Configuration Manual

MSc Internship
Cyber Security

Sharad Rajendra Parmar
Student ID: x18176381

School of Computing
National College of Ireland

Supervisor: Niall Heffernan

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	**Sharad Rajendra Parmar**
Student ID:	**X18176381**
Programme:	**Cyber Security** **Year: 2020**
Module:	**MSc Internship**
Lecturer:	**Niall Heffernan**
Submission Due Date:	**17/08/2020**
Project Title:	**Detection of Phishing URL using Ensemble Learning Techniques**
Word Count: 858	**Page Count: 16**

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.
I agree to an electronic copy of my thesis being made publicly available on NORMA the National College of Ireland's Institutional Repository for consultation.

Signature: Sharad Rajendra Parmar

Date: 17/08/2020

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

Sharad Rajendra Parmar
Student ID: x18176381

1. Introduction

This Configuration Manual document includes details on the technical resources and tools and technologies required to execute the project. This manual also provides step-by - step procedure of implementation. The approach involves downloading and installing the necessary applications and services, and the minimal setup needed to operate the project smoothly.

2. System Environment

It is very important to consider System with good hardware specifications which are capable of handling Machine Learning Tasks. The System Specification used for the research are as follows:
Processor: Intel(R) Core(TM) i5-5200 CPU @ 2.20GHz
RAM: 16 GB
Storage: 1 TB HDD
Operating System: Windows 10 Pro (64 bit)

3. Tools and Technologies

Python 3.8.5
Anaconda 2020.02
Jupyter Notebook 6.0.3

4. Download and Installation

4.1 Downloading and Installing Python

The Latest Version of Python has been installed from the official website for the purpose of Development. It is Open Source and free to download. [1] Link for Downloading Python: **https://www.python.org/downloads/**

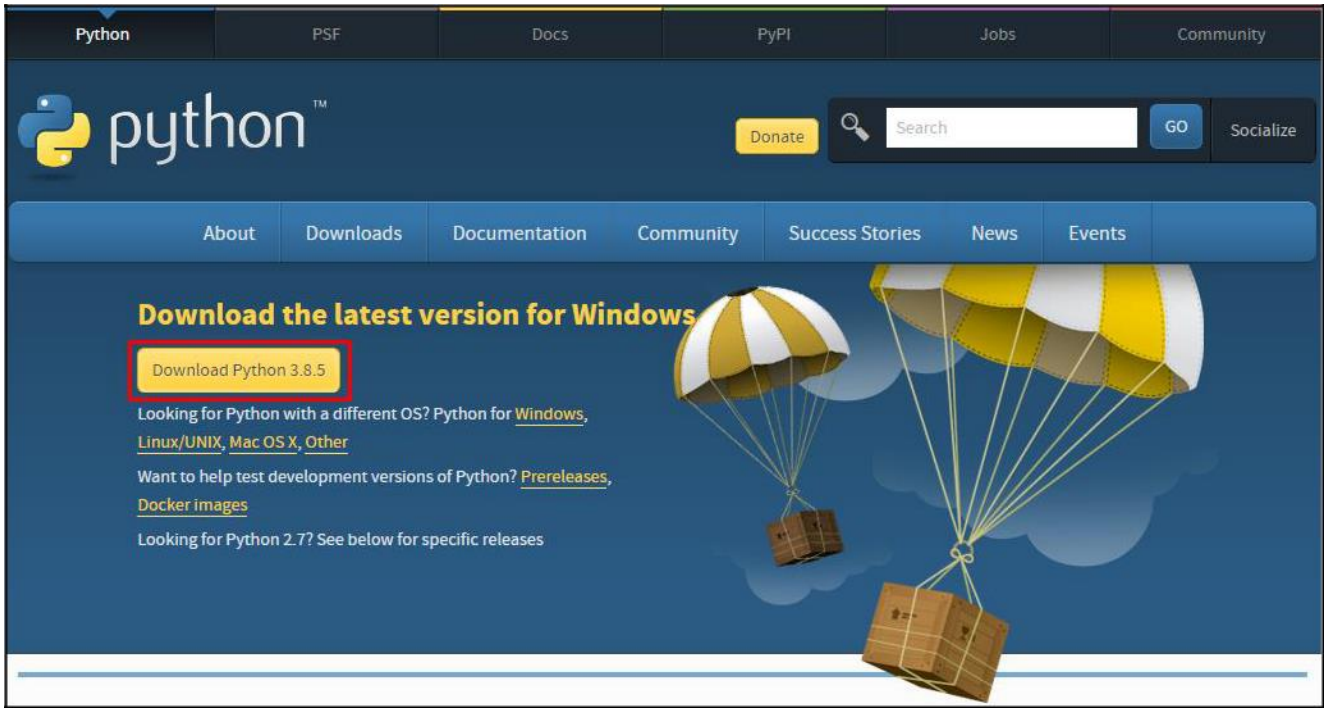

Figure 1: Installation of Python

4.2 Downloading and Installing Anaconda

Anaconda is a free and open source distribution of the Python and R Programming languages for data science and machine learning applications and many more for simplification of package management. The Latest Version of Anaconda has been installed from the official website for the purpose of package management. [2] Link for Downloading Anaconda: https://www.anaconda.com/products/individual

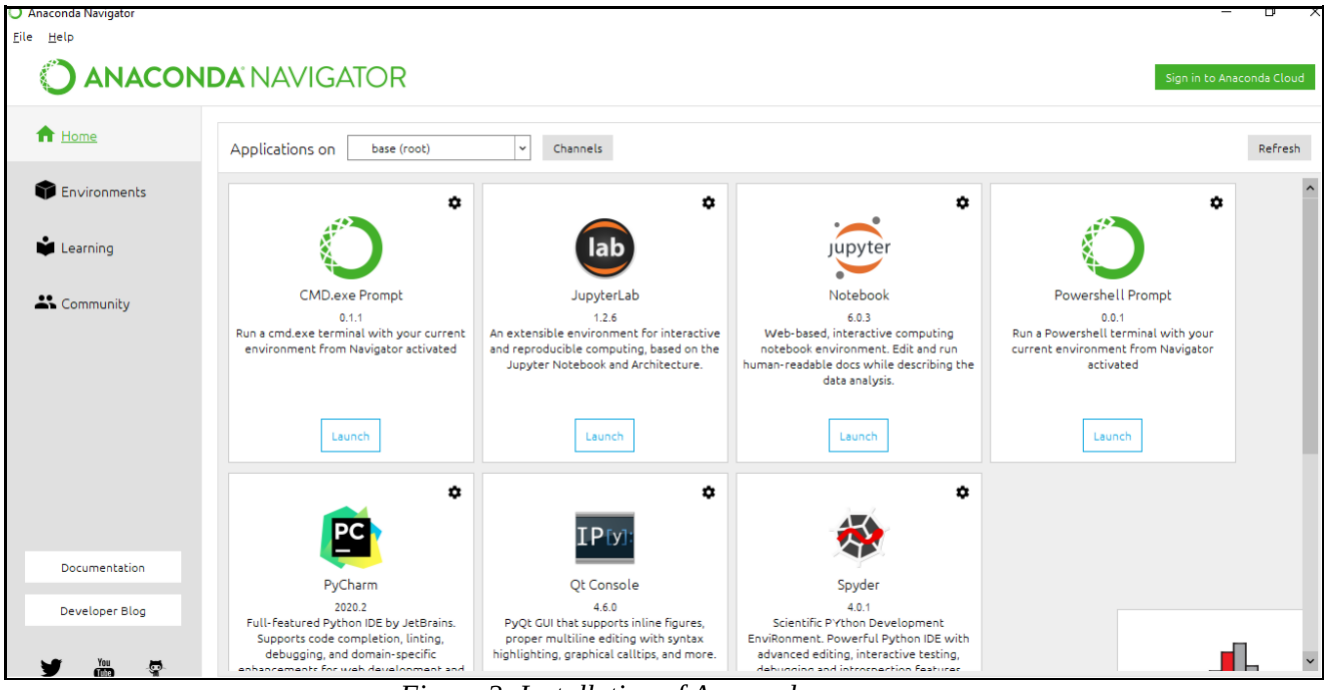

Figure 2: Installation of Anaconda

4.3 Launch Jupyter Notebook

The Jupyter Notebook is a web-based open-source application that helps you to build and exchange documents that include live data, calculations, visualizations and many more. It is used for data cleaning and transformation, numerical simulation, statistical modelling, data visualization, machine learning, etc. Home page of Jupyter Notebook is shown in figure 3. This is the base directory, if we want to create a new file in this directory, we can do it by clicking on new and select Python 3, by this new jupyter notebook will open and then we can start development.

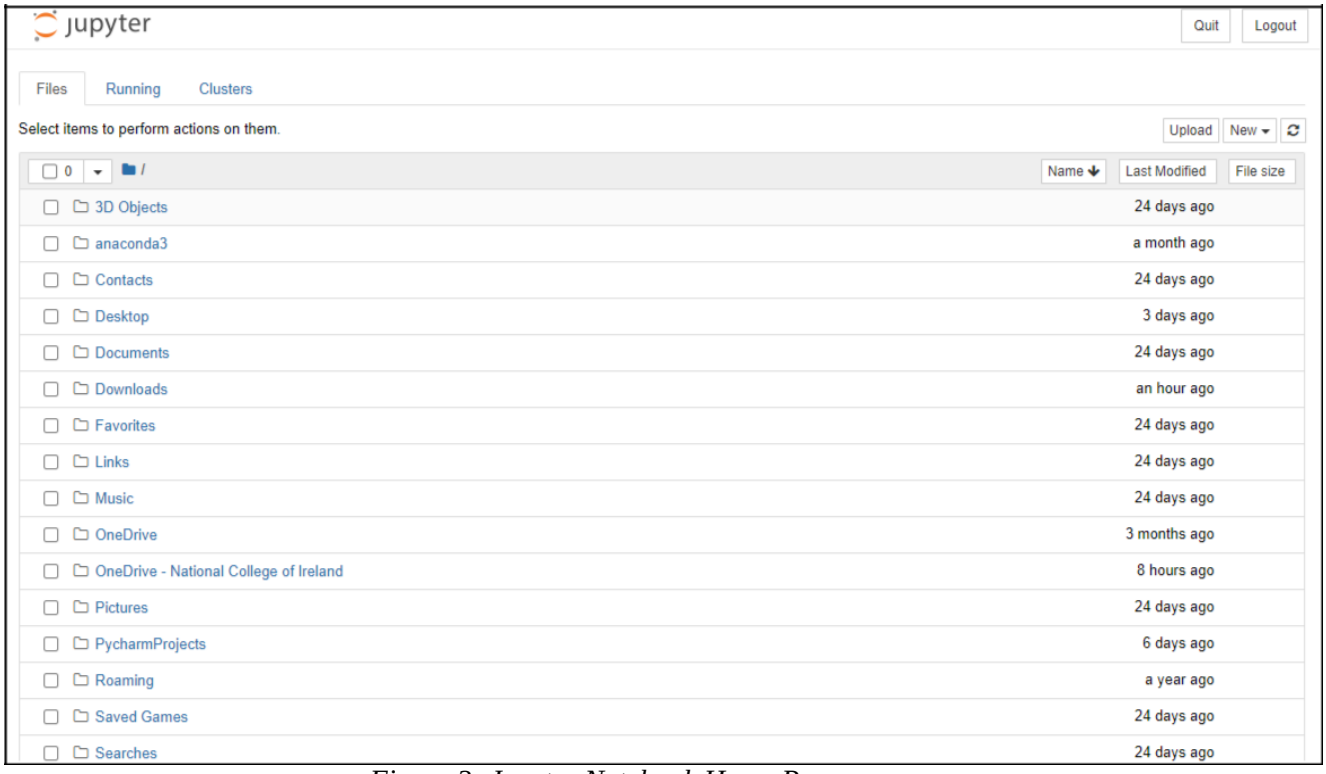

Figure 3: Jupyter Notebook Home Page

5. Project Development

Step 1: Firstly importing all required Libraries of python for development.

```
import numpy as np
import pandas as pd
from scipy.io import arff
import seaborn as sns
sns.set()

import matplotlib.pyplot as plt
```

Figure 4: Python Libraries

Step 2: Importing Dataset in Python using read_csv function.

```
df1 = pd.read_csv("dataset.csv")
#d2 = arff.loadarff("d2.arff")
#df2 = pd.DataFrame(d2)
#d3 = arff.loadarff("d3.arff")
#df3 = pd.DataFrame(d3)
```

Figure 5: Importing Dataset in python

df1

	index	having_IPhaving_IP_Address	URLURL_Length	Shortining_Service	having_At_Symbol	double_slash_redirecting	Prefix_Suffix	having_Sub_Domain
0	1	-1	1	1	1	-1	-1	-1
1	2	1	1	1	1	1	-1	0
2	3	1	0	1	1	1	-1	-1
3	4	1	0	1	1	1	-1	-1
4	5	1	0	-1	1	1	-1	1
...	...	...	...	...	...	...	...	...
11050	11051	1	-1	1	-1	1	1	1
11051	11052	-1	1	1	-1	-1	-1	1
11052	11053	1	-1	1	1	1	-1	1
11053	11054	-1	-1	1	1	1	-1	-1
11054	11055	-1	-1	1	1	1	-1	-1

11055 rows × 32 columns

Figure 6: Glimpse of Dataset imported

Step 3: Plotted Histograms of every variables present in the dataset with respect to frequency.

```
import matplotlib.pyplot as plt

for col in df1:
    plt.hist(df1[col],bins = 3)
    plt.xlabel(col)
    plt.ylabel("Frequency")
    plt.title(col)
    plt.show()
```

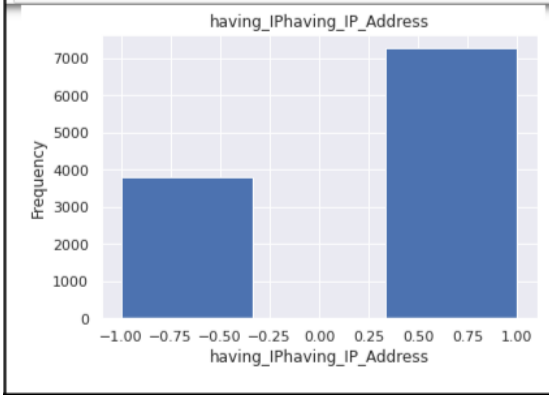

Figure 7: Histograms of all Variables

Step 4: Imported all the classifiers which were considered for development from sklearn python library.

```
from sklearn import metrics
from sklearn.ensemble import BaggingClassifier, RandomForestClassifier, AdaBoostClassifier, GradientBoostingClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from datetime import datetime
```

Figure 8: importing classification models

Step 5: Correlation of all variables were outputted.

	Index	having_IPhaving_IP_Address	URLURL_Length	Shortining_Service	having_At_Symbol	double_slash_redirecting	Prefix_
Index	1.000000	-0.388317	0.006105	-0.006281	-0.169478	-0.003363	-0.0(
having_IPhaving_IP_Address	-0.388317	1.000000	-0.052411	0.403461	0.158699	0.397389	-0.0(
URLURL_Length	0.006105	-0.052411	1.000000	-0.097881	-0.075108	-0.081247	0.0(
Shortining_Service	-0.006281	0.403461	-0.097881	1.000000	0.104447	0.842796	-0.0(
having_At_Symbol	-0.169478	0.158699	-0.075108	0.104447	1.000000	0.086960	-0.0
double_slash_redirecting	-0.003363	0.397389	-0.081247	0.842796	0.086960	1.000000	-0.0(
Prefix_Suffix	-0.007340	-0.005257	0.055247	-0.080471	-0.011726	-0.085590	1.0(
having_Sub_Domain	0.234091	-0.080745	0.003997	-0.041916	-0.058976	-0.043079	0.0(
SSLfinal_State	-0.006682	0.071414	0.048754	-0.061426	0.031220	-0.036200	0.2(
Domain_registeration_length	-0.001180	-0.022739	-0.221892	0.060923	0.015522	0.047464	-0.0(
Favicon	0.007293	0.087025	-0.042497	0.006101	0.304899	0.035100	-0.0(
port	0.001656	0.060979	0.000323	0.002201	0.364891	0.025060	-0.0
HTTPS_token	0.002916	0.363534	-0.089383	0.757838	0.104561	0.760799	-0.0
Request_URL	-0.000862	0.029773	0.246348	-0.037235	0.027909	-0.026368	0.0(
URL_of_Anchor	-0.005071	0.099847	-0.023396	0.000561	0.057914	-0.005036	0.3
Links_in_tags	-0.028865	0.006212	0.052869	-0.133379	-0.070861	-0.125583	0.1(
SFH	0.085354	-0.010962	0.414196	-0.022723	-0.008672	-0.041672	0.0(
Submitting_to_email	0.005828	0.077989	-0.014457	0.049328	0.370123	0.031898	-0.0
Abnormal_URL	0.003228	0.336549	-0.106761	0.739290	0.203945	0.723724	-0.0
Redirect	0.016804	-0.321181	0.046832	-0.534530	-0.028160	-0.591478	0.0
on_mouseover	0.003649	0.084059	-0.045103	0.062383	0.279697	0.086635	0.0
RightClick	-0.005265	0.042881	-0.013613	0.038118	0.219503	0.025863	-0.0
popUpWidnow	0.006515	0.096882	-0.049381	0.036616	0.290893	0.054463	-0.0
Iframe	0.002533	0.054694	-0.013838	0.016581	0.284410	0.010459	-0.0
age_of_domain	0.115320	-0.010446	0.179426	-0.052596	-0.005499	-0.050107	0.0
DNSRecord	0.400890	-0.050733	-0.040823	0.436064	-0.047872	0.431409	-0.0
web_traffic	-0.014900	0.002922	0.008993	-0.047074	0.032918	-0.062369	0.1
Page_Rank	0.065117	-0.091774	0.183518	0.014591	-0.064735	-0.003132	-0.0(
Google_Index	-0.012527	0.029153	0.002902	0.155844	0.037061	0.178415	0.0(
Links_pointing_to_page	0.002442	-0.339065	-0.022987	-0.198410	-0.006080	-0.194165	0.0(
Statistical_report	0.163799	-0.019103	-0.067153	0.085461	-0.080357	0.070390	-0.0(
Result	0.000978	0.094160	0.057430	-0.067966	0.052948	-0.038608	0.3

Figure 9: Correlation of variables

Step 6: Next step is Data Pre-processing and splitting the data in training set and testing set.

Preprocessing

```
#df2 = df1.iloc[:,[6,7,8,9,13,14,15,16,24,26,31]]
x = df1.iloc[:,0:31]
y = df1.loc[:,"Result"]
train_x, test_x, train_y, test_y = train_test_split(x,y,shuffle = True,test_size = 0.2)
```

Figure 10: Pre-Processing and Splitting od data

Step 7: Analysis of Ensemble Models.

Bagging Classifier

```
time_st_bc = datetime.now()
bc = BaggingClassifier(n_estimators=15)
bc.fit(train_x, train_y)
time_fn_bc = datetime.now()
predict_y_bc = bc.predict(test_x)
accuracy_bc = metrics.accuracy_score(test_y, predict_y_bc)
precision_bc = metrics.precision_score(test_y, predict_y_bc)
recall_bc = metrics.recall_score(test_y, predict_y_bc)
f1_score_bc = metrics.f1_score(test_y, predict_y_bc)
confusion_matrix_bc = metrics.plot_confusion_matrix(bc,test_x, test_y)
plt.title("Bagging Classifier")
plt.show()
time_bc = time_fn_bc - time_st_bc
```

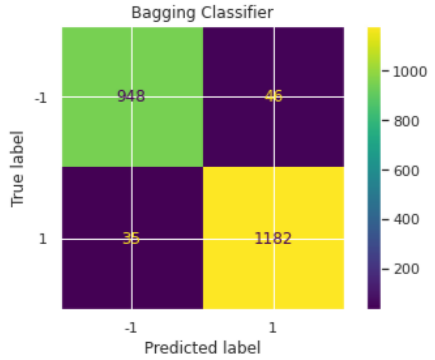

Figure 11: Bagging Classifier

Random Forest Classifier

```
time_st_rfc = datetime.now()
rfc = RandomForestClassifier(n_estimators=15)
rfc.fit(train_x, train_y)
time_fn_rfc = datetime.now()
predict_y_rfc = rfc.predict(test_x)
accuracy_rfc = metrics.accuracy_score(test_y, predict_y_rfc)
precision_rfc = metrics.precision_score(test_y, predict_y_rfc)
recall_rfc = metrics.recall_score(test_y, predict_y_rfc)
f1_score_rfc = metrics.f1_score(test_y, predict_y_rfc)
confusion_matrix_rfc = metrics.confusion_matrix(test_y, predict_y_rfc)
confusion_matrix_rfc = metrics.plot_confusion_matrix(rfc,test_x, test_y)
plt.title("Random Forest Classifier")
plt.show()
time_rfc = time_fn_rfc - time_st_rfc
```

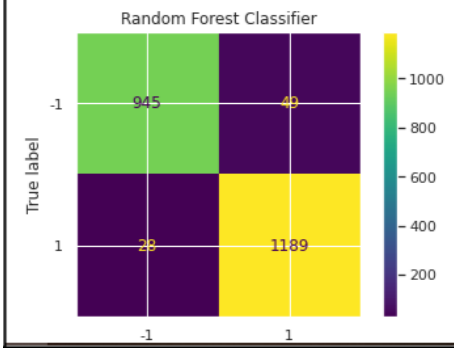

Figure 12: Random Forest Classifier

Adaboost Classifier

```
time_st_ac = datetime.now()
ac = AdaBoostClassifier(n_estimators=15)
ac.fit(train_x, train_y)
time_fn_ac = datetime.now()
predict_y_ac = ac.predict(test_x)
accuracy_ac = metrics.accuracy_score(test_y, predict_y_ac)
precision_ac = metrics.precision_score(test_y, predict_y_ac)
recall_ac = metrics.recall_score(test_y, predict_y_ac)
f1_score_ac = metrics.f1_score(test_y, predict_y_ac)
confusion_matrix_ac = metrics.confusion_matrix(test_y, predict_y_ac)
confusion_matrix_ac = metrics.plot_confusion_matrix(ac,test_x, test_y)
plt.title("Adaboost Classifier")
plt.show()
time_ac = time_fn_ac - time_st_ac
```

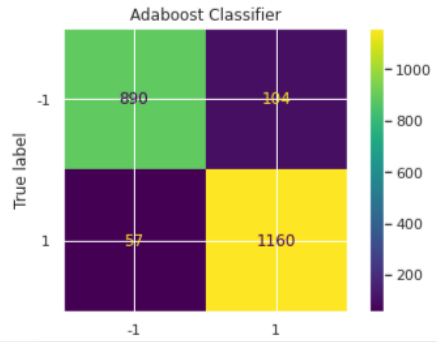

Figure 13: Adaboost Classifier

Gradient Boosting Classifier

```
time_st_gbc = datetime.now()
gbc = GradientBoostingClassifier(n_estimators=15)
gbc.fit(train_x, train_y)
time_fn_gbc = datetime.now()
predict_y_gbc = gbc.predict(test_x)
accuracy_gbc = metrics.accuracy_score(test_y, predict_y_gbc)
precision_gbc = metrics.precision_score(test_y, predict_y_gbc)
recall_gbc = metrics.recall_score(test_y, predict_y_gbc)
f1_score_gbc = metrics.f1_score(test_y, predict_y_gbc)
confusion_matrix_gbc = metrics.confusion_matrix(test_y, predict_y_gbc)
confusion_matrix_gbc = metrics.plot_confusion_matrix(gbc,test_x, test_y)
plt.title("Gradient Boosting Classifier")
plt.show()
time_gbc = time_fn_gbc - time_st_gbc
```

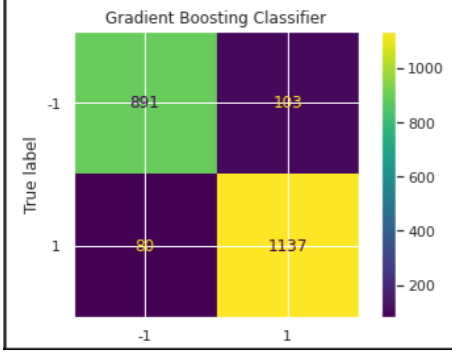

Figure 14: Gradient Boosting Classifier

Step 8: Analysis of Non-Ensemble Models.

Decision Tree Classifier

```
time_st_dt = datetime.now()
dt = DecisionTreeClassifier(max_depth=2)
dt.fit(train_x, train_y)
time_fn_dt = datetime.now()
predict_y_dt = dt.predict(test_x)
accuracy_dt = metrics.accuracy_score(test_y, predict_y_dt)
precision_dt = metrics.precision_score(test_y, predict_y_dt)
recall_dt = metrics.recall_score(test_y, predict_y_dt)
f1_score_dt = metrics.f1_score(test_y, predict_y_dt)
confusion_matrix_dt = metrics.confusion_matrix(test_y, predict_y_dt)
confusion_matrix_dt = metrics.plot_confusion_matrix(dt,test_x, test_y)
plt.title("Decision Tree Classifier")
plt.show()
time_dt = time_fn_dt - time_st_dt
```

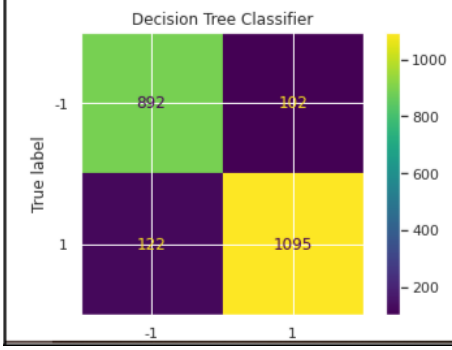

Figure 15: Decission Tree Classifier

KNeighbors Classification

```
time_st_kn = datetime.now()
kn = KNeighborsClassifier(n_neighbors=10)
kn.fit(train_x, train_y)
time_fn_kn = datetime.now()
predict_y_kn = kn.predict(test_x)
accuracy_kn = metrics.accuracy_score(test_y, predict_y_kn)
precision_kn = metrics.precision_score(test_y, predict_y_kn)
recall_kn = metrics.recall_score(test_y, predict_y_kn)
f1_score_kn = metrics.f1_score(test_y, predict_y_kn)
confusion_matrix_kn = metrics.confusion_matrix(test_y, predict_y_kn)
confusion_matrix_kn = metrics.plot_confusion_matrix(kn,test_x, test_y)
plt.title("KNeighbors Classifier")
plt.show()
time_kn = time_fn_kn - time_st_kn
```

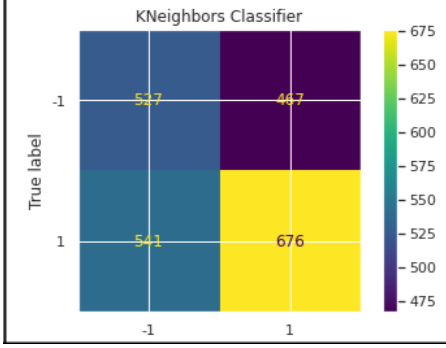

Figure 16: KNeighbors Classifier

Logistic Regression

```
time_st_lr = datetime.now()
lr = LogisticRegression(max_iter=1000)
lr.fit(train_x, train_y)
time_fn_lr = datetime.now()
predict_y_lr = lr.predict(test_x)
accuracy_lr = metrics.accuracy_score(test_y, predict_y_lr)
precision_lr = metrics.precision_score(test_y, predict_y_lr)
recall_lr = metrics.recall_score(test_y, predict_y_lr)
f1_score_lr = metrics.f1_score(test_y, predict_y_lr)
confusion_matrix_lr = metrics.confusion_matrix(test_y, predict_y_lr)
confusion_matrix_lr = metrics.plot_confusion_matrix(lr,test_x, test_y)
plt.title("Logistic Regression")
plt.show()
time_lr = time_fn_lr - time_st_lr
```

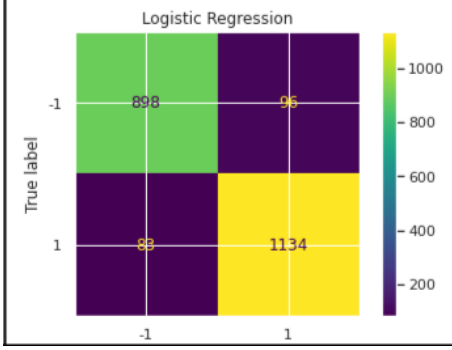

Figure 17: Logistic Regression

Step 9: Generated Confusion Matrices for Ensemble and Non-Ensemble Models.

Confusion Matrices

Bagging Classifier

```
In [ ]: confusion_matrix_bc
```

```
Out[65]: array([[ 915,   48],
                [  49, 1199]])
```

Random Forest Classifier

```
In [ ]: confusion_matrix_rfc
```

```
Out[66]: array([[ 927,   36],
                [  49, 1199]])
```

Adaboost Classifier

```
In [ ]: confusion_matrix_ac
```

```
Out[67]: array([[ 883,   80],
                [  69, 1179]])
```

Gradient Boosting Classifier

```
In [ ]: confusion_matrix_gbc
```

```
Out[68]: array([[ 869,   94],
                [  72, 1176]])
```

Decision Tree Classifier

```
In [ ]: confusion_matrix_dt
```

```
Out[74]: array([[ 873,   90],
                [ 120, 1128]])
```

KNeigbors Classifier

```
In [ ]: confusion_matrix_kn
```

```
Out[70]: array([[516, 447],
                [563, 685]])
```

Logistic Regression

```
In [ ]: confusion_matrix_lr
```

```
Out[71]: array([[ 874,   89],
                [  79, 1169]])
```

Figure 18: Confusion Matrices

Step 10: Calculated Accuracy measures for all the classifiers

```
print("accuracy")
print(pd.DataFrame(accuracy.items()))
print("precision")
print(pd.DataFrame(precision.items()))
print("recall")
print(pd.DataFrame(recall.items()))
print("f1_score")
print(pd.DataFrame(f1_score.items()))
print("times")
print(pd.DataFrame(times.items()))
```

```
accuracy
                             0         1
0            Bagging Classifier  0.956128
1      Random Forest Classifier  0.961556
2           Adaboost Classifier  0.932610
3  Gradient Boosting Classifier  0.924921
4      Decision Tree Classifier  0.905020
5          KNeigbors Classifier  0.543193
6           Logistic Regression  0.924016
precision
                             0         1
0            Bagging Classifier  0.961508
1      Random Forest Classifier  0.970850
2           Adaboost Classifier  0.936458
3  Gradient Boosting Classifier  0.925984
4      Decision Tree Classifier  0.926108
5          KNeigbors Classifier  0.605124
6           Logistic Regression  0.929253
recall
                             0         1
0            Bagging Classifier  0.960737
1      Random Forest Classifier  0.960737
2           Adaboost Classifier  0.944712
3  Gradient Boosting Classifier  0.942308
4      Decision Tree Classifier  0.903846
5          KNeigbors Classifier  0.548878
6           Logistic Regression  0.936699
f1_score
                             0         1
0            Bagging Classifier  0.961122
1      Random Forest Classifier  0.965767
2           Adaboost Classifier  0.940566
3  Gradient Boosting Classifier  0.934075
4      Decision Tree Classifier  0.914842
5          KNeigbors Classifier  0.575630
6           Logistic Regression  0.932961
times
                             0       1
0            Bagging Classifier  275728
1      Random Forest Classifier  105795
2           Adaboost Classifier  123008
3  Gradient Boosting Classifier  181806
4      Decision Tree Classifier   14589
5          KNeigbors Classifier   41833
6           Logistic Regression  559801
```

Figure 19: Accuracy Measures

Step 11: Final Step is to do Analysis and comparison of all the measures mentioned below:

a. Accuracy:

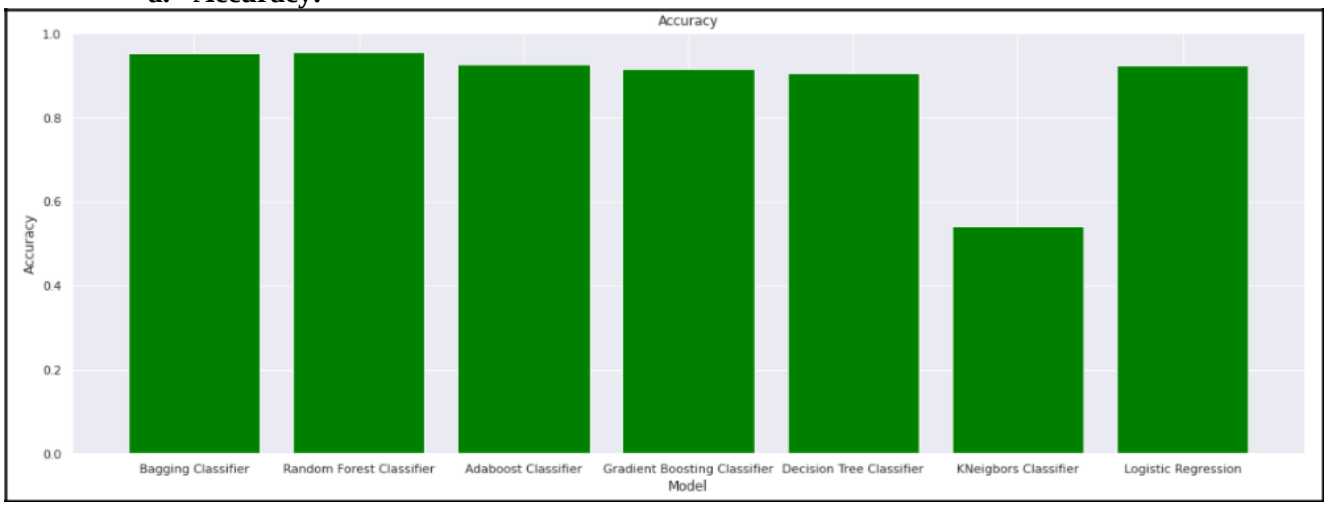

Figure 20: Accuracy

b. Precision:

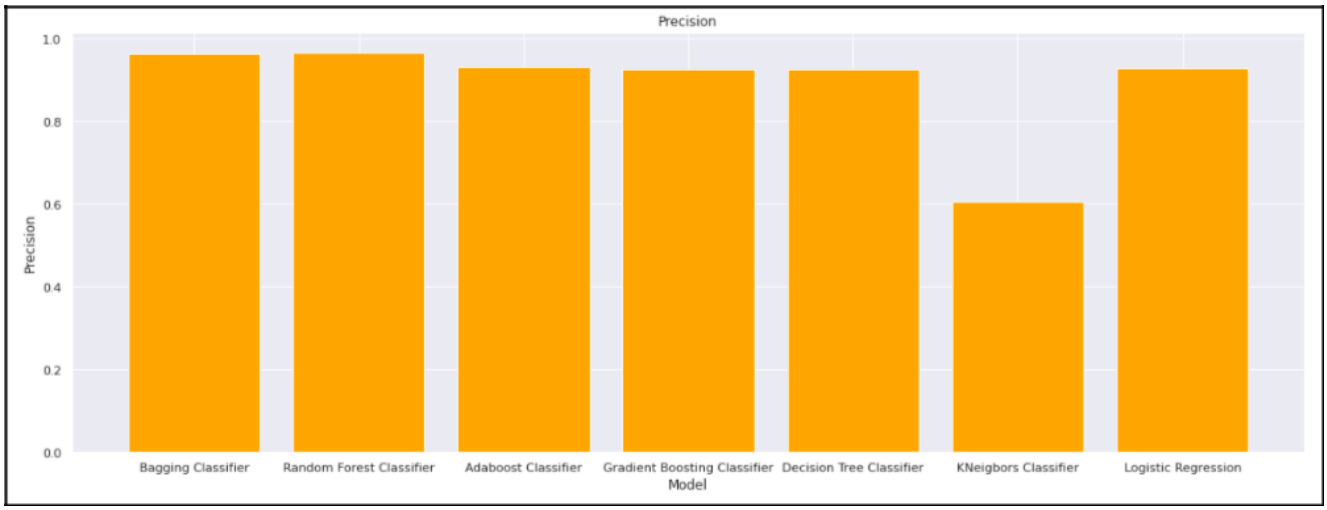

Figure 21: Precision

c. Recall:

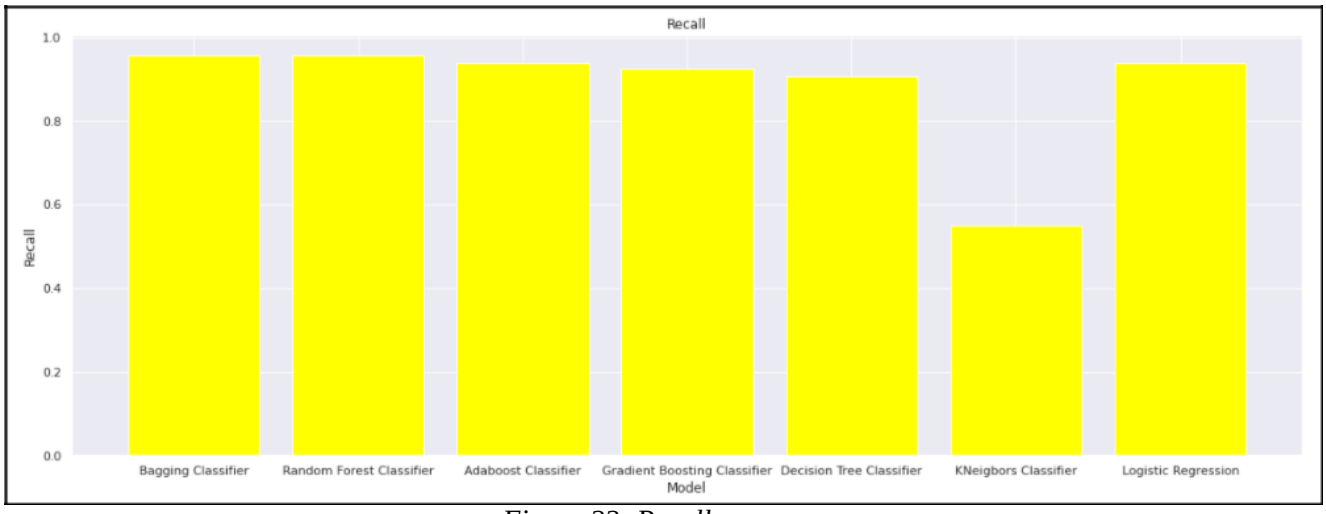

Figure 22: Recall

d. F1-Score:

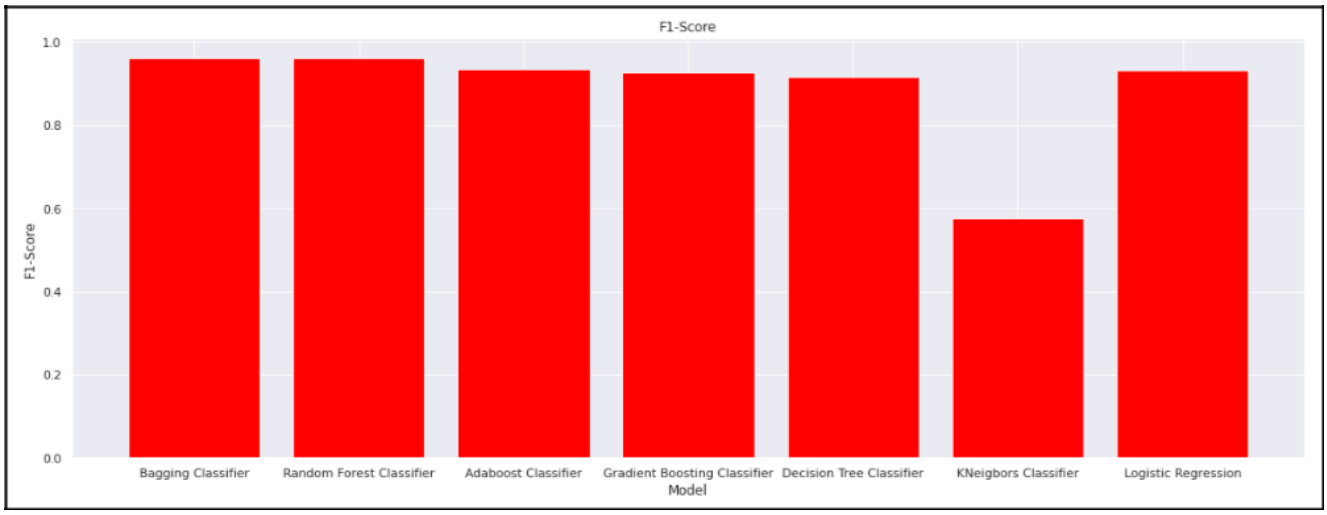

Figure 23: F1-Score

e. Time:

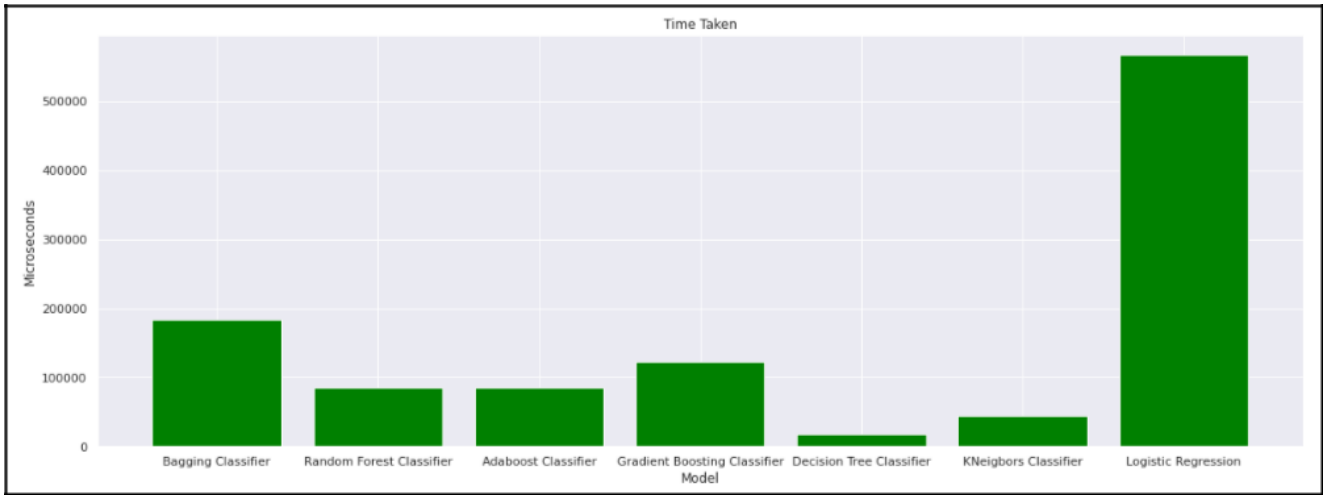

Figure 24: Time Taken

References

[1] ‘Download Python’. Python.Org, https://www.python.org/downloads/.

[2] ‘Anaconda | Individual Edition’. Anaconda, https://www.anaconda.com/products/individual.