National College of Ireland

Configuration Manual

MSc Research Project
MSc in Cyber Security

Ashutosh Datta Ganeshkar
x23142171

School of Computing
National College of Ireland

Supervisor: Mr. Eugene McLaughlin

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Ashutosh Datta Ganeshkar
Student ID:	x23142171
Programme:	MSc in Cyber Security **Year:** 2023-24
Module:	MSc Research Project
Lecturer:	Mr. Eugene McLaughlin
Submission Due Date:	12/08/2024
Project Title:	Flow-Based Network Intrusion Detection system using Hybrid Machine Learning Techniques
Word Count:	1245 **Page Count:** 11

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:	Ashutosh Datta Ganeshkar
Date:	12/08/2024

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):	

Table of Contents

Configuration Manual

Ashutosh Datta Ganeshkar
x23142171

1 Introduction

This Configuration handbook will provide the setup and equipments which were required to do this research project. It offers a detailed information on all the Machine Learning Models developed and Hybrid Ensemble Method/ Stacking classifier using over sampling Technique (SMOTE-Synthetic Minority Oversampling technique) on Flow-Based Network Data. The UNSW-NB15 Dataset was used in this research study, which has nine different types of attack in it. Hence, the configuration manual is very necessary, and it will include all the hardware and software which were required, the implementation methods which were developed for this project.

2 System Specification

2.1 Hardware Requirements

The below mentioned are the hardware required to perform this work:

Operating System: Windows 11
RAM: 8 GB
Processor: Intel(R) Core (TM) i5-8250U CPU @1.60GHz 1.80 GHz
Storage:225 GB SSD
System Type:64-bit operating system, x64-based processor

2.2 Software Requirement

The below are the details of software requirements to perform this work:

Python 3.6.3 version
Google Colab as my testing environment.
Google Drive to access and store my UNSW-NB15 dataset.

In my research project, the programming language which I have used is python. The **python version 3.6.3** was selected. The python programming language was selected because it is very simple to code, and it also helps us by providing large number of libraries and the frameworks which are designed for Machine Learning methods. The **Google colab** was selected for my **testing environment** as it is a cloud-based platform and we don't need to do any setup, we can start coding directly. It has free usage of GPU (Graphical Processing unit) and TPU (Tensor Processing Unit).

Mounted my google drive with **colab** to access the dataset.

```
#connecting colab with drive
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

3 Installation of Python Libraries

Here I will discuss the python libraries which I have installed into my environment using the import command:

NumPy

NumPy is the most common library used in python for numerical operation. Mathematical functions and Array operations are done using this library.

Pandas

Pandas are the most important library used for data analysis and manipulation. It offers a wide range of data structure technique to work with time series and numerical data.

Seaborn

The **seaborn** library is used for making statistical graphs.

Matplotlib

The **matplotlib** library is used for plotting the bar charts, graphs, histograms, scatterplot and pie charts.

Scikit-learn or sklearn

The **Scikit-learn** library is known as the most important library in python for complex data. It is supports Machine Learning models. This library includes different machine learning and statistical methods which includes classification, regression and clustering.

Imblearn

This library is used for handling the imbalanced datasets and it offers different resampling techniques, one of them which is used in my project is SMOTE (Synthetic Minority Oversampling Technique). (GitHub, 2020)

mlxtend

This library known as **mlxtend** is used for stacking ensemble model in which it uses different predictions from the base classifiers used. (parthmanchanda81, 2021)

```
#importing all required libraries
import pickle
import imblearn
import numpy as np
import pandas as pd
import seaborn as sns
from sklearn import metrics
from sklearn import ensemble
import matplotlib.pyplot as plt
from sklearn import preprocessing
from xgboost import XGBClassifier
from imblearn.over_sampling import SMOTE
from sklearn.preprocessing import LabelEncoder
from mlxtend.classifier import StackingClassifier
from sklearn.metrics import classification_report
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix,accuracy_score
from sklearn.metrics import precision_recall_fscore_support
from sklearn.ensemble import GradientBoostingClassifier,AdaBoostClassifier
from sklearn.feature_selection import mutual_info_classif,SelectPercentile
import warnings
warnings.filterwarnings('ignore')
```

4 Description of Dataset

The dataset which I decided for this research project is UNSW-NB15 dataset from Kaggle. The UNSW-NB15 dataset is stored in my google drive. The dataset is generated by the IXIA perfectstorm pgrogram in the cyber range lab of Australian centre for cyber security (ACCS). The dataset has nine different types of network attacks in it such as worms, Denial of Service (DoS), Fuzzers, shellcode, generic, backdoor, Reconnaissance, exploits, analysis. It has around 2,540,044 records stored in it. The 82,332 records are for testing set and the 175,341 records are for training set.

5 Data Pre-processing

The dataset was imported from the CSV file which is stored in my google drive and loaded into the pandas framework. The below figure displays the first 10 rows form the dataset.

```
[ ] #loading dataset into pandas dataframe
    dataset = pd.read_csv('/content/drive/My Drive/network_intrusion_detection/Data/UNSW_NB15_training-set.csv')
    dataset.head(10)
```

	id	dur	proto	service	state	spkts	dpkts	sbytes	dbytes	rate	...	ct_dst_sport_ltm	ct_dst_src_ltm	is_ftp_login	ct_ftp_cmd	ct_flw_http_mt
0	1	0.000011	udp	-	INT	2	0	496	0	90909.09020	...	1	2	0	0	
1	2	0.000008	udp	-	INT	2	0	1762	0	125000.00030	...	1	2	0	0	
2	3	0.000005	udp	-	INT	2	0	1068	0	200000.00510	...	1	3	0	0	
3	4	0.000006	udp	-	INT	2	0	900	0	166666.66080	...	1	3	0	0	
4	5	0.000010	udp	-	INT	2	0	2126	0	100000.00250	...	1	3	0	0	
5	6	0.000003	udp	-	INT	2	0	784	0	333333.32150	...	1	2	0	0	
6	7	0.000006	udp	-	INT	2	0	1960	0	166666.66080	...	1	2	0	0	
7	8	0.000028	udp	-	INT	2	0	1384	0	35714.28522	...	1	3	0	0	
8	9	0.000000	arp	-	INT	1	0	46	0	0.00000	...	2	2	0	0	
9	10	0.000000	arp	-	INT	1	0	46	0	0.00000	...	2	2	0	0	

10 rows × 45 columns

```
#shape of dataset
dataset.shape
```

```
(82332, 45)
```

The below figure displays the different attack categories and their count.

ashutosh_new_version.ipynb

File Edit View Insert Runtime Tools Help All changes saved

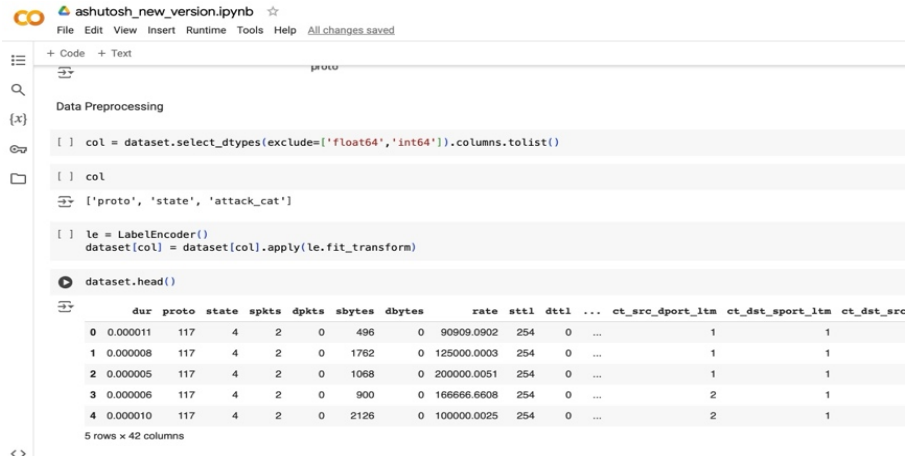
+ Code + Text

```
#value count of target class
dataset['attack_cat'].value_counts()
```

attack_cat	count
Normal	37000
Generic	18871
Exploits	11132
Fuzzers	6062
DoS	4089
Reconnaissance	3496
Backdoor	583
Shellcode	378
Worms	44

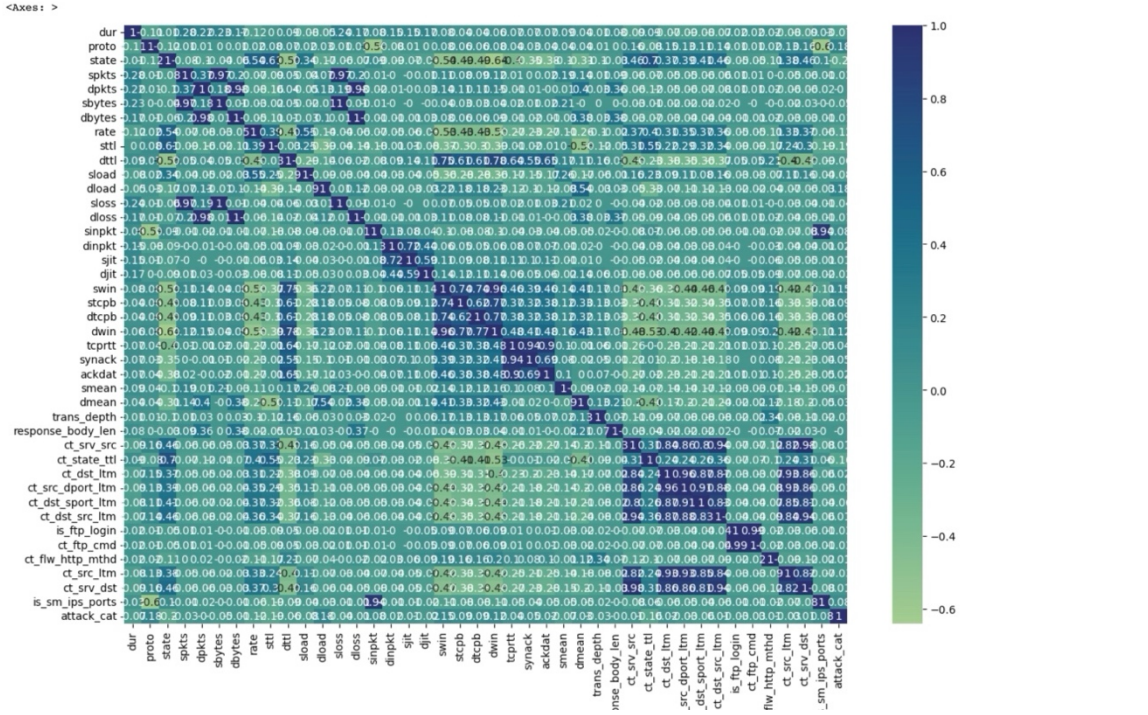
dtype: int64

The below figure displays the encoding method used to encode the categorial variables into numerical values using **"LabelEncoder"**



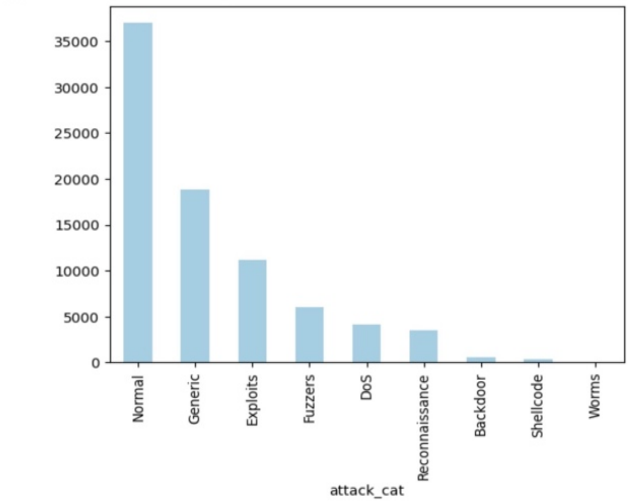
The below Figure is showing the **Correlation Matrix** using **Heatmap** for different features. The correlation between the **two features** is displayed using **-1 and 1**. The **1** states **positive correlation** and **-1** states **negative correlation**. If it is **0** then it is **no correlation**.

```
correlation = dataset.corr().round(2)
plt.figure(figsize = (15,10))
sns.heatmap(correlation, annot = True, cmap = 'crest')
```



The below shows the count of different attack categories. The data is imbalanced in this picture.

```
#value count of target class(from the below we identified the data is imbalanced)
dataset["attack_cat"].value_counts().plot(kind="bar", label='index', colormap='Paired')
```

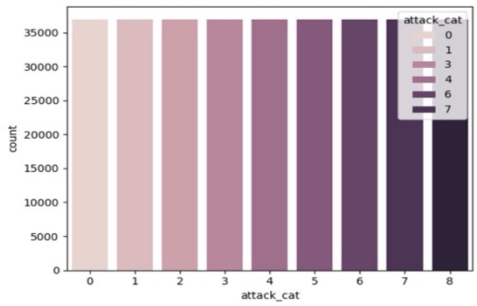


The below figure displays the data after applying the SMOTE (Synthetic Minority oversampling technique).

```
[ ] #using smote over sampling to handle data imbalanced.
    oversample = SMOTE()
    X, y = oversample.fit_resample(X,y)

    sns.countplot(x=y, hue=y)

    <Axes: xlabel='attack_cat', ylabel='count'>
```




```
[ ] X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.2, random_state=20, shuffle=True)
```

At last the in the above figure it shows that the data is split into training and testing, 80% is split for training the model and 20% is split for testing the model.

6 Model Training and Testing

6.1 AdaBoost Classifier

The below Figures shows about the adaboost classifier and displays the accuracy score given by the model, confusion matrix and classification report.

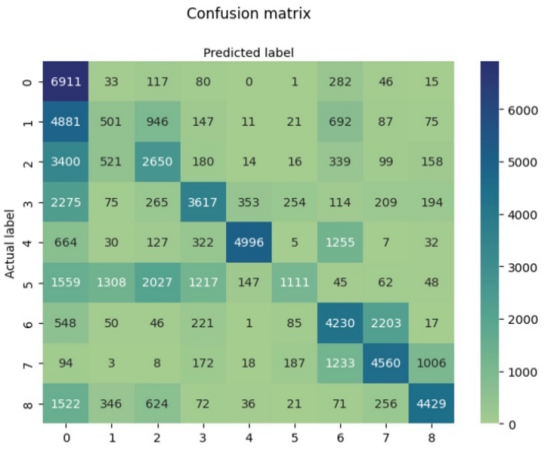
AdaBoost Classifier

```
#AdaBoost Classifier
adaboost_clf = AdaBoostClassifier(n_estimators=100)
adb = adaboost_clf
adb.fit(X_train,y_train)
y_pred = adb.predict(X_test)
```

```
[ ] #accuracy score
    print("Accuracy Score: ",accuracy_score(y_test,y_pred))

    Accuracy Score:  0.4955705705705706
```

```
#confusion Matrix
matrix =confusion_matrix(y_test, y_pred)
class_names=[0,1,2,3,4,5,6,7,8]
fig, ax = plt.subplots()
tick_marks = np.arange(len(class_names))
plt.xticks(tick_marks, class_names)
plt.yticks(tick_marks, class_names)
sns.heatmap(pd.DataFrame(matrix), annot=True, cmap="crest" ,fmt='g')
ax.xaxis.set_label_position("top")
plt.tight_layout()
plt.title('Confusion matrix', y=1.1)
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
plt.show()
```



```
[ ] #Classification Report
    print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.32      0.92      0.47      7485
           1       0.17      0.07      0.10      7361
           2       0.39      0.36      0.37      7377
           3       0.60      0.49      0.54      7356
           4       0.90      0.67      0.77      7438
           5       0.65      0.15      0.24      7524
           6       0.51      0.57      0.54      7401
           7       0.61      0.63      0.62      7281
           8       0.74      0.60      0.66      7377

    accuracy                           0.50     66600
   macro avg       0.54      0.50      0.48     66600
weighted avg       0.54      0.50      0.48     66600
```

6.2 XGBoost Classifier

The below Figures shows about the XGBoost classifier and displays the accuracy score given by the model, confusion matrix and classification report.

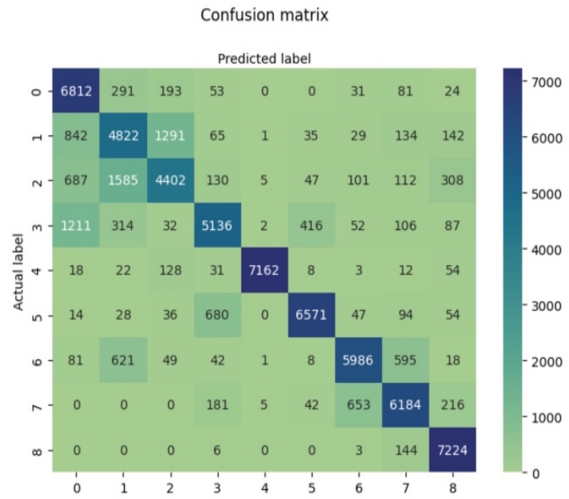
XGBoost Classifier

```
[ ] #XGBoost Classifier
    xgb_model = XGBClassifier(n_estimators=5)
    xgb = xgb_model
    xgb.fit(X_train,y_train)
    y_pred = xgb.predict(X_test)
```

```
[ ] #accuracy score
    print("Accuracy Score: ",accuracy_score(y_test,y_pred))
```

```
Accuracy Score:  0.8153003003003003
```

```
    #confusion Matrix
    matrix =confusion_matrix(y_test, y_pred)
    class_names=[0,1,2,3,4,5,6,7,8]
    fig, ax = plt.subplots()
    tick_marks = np.arange(len(class_names))
    plt.xticks(tick_marks, class_names)
    plt.yticks(tick_marks, class_names)
    sns.heatmap(pd.DataFrame(matrix), annot=True, cmap="crest" ,fmt='g')
    ax.xaxis.set_label_position("top")
    plt.tight_layout()
    plt.title('Confusion matrix', y=1.1)
    plt.ylabel('Actual label')
    plt.xlabel('Predicted label')
    plt.show()
```



```
#Classification Report
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.70      0.91      0.79      7485
           1       0.63      0.66      0.64      7361
           2       0.72      0.60      0.65      7377
           3       0.81      0.70      0.75      7356
           4       1.00      0.96      0.98      7438
           5       0.92      0.87      0.90      7524
           6       0.87      0.81      0.84      7401
           7       0.83      0.85      0.84      7281
           8       0.89      0.98      0.93      7377

    accuracy                           0.82     66600
   macro avg       0.82      0.81      0.81     66600
weighted avg       0.82      0.82      0.81     66600
```

6.3 Gradient Boosting Classifier

The below Figures shows about the Gradient Boosting classifier and displays the accuracy score given by the model, confusion matrix and classification report.

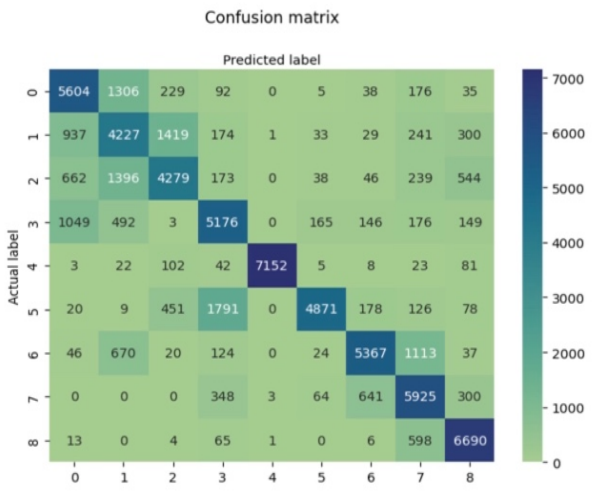
Gradient Boosting Classifier

```
#Gradient Boosting Classifier
gbc_model = GradientBoostingClassifier(n_estimators=10)
gbc = gbc_model
gbc.fit(X_train,y_train)
y_pred = gbc.predict(X_test)
```

```
#accuracy score
print("Accuracy Score: ",accuracy_score(y_test,y_pred))
```

```
Accuracy Score:  0.7401051051051051
```

```
#confusion Matrix
matrix =confusion_matrix(y_test, y_pred)
class_names=[0,1,2,3,4,5,6,7,8]
fig, ax = plt.subplots()
tick_marks = np.arange(len(class_names))
plt.xticks(tick_marks, class_names)
plt.yticks(tick_marks, class_names)
sns.heatmap(pd.DataFrame(matrix), annot=True, cmap="crest" ,fmt='g')
ax.xaxis.set_label_position("top")
plt.tight_layout()
plt.title('Confusion matrix', y=1.1)
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
plt.show()
```



```
#Classification Report
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.67      0.75      0.71      7485
           1       0.52      0.57      0.55      7361
           2       0.66      0.58      0.62      7377
           3       0.65      0.70      0.67      7356
           4       1.00      0.96      0.98      7438
           5       0.94      0.65      0.77      7524
           6       0.83      0.73      0.77      7401
           7       0.69      0.81      0.75      7281
           8       0.81      0.91      0.86      7377

    accuracy                           0.74     66600
   macro avg       0.75      0.74      0.74     66600
weighted avg       0.75      0.74      0.74     66600
```

6.4 Hybrid Ensemble Method/Stacking Classifier

The below Figures shows about the Hybrid Ensemble Method/Stacking classifier, the novel method in this research study. It displays the accuracy score given by the model, confusion matrix and classification report.

The base classifiers are XGBoost & AdaBoost, and the meta classifier is the Gradient Boosting Algorithm.

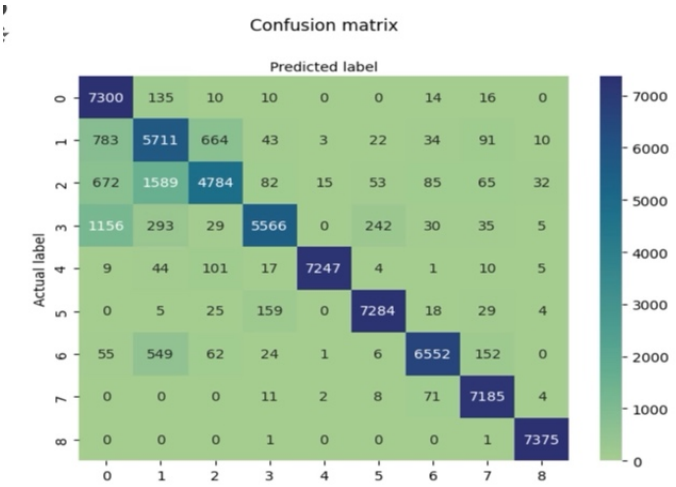
Hybrid Ensemble Model

```
#Stacking Classifier
stc_model = StackingClassifier(classifiers = [XGBClassifier(), AdaBoostClassifier()], meta_classifier=GradientBoostingClassifier())
stc = stc_model
stc.fit(X_train,y_train)
y_pred = stc.predict(X_test)
```

```
#accuracy score
print("Accuracy Score: ",accuracy_score(y_test,y_pred))
```

```
Accuracy Score:  0.885945945946
```

```
#confusion Matrix
matrix =confusion_matrix(y_test, y_pred)
class_names=[0,1,2,3,4,5,6,7,8]
fig, ax = plt.subplots()
tick_marks = np.arange(len(class_names))
plt.xticks(tick_marks, class_names)
plt.yticks(tick_marks, class_names)
sns.heatmap(pd.DataFrame(matrix), annot=True, cmap="crest" ,fmt='g')
ax.xaxis.set_label_position("top")
plt.tight_layout()
plt.title('Confusion matrix', y=1.1)
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
plt.show()
```



```
#Classification Report
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.73      0.98      0.84      7485
           1       0.69      0.78      0.73      7361
           2       0.84      0.65      0.73      7377
           3       0.94      0.76      0.84      7356
           4       1.00      0.97      0.99      7438
           5       0.96      0.97      0.96      7524
           6       0.96      0.89      0.92      7401
           7       0.95      0.99      0.97      7281
           8       0.99      1.00      1.00      7377

    accuracy                           0.89     66600
   macro avg       0.90      0.89      0.89     66600
weighted avg       0.90      0.89      0.89     66600
```

References

GitHub. (2020). *scikit-learn-contrib/imbalanced-learn*. [online] Available at: https://github.com/scikit-learn-contrib/imbalanced-learn.

parthmanchanda81, G. (2021). *Libraries in Python*. [online] GeeksforGeeks. Available at: https://www.geeksforgeeks.org/libraries-in-python/.