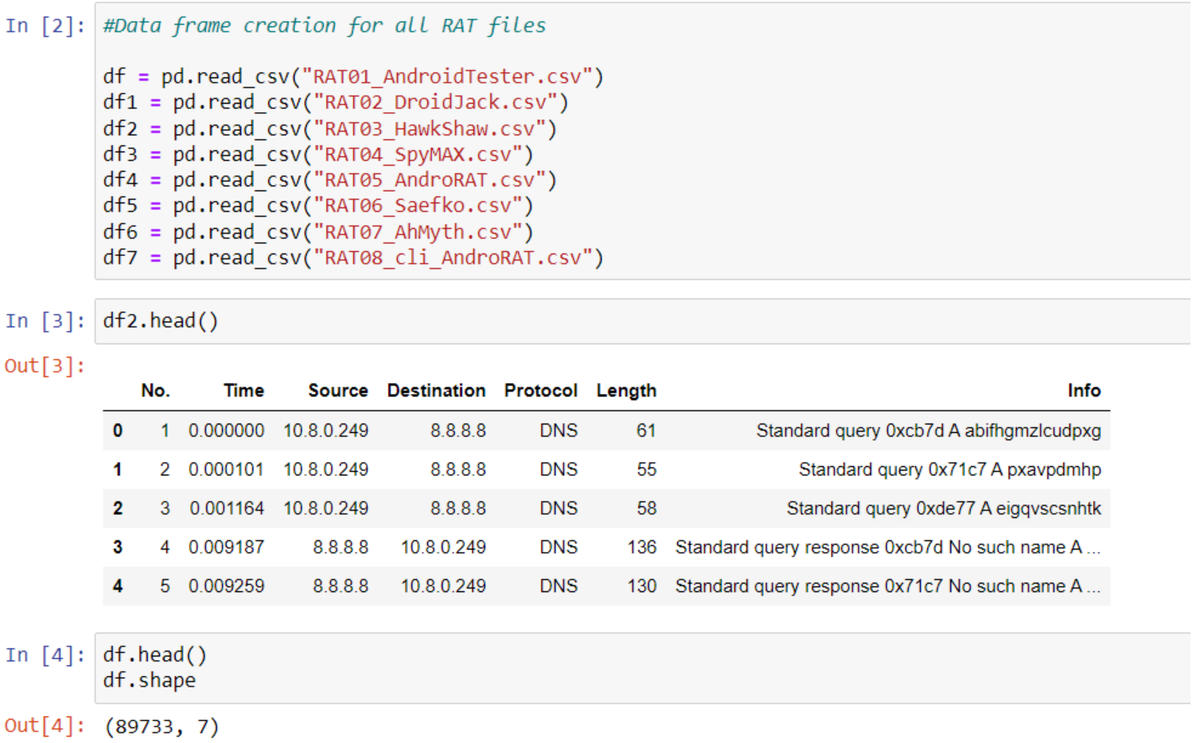
5 Description of Dataset and Data preprocessing

Android Mischief Dataset was used to train the machine learning model in this project and this dataset has 8 types of remote access trojan (RAT). This dataset is created in Stratosphere Laboratory at the Czech Technical University in Prague and the latest version of the dataset can be downloaded and saved on the local system from below mentioned link.(*Android Mischief Dataset*, no date)

https://mcfp.felk.cvut.cz/publicDatasets/Android-Mischief-Dataset/

Note: Please note that the dataset contains the **pcap** format of the file that is packet capture of the network which needs to be converted to **.csv** using the **Wireshark software** as Python does not accept the pcap format of the file for processing.

Data importing and Processing



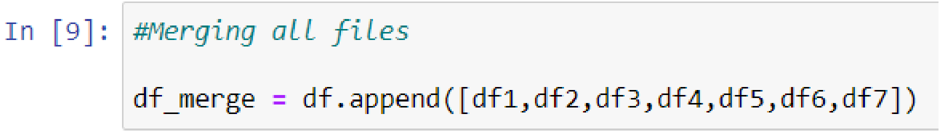
```
In [2]:  #Data frame creation for all RAT files

         df = pd.read_csv("RAT01_AndroidTester.csv")
         df1 = pd.read_csv("RAT02_DroidJack.csv")
         df2 = pd.read_csv("RAT03_HawkShaw.csv")
         df3 = pd.read_csv("RAT04_SpyMAX.csv")
         df4 = pd.read_csv("RAT05_AndroRAT.csv")
         df5 = pd.read_csv("RAT06_Saefko.csv")
         df6 = pd.read_csv("RAT07_AhMyth.csv")
         df7 = pd.read_csv("RAT08_cli_AndroRAT.csv")
```

```
In [3]:  df2.head()
```

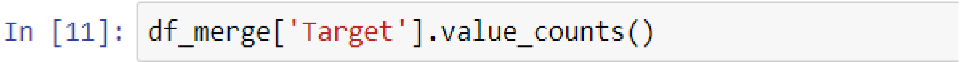
Out[3]:

	No.	Time	Source	Destination	Protocol	Length	Info
0	1	0.000000	10.8.0.249	8.8.8.8	DNS	61	Standard query 0xcb7d A abifhgmzlcudpxg
1	2	0.000101	10.8.0.249	8.8.8.8	DNS	55	Standard query 0x71c7 A pxavpdmhp
2	3	0.001164	10.8.0.249	8.8.8.8	DNS	58	Standard query 0xde77 A eigqvscsnhtk
3	4	0.009187	8.8.8.8	10.8.0.249	DNS	136	Standard query response 0xcb7d No such name A ...
4	5	0.009259	8.8.8.8	10.8.0.249	DNS	130	Standard query response 0x71c7 No such name A ...

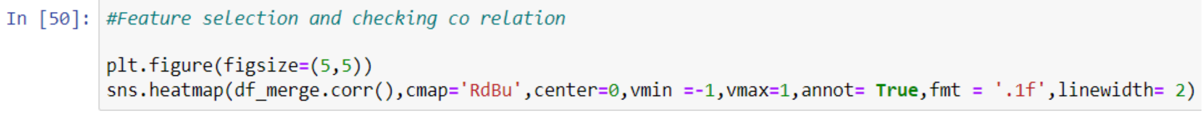
```
In [4]:  df.head()
         df.shape

Out[4]:  (89733, 7)
```



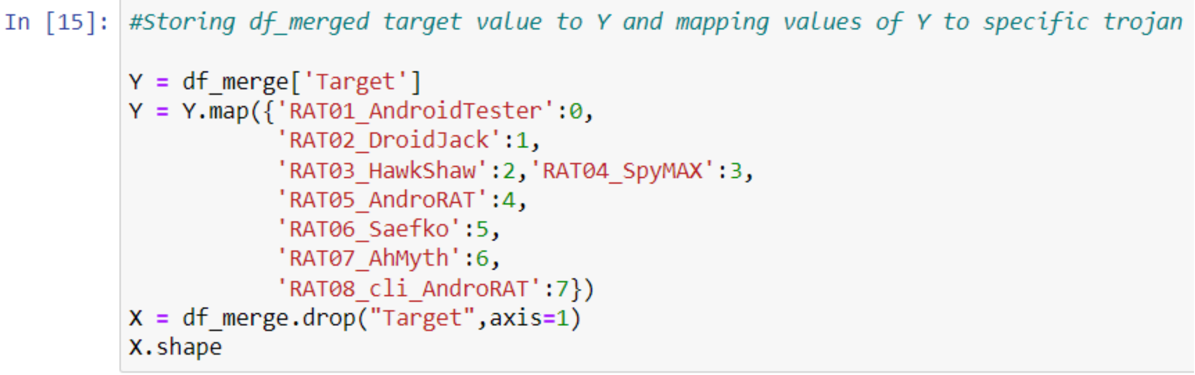
```
In [9]:  #Merging all files

          df_merge = df.append([df1,df2,df3,df4,df5,df6,df7])
```



```
In [11]:  df_merge['Target'].value_counts()
```

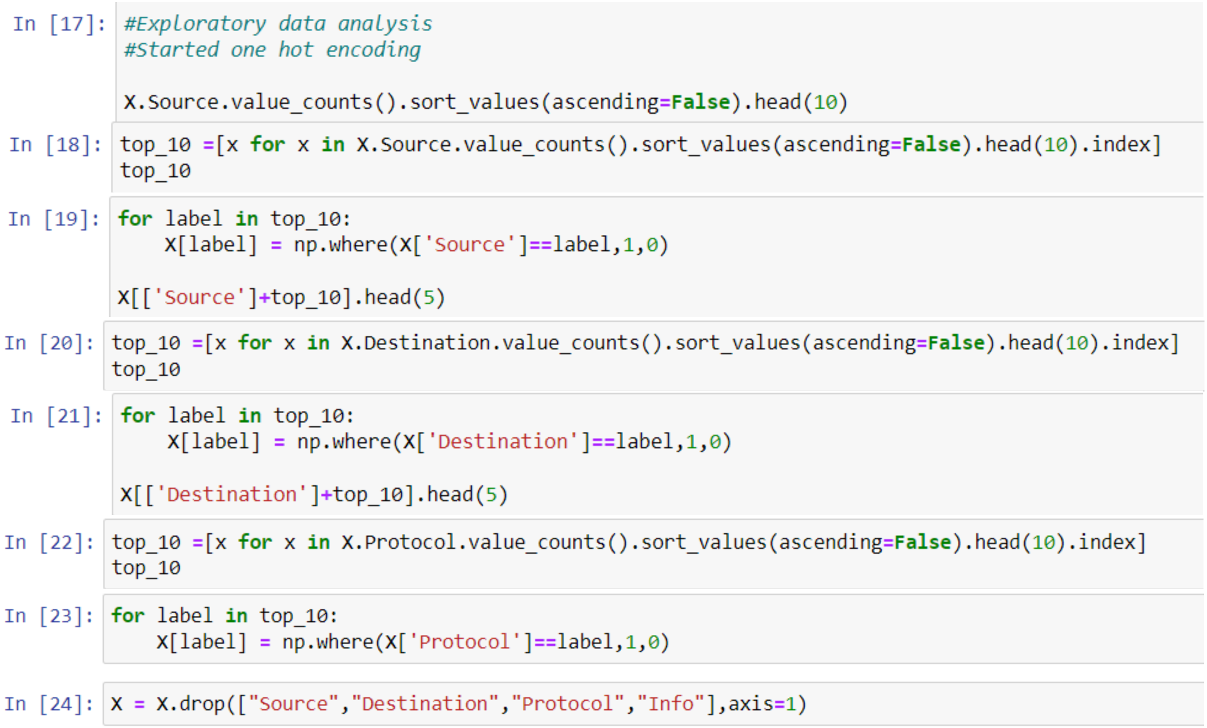


```
In [50]:  #Feature selection and checking co relation

           plt.figure(figsize=(5,5))
           sns.heatmap(df_merge.corr(),cmap='RdBu',center=0,vmin =-1,vmax=1,annot= True,fmt = '.1f',linewidth= 2)
```

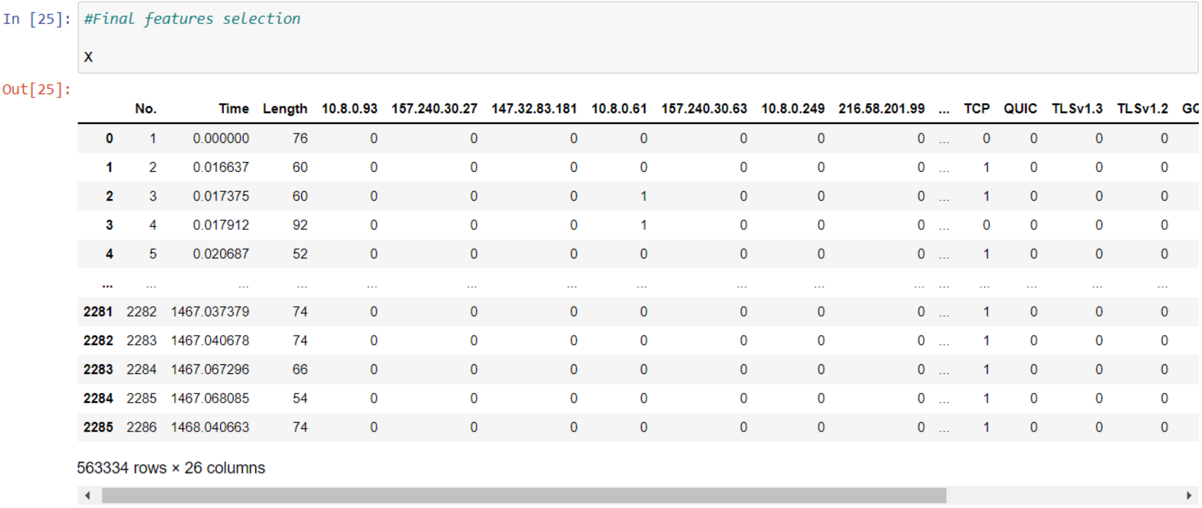


```
In [15]:  #Storing df_merged target value to Y and mapping values of Y to specific trojan

           Y = df_merge['Target']
           Y = Y.map({'RAT01_AndroidTester':0,
                      'RAT02_DroidJack':1,
                      'RAT03_HawkShaw':2,'RAT04_SpyMAX':3,
                      'RAT05_AndroRAT':4,
                      'RAT06_Saefko':5,
                      'RAT07_AhMyth':6,
                      'RAT08_cli_AndroRAT':7})
           X = df_merge.drop("Target",axis=1)
           X.shape
```

Exploratory Data analysis and One hot encoding



```
In [17]:  #Exploratory data analysis
           #Started one hot encoding

           X.Source.value_counts().sort_values(ascending=False).head(10)

In [18]:  top_10 =[x for x in X.Source.value_counts().sort_values(ascending=False).head(10).index]
           top_10

In [19]:  for label in top_10:
               X[label] = np.where(X['Source']==label,1,0)

           X[['Source']+top_10].head(5)

In [20]:  top_10 =[x for x in X.Destination.value_counts().sort_values(ascending=False).head(10).index]
           top_10

In [21]:  for label in top_10:
               X[label] = np.where(X['Destination']==label,1,0)

           X[['Destination']+top_10].head(5)

In [22]:  top_10 =[x for x in X.Protocol.value_counts().sort_values(ascending=False).head(10).index]
           top_10

In [23]:  for label in top_10:
               X[label] = np.where(X['Protocol']==label,1,0)

In [24]:  X = X.drop(["Source","Destination","Protocol","Info"],axis=1)
```



```
In [25]: #Final features selection
         x
```

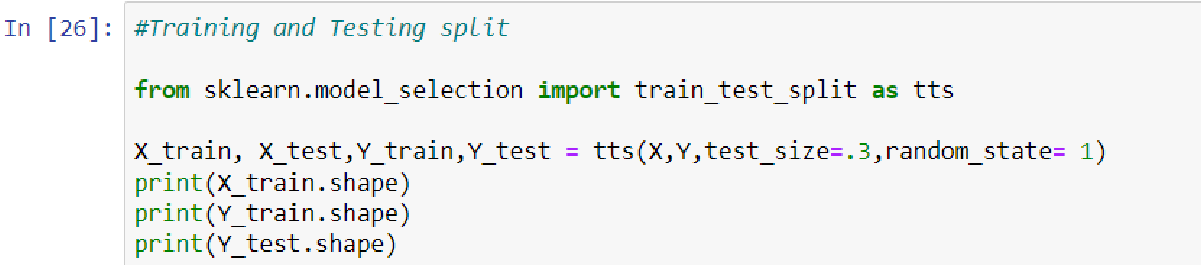
Out[25]:

	No.	Time	Length	10.8.0.93	157.240.30.27	147.32.83.181	10.8.0.61	157.240.30.63	10.8.0.249	216.58.201.99	...	TCP	QUIC	TLSv1.3	TLSv1.2	GC
0	1	0.000000	76	0	0	0	0	0	0	0	...	0	0	0	0	
1	2	0.016637	60	0	0	0	0	0	0	0	...	1	0	0	0	
2	3	0.017375	60	0	0	0	1	0	0	0	...	1	0	0	0	
3	4	0.017912	92	0	0	0	1	0	0	0	...	0	0	0	0	
4	5	0.020687	52	0	0	0	0	0	0	0	...	1	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
2281	2282	1467.037379	74	0	0	0	0	0	0	0	...	1	0	0	0	
2282	2283	1467.040678	74	0	0	0	0	0	0	0	...	1	0	0	0	
2283	2284	1467.067296	66	0	0	0	0	0	0	0	...	1	0	0	0	
2284	2285	1467.068085	54	0	0	0	0	0	0	0	...	1	0	0	0	
2285	2286	1468.040663	74	0	0	0	0	0	0	0	...	1	0	0	0	

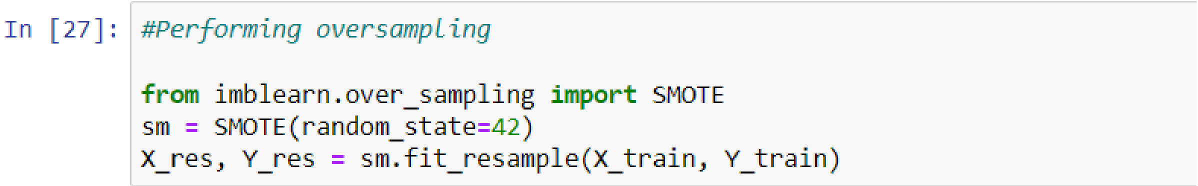
563334 rows × 26 columns

6 Model Training and Testing Summary

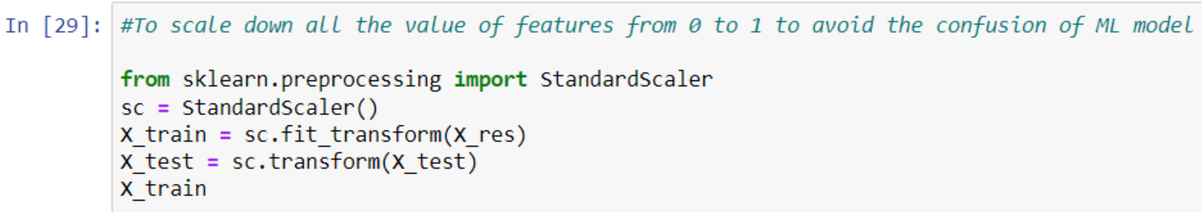
Random Forest Classifier and Decision Tree Classifier



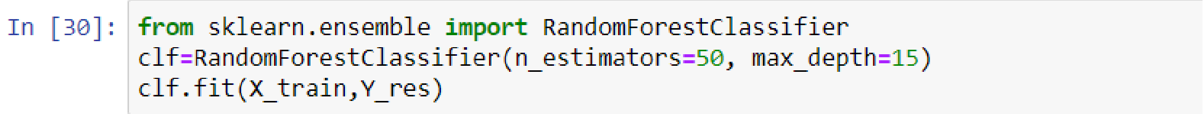
```
In [26]: #Training and Testing split

         from sklearn.model_selection import train_test_split as tts

         X_train, X_test,Y_train,Y_test = tts(X,Y,test_size=.3,random_state= 1)
         print(X_train.shape)
         print(Y_train.shape)
         print(Y_test.shape)
```



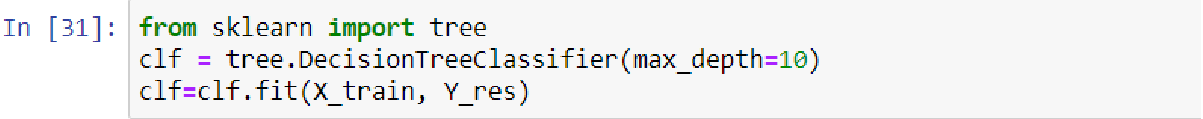
```
In [27]: #Performing oversampling

         from imblearn.over_sampling import SMOTE
         sm = SMOTE(random_state=42)
         X_res, Y_res = sm.fit_resample(X_train, Y_train)
```



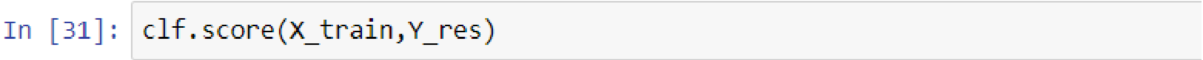
```
In [29]: #To scale down all the value of features from 0 to 1 to avoid the confusion of ML model

         from sklearn.preprocessing import StandardScaler
         sc = StandardScaler()
         X_train = sc.fit_transform(X_res)
         X_test = sc.transform(X_test)
         X_train
```



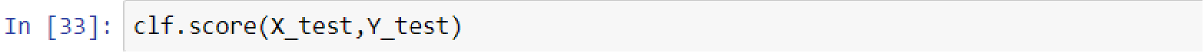
```
In [30]: from sklearn.ensemble import RandomForestClassifier
         clf=RandomForestClassifier(n_estimators=50, max_depth=15)
         clf.fit(X_train,Y_res)
```



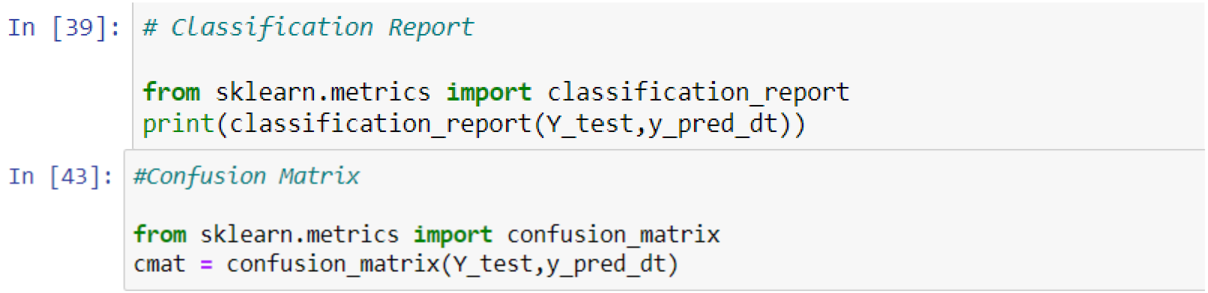
```
In [31]: from sklearn import tree
         clf = tree.DecisionTreeClassifier(max_depth=10)
         clf=clf.fit(X_train, Y_res)
```



```
In [31]: clf.score(X_train,Y_res)
```



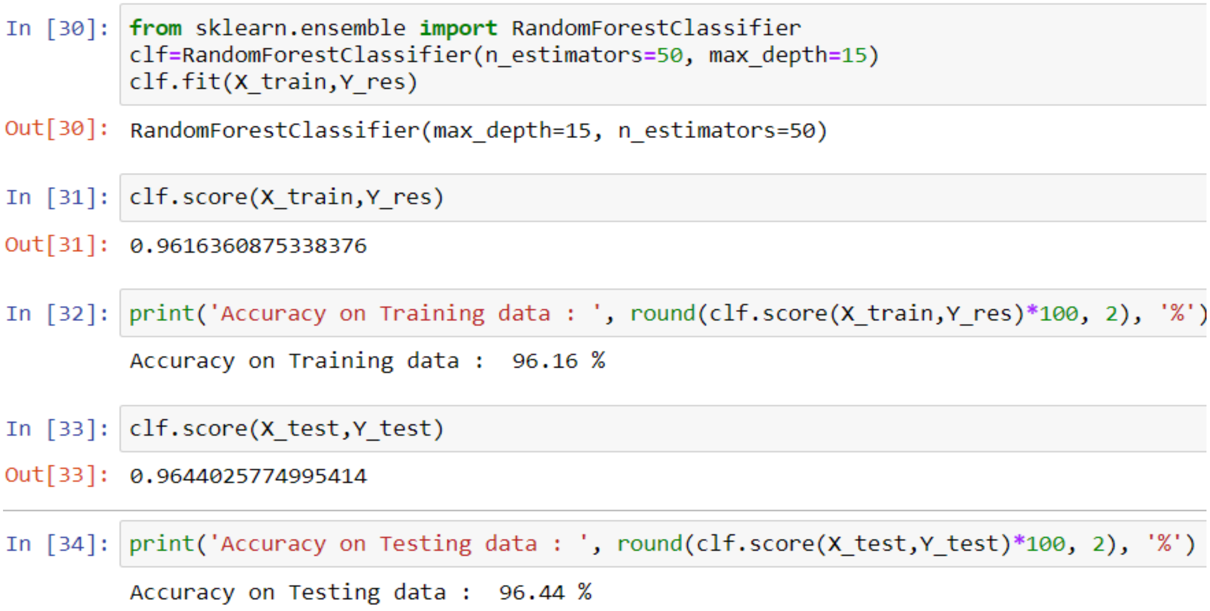
```
In [33]: clf.score(X_test,Y_test)
```



```
In [39]: # Classification Report

         from sklearn.metrics import classification_report
         print(classification_report(Y_test,y_pred_dt))

In [43]: #Confusion Matrix

         from sklearn.metrics import confusion_matrix
         cmat = confusion_matrix(Y_test,y_pred_dt)
```

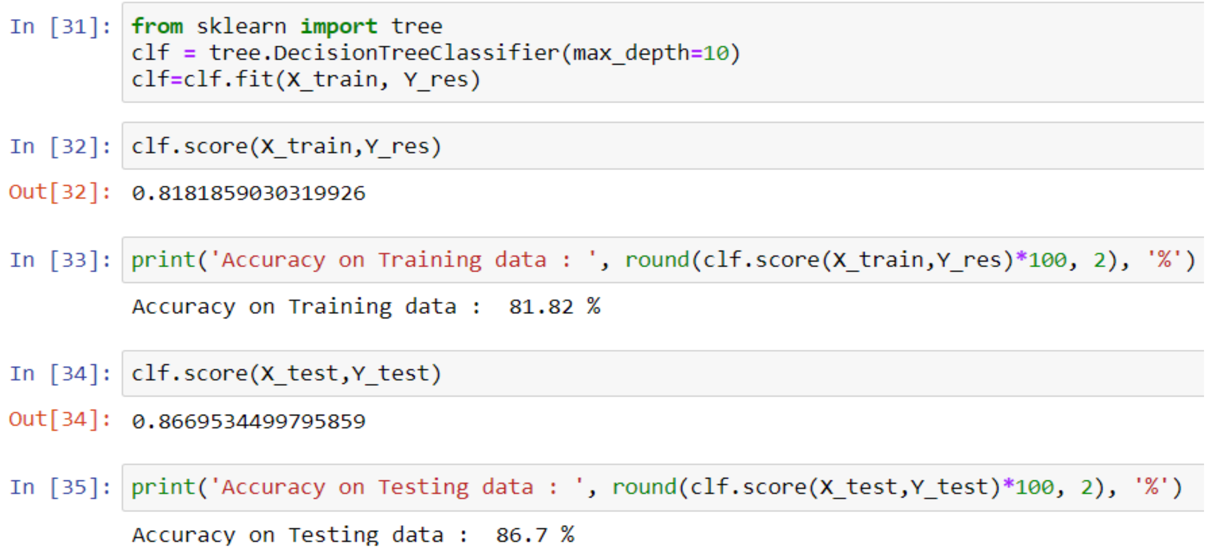
Models Accuracy Summary

Random Forest



```
In [30]: from sklearn.ensemble import RandomForestClassifier
         clf=RandomForestClassifier(n_estimators=50, max_depth=15)
         clf.fit(X_train,Y_res)

Out[30]: RandomForestClassifier(max_depth=15, n_estimators=50)

In [31]: clf.score(X_train,Y_res)

Out[31]: 0.9616360875338376

In [32]: print('Accuracy on Training data : ', round(clf.score(X_train,Y_res)*100, 2), '%')

         Accuracy on Training data :  96.16 %

In [33]: clf.score(X_test,Y_test)

Out[33]: 0.9644025774995414

In [34]: print('Accuracy on Testing data : ', round(clf.score(X_test,Y_test)*100, 2), '%')

         Accuracy on Testing data :  96.44 %
```

Decision Tree



```
In [31]: from sklearn import tree
         clf = tree.DecisionTreeClassifier(max_depth=10)
         clf=clf.fit(X_train, Y_res)

In [32]: clf.score(X_train,Y_res)

Out[32]: 0.8181859030319926

In [33]: print('Accuracy on Training data : ', round(clf.score(X_train,Y_res)*100, 2), '%')

         Accuracy on Training data :  81.82 %

In [34]: clf.score(X_test,Y_test)

Out[34]: 0.8669534499795859

In [35]: print('Accuracy on Testing data : ', round(clf.score(X_test,Y_test)*100, 2), '%')

         Accuracy on Testing data :  86.7 %
```

References

Anaconda | Anaconda Distribution (no date) *Anaconda*. Available at: https://www.anaconda.com/products/distribution (Accessed: 14 December 2022).

Android Mischief Dataset (no date) *Stratosphere IPS*. Available at: https://www.stratosphereips.org/android-mischief-dataset (Accessed: 10 December 2022).

Libraries in Python - GeeksforGeeks (no date). Available at: https://www.geeksforgeeks.org/libraries-in-python/ (Accessed: 14 December 2022).