IMPLEMENTING MACHINE LEARNING ALGORITHMS FOR ADVANCED PERSISTENT THREAT (APT) DETECTION AND RESPONSE

MSc Research Project

Msc in Cyber Security

Olamide Eniola Ogunbanjo

X21231125

School of Computing

National College of Ireland

Supervisor: Mr Micheal Prior

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Olamide Eniola Ogunbanjo …………………………………………………………………………
Student ID:	X21231125……………………………………………………………………………………..……
Programme:	Cybersecurity…………………………………………… **Year:** 2023………………..
Module:	Research Project……..……………………………………………………………………….………
Supervisor:	Mr Micheal Prior…………………………………………………………………………………..………
Submission Due Date:	14/08/202…………………………………………………………..………
Project Title:	Implementation of machine learning algorithms for advanced persistent threat detection and response
Word Count: 7280	……………………………………… **Page Count 27**……………………………………….……..

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: Olamide Eniola Ogunbanjo……………..…………………………………………………………

Date: 11/08/2023………………………………………………………………………………………………

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

Olamide Eniola Ogunbanjo

X21231125

1. Introduction

This project is aimed at utilizing machine learning algorithms for the detection and response of Advanced Persistent Threats (APTs) in the cybersecurity domain. The project makes use of the BETH cybersecurity dataset and implements a range of machine learning models for threat detection. These models are then evaluated and compared using performance metrics.

2. Prerequisites

To run the project, the following software and hardware prerequisites are required:

- Google Colab or a local Python environment with necessary libraries installed
- Python 3.7 or later
- Libraries: pandas, numpy, seaborn, matplotlib, scikit-learn, xgboost, joblib
- Internet access (for Google Colab and to download the BETH dataset)

3. Installation Instructions

If you are using a local Python environment, you should install the necessary libraries using

```
pip install pandas numpy seaborn matplotlib scikit-learn xgboost joblib
```

4. Configuration

The project does not require any special configuration settings.

Usage

The project is implemented in a Jupyter notebook. To run the project, follow these steps:

1. **Data Import and Preprocessing**: Load the BETH cybersecurity dataset into pandas dataframes. Perform initial data exploration to identify any null values or other issues. Feature engineering steps include dropping unnecessary columns, label encoding of

categorical variables, and scaling numerical variables. Finally, the data is split into training and testing sets.

2. **Model Training and Evaluation**: Train various machine learning models including Random Forests, Gradient Boosting, XGBoost, K-Nearest Neighbors, Naive Bayes, Decision Trees, and AdaBoost. Each trained model is saved using joblib for future use. The performance of each model is then evaluated using the classification report, which provides precision, recall, and F1-score.
3. **Model Comparison**: Plot ROC curves for the different models to compare their performance visually. This helps in identifying the model that performs best.

5. Visual Work Through

Step 1 Importing necessary Python libraries such as pandas, numpy, seaborn, and matplotlib

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

Step 2 Importing dataset and unzipping the 'archive.zip' file to access the data

```
# Importing the zipfile module
import zipfile

# Creating a ZipFile object with the file name
zip_file = zipfile.ZipFile("/content/archive.zip")

# Extracting all the files in the current directory
zip_file.extractall()

# Closing the ZipFile object
zip_file.close()

```

Loading the 'labelled_testing_data.csv' file into a pandas DataFrame

```
#beth_df_1 = pd.read_csv("labelled_training_data.csv")
import pandas as pd
beth_df_1=pd.read_csv('/content/labelled_testing_data.csv',  encoding='ISO-8859-1')
beth_df_1.head()
```

	timestamp	processId	threadId	parentProcessId	userId	mountNamespace	processName	hostName	eventId	eventName	stackAddresses	argsNum	returnValue	args	sus	evil
0	129.050634	382	382	1	101	4026532232	systemd-resolve	ip-10-100-1-217	41	socket	[140159195621643, 140159192455417, 94656731598...	3	15	[{'name': 'domain', 'type': 'int', 'value': 'A...	0	0
1	129.051238	379	379	1	100	4026532231	systemd-network	ip-10-100-1-217	41	socket	[139853228042507, 93935071185801, 93935080775184]	3	15	[{'name': 'domain', 'type': 'int', 'value': 'A...	0	0
2	129.051434	1	1	0	0	4026531840	systemd	ip-10-100-1-217	1005	security_file_open	[140362867191588, 8103505641674583858]	4	0	[{'name': 'pathname', 'type': 'const char*', '...	0	0
3	129.051481	1	1	0	0	4026531840	systemd	ip-10-100-1-217	257	openat	[]	4	17	[{'name': 'dirfd', 'type': 'int', 'value': -10...	0	0
4	129.051522	1	1	0	0	4026531840	systemd	ip-10-100-1-217	5	fstat	[140362867189385]	2	0	[{'name': 'fd', 'type': 'int', 'value': 17}, {...	0	0

Loading the 'labelled_training_data.csv' and 'labelled_validation_data.csv' files into pandas DataFrames

```
[ ]
#beth_df_2 = pd.read_csv("labelled_testing_data.csv")
#beth_df_3 = pd.read_csv("labelled_validation_data.csv")
import pandas as pd
beth_df_2=pd.read_csv('/content/labelled_training_data.csv',  encoding='ISO-8859-1')
beth_df_2.head()
beth_df_3=pd.read_csv('/content/labelled_validation_data.csv',  encoding='ISO-8859-1')
beth_df_3.head()
```

Checking the shape (number of rows and columns) of the three DataFrames

```
[ ]
beth_df_1.shape
```

```
(188967, 16)
```

```
[ ]
beth_df_2.shape
```

```
(763144, 16)
```

```
[ ]
beth_df_3.shape
```

```
(188967, 16)
```

Merging the three DataFrames into one and check the shape of the merged DataFrame

```
[ ]
#Merge Dataframe
beth_df = pd.concat ([beth_df_1, beth_df_2, beth_df_3])
beth_df.shape
```

```
(1141078, 16)
```

Shuffling the merged DataFrame and resetting the index

```
[ ]  #shuffle Dataframe
     df = beth_df.sample(frac=1).reset_index(drop=True)
```

Displaying the first few rows of the DataFrame

```
df.head(2)
```

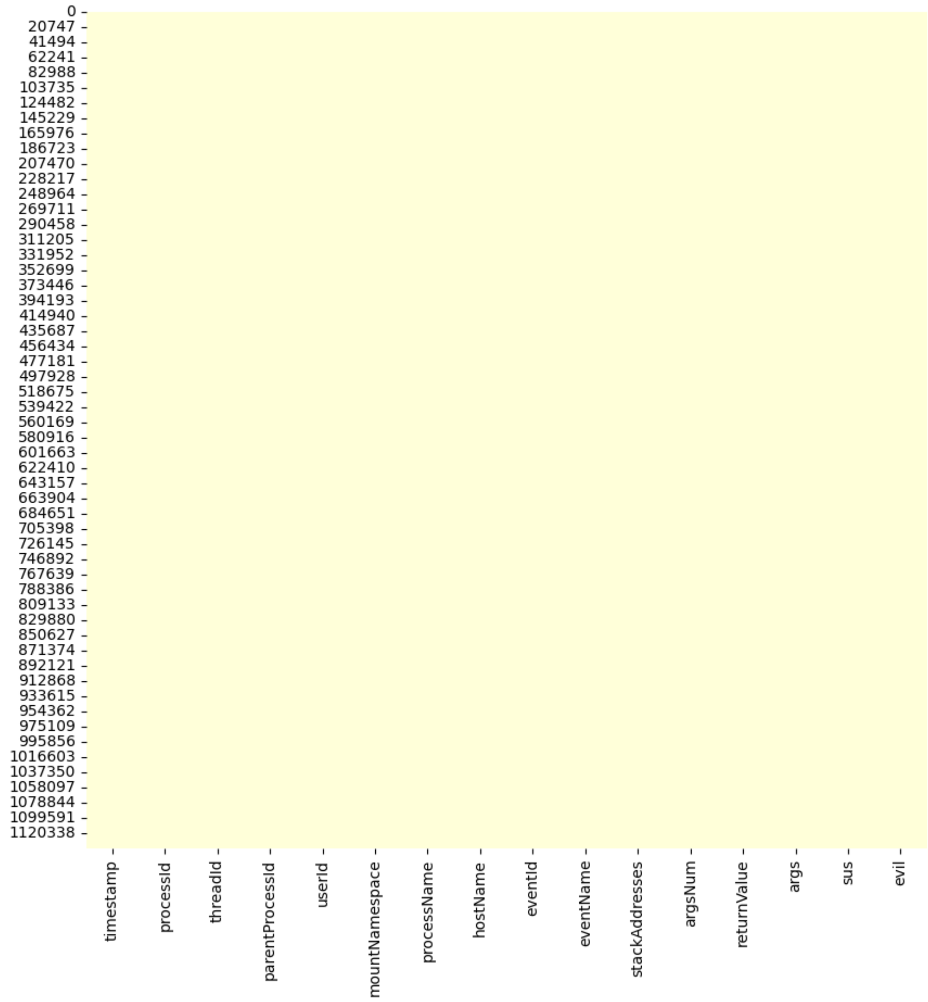
	timestamp	processId	threadId	parentProcessId	userId	mountNamespace	processName	hostName	eventId	eventName	stackAddresses	argsNum	returnValue	args	sus	evil
0	412.414600	7292	7292	187	0	4026532217	systemd-udevd	ubuntu	5	fstat	[]	2	0	[{'name': 'fd', 'type': 'int', 'value': 6}, {'...	0	0
1	138.932981	1	1	0	0	4026531840	systemd	ip-10-100-1-129	1005	security_file_open	[]	4	0	[{'name': 'pathname', 'type': 'const char*', '...	0	0

STEP 3 Exploring the Dataset

Using a heatmap to visualize the presence of any null values in the DataFrame

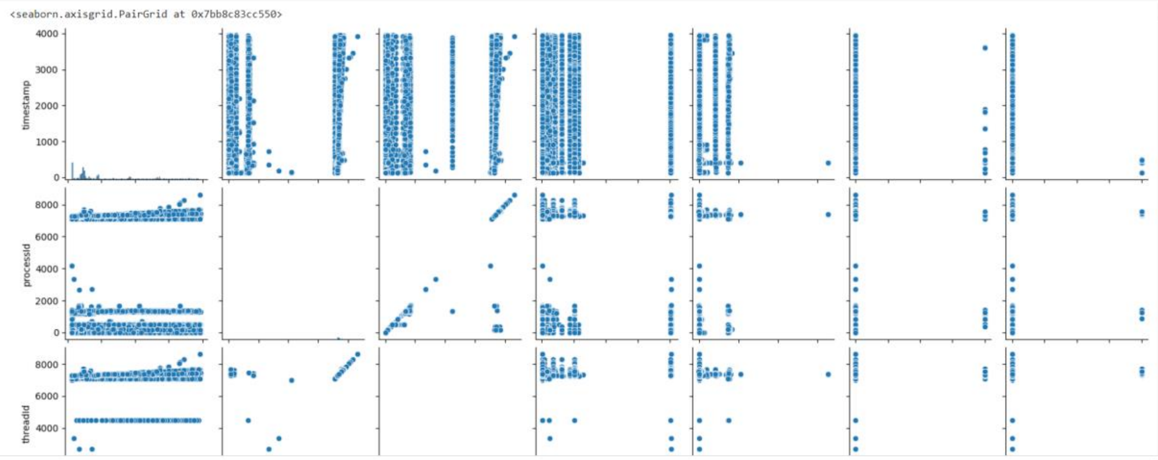
```
[ ]  #check null value in the dataset
     plt.figure(figsize = (10,10))
     sns.heatmap(df.isnull(), cbar = False, cmap="YlGnBu")
```

Display of the Heatmap



Visualizing the pairwise relationships in the DataFrame using a pair plot

```
[ ]  1 sns.pairplot(new)
```



Displaying information about the DataFrame including the index dtype and column dtypes, non-null values, and memory usage

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1141078 entries, 0 to 1141077
Data columns (total 16 columns):
 #   Column           Non-Null Count    Dtype  
---  ------           --------------    -----  
 0   timestamp        1141078 non-null  float64
 1   processId        1141078 non-null  int64  
 2   threadId         1141078 non-null  int64  
 3   parentProcessId  1141078 non-null  int64  
 4   userId           1141078 non-null  int64  
 5   mountNamespace   1141078 non-null  int64  
 6   processName      1141078 non-null  object 
 7   hostName         1141078 non-null  object 
 8   eventId          1141078 non-null  int64  
 9   eventName        1141078 non-null  object 
 10  stackAddresses   1141078 non-null  object 
 11  argsNum          1141078 non-null  int64  
 12  returnValue      1141078 non-null  int64  
 13  args             1141078 non-null  object 
 14  sus              1141078 non-null  int64  
 15  evil             1141078 non-null  int64  
dtypes: float64(1), int64(10), object(5)
memory usage: 139.3+ MB
```

Displaying descriptive statistics of the DataFrame that summarize the central tendency, dispersion, and shape of a dataset's distribution

```
[ ] df.describe()
```

	timestamp	processId	threadId	parentProcessId	userId	mountNamespace	eventId	argsNum	returnValue	sus	evil
count	1.141078e+06	1.141078e+06	1.141078e+06	1.141078e+06	1.141078e+06	1.141078e+06	1.141078e+06	1.141078e+06	1.141078e+06	1.141078e+06	1.141078e+06
mean	1.367449e+03	6.909070e+03	6.913038e+03	2.467229e+03	1.437311e+02	4.026532e+09	2.372977e+02	2.671557e+00	3.018248e+00	1.520615e-01	1.388441e-01
std	1.154433e+03	1.816699e+03	1.807393e+03	2.862640e+03	3.500947e+02	1.726697e+02	3.548319e+02	1.250393e+00	3.223468e+02	3.590806e-01	3.457840e-01
min	1.244392e+02	1.000000e+00	1.000000e+00	0.000000e+00	0.000000e+00	4.026532e+09	2.000000e+00	0.000000e+00	-1.150000e+02	0.000000e+00	0.000000e+00
25%	4.612974e+02	7.301000e+03	7.301000e+03	1.870000e+02	0.000000e+00	4.026532e+09	4.000000e+00	1.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
50%	9.033516e+02	7.366000e+03	7.366000e+03	1.385000e+03	0.000000e+00	4.026532e+09	4.200000e+01	3.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
75%	2.327305e+03	7.461000e+03	7.461000e+03	4.489000e+03	0.000000e+00	4.026532e+09	2.570000e+02	4.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
max	3.954588e+03	8.619000e+03	8.619000e+03	7.672000e+03	1.001000e+03	4.026532e+09	1.010000e+03	5.000000e+00	3.276800e+04	1.000000e+00	1.000000e+00

STEP 4 ENCODING

Encoding the 'hostName' column of the DataFrame using label encoding

```
#Target column encoding
def encode_text_index(df,name):
  le = preprocessing.LabelEncoder()
  df[name] = le.fit_transform(df[name])
  return le.classes_
```

Normalizing the numeric columns in the DataFrame using z-score normalization

```
#Target column encoding
from sklearn import preprocessing
#def encode_text_index(df,name):
le = preprocessing.LabelEncoder()
le.fit_transform(df['hostName'])
le_name_mapping = dict(zip(le.classes_, le.transform(le.classes_)))
print(le_name_mapping)
```

```
#Encoding the numeric column
def  encode_numeric_zscore(df, name, mean=None, sd=None):
  if mean is None:
    mean= df[name].mean()

  if sd is None:
    sd=df[name].std()

  df[name] = (df[name] - mean) / sd
```

STEP 5 Training Testing Spliting

Copying the 'evil' column from the DataFrame into a new variable 'y

```
y=df[['evil']].copy()
```

Selecting the 'timestamp', 'processId', 'threadId', 'eventId', and 'returnValue' columns for feature selection

```
feature_selection=['timestamp','processId','threadId','eventId','returnValue']
```

```
X=df[feature_selection].copy()
```

Splitinng the data into training and testing sets using a test size of 0.22 and a random state of 350

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.22, random_state=350)
```

Training a Random Forest Classifier on the training data and save the model using 'joblib'.

```
from sklearn.ensemble import RandomForestClassifier

RF_classifier = RandomForestClassifier(n_estimators = 40, criterion = 'entropy', max_leaf_nodes=30, random_state = 350)
RF_classifier.fit(X_train, y_train)

# Save the model
joblib.dump(RF_classifier, "/content/RF_classifier.pkl")

```

```
from sklearn.metrics import classification_report, confusion_matrix

y_predict_train = RF_classifier.predict(X_train)
y_predict_train
#cm = confusion_matrix(y_train, y_predict_train)
#sns.heatmap(cm, annot=True)
```

```
# Predicting the Test set results
y_predict_test = RF_classifier.predict(X_test)
#cm = confusion_matrix(y_test, y_predict_test)
#sns.heatmap(cm, annot=True)
```

```
print(classification_report(y_test, y_predict_test))
```

Training a Gradient Boosting Classifier on the training data and save the model using 'joblib

```
from sklearn.ensemble import GradientBoostingClassifier

GBDT_classifier = GradientBoostingClassifier()
GBDT_classifier.fit(X_train, y_train)

# Save the model
joblib.dump(GBDT_classifier, "/content/GBDT_classifier.pkl")

```

```
from sklearn.metrics import classification_report, confusion_matrix

y_predict_train = GBDT_classifier.predict(X_train)
y_predict_train
#cm = confusion_matrix(y_train, y_predict_train)
#sns.heatmap(cm, annot=True)
```

```
# Predicting the Test set results
y_predict_test = GBDT_classifier.predict(X_test)
#cm = confusion_matrix(y_test, y_predict_test)
#sns.heatmap(cm, annot=True)
```

```
print(classification_report(y_test, y_predict_test))
```

Training an XGBoost Classifier on the training data and save the model using 'joblib'.

```
from xgboost import XGBClassifier

XGB_classifier = XGBClassifier()
XGB_classifier.fit(X_train, y_train)

# Save the model
joblib.dump(XGB_classifier, "/content/XGB_classifier.pkl")

```

```
['/content/XGB_classifier.pkl']
```

```
[ ]
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score

y_predict_train = XGB_classifier.predict(X_train)
y_predict_train
#cm = confusion_matrix(y_train, y_predict_train)
#sns.heatmap(cm, annot=True)
```

```
array([0, 0, 1, ..., 0, 0, 0])
```

```
[ ]
# Predicting the Test set results
y_predict_test = XGB_classifier.predict(X_test)
#cm = confusion_matrix(y_test, y_predict_test)
#sns.heatmap(cm, annot=True)
```

```
[ ]
print(classification_report(y_test, y_predict_test))
```

Training a K-Nearest Neighbors Classifier on the training data and save the model using 'joblib'

```
[ ]
from sklearn.neighbors import KNeighborsClassifier

neigh = KNeighborsClassifier()
neigh.fit(X_train, y_train)

# Save the model
joblib.dump(neigh, "/content/KNN_classifier.pkl")

```

```
/usr/local/lib/python3.10/dist-packages/sklearn/neighbors/_classification.py:215: DataConversionWarning: A column-vector y was
  return self._fit(X, y)
['/content/KNN_classifier.pkl']
```

```
[ ]
from sklearn.metrics import classification_report, confusion_matrix

y_predict_train = neigh.predict(X_train)
y_predict_train


```

```
array([0, 0, 1, ..., 0, 0, 0])
```

```
[ ]
y_predict_test = neigh.predict(X_test)
```

```
[ ]
print(classification_report(y_test, y_predict_test))
```

- Training a Naive Bayes Classifier on the training data and save the model using 'joblib'

```
from sklearn.naive_bayes import GaussianNB

neigh = GaussianNB()
neigh.fit(X_train, y_train)

# Save the model
joblib.dump(neigh, "/content/GaussianNB_classifier.pkl")


```

```
/usr/local/lib/python3.10/dist-packages/sklearn/utils/validation.py:1143: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_s
  y = column_or_1d(y, warn=True)
['/content/GaussianNB_classifier.pkl']
```

```
from sklearn.metrics import classification_report, confusion_matrix

y_predict_train = neigh.predict(X_train)
y_predict_train

```

```
array([0, 0, 1, ..., 0, 0, 0])
```

```
y_predict_test = neigh.predict(X_test)
```

```
print(classification_report(y_test, y_predict_test))
```

Training a Decision Tree Classifier on the training data and save the model using 'joblib'

```
from sklearn.tree import DecisionTreeClassifier

DT_classifier = DecisionTreeClassifier(max_leaf_nodes=30, random_state = 350)
DT_classifier.fit(X_train, y_train)

# Save the model
joblib.dump(DT_classifier, "/content/DT_classifier.pkl")

```

```
['/content/DT_classifier.pkl']
```

```
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score

y_predict_train = DT_classifier.predict(X_train)
y_predict_train
#cm = confusion_matrix(y_train, y_predict_train)
#sns.heatmap(cm, annot=True)
```

```
array([0, 0, 1, ..., 0, 0, 0])
```

```
# Predicting the Test set results
y_predict_test = DT_classifier.predict(X_test)
#cm = confusion_matrix(y_test, y_predict_test)
#sns.heatmap(cm, annot=True)
```

```
print(classification_report(y_test, y_predict_test))
```

Training an AdaBoost Classifier on the training data and save the model using 'joblib'

```
from sklearn.ensemble import AdaBoostClassifier

AB_classifier = AdaBoostClassifier(n_estimators = 40,  random_state = 350)
AB_classifier.fit(X_train, y_train)

# Save the model
joblib.dump(AB_classifier, "/content/AB_classifier.pkl")

```

```
/usr/local/lib/python3.10/dist-packages/sklearn/utils/validation.py:1143: DataConversionWarning: A column-vector y was passed wh
  y = column_or_1d(y, warn=True)
['/content/AB_classifier.pkl']
```

```
from sklearn.metrics import classification_report, confusion_matrix

y_predict_train = AB_classifier.predict(X_train)
y_predict_train
```

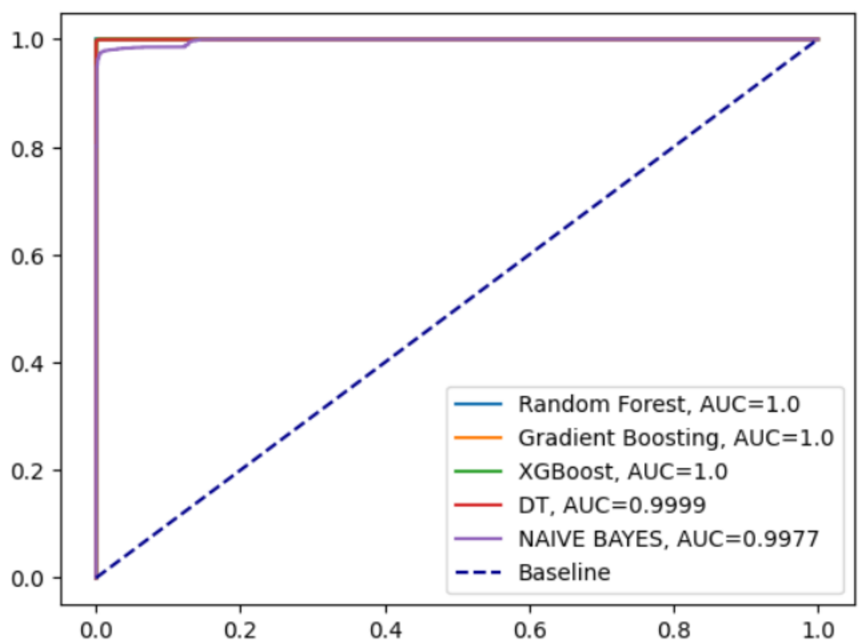
```
array([0, 0, 1, ..., 0, 0, 0])
```

```
y_predict_test = AB_classifier.predict(X_test)
```

```
print(classification_report(y_test, y_predict_test))
```

Results and Comparison

Plotting ROC curves for the Random Forest, Gradient Boosting, XGBoost, Decision Tree, and Naive Bayes classifiers and calculating the AUC for each



6. Troubleshooting

If you encounter issues while running the project, refer to the following solutions:

- **Data Loading Issues**: Make sure the BETH dataset is correctly uploaded and the path specified is correct.
- **Library Import Errors**: Ensure that all required Python libraries are correctly installed.
- **Model Training Errors**: Verify that the data is correctly preprocessed and split into training and testing sets before training the models.

7. Code Documentation

Each code block in the notebook is documented with comments to explain the purpose of the code. Further, markdown cells are used to provide detailed explanations of the steps being performed.

8. Contact Information

For any further queries or issues, please contact the project developer.

10. Version History

This initial version of the project includes the implementation of seven machine learning models and their evaluation.