Botnet Detection in IoT Devices using Gradient and Ada Boosting Algorithm

MSc Research Project
Cybersecurity

Sririshi Veeranam Shanmugam
Student ID: x21167672

School of Computing
National College of Ireland

Supervisor: Michael Pantridge

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Sririshi Veeranam Shanmugam
Student ID:	X21167672
Programme:	Cybersecurity
Year:	2022
Module:	MSc Research Project
Supervisor:	Michael Pantridge
Submission Due Date:	15/12/2018
Project Title:	Botnet Detection in IoT Devices using Gradient Boosting and Ada Boosting Algorithm
Word Count:	672
Page Count:	10

Signature:	Sririshi Veeranam Shanmugam
Date:	

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.	

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

Botnet Detection in IoT Devices using Gradient and Ada Boosting Algorithm

Sririshi Veeranam Shanmugam
X21167672

1 Introduction

The setting guide provides a detailed account of the whole setup procedure. Specifications for the hardware and software used in the study titled "Botnet Detection in IoT Devices using Gradient and Ada Boosting Algorithm" are provided. Using machine learningmodels like Gradient boosting and the Ada Boosting Algorithm, this research aims to achieve high precision.

2 System Specification

Google Collaboratory, often known as Colab, a cloud-based platform, hosted this project. Both graphical processing units and tensor processing units are welcome in the shared workspace. Bisong (2019)

2.1 Hardware

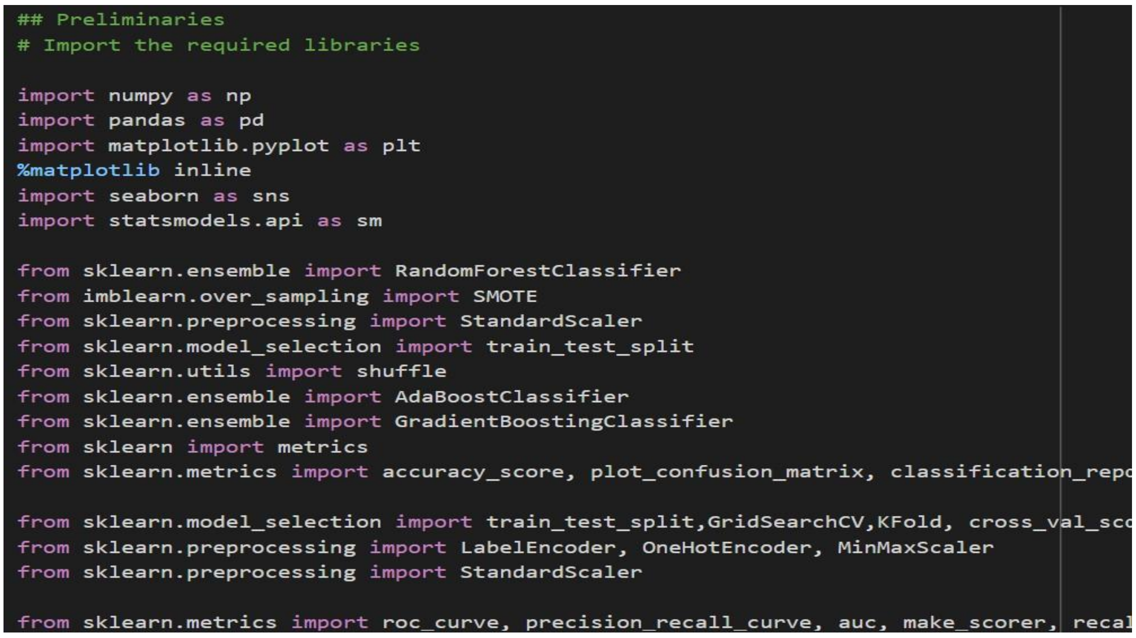
- Google Colab: 2vCPU @ 2.2GHz
- The GPU Instance was 250GB
- The RAM was 13 GB
- The Disk Space was 32GB

2.2 Software

In order to put the project into action, the Python programming language was utilized. Python was used to carry out all of the operations necessary for the pre-processing stage, including cleaning, encoding, dimension reduction implementation, and evaluation.

3 Importing Libraries

The cloud platform already has pre-defined versions of several of the necessary libraries. When it was necessary, the import of the other relevant libraries was performed. Importing the necessary libraries is the task at hand for this stage.



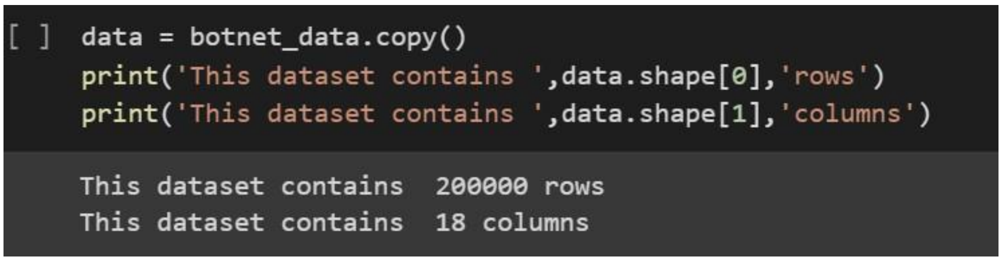
```
## Preliminaries
# Import the required libraries

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import statsmodels.api as sm

from sklearn.ensemble import RandomForestClassifier
from imblearn.over_sampling import SMOTE
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.utils import shuffle
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn import metrics
from sklearn.metrics import accuracy_score, plot_confusion_matrix, classification_repo

from sklearn.model_selection import train_test_split,GridSearchCV,KFold, cross_val_sco
from sklearn.preprocessing import LabelEncoder, OneHotEncoder, MinMaxScaler
from sklearn.preprocessing import StandardScaler

from sklearn.metrics import roc_curve, precision_recall_curve, auc, make_scorer, recal
```

Figure 1: Importing Libraries

4 Data Extraction

4.1 Importing Files

The next phase involves loading the data set from the hard drive and loading the file from the downloaded directory.

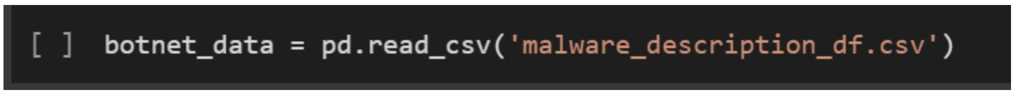


```
[ ]  data = botnet_data.copy()
     print('This dataset contains ',data.shape[0],'rows')
     print('This dataset contains ',data.shape[1],'columns')

     This dataset contains  200000 rows
     This dataset contains  18 columns
```

Figure 2: Importing Data

4.2 Set Path

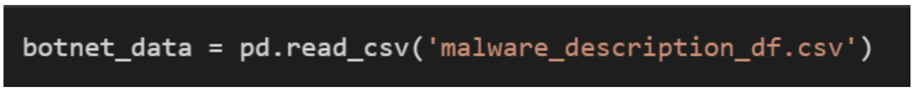
Here, we specify the location of the dataset and retrieve the information.



```
[ ]  botnet_data = pd.read_csv('malware_description_df.csv')
```

Figure 3: Working directory path

4.3 Reading the data

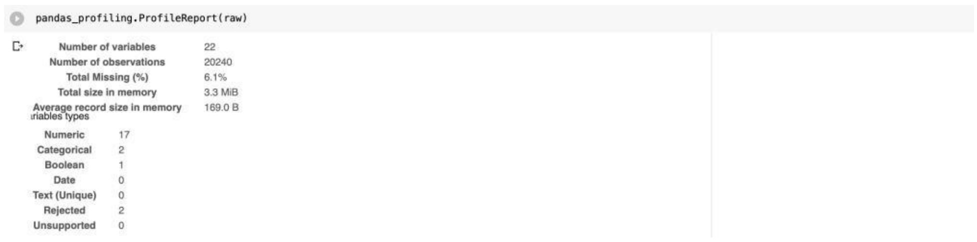


```
botnet_data = pd.read_csv('malware_description_df.csv')
```

Figure 4: Reading Data

5 Exploratory Data Analysis (EDA)

Python's pandas profiling was used for the exploratory data analysis. Pandas profiling is a single line of code that unlocks new insights from your data. It examines the data and generates a report in HTML format detailing any discrepancies, outliers, class balance, correlations, etc.



```
pandas_profiling.ProfileReport(raw)
```

Number of variables	22
Number of observations	20240
Total Missing (%)	6.1%
Total size in memory	3.3 MiB
Average record size in memory	169.0 B

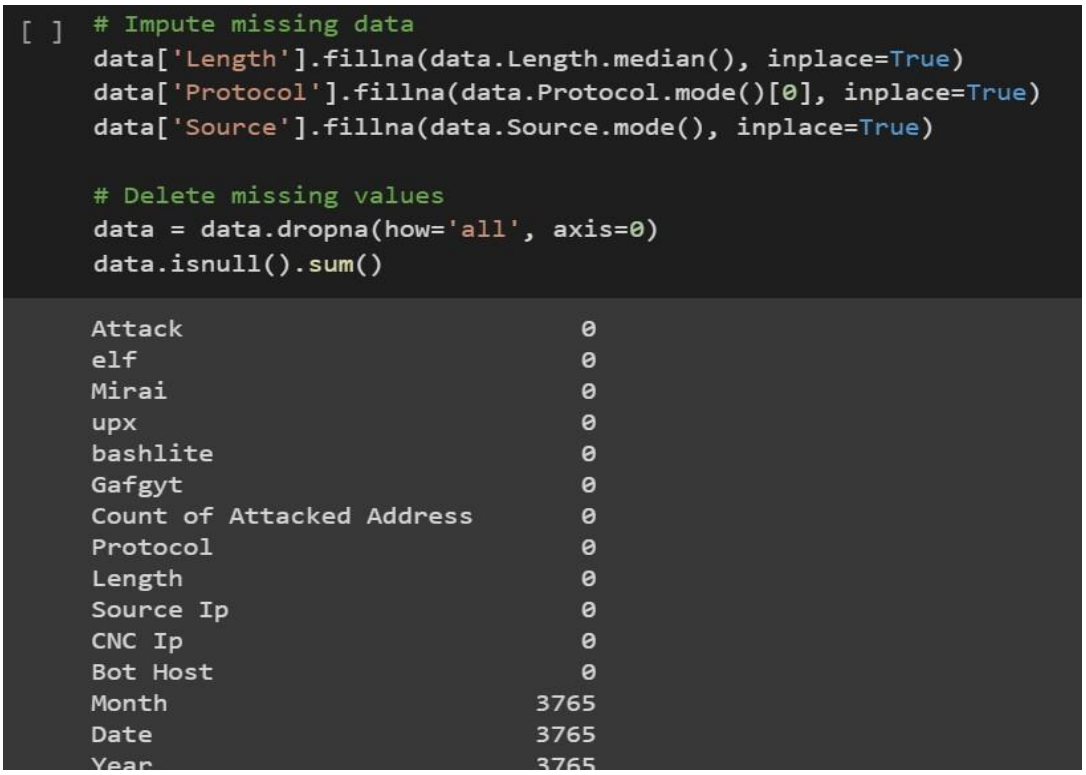
ıriables types

Numeric	17
Categorical	2
Boolean	1
Date	0
Text (Unique)	0
Rejected	2
Unsupported	0

Figure 5: Exploratory Data Analysis

5.1 Removing null values

At this point in the process, the null values in the raw dataset are going to be removed from the database. This will result in an enhancement in the quality of the dataset, which, in turn, will help deliver findings that are more accurate.



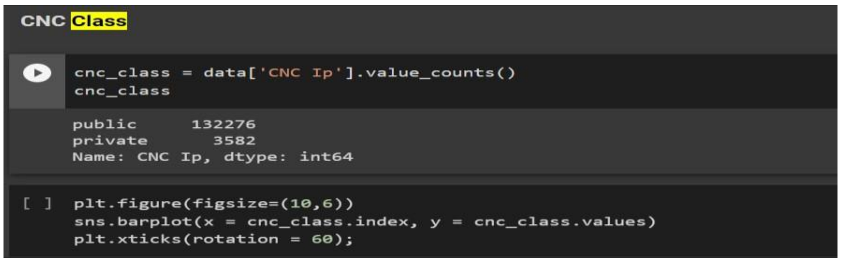
```
[ ]  # Impute missing data
     data['Length'].fillna(data.Length.median(), inplace=True)
     data['Protocol'].fillna(data.Protocol.mode()[0], inplace=True)
     data['Source'].fillna(data.Source.mode(), inplace=True)

     # Delete missing values
     data = data.dropna(how='all', axis=0)
     data.isnull().sum()
```

```
Attack                          0
elf                             0
Mirai                           0
upx                             0
bashlite                        0
Gafgyt                          0
Count of Attacked Address       0
Protocol                        0
Length                          0
Source Ip                       0
CNC Ip                          0
Bot Host                        0
Month                        3765
Date                         3765
Year                         3765
```

Figure 6: Removing null values

5.2 Checking Class Imbalance

Under sampling or oversampling may occur depending on the data, however for optimal results the class size should be about right. The distribution of abilities was typical of the class in this data set.



CNC Class

```
cnc_class = data['CNC Ip'].value_counts()
cnc_class
```

```
public      132276
private       3582
Name: CNC Ip, dtype: int64
```

```
[ ]  plt.figure(figsize=(10,6))
     sns.barplot(x = cnc_class.index, y = cnc_class.values)
     plt.xticks(rotation = 60);
```

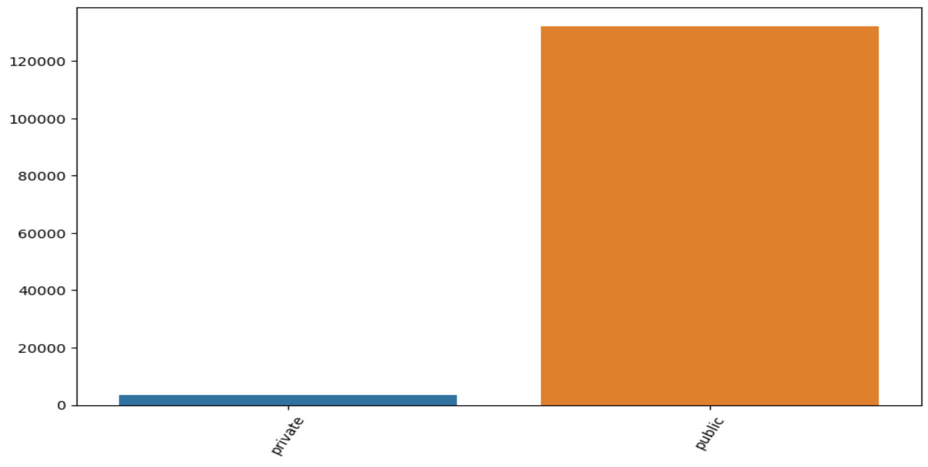
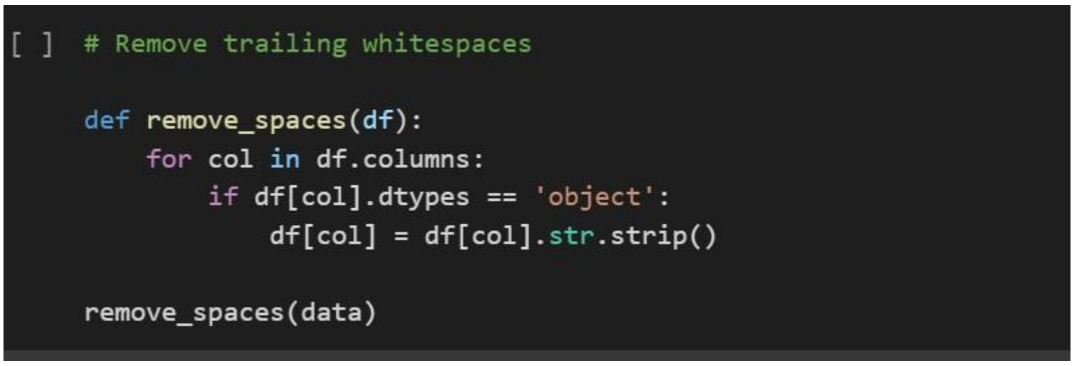


Figure 7: Class Imbalance check

5.3 Dropping the unwanted columns

The columns that include special characters and those that are deemed to be irrelevant are eliminated.



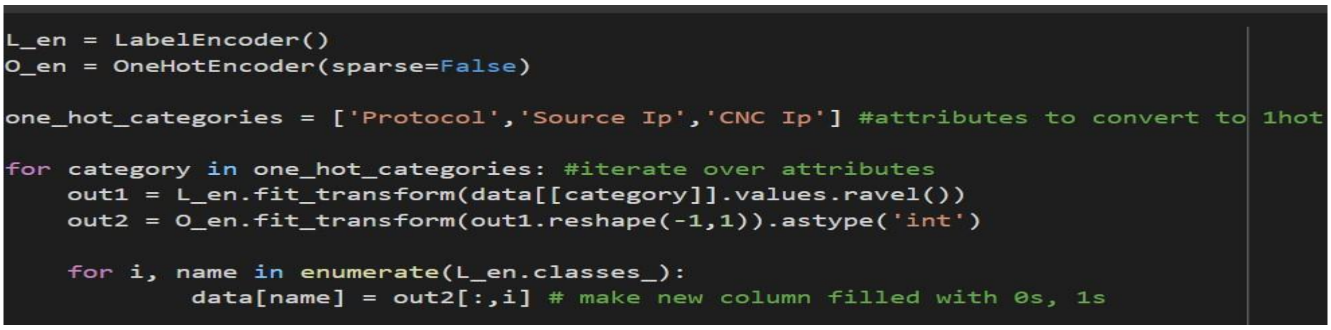
```
[ ]  # Remove trailing whitespaces

     def remove_spaces(df):
         for col in df.columns:
             if df[col].dtypes == 'object':
                 df[col] = df[col].str.strip()

     remove_spaces(data)
```

Figure 8: Removing unwanted columns

6 Pre-processing

6.1 Encoding the data

6.2

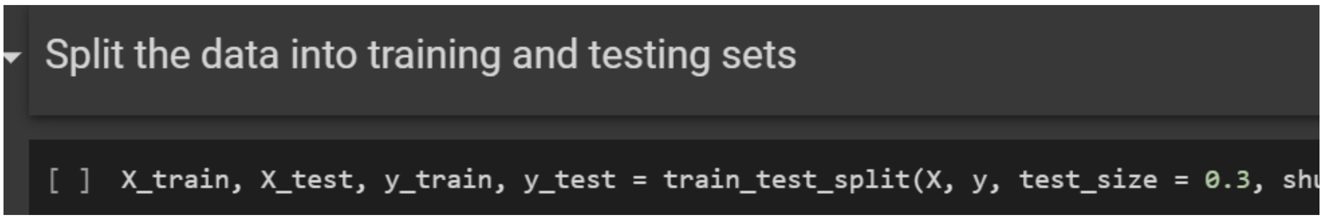


```
L_en = LabelEncoder()
O_en = OneHotEncoder(sparse=False)

one_hot_categories = ['Protocol','Source Ip','CNC Ip'] #attributes to convert to 1hot

for category in one_hot_categories: #iterate over attributes
    out1 = L_en.fit_transform(data[[category]].values.ravel())
    out2 = O_en.fit_transform(out1.reshape(-1,1)).astype('int')

    for i, name in enumerate(L_en.classes_):
            data[name] = out2[:,i] # make new column filled with 0s, 1s
```

Figure 10: Label Encoding

7 Training and testing dataset

In this step, a 50/50 split was made between the data set used for training and the one used for testing.



Split the data into training and testing sets

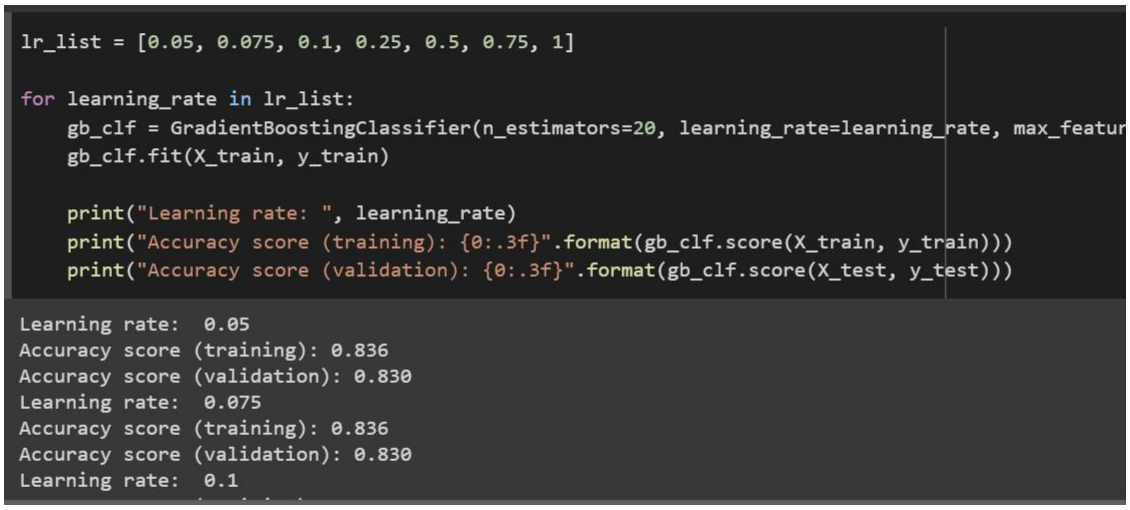
```
[ ] X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, shu
```

Figure 11: Training and testing data

8 Machine learning models

There are two machine learning models implemented in this project Like the Ada Boosting algorithm, the Gradient boosting algorithm.

8.1 Ada Boosting algorithm.

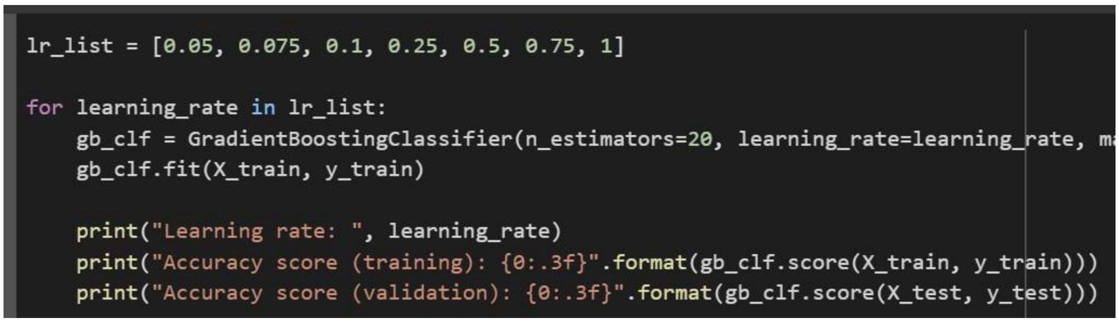


```
lr_list = [0.05, 0.075, 0.1, 0.25, 0.5, 0.75, 1]

for learning_rate in lr_list:
    gb_clf = GradientBoostingClassifier(n_estimators=20, learning_rate=learning_rate, max_featur
    gb_clf.fit(X_train, y_train)

    print("Learning rate: ", learning_rate)
    print("Accuracy score (training): {0:.3f}".format(gb_clf.score(X_train, y_train)))
    print("Accuracy score (validation): {0:.3f}".format(gb_clf.score(X_test, y_test)))
```

```
Learning rate:  0.05
Accuracy score (training): 0.836
Accuracy score (validation): 0.830
Learning rate:  0.075
Accuracy score (training): 0.836
Accuracy score (validation): 0.830
Learning rate:  0.1
```

Figure 13: Ada Boosting Algorithm

8.2 Gradient Boosting Algorithm



```
lr_list = [0.05, 0.075, 0.1, 0.25, 0.5, 0.75, 1]

for learning_rate in lr_list:
    gb_clf = GradientBoostingClassifier(n_estimators=20, learning_rate=learning_rate, m
    gb_clf.fit(X_train, y_train)

    print("Learning rate: ", learning_rate)
    print("Accuracy score (training): {0:.3f}".format(gb_clf.score(X_train, y_train)))
    print("Accuracy score (validation): {0:.3f}".format(gb_clf.score(X_test, y_test)))
```

Figure 14: Gradient Boosting Algorithm

References

Google colaboratory (no date) *Google Colab*. Google. Available at: https://colab.research.google.com/drive/1GJjmc_gAD2hRyVBXynWuvEP28FnysG76#scrollTo=7l8RpL-Nf5lH (Accessed: December 15, 2022).