National College of Ireland

Configuration Manual

MSc Research Project

MSc Cybersecurity

Tanmay Dharmaraj Shukla

Student ID: x22112421

School of Computing

National College of Ireland

Supervisor: Eugene McLaughlin

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Tanmay Dharmaraj Shukla
Student ID:	x22112421
Programme:	MSc Cybersecurity **Year:** 2023-2024
Module:	MSc Research Project
Lecturer:	Eugene McLaughlin
Submission Due Date:	25th April 2024
Project Title:	Unveiling the Power of CNNs with Attention for URL Phishing Detection.
Word Count:	940 **Page Count:** 13

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: Tanmay Dharmaraj Shukla

Date: 25th April 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):	

Configuration Manual

Tanmay Dharmaraj Shukla
x22112421

1 INTRODUCTION

This manual is created with the stepwise implementation of how the project implement worked and how the model is created. In addition to it information and process of used libraries and tools required to carry out for project implementation is mentioned. Furthermore, information regarding the specifications of the local machine is mentioned in the manual. Each model information is mentioned in this configure manual.

.

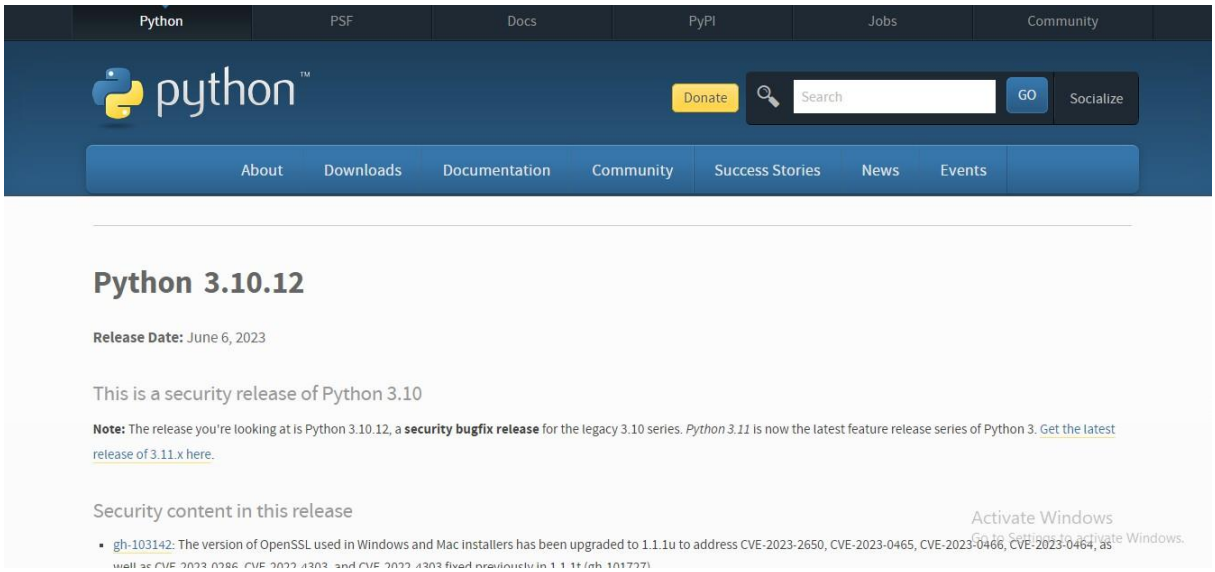
2 HARDWARE CONFIGURATION

- Operating system: Windows 11
- Processor: > 11th Gen Intel(R) Core(TM) i3-1115G4 @ 3.00GHz 3.00 GHz
- System Compatibility: 64-bit
- Hard Disk: 512GB
- RAM: 16 GB

3 SOFTWARE CONFIGURATIONS

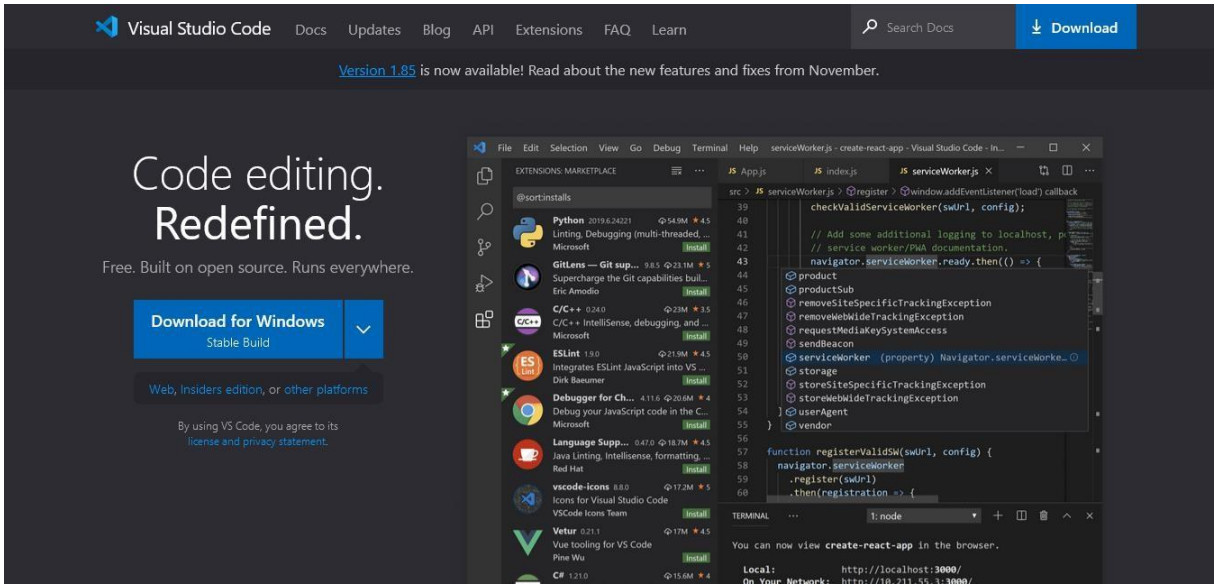
a) Python==3.10.12 :-

Python is an interpreted , high-level programming language . Its high-level built in data structures , combined with dynamic typing and dynamic binding , make it very attractive for Rapid Application Development , as well as for use as a scripting or glue language to connect existing components together. Its language concept and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.



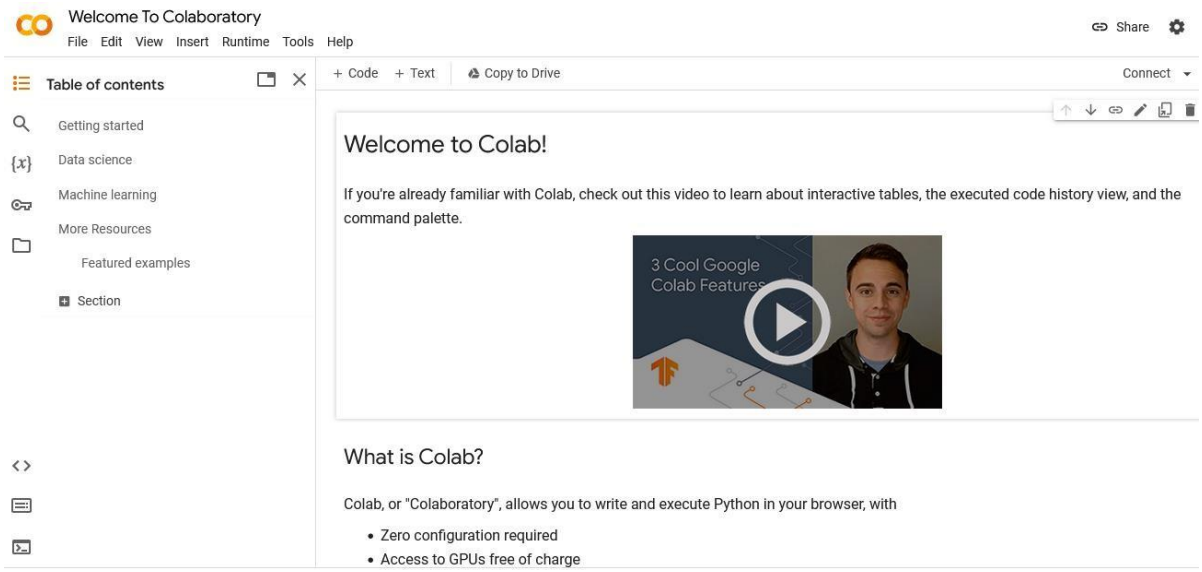
b) Visual Studio Code:-

Visual Studio Code (famously known as VS Code) is a free open source text editor by Microsoft. VS Code is available for Windows, Linux, and macOS.(Visual Studio Code - Code Editing. Redefined,)



c) Google Colab:-

Colaboratory, or "Colab" for short, is a product from Google Research. Colab allows anybody to write and execute arbitrary python code through the browser, and is especially well suited to machine learning, data analysis and education. More technically, Colab is a hosted Jupyter notebook service that requires no setup to use, while providing access free of charge to computing resources including GPUs.(colab.google,)



4 Libraries Configuration

1. Pandas
2. Numpy
3. Matplotlib
4. Seaborn
5. Plotly
6. Scikit-learn
7. Tensorflow
8. Flask
9. Keras

(*Best Python Libraries for Machine Learning - GeeksforGeeks*, 2023.)

5 Implementation

Implementation is divided into 3 modules which is mentioned below:

Module1: Visual Studio Code
First Loading Urls Dataset from each class,Extracting 19 Features and one Labelling Feature(target class) and last step is to save extracted features into csv file. i.e final_dataframe.csv.(*URL 2016 | Datasets | Research | Canadian Institute for Cybersecurity | UNB*, no date)

Module2: Next step is on google colab for training of model .

Saving the extracted features into csv file on drive followed by Importing & Installing all required libraries.
Dataset Loading using Pandas

Data Cleaning: null values are checked and unnecessary columns are dropped. We removed unnecessary data from the dataset, 2000 url we took from which 19+1 is feature extraction and 1 is label (target column).Then pre-processing is done where we are converting categorical data into numerical.
Data Visualization(EDA) : creating graphs and plots to understand data.
Splitting data (Features and target split in to and y) and we also split data for training 90% and 10 % for testing,we always train model with a higher volume of data,as we have used 1000 of each benign and phishing URLs.
This data is used to train ML and DL models.(Fit trains data and predict from test data)
Later Performance evaluation is done using the Confusion Matrix and Classification Report.

Module 3: Web application for detection of Phishing URL. There is web framework inside python here in this step two modules are merged.(Welcome to Flask — Flask Documentation (3.0.x),)

Step 1: Select the relevant dataset here we are using only two classes benign and phishing URLs.
Dataset Link :- https://www.unb.ca/cic/datasets/url-2016.html.

Step 2: After Installation of Visual Studio Code we have created new project and we have imported the necessary libraries for feature extraction and selected desired columns which are features mentioned in figure 1.

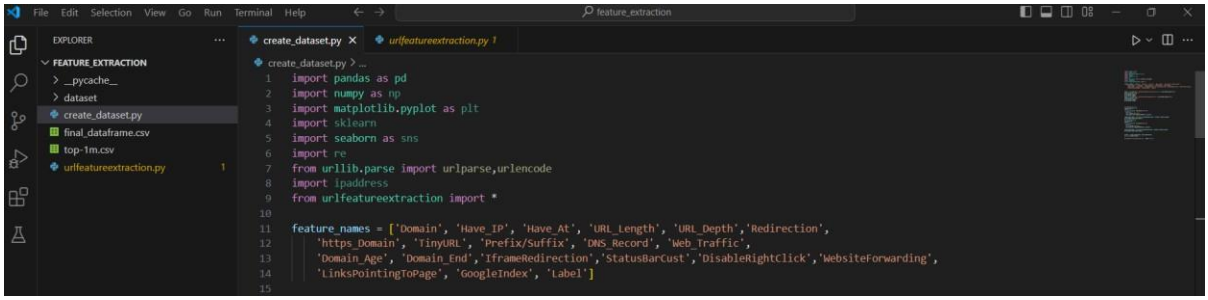


Fig. 1

Step 3: All the functions for feature extraction is in python where we are using regular expressions library which will extract features of NLP concept mentioned in figure 2 and 3.

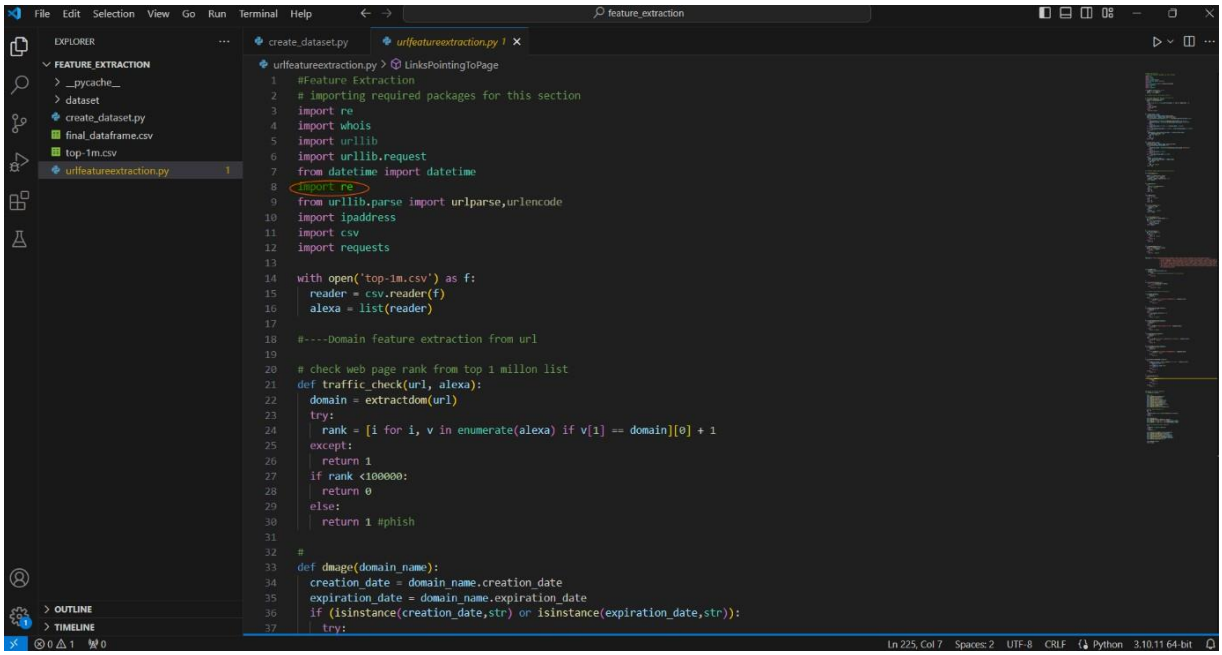


Fig. 2

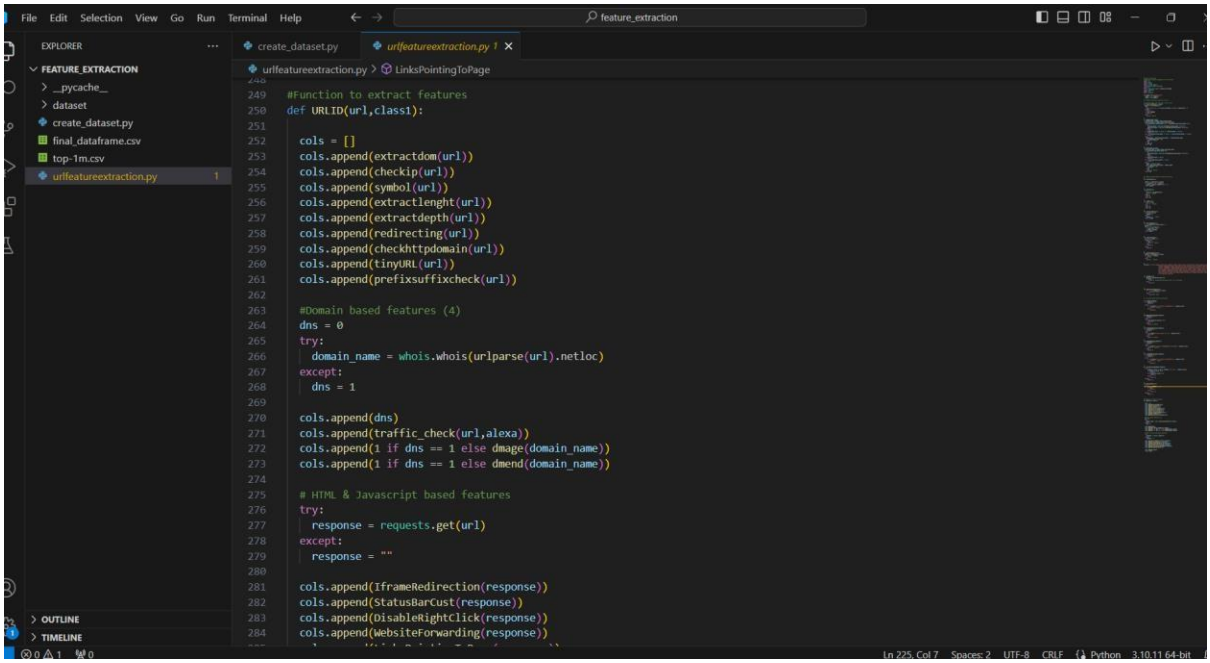


Fig. 3

Step 4.After running the create_dataset ,it will start checking each URL and later it will save the final data as mentioned in below image.

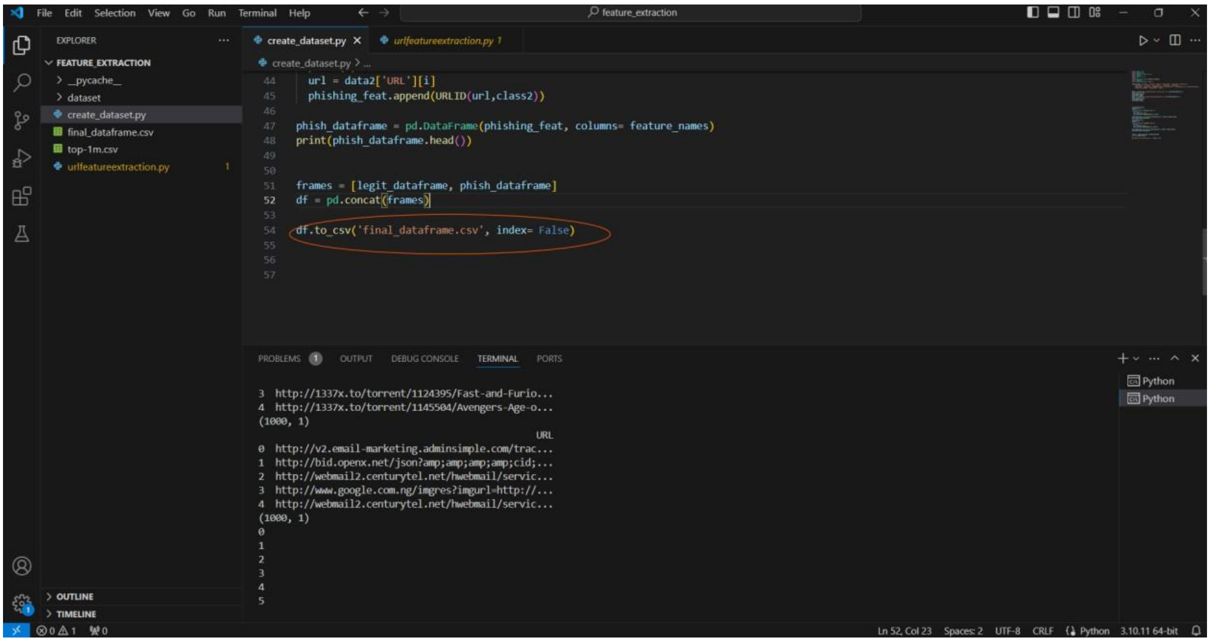
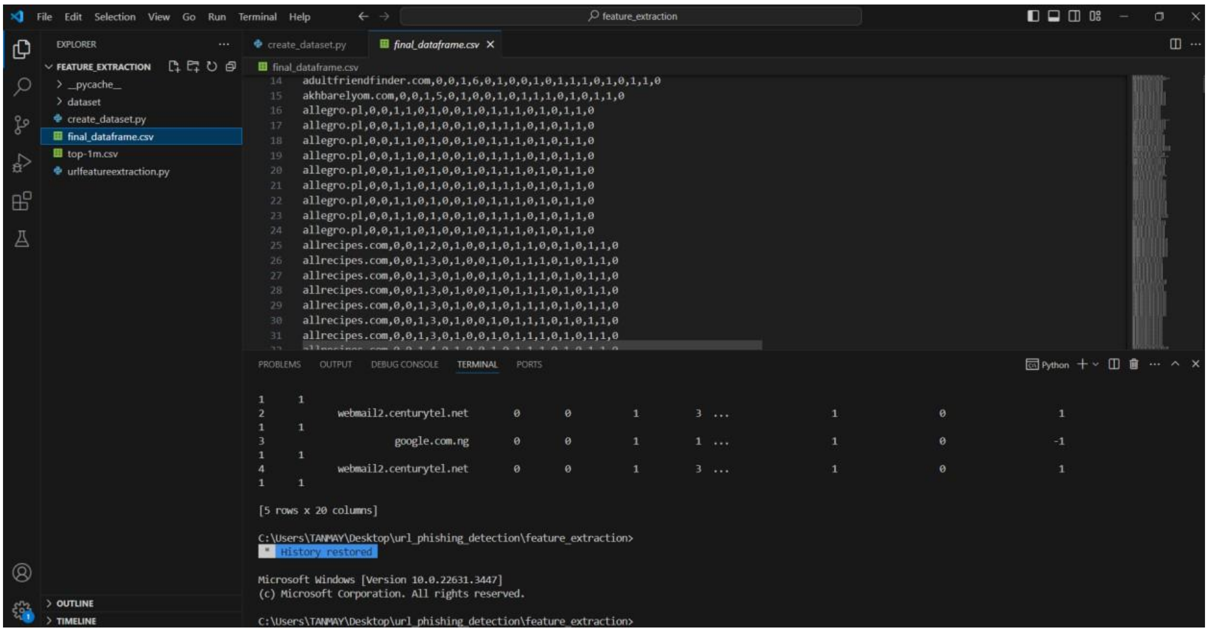


Fig. 4

In Below figure **Final feature extraction.csv file of visual studio code is mentioned below.** Which is final .csv file after feature extraction which will be used on google colab.



Step 5: All the below steps and image illustrates the work done in google colab which is explained in module 2 mentioned above. Like figure 5 states that we are Importing Libraries for visualization like plotly, sklearn, and seaborn. TensorFlow is for deep learning to speed up the process.

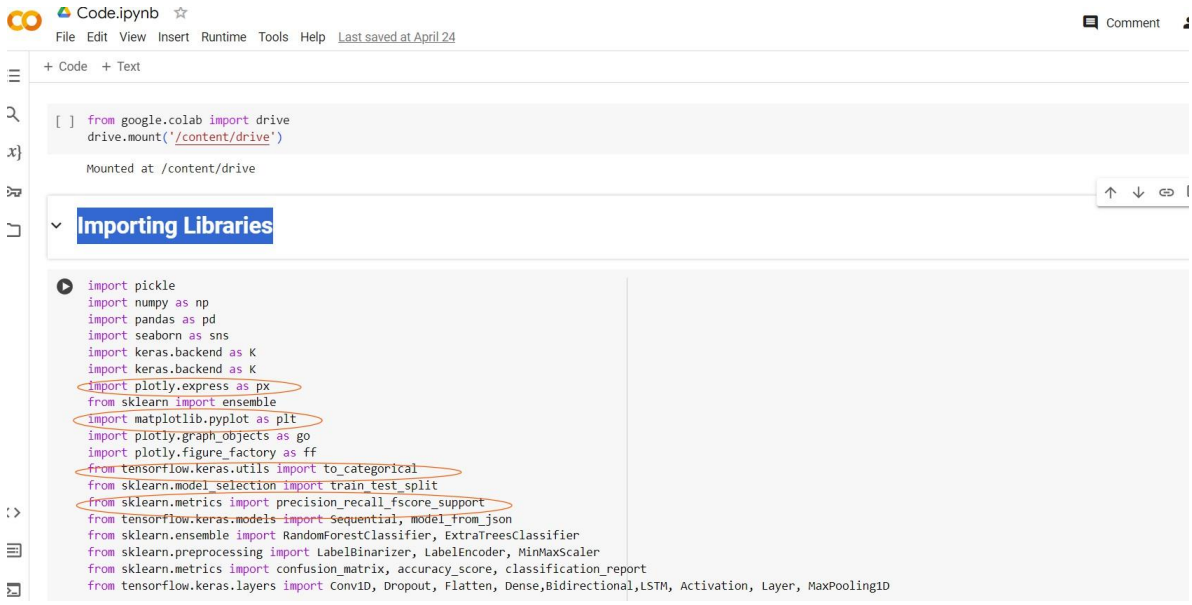


Fig. 5

Fig.6

	Have_IP	Have_At	URL_Length	URL_Depth	Redirection	https_Domain	TinyURL	Prefix/Suffix	DNS_Record	Web_Traffic	Domain_Age	Domain_End	IframeRedirection	StatusBarC
count	2000.0	2000.000000	2000.0	2000.000000	2000.00000	2000.0	2000.000000	2000.000000	2000.0	2000.000000	2000.0	2000.0	2000.000000	2000.000
mean	0.0	0.005500	1.0	2.782000	0.01250	1.0	0.042500	0.337000	1.0	0.596500	1.0	1.0	0.535000	0.459
std	0.0	0.073976	0.0	2.101591	0.11113	0.0	0.201777	0.472803	0.0	0.490722	0.0	0.0	0.498898	0.498
min	0.0	0.000000	1.0	0.000000	0.00000	1.0	0.000000	0.000000	1.0	0.000000	1.0	1.0	0.000000	0.000
25%	0.0	0.000000	1.0	1.000000	0.00000	1.0	0.000000	0.000000	1.0	0.000000	1.0	1.0	0.000000	0.000
50%	0.0	0.000000	1.0	2.000000	0.00000	1.0	0.000000	0.000000	1.0	1.000000	1.0	1.0	1.000000	0.000
75%	0.0	0.000000	1.0	4.000000	0.00000	1.0	0.000000	1.000000	1.0	1.000000	1.0	1.0	1.000000	1.000
max	0.0	1.000000	1.0	16.000000	1.00000	1.0	1.000000	1.000000	1.0	1.000000	1.0	1.0	1.000000	1.000

Fig. 7

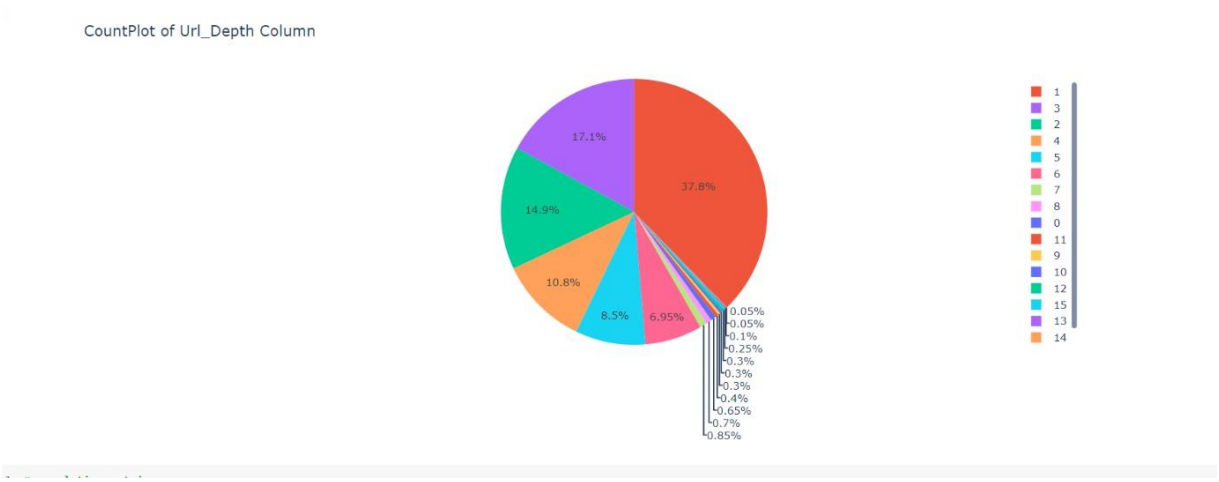


Fig . 8

```
[ ] #model feature importance
    fearture_name = X_train.columns.values
    model = ensemble.ExtraTreesClassifier()
    model.fit(X_train,y_train)
    #plot imp
    importance = model.feature_importances_
    std = np.std([tree.feature_importances_ for tree in model.estimators_],axis=0)
    indices = np.argsort(importance)[::-1][:5]
    plt.figure(figsize=(7,7))
    plt.title("Top 5 Important Features")
    plt.bar(range(len(indices)), importance[indices], color="lightgreen")
    plt.xticks(range(len(indices)), fearture_name[indices], rotation='vertical')
    plt.xlim([-1, len(indices)])
    plt.show()
```

Top 5 Important Features

Fig . 9

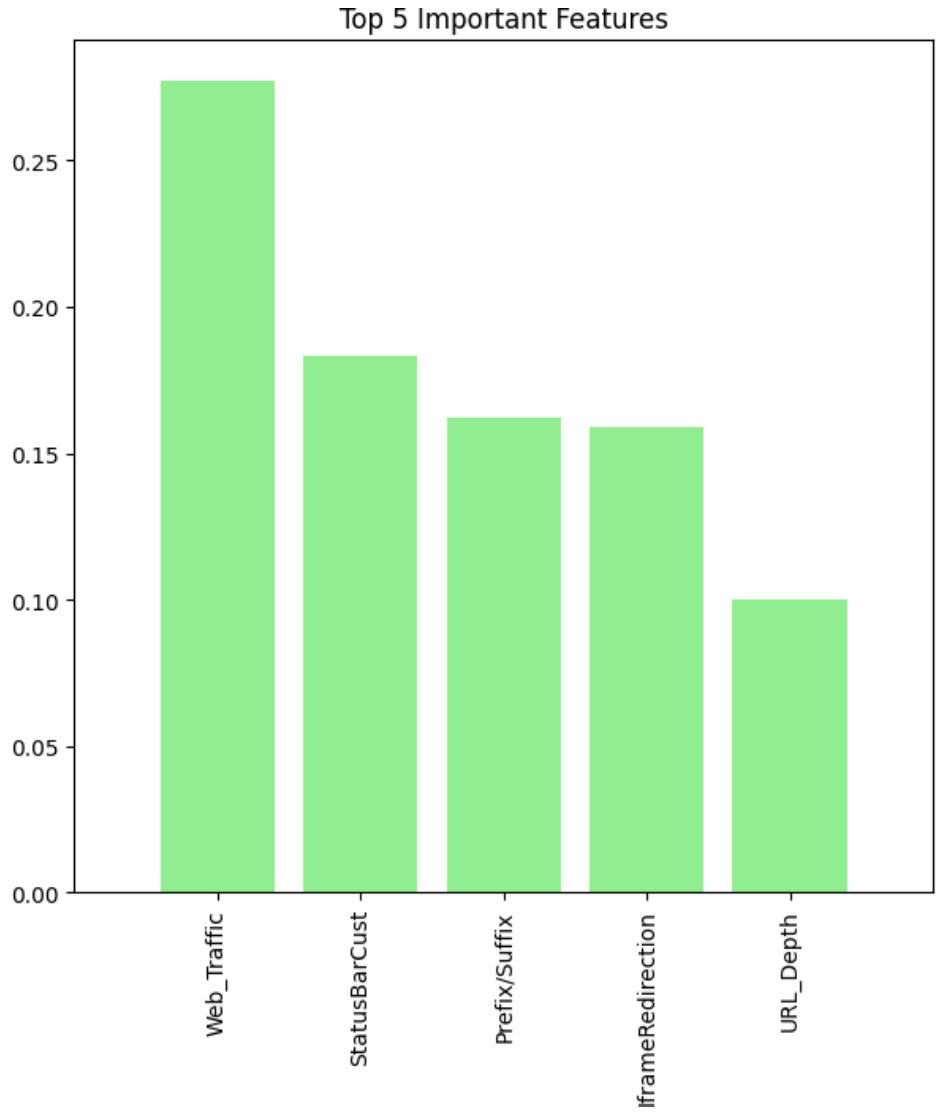
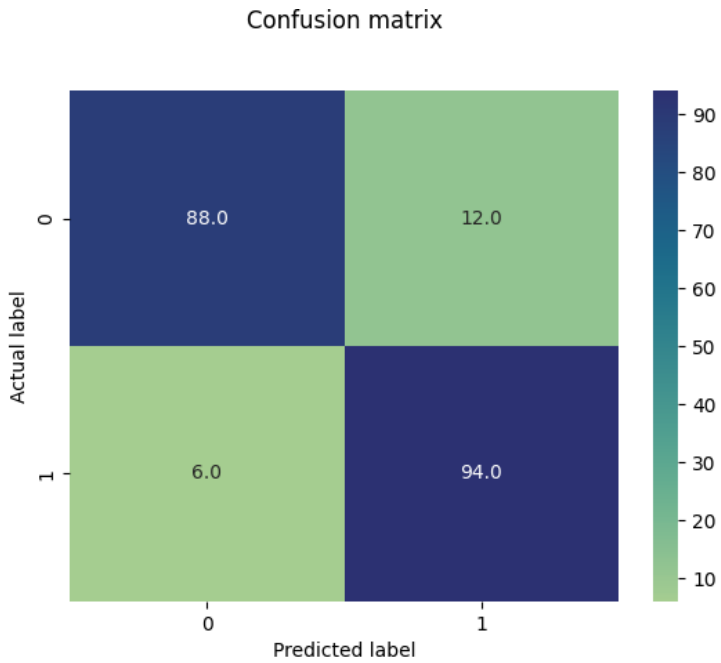


Fig. 10

Step 6:figure 9 and 10 tells us that we what are the impactful fature for detecting phishing URL which are correlated from 19 features.

Step 7: We get 91% accuracy using CNN model with attention mechanism which is evaluated using confusion matrix and classification report mentioned in below image.




```
[ ]  print(classification_report(yorg, ypred))
```

	precision	recall	f1-score	support
0	0.94	0.88	0.91	100
1	0.89	0.94	0.91	100
accuracy			0.91	200
macro avg	0.91	0.91	0.91	200
weighted avg	0.91	0.91	0.91	200

Step 8.We have analysed from the evaluation that CNN with attention mechanism is best model and we save it for further integration with flask web application.

```
history = model.fit(X_train, y_train, batch_size=32, epochs=5, validation_data=(X_test, y_test), verbose=1)
```

```
Epoch 1/5
57/57 [==============================] - 3s 14ms/step - loss: 0.6467 - accuracy: 0.6689 - val_loss: 0.4941 - val_accuracy: 0.8400
Epoch 2/5
57/57 [==============================] - 0s 8ms/step - loss: 0.3926 - accuracy: 0.8350 - val_loss: 0.2541 - val_accuracy: 0.8800
Epoch 3/5
57/57 [==============================] - 1s 11ms/step - loss: 0.2793 - accuracy: 0.8744 - val_loss: 0.2563 - val_accuracy: 0.8950
Epoch 4/5
57/57 [==============================] - 1s 13ms/step - loss: 0.2641 - accuracy: 0.8894 - val_loss: 0.2400 - val_accuracy: 0.8950
Epoch 5/5
57/57 [==============================] - 0s 8ms/step - loss: 0.2565 - accuracy: 0.8922 - val_loss: 0.1936 - val_accuracy: 0.9100
```

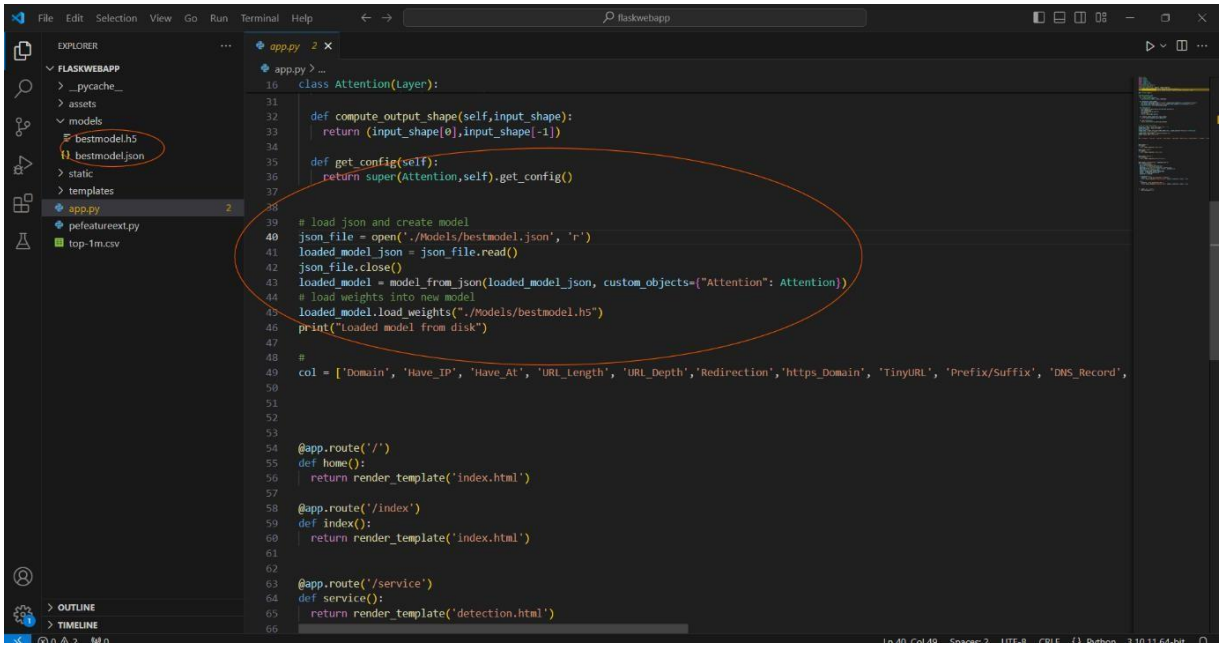
```
ypred = model.predict(X_test)
ypred = np.argmax(ypred,axis=1)
yorg = np.argmax(y_test,axis=1)
```

```
7/7 [==============================] - 0s 3ms/step
```

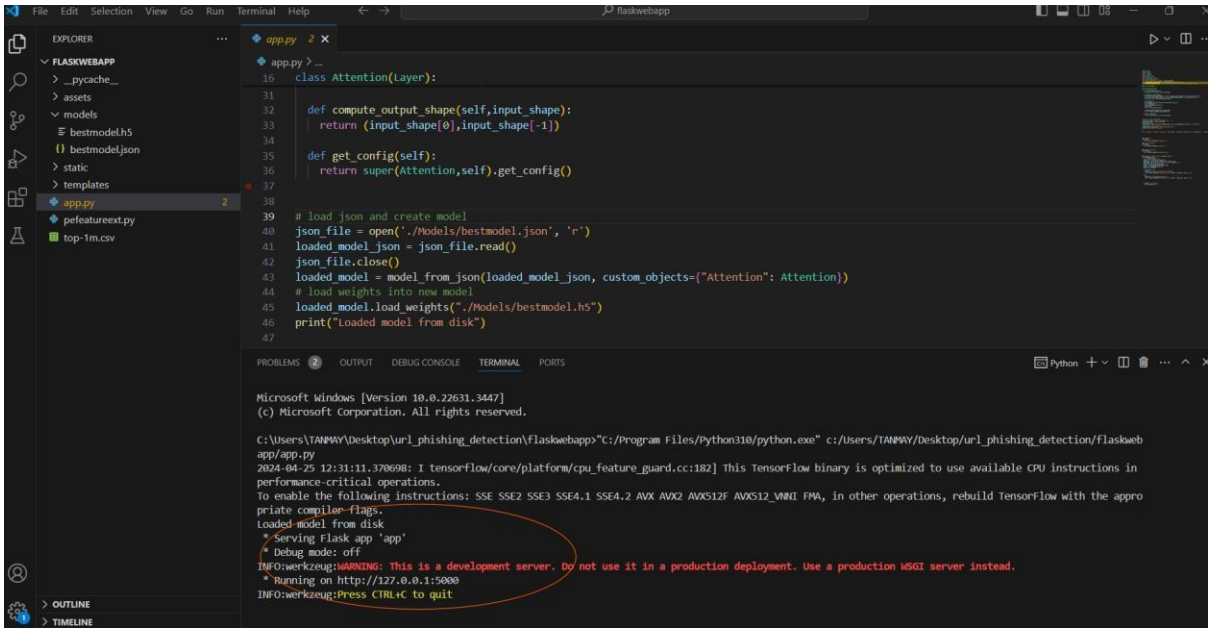
```
#Train and Validation Accuracy
plt.plot(history.history['accuracy'], 'b', marker='*', label='Train Accuracy')
plt.plot(history.history['val_accuracy'], 'g', marker='*', label='Val Accuracy')
plt.title('Train and Val Accuracy')
plt.legend()
plt.figure()

#Train and Validation Loss
plt.plot(history.history['loss'], 'b', marker='*', label='Train Loss')
plt.plot(history.history['val_loss'], 'g', marker='*', label='Val Loss')
plt.title('Train and Val Loss')
plt.legend()
plt.show()
```

Step 9: Web Application for testing the phishing URLs is run on visual studio code where we load the best model and integrate all 3 modules. Here in assets we have screenshot of web application, models have best model selected in google colab i.e. CNN with attention mechanism. Which focuses constantly on impactful layers features. Static is having design.

All this is shared in artefacts upload.



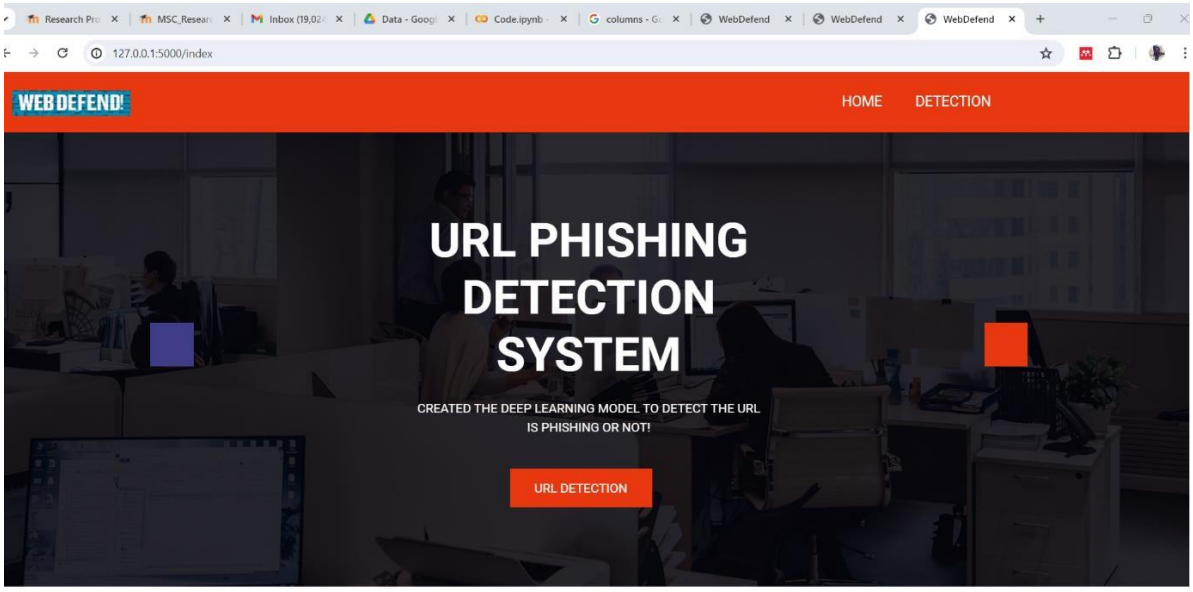
Step 10: Output of the web page once we receive the IP after running app.py.

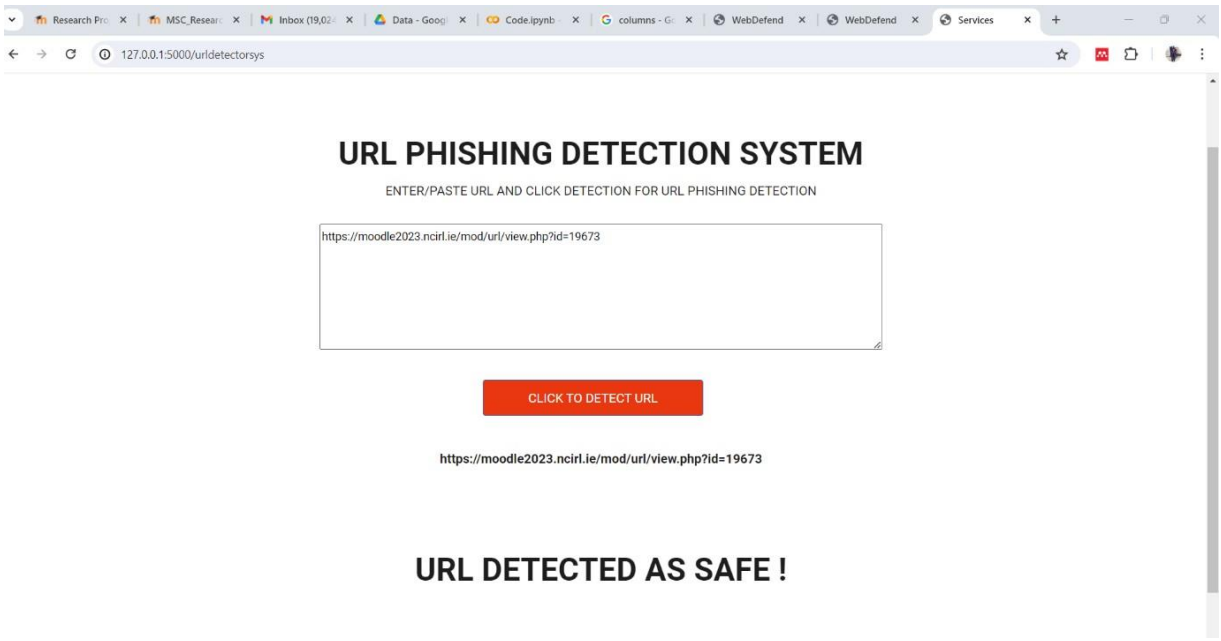


Step 11:

Copy URL from data set to identify whether its safe or phishing.

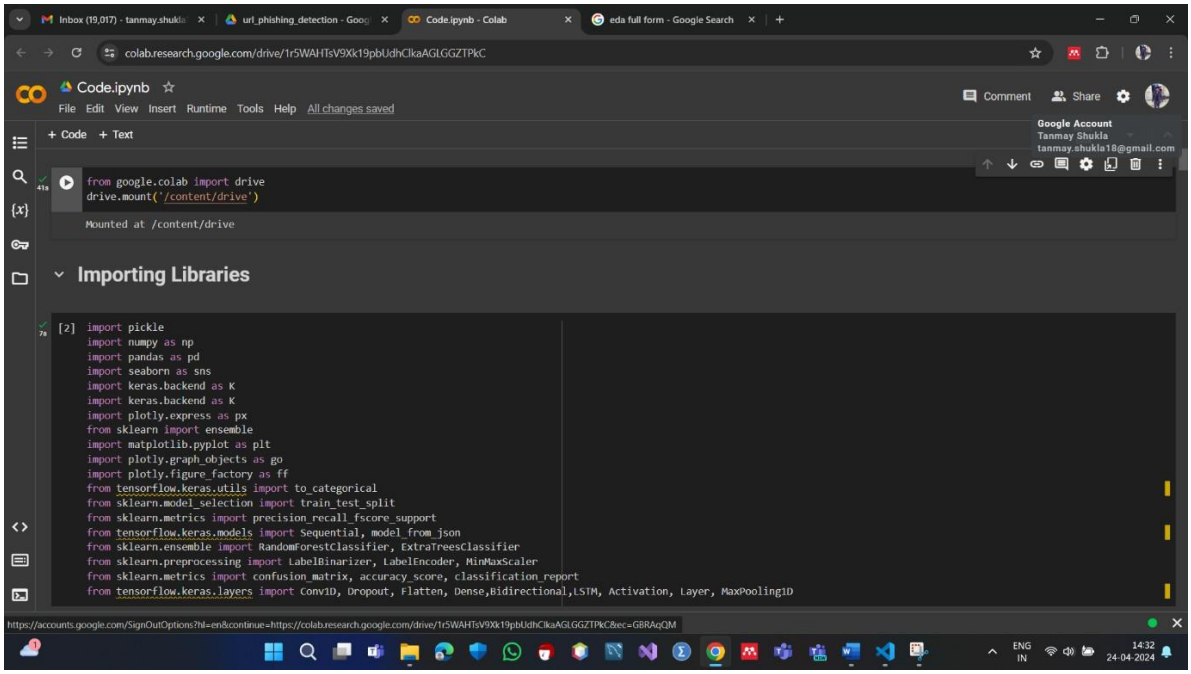
Below figure represents GUI of this App.

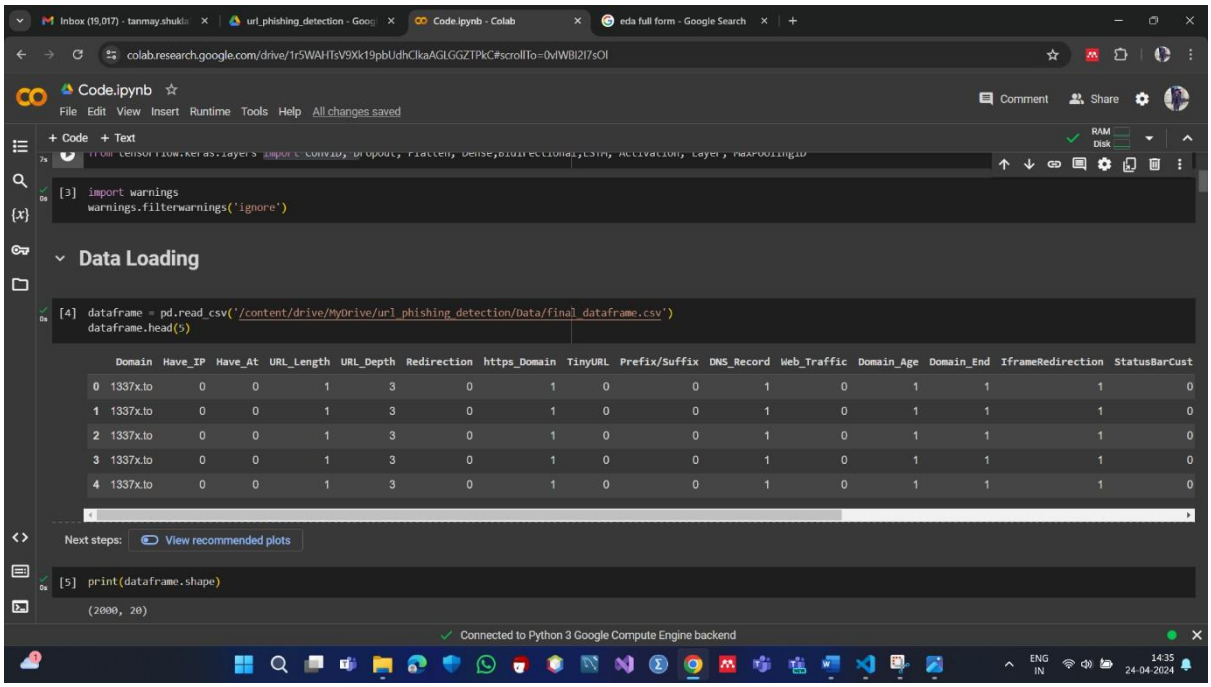


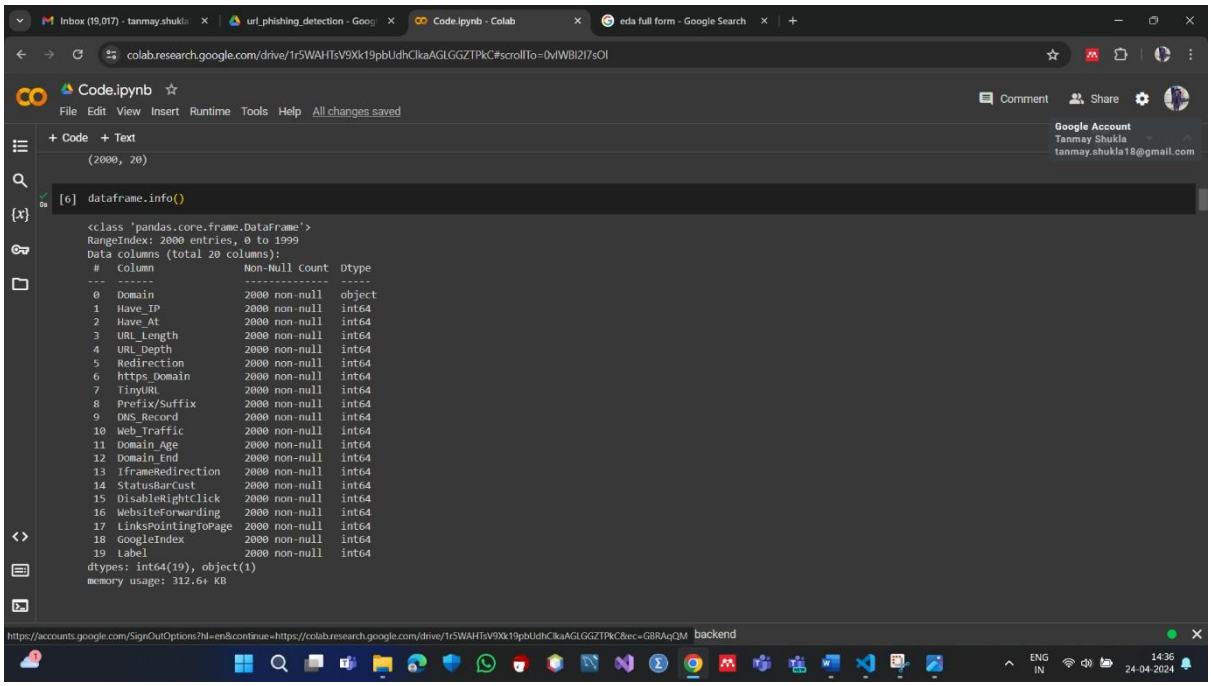


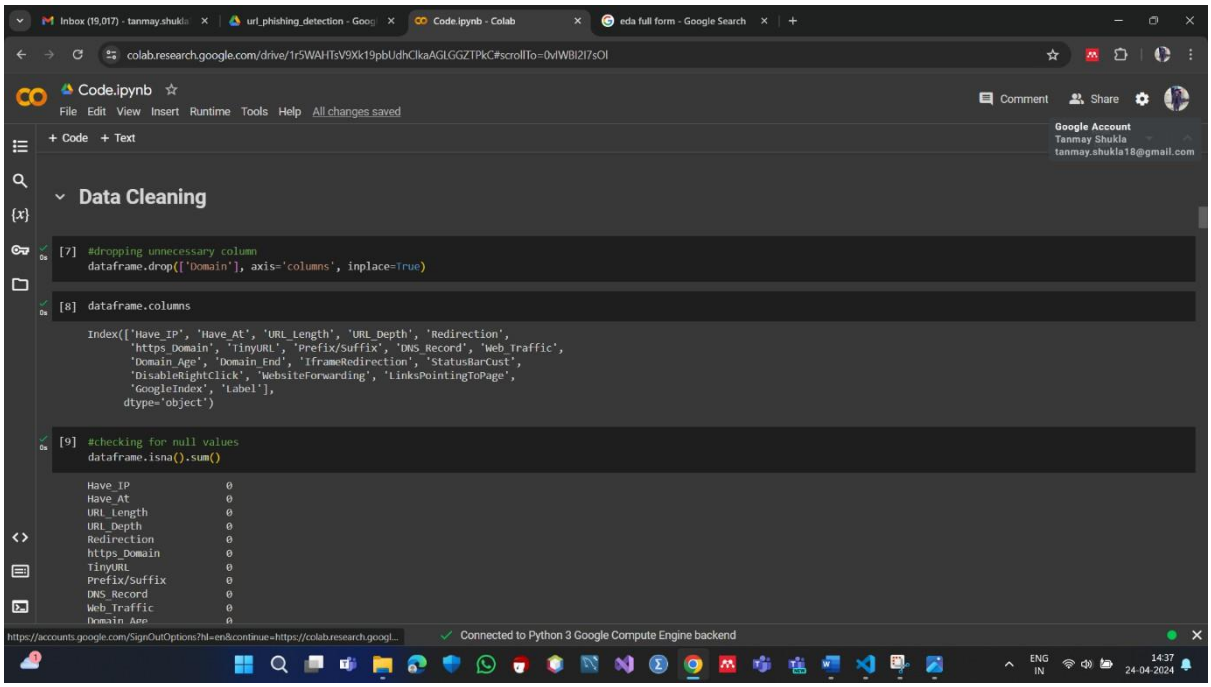
The above figure represents that the URL is safe.

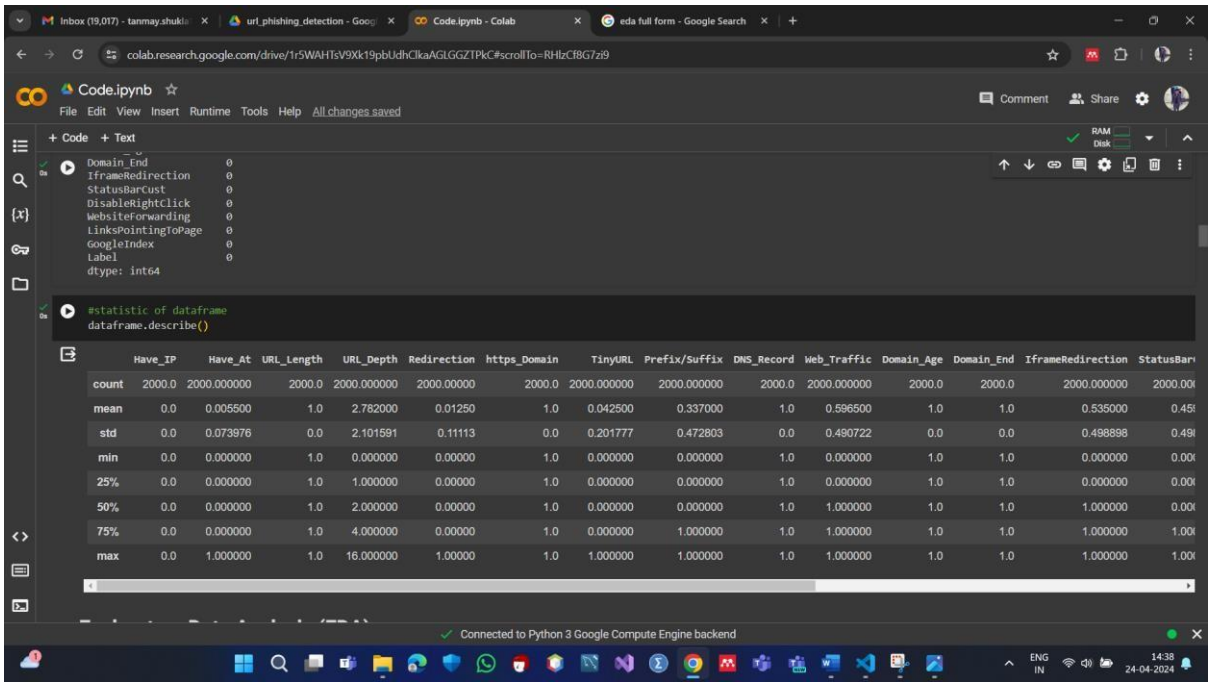
Appendix:

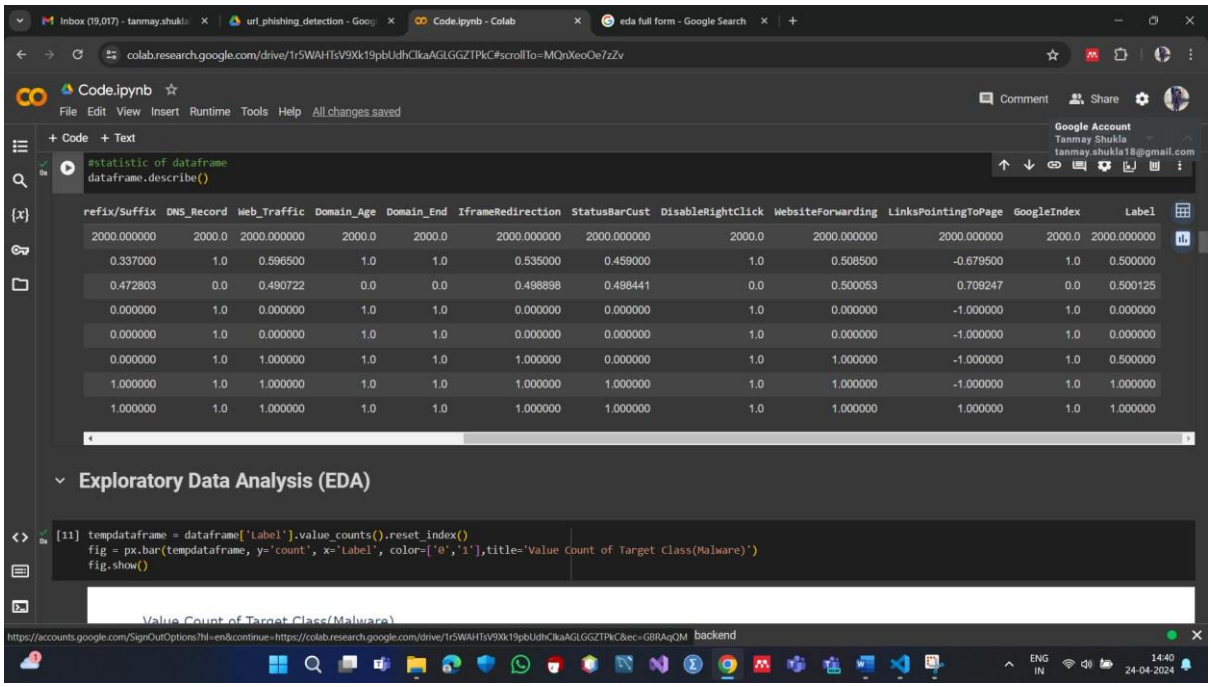


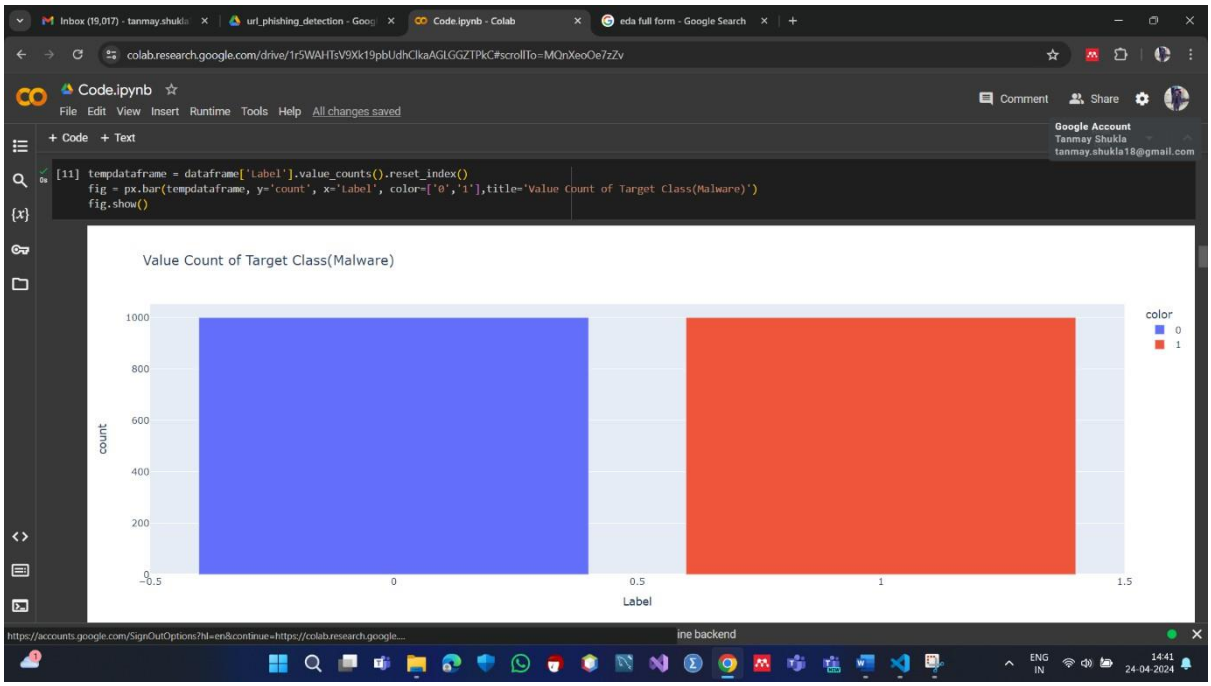

Code.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
[3] import warnings
warnings.filterwarnings('ignore')
Data Loading
[4] dataframe = pd.read_csv('/content/drive/MyDrive/url_phishing_detection/Data/final_dataframe.csv')
dataframe.head(5)
Domain Have_IP Have_At URL_Length URL_Depth Redirection https_Domain TinyURL Prefix/Suffix DNS_Record Web_Traffic Domain_Age Domain_End IframeRedirection StatusBarCust
0 1337x.to 0 0 1 3 0 1 0 0 1 0 1 1 1 0
1 1337x.to 0 0 1 3 0 1 0 0 1 0 1 1 1 0
2 1337x.to 0 0 1 3 0 1 0 0 1 0 1 1 1 0
3 1337x.to 0 0 1 3 0 1 0 0 1 0 1 1 1 0
4 1337x.to 0 0 1 3 0 1 0 0 1 0 1 1 1 0
Next steps: View recommended plots
[5] print(dataframe.shape)
(2000, 20)
Connected to Python 3 Google Compute Engine backend

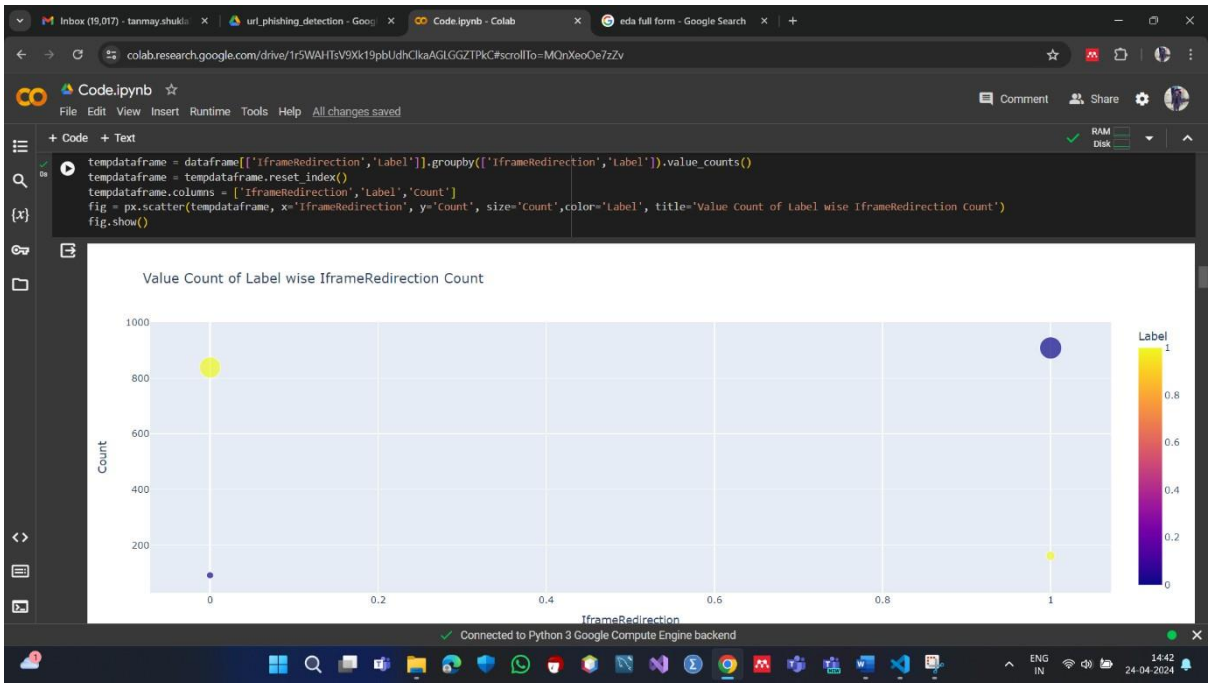

Code.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
Google Account
Tanmay Shukla
tanmay.shukla18@gmail.com
(2000, 20)
[6] dataframe.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2000 entries, 0 to 1999
Data columns (total 20 columns):
Column Non-Null Count Dtype
0 Domain 2000 non-null object
1 Have_IP 2000 non-null int64
2 Have_At 2000 non-null int64
3 URL_Length 2000 non-null int64
4 URL_Depth 2000 non-null int64
5 Redirection 2000 non-null int64
6 https_Domain 2000 non-null int64
7 TinyURL 2000 non-null int64
8 Prefix/Suffix 2000 non-null int64
9 DNS_Record 2000 non-null int64
10 Web_Traffic 2000 non-null int64
11 Domain_Age 2000 non-null int64
12 Domain_End 2000 non-null int64
13 IframeRedirection 2000 non-null int64
14 StatusBarCust 2000 non-null int64
15 DisableRightClick 2000 non-null int64
16 WebsiteForwarding 2000 non-null int64
17 LinksPointingToPage 2000 non-null int64
18 GoogleIndex 2000 non-null int64
19 Label 2000 non-null int64
dtypes: int64(19), object(1)
memory usage: 312.6+ KB
backend

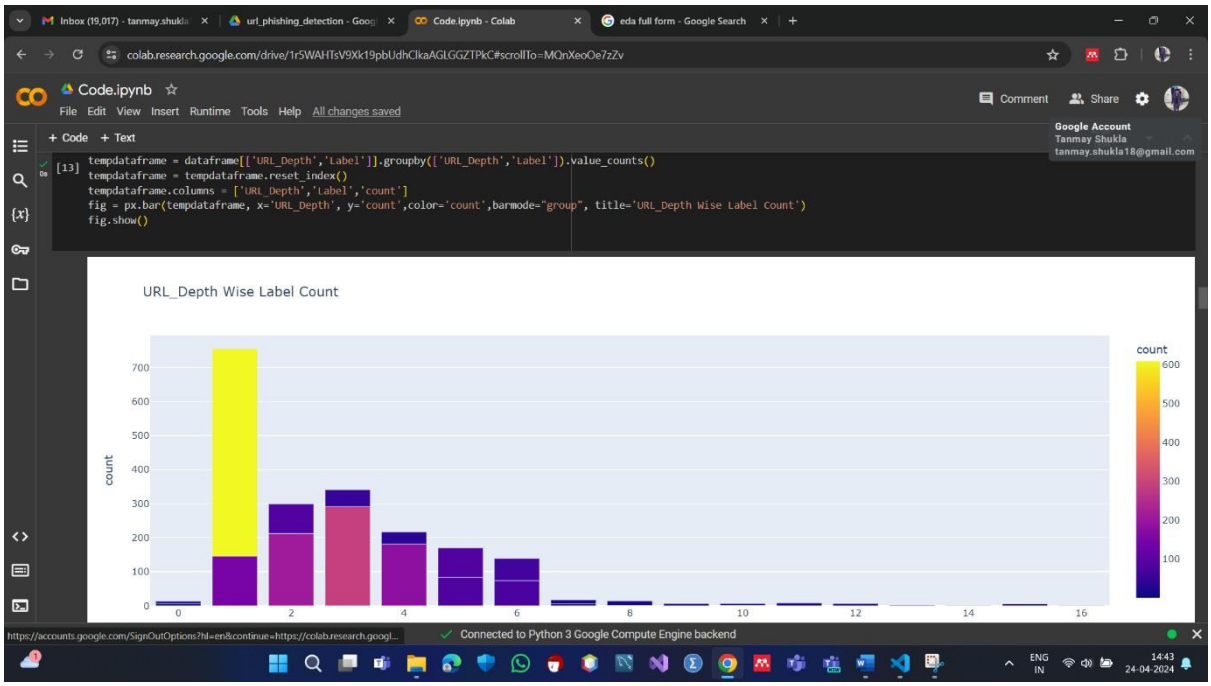

Code.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
Data Cleaning
[7] #dropping unnecessary column
dataframe.drop(['Domain'], axis='columns', inplace=True)
[8] dataframe.columns
Index(['Have_IP', 'Have_At', 'URL_Length', 'URL_Depth', 'Redirection',
'https_Domain', 'TinyURL', 'Prefix/Suffix', 'DNS_Record', 'Web_Traffic',
'Domain_Age', 'Domain_End', 'IframeRedirection', 'StatusBarCust',
'DisableRightClick', 'WebsiteForwarding', 'LinksPointingToPage',
'GoogleIndex', 'Label'],
dtype='object')
[9] #checking for null values
dataframe.isna().sum()
Have_IP 0
Have_At 0
URL_Length 0
URL_Depth 0
Redirection 0
https_Domain 0
TinyURL 0
Prefix/Suffix 0
DNS_Record 0
Web_Traffic 0
Connected to Python 3 Google Compute Engine backend

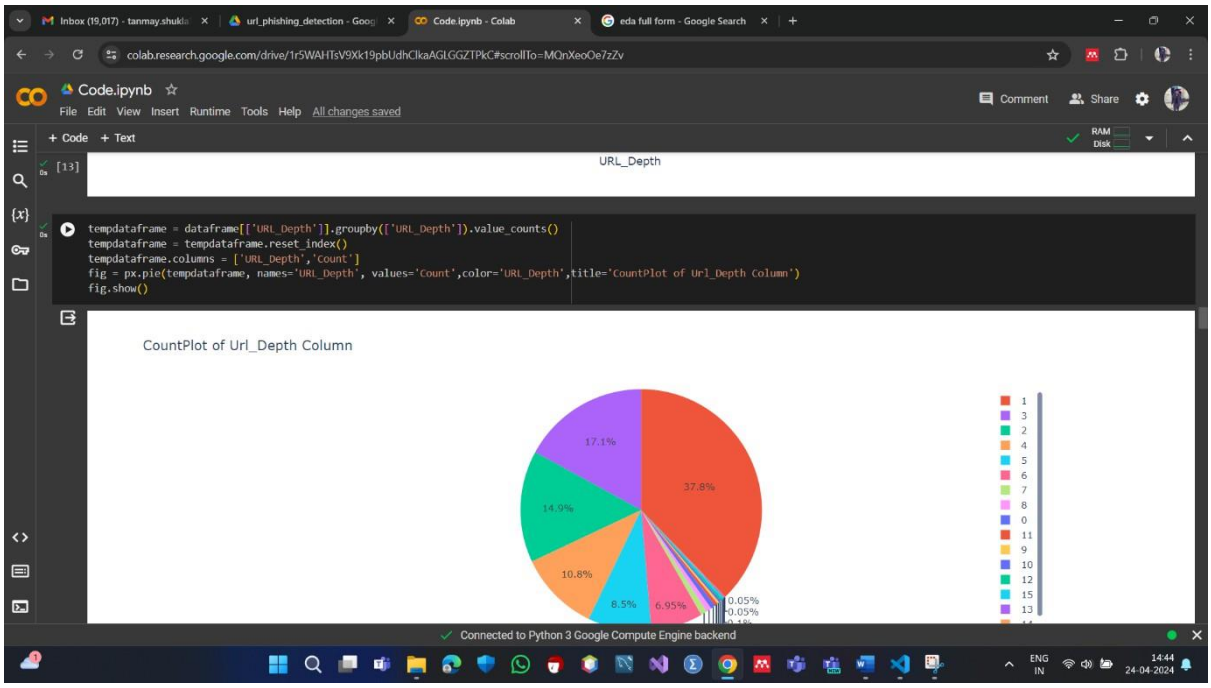

Code.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
Domain_End 0
IframeRedirection 0
StatusBarCust 0
DisableRightClick 0
WebsiteForwarding 0
LinksPointingToPage 0
GoogleIndex 0
Label 0
dtype: int64
#statistic of dataframe
dataframe.describe()
Have_IP Have_At URL_Length URL_Depth Redirection https_Domain TinyURL Prefix/Suffix DNS_Record Web_Traffic Domain_Age Domain_End IframeRedirection
count 2000.0 2000.000000 2000.0 2000.000000 2000.00000 2000.0 2000.000000 2000.000000 2000.0 2000.000000 2000.0 2000.0 2000.000000
mean 0.0 0.005500 1.0 2.782000 0.01250 1.0 0.042500 0.337000 1.0 0.596500 1.0 1.0 0.535000
std 0.0 0.073976 0.0 2.101591 0.11113 0.0 0.201777 0.472803 0.0 0.490722 0.0 0.0 0.498898
min 0.0 0.000000 1.0 0.000000 0.00000 1.0 0.000000 0.000000 1.0 0.000000 1.0 1.0 0.000000
25% 0.0 0.000000 1.0 1.000000 0.00000 1.0 0.000000 0.000000 1.0 0.000000 1.0 1.0 0.000000
50% 0.0 0.000000 1.0 2.000000 0.00000 1.0 0.000000 0.000000 1.0 1.000000 1.0 1.0 1.000000
75% 0.0 0.000000 1.0 4.000000 0.00000 1.0 0.000000 1.000000 1.0 1.000000 1.0 1.0 1.000000
max 0.0 1.000000 1.0 16.000000 1.00000 1.0 1.000000 1.000000 1.0 1.000000 1.0 1.0 1.000000
Connected to Python 3 Google Compute Engine backend

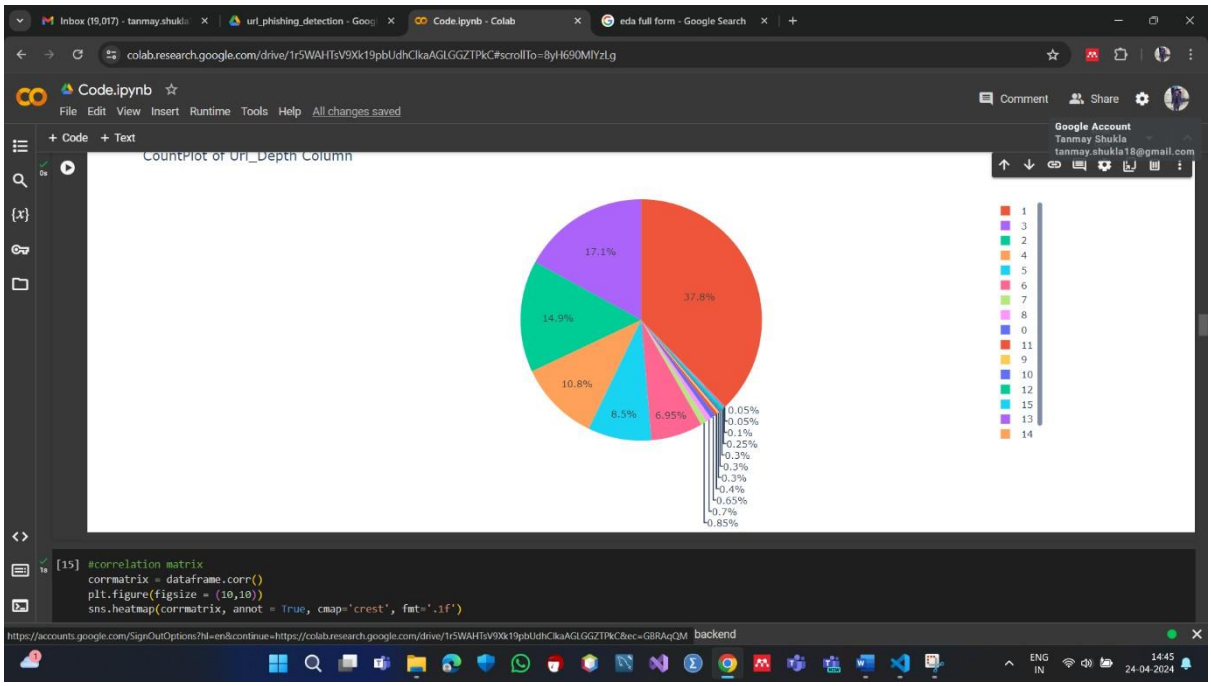

Code.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
#statistic of dataframe
dataframe.describe()
refix/Suffix DNS_Record Web_Traffic Domain_Age Domain_End IframeRedirection StatusBarCust DisableRightClick WebsiteForwarding LinksPointingToPage GoogleIndex Label
2000.000000 2000.0 2000.000000 2000.0 2000.0 2000.000000 2000.000000 2000.0 2000.000000 2000.000000 2000.0 2000.000000
0.337000 1.0 0.596500 1.0 1.0 0.535000 0.459000 1.0 0.508500 -0.679500 1.0 0.500000
0.472803 0.0 0.490722 0.0 0.0 0.498898 0.498441 0.0 0.500053 0.709247 0.0 0.500125
0.000000 1.0 0.000000 1.0 1.0 0.000000 0.000000 1.0 0.000000 -1.000000 1.0 0.000000
0.000000 1.0 0.000000 1.0 1.0 0.000000 0.000000 1.0 0.000000 -1.000000 1.0 0.000000
0.000000 1.0 1.000000 1.0 1.0 1.000000 0.000000 1.0 1.000000 -1.000000 1.0 0.500000
1.000000 1.0 1.000000 1.0 1.0 1.000000 1.000000 1.0 1.000000 -1.000000 1.0 1.000000
1.000000 1.0 1.000000 1.0 1.0 1.000000 1.000000 1.0 1.000000 1.000000 1.0 1.000000
Exploratory Data Analysis (EDA)
[11] tempdataframe = dataframe['Label'].value_counts().reset_index()
fig = px.bar(tempdataframe, y='count', x='Label', color=['0','1'],title='Value Count of Target Class(Malware)')
fig.show()
Value Count of Target Class(Malware)
backend

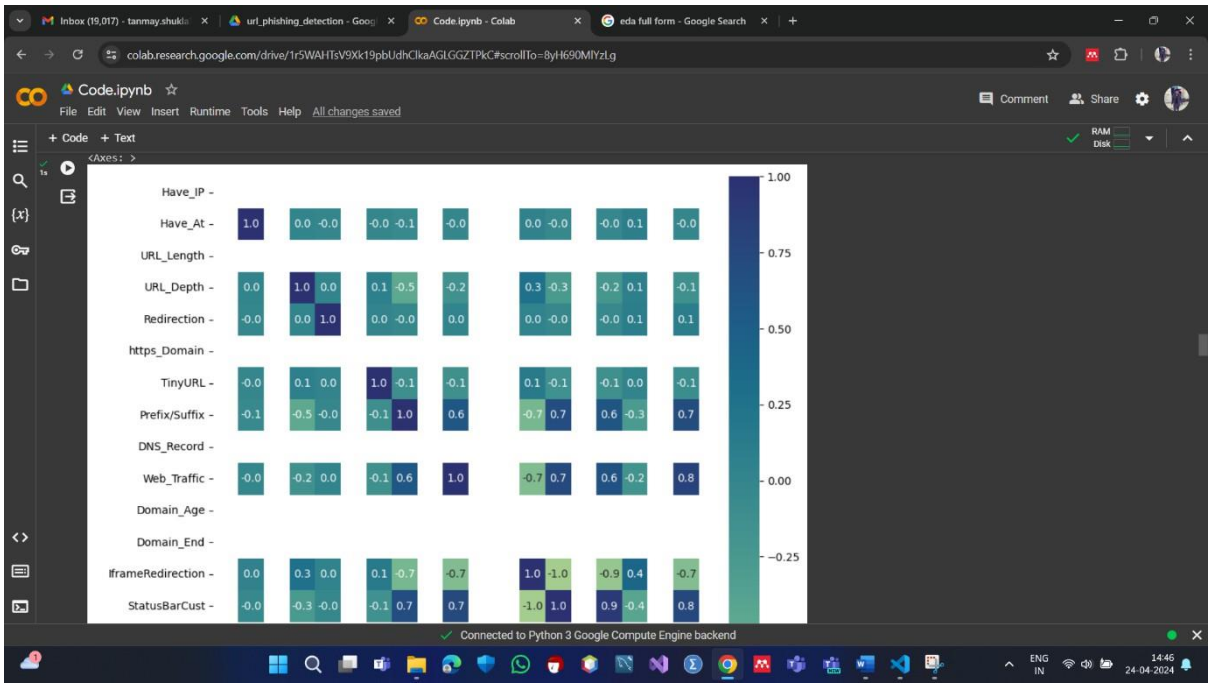

Code.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
[11] tempdataframe = dataframe['Label'].value_counts().reset_index()
fig = px.bar(tempdataframe, y='count', x='Label', color=['0','1'],title='Value Count of Target Class(Malware)')
fig.show()
Value Count of Target Class(Malware)
color
0
1
count
1000
800
600
400
200
0
−0.5
0
0.5
1
1.5
Label
ine backend

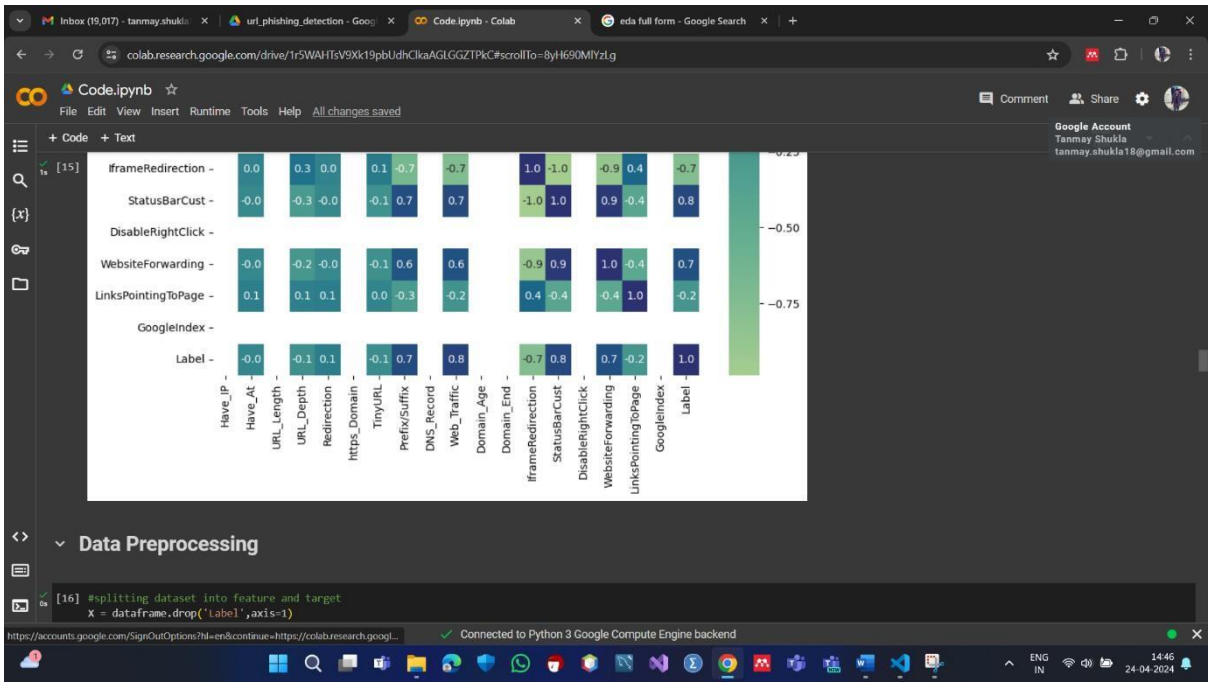

Code.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
tempdataframe = dataframe[['IframeRedirection','Label']].groupby(['IframeRedirection','Label']).value_counts()
tempdataframe = tempdataframe.reset_index()
tempdataframe.columns = ['IframeRedirection','Label','Count']
fig = px.scatter(tempdataframe, x='IframeRedirection', y='Count', size='Count',color='Label', title='Value Count of Label wise IframeRedirection Count')
fig.show()
Value Count of Label wise IframeRedirection Count
Count
IframeRedirection
Label
Connected to Python 3 Google Compute Engine backend

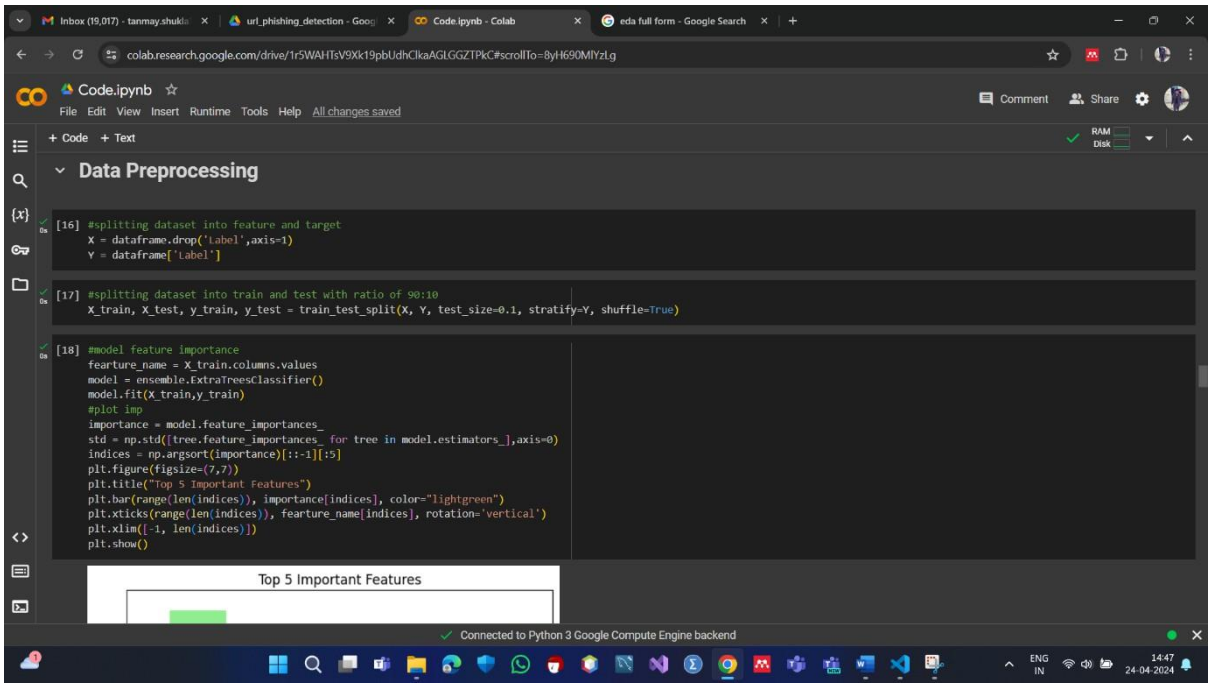

Code.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
[13] tempdataframe = dataframe[['URL_Depth','Label']].groupby(['URL_Depth','Label']).value_counts()
tempdataframe = tempdataframe.reset_index()
tempdataframe.columns = ['URL_Depth','Label','count']
fig = px.bar(tempdataframe, x='URL_Depth', y='count',color='count',barmode="group", title='URL_Depth Wise Label Count')
fig.show()
URL_Depth Wise Label Count
count
count
Connected to Python 3 Google Compute Engine backend

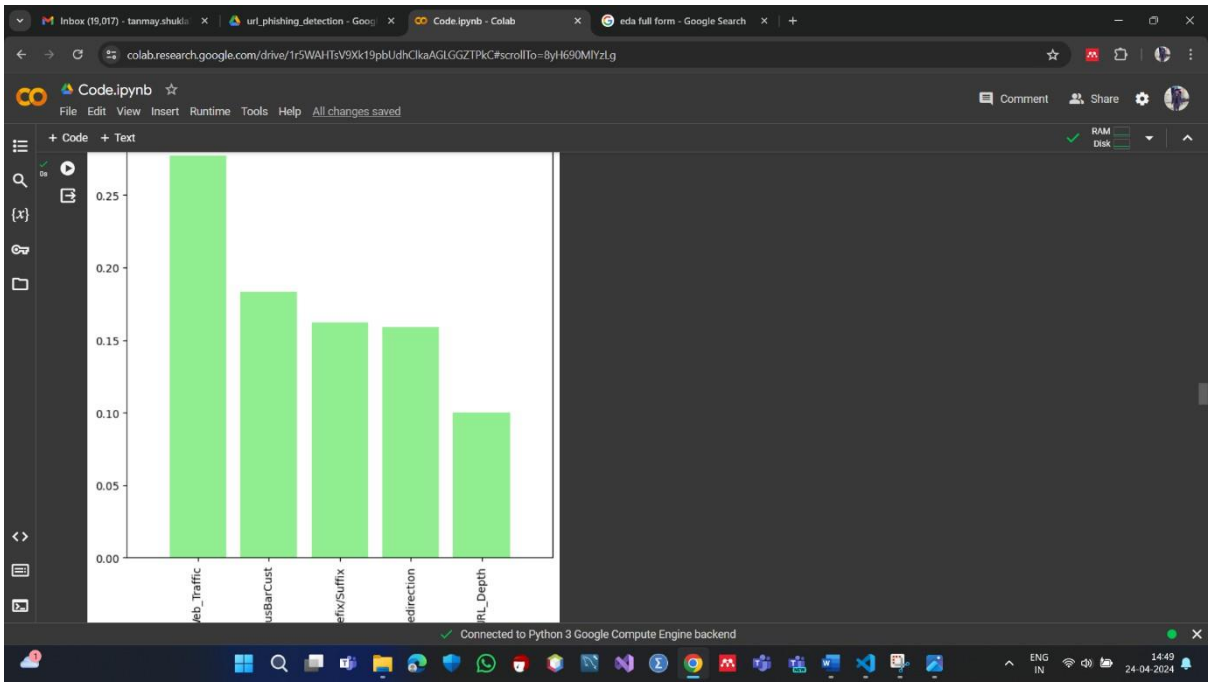

Code.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
[13]
URL_Depth
tempdataframe = dataframe[['URL_Depth']].groupby(['URL_Depth']).value_counts()
tempdataframe = tempdataframe.reset_index()
tempdataframe.columns = ['URL_Depth','Count']
fig = px.pie(tempdataframe, names='URL_Depth', values='Count',color='URL_Depth',title='CountPlot of Url_Depth Column')
fig.show()
CountPlot of Url_Depth Column
37.8%
17.1%
14.9%
10.8%
8.5%
6.95%
0.05%
0.05%
1
3
2
4
5
6
7
8
0
11
9
10
12
15
13
Connected to Python 3 Google Compute Engine backend

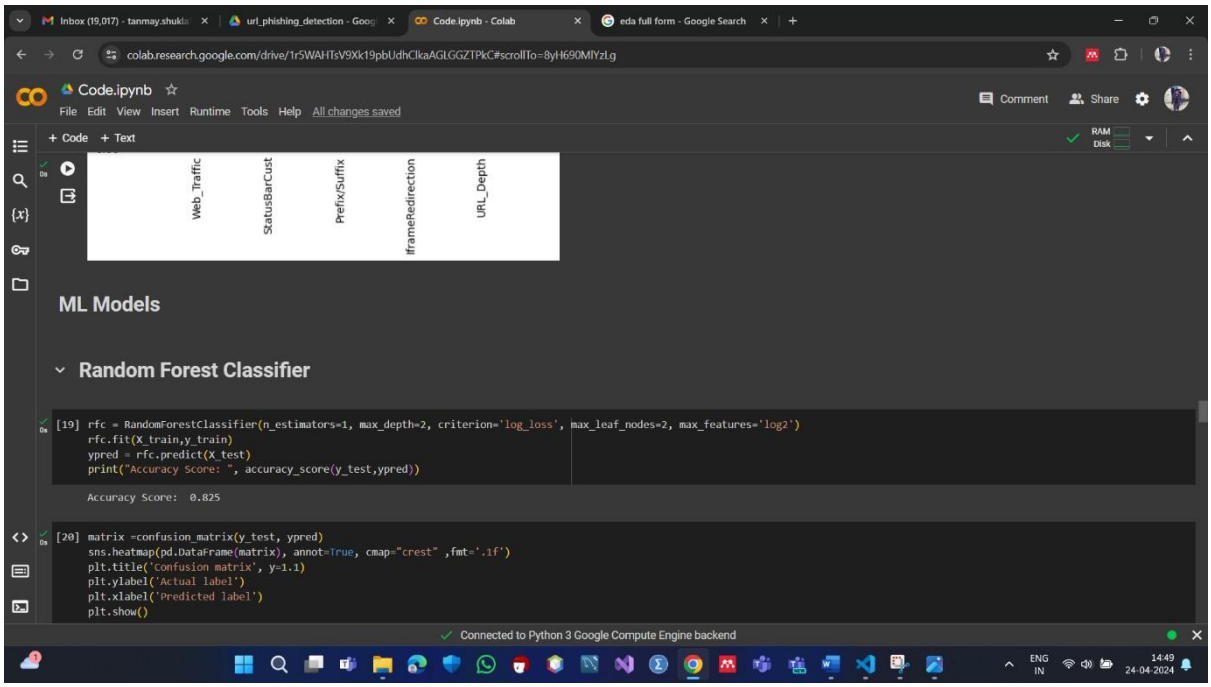

Code.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
Google Account
Tanmay Shukla
tanmay.shukla18@gmail.com
CountPlot of Url_Depth Column
37.8%
17.1%
14.9%
10.8%
8.5%
6.95%
0.05%
0.05%
0.1%
0.25%
0.3%
0.3%
0.3%
0.4%
0.65%
0.7%
0.85%
1
3
2
4
5
6
7
8
0
11
9
10
12
15
13
14
[15] #correlation matrix
corrmatrix = dataframe.corr()
plt.figure(figsize = (10,10))
sns.heatmap(corrmatrix, annot = True, cmap='crest', fmt='.1f')
backend

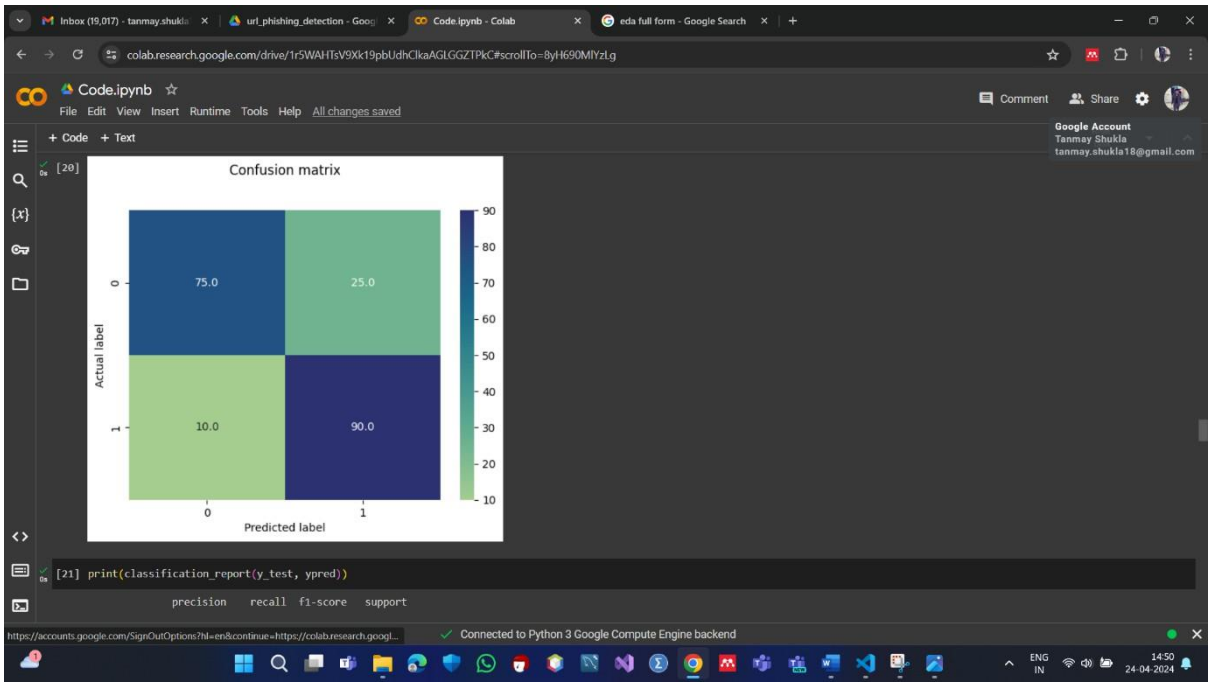

Code.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
<Axes: >
Have_IP
Have_At
URL_Length
URL_Depth
Redirection
https_Domain
TinyURL
Prefix/Suffix
DNS_Record
Web_Traffic
Domain_Age
Domain_End
IframeRedirection
StatusBarCust
1.00
0.75
0.50
0.25
0.00
−0.25
Connected to Python 3 Google Compute Engine backend


Code.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
Google Account
Tanmay Shukla
tanmay.shukla18@gmail.com
[15]
IframeRedirection
StatusBarCust
DisableRightClick
WebsiteForwarding
LinksPointingToPage
GoogleIndex
Label
−0.50
−0.75
Have_IP
Have_At
URL_Length
URL_Depth
Redirection
https_Domain
TinyURL
Prefix/Suffix
DNS_Record
Web_Traffic
Domain_Age
Domain_End
IframeRedirection
StatusBarCust
DisableRightClick
WebsiteForwarding
LinksPointingToPage
GoogleIndex
Label
Data Preprocessing
[16] #splitting dataset into feature and target
X = dataframe.drop('Label',axis=1)
Connected to Python 3 Google Compute Engine backend


Code.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
Data Preprocessing
[16] #splitting dataset into feature and target
X = dataframe.drop('label',axis=1)
Y = dataframe['label']
[17] #splitting dataset into train and test with ratio of 90:10
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.1, stratify=Y, shuffle=True)
[18] #model feature importance
fearture_name = X_train.columns.values
model = ensemble.ExtraTreesClassifier()
model.fit(X_train,y_train)
#plot imp
importance = model.feature_importances_
std = np.std([tree.feature_importances_ for tree in model.estimators_],axis=0)
indices = np.argsort(importance)[::-1][:5]
plt.figure(figsize=(7,7))
plt.title("Top 5 Important Features")
plt.bar(range(len(indices)), importance[indices], color="lightgreen")
plt.xticks(range(len(indices)), fearture_name[indices], rotation='vertical')
plt.xlim([-1, len(indices)])
plt.show()
Top 5 Important Features
Connected to Python 3 Google Compute Engine backend


Code.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
0.25
0.20
0.15
0.10
0.05
0.00
Connected to Python 3 Google Compute Engine backend


Code.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
Web_Traffic
StatusBarCust
Prefix/Suffix
IframeRedirection
URL_Depth
ML Models
Random Forest Classifier
[19] rfc = RandomForestClassifier(n_estimators=1, max_depth=2, criterion='log_loss', max_leaf_nodes=2, max_features='log2')
rfc.fit(X_train,y_train)
ypred = rfc.predict(X_test)
print("Accuracy Score: ", accuracy_score(y_test,ypred))
Accuracy Score: 0.825
[20] matrix =confusion_matrix(y_test, ypred)
sns.heatmap(pd.DataFrame(matrix), annot=True, cmap="crest" ,fmt='.1f')
plt.title('Confusion matrix', y=1.1)
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
plt.show()
Connected to Python 3 Google Compute Engine backend


Code.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
Google Account
Tanmay Shukla
tanmay.shukla18@gmail.com
[20]
Confusion matrix
75.0
25.0
10.0
90.0
Actual label
Predicted label
[21] print(classification_report(y_test, ypred))
precision recall f1-score support
Connected to Python 3 Google Compute Engine backend

```
print(classification_report(y_test, ypred))
```

```
              precision    recall  f1-score   support

           0       0.88      0.75      0.81       100
           1       0.78      0.90      0.84       100

    accuracy                           0.82       200
   macro avg       0.83      0.82      0.82       200
weighted avg       0.83      0.82      0.82       200
```

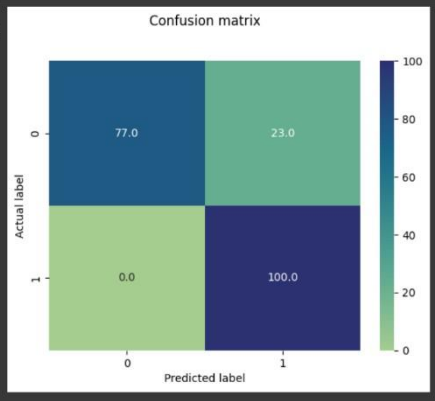
Extra Tree Classifier

```
[22] etc = ExtraTreesClassifier(n_estimators=1, max_depth=2, criterion='log_loss', max_leaf_nodes=2, max_features='log2')
     etc.fit(X_train,y_train)
     ypred = etc.predict(X_test)
     print("Accuracy Score: ", accuracy_score(y_test,ypred))
```

```
Accuracy Score:  0.885
```

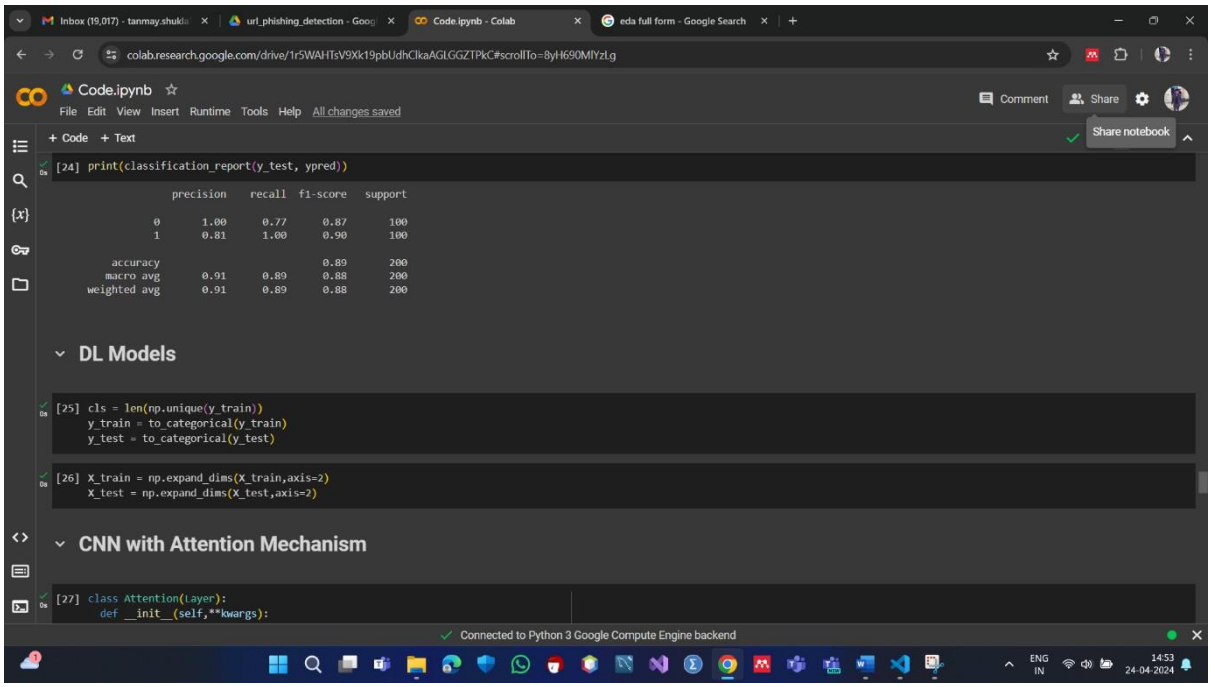
```
[23] matrix =confusion_matrix(y_test, ypred)
     sns.heatmap(pd.DataFrame(matrix), annot=True, cmap="crest" ,fmt='.1f')
     plt.title('Confusion matrix', y=1.1)
     plt.ylabel('Actual label')
     plt.xlabel('Predicted label')
     plt.show()
```

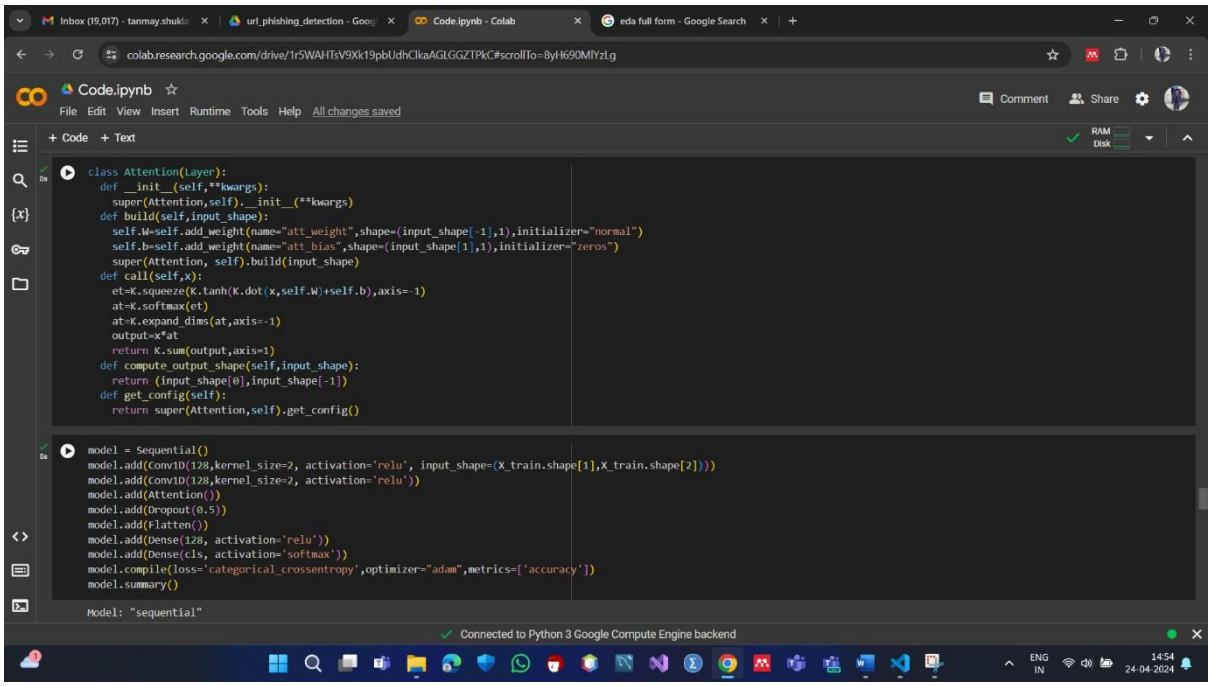
Connected to Python 3 Google Compute Engine backend

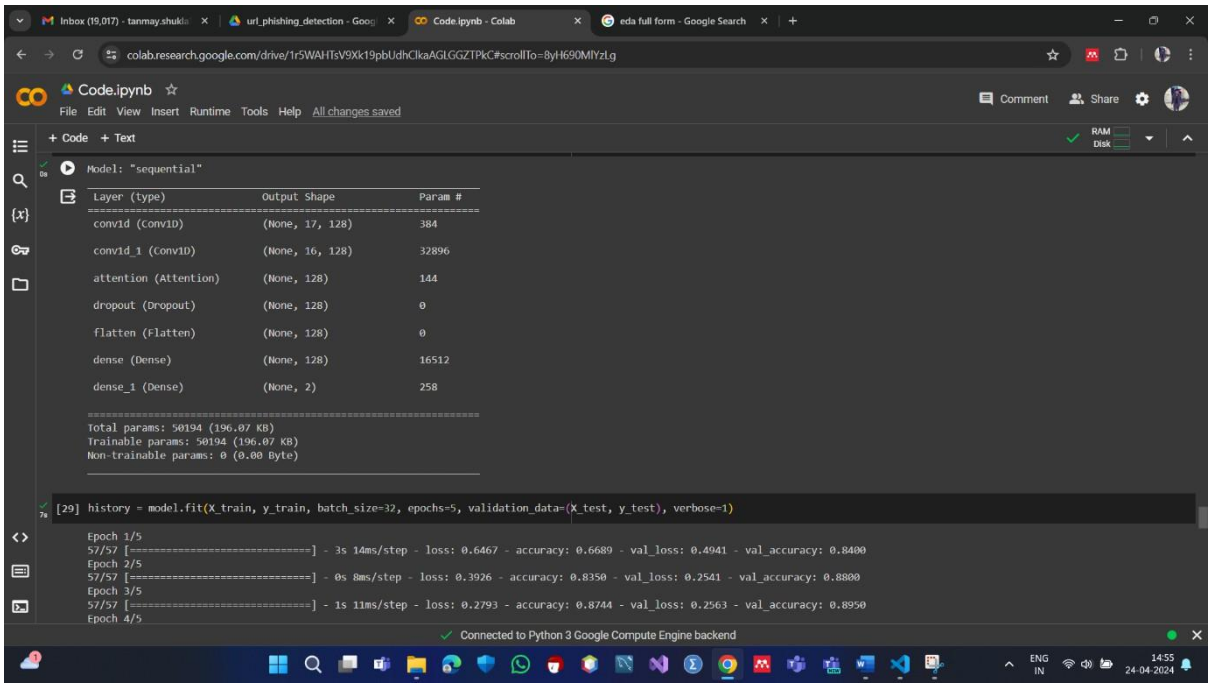


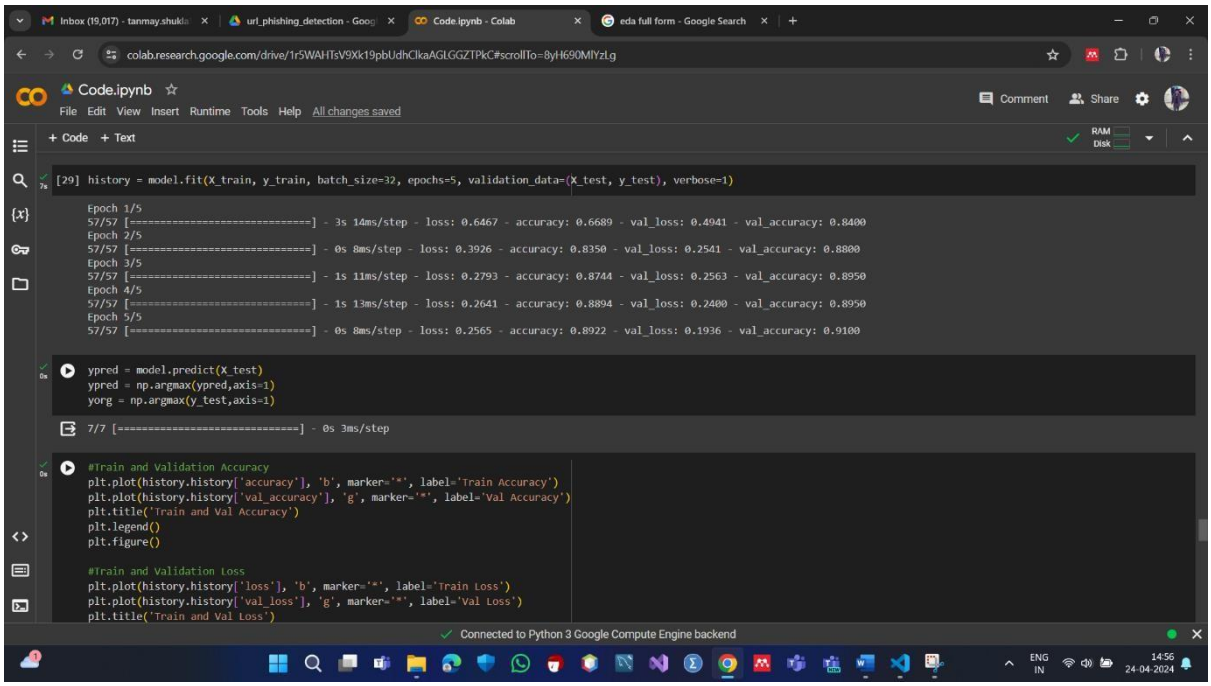

```
[24] print(classification_report(y_test, ypred))
```

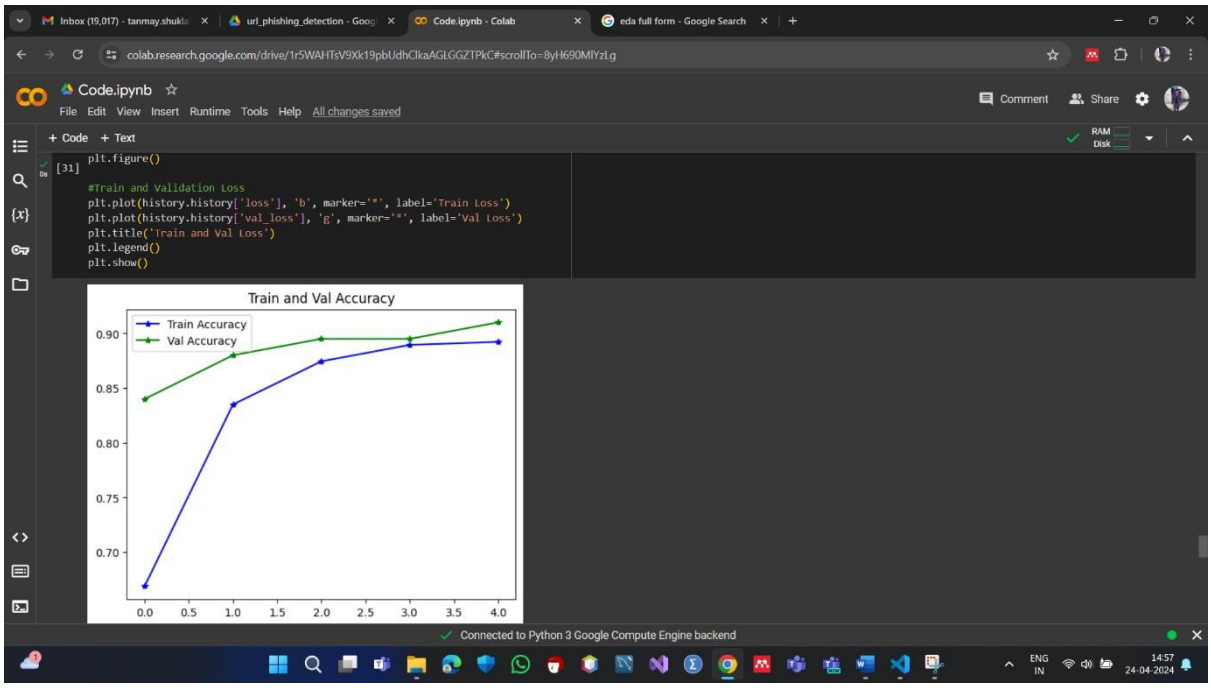
Connected to Python 3 Google Compute Engine backend

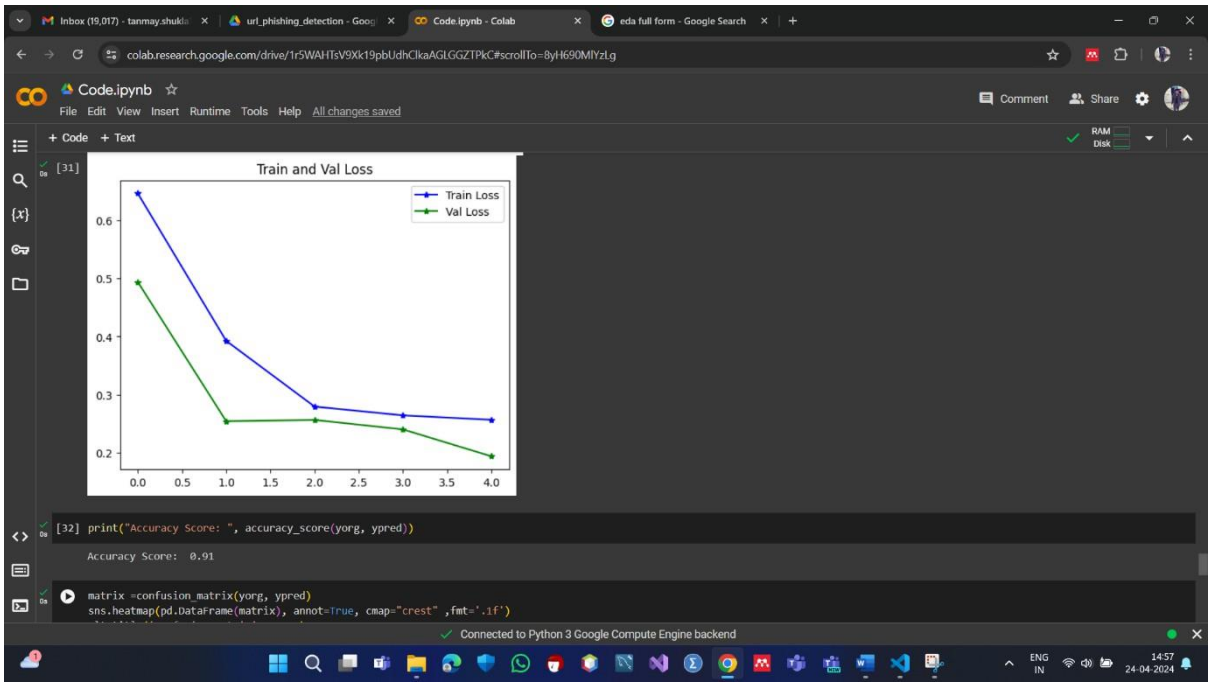

Code.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
Share notebook
+ Code + Text
[24] print(classification_report(y_test, ypred))
precision recall f1-score support
0 1.00 0.77 0.87 100
1 0.81 1.00 0.90 100
accuracy 0.89 200
macro avg 0.91 0.89 0.88 200
weighted avg 0.91 0.89 0.88 200
DL Models
[25] cls = len(np.unique(y_train))
y_train = to_categorical(y_train)
y_test = to_categorical(y_test)
[26] X_train = np.expand_dims(X_train,axis=2)
X_test = np.expand_dims(X_test,axis=2)
CNN with Attention Mechanism
[27] class Attention(Layer):
def __init__(self,**kwargs):
Connected to Python 3 Google Compute Engine backend

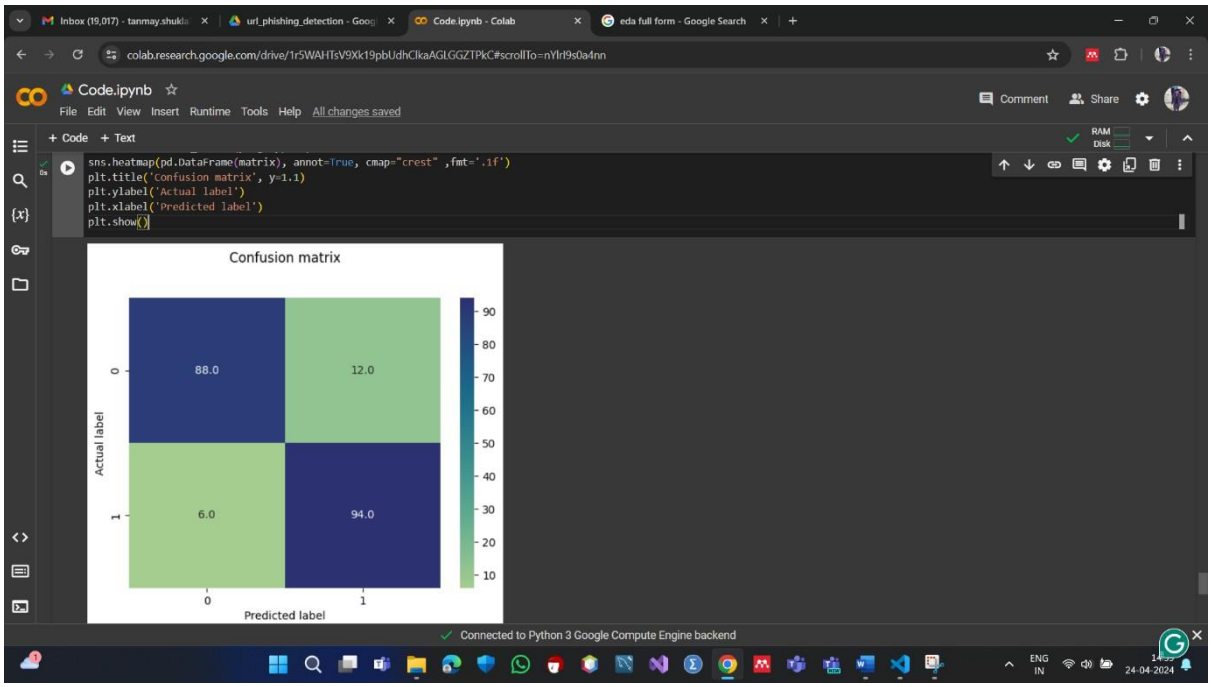

Code.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
RAM
Disk
class Attention(Layer):
def __init__(self,**kwargs):
super(Attention,self).__init__(**kwargs)
def build(self,input_shape):
self.W=self.add_weight(name="att_weight",shape=(input_shape[-1],1),initializer="normal")
self.b=self.add_weight(name="att_bias",shape=(input_shape[1],1),initializer="zeros")
super(Attention, self).build(input_shape)
def call(self,x):
et=K.squeeze(K.tanh(K.dot(x,self.W)+self.b),axis=-1)
at=K.softmax(et)
at=K.expand_dims(at,axis=-1)
output=x*at
return K.sum(output,axis=1)
def compute_output_shape(self,input_shape):
return (input_shape[0],input_shape[-1])
def get_config(self):
return super(Attention,self).get_config()
model = Sequential()
model.add(Conv1D(128,kernel_size=2, activation='relu', input_shape=(X_train.shape[1],X_train.shape[2])))
model.add(Conv1D(128,kernel_size=2, activation='relu'))
model.add(Attention())
model.add(Dropout(0.5))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dense(cls, activation='softmax'))
model.compile(loss='categorical_crossentropy',optimizer="adam",metrics=['accuracy'])
model.summary()
Model: "sequential"
Connected to Python 3 Google Compute Engine backend

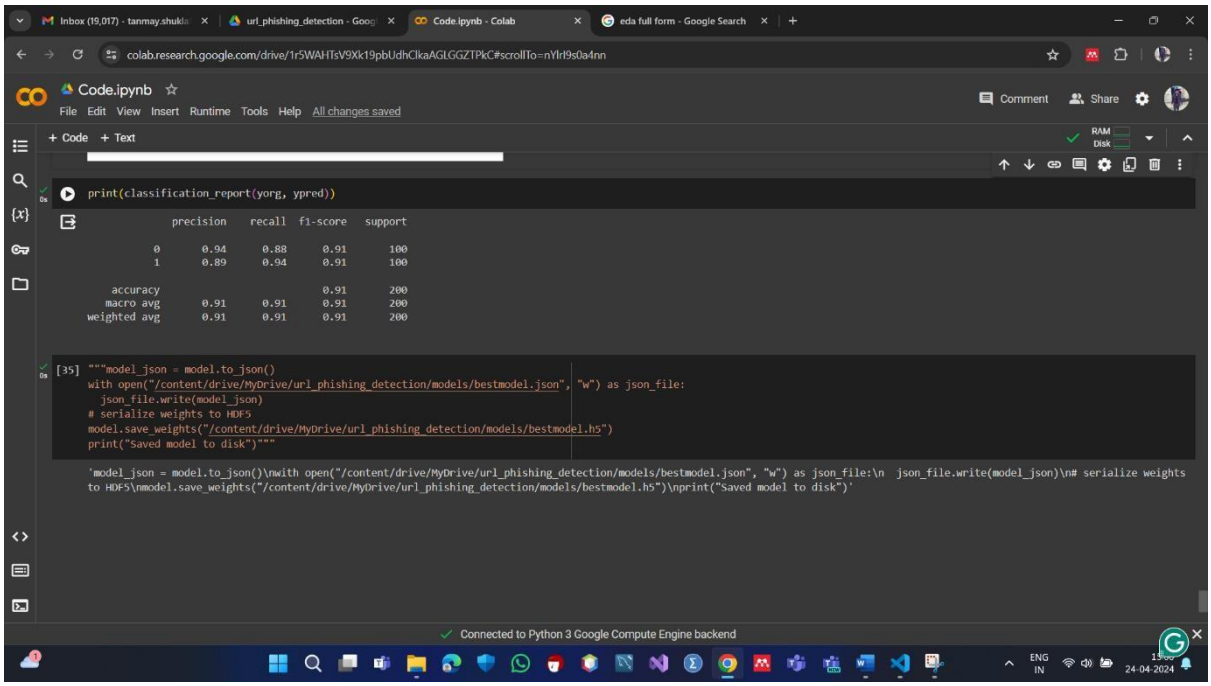

Code.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
Model: "sequential"
Layer (type) Output Shape Param #
conv1d (Conv1D) (None, 17, 128) 384
conv1d_1 (Conv1D) (None, 16, 128) 32896
attention (Attention) (None, 128) 144
dropout (Dropout) (None, 128) 0
flatten (Flatten) (None, 128) 0
dense (Dense) (None, 128) 16512
dense_1 (Dense) (None, 2) 258
Total params: 50194 (196.07 KB)
Trainable params: 50194 (196.07 KB)
Non-trainable params: 0 (0.00 Byte)
[29] history = model.fit(X_train, y_train, batch_size=32, epochs=5, validation_data=(X_test, y_test), verbose=1)
Epoch 1/5
57/57 [==========] - 3s 14ms/step - loss: 0.6467 - accuracy: 0.6689 - val_loss: 0.4941 - val_accuracy: 0.8400
Epoch 2/5
57/57 [==========] - 0s 8ms/step - loss: 0.3926 - accuracy: 0.8350 - val_loss: 0.2541 - val_accuracy: 0.8800
Epoch 3/5
57/57 [==========] - 1s 11ms/step - loss: 0.2793 - accuracy: 0.8744 - val_loss: 0.2563 - val_accuracy: 0.8950
Epoch 4/5
Connected to Python 3 Google Compute Engine backend

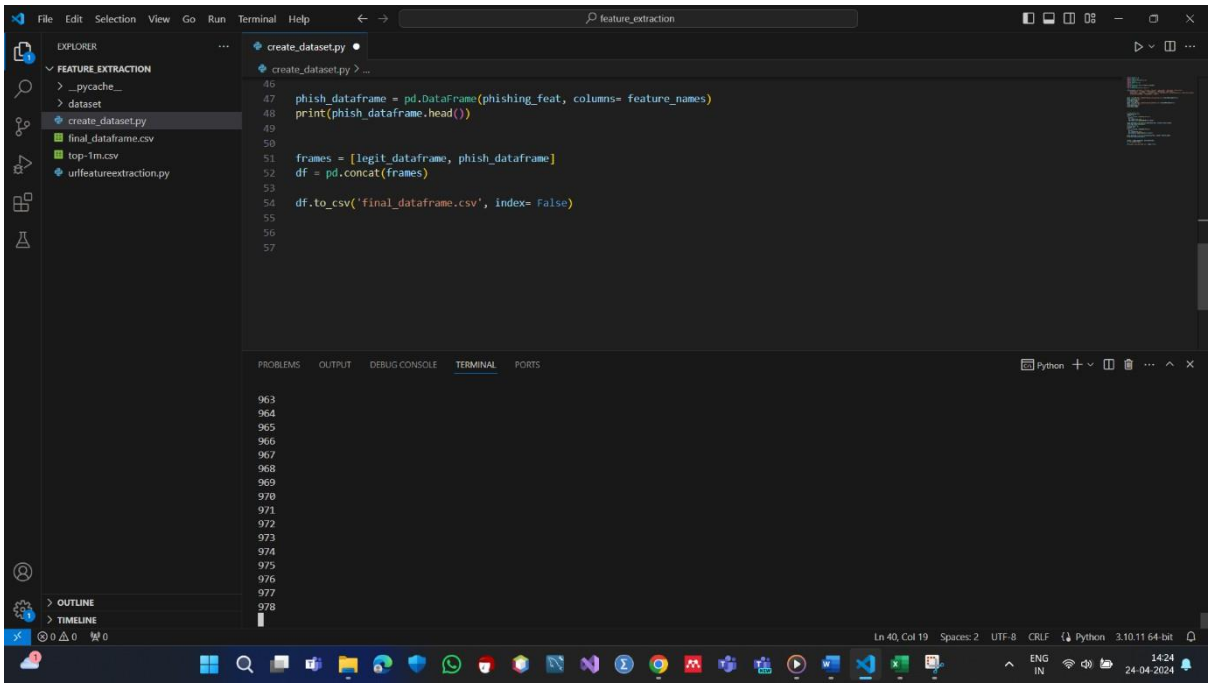

Code.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
[29] history = model.fit(X_train, y_train, batch_size=32, epochs=5, validation_data=(X_test, y_test), verbose=1)
Epoch 1/5
57/57 [==========] - 3s 14ms/step - loss: 0.6467 - accuracy: 0.6689 - val_loss: 0.4941 - val_accuracy: 0.8400
Epoch 2/5
57/57 [==========] - 0s 8ms/step - loss: 0.3926 - accuracy: 0.8350 - val_loss: 0.2541 - val_accuracy: 0.8800
Epoch 3/5
57/57 [==========] - 1s 11ms/step - loss: 0.2793 - accuracy: 0.8744 - val_loss: 0.2563 - val_accuracy: 0.8950
Epoch 4/5
57/57 [==========] - 1s 13ms/step - loss: 0.2641 - accuracy: 0.8894 - val_loss: 0.2400 - val_accuracy: 0.8950
Epoch 5/5
57/57 [==========] - 0s 8ms/step - loss: 0.2565 - accuracy: 0.8922 - val_loss: 0.1936 - val_accuracy: 0.9100
ypred = model.predict(X_test)
ypred = np.argmax(ypred,axis=1)
yorg = np.argmax(y_test,axis=1)
7/7 [==========] - 0s 3ms/step
#Train and Validation Accuracy
plt.plot(history.history['accuracy'], 'b', marker='*', label='Train Accuracy')
plt.plot(history.history['val_accuracy'], 'g', marker='*', label='Val Accuracy')
plt.title('Train and Val Accuracy')
plt.legend()
plt.figure()
#Train and Validation Loss
plt.plot(history.history['loss'], 'b', marker='*', label='Train Loss')
plt.plot(history.history['val_loss'], 'g', marker='*', label='Val Loss')
plt.title('Train and Val Loss')
Connected to Python 3 Google Compute Engine backend

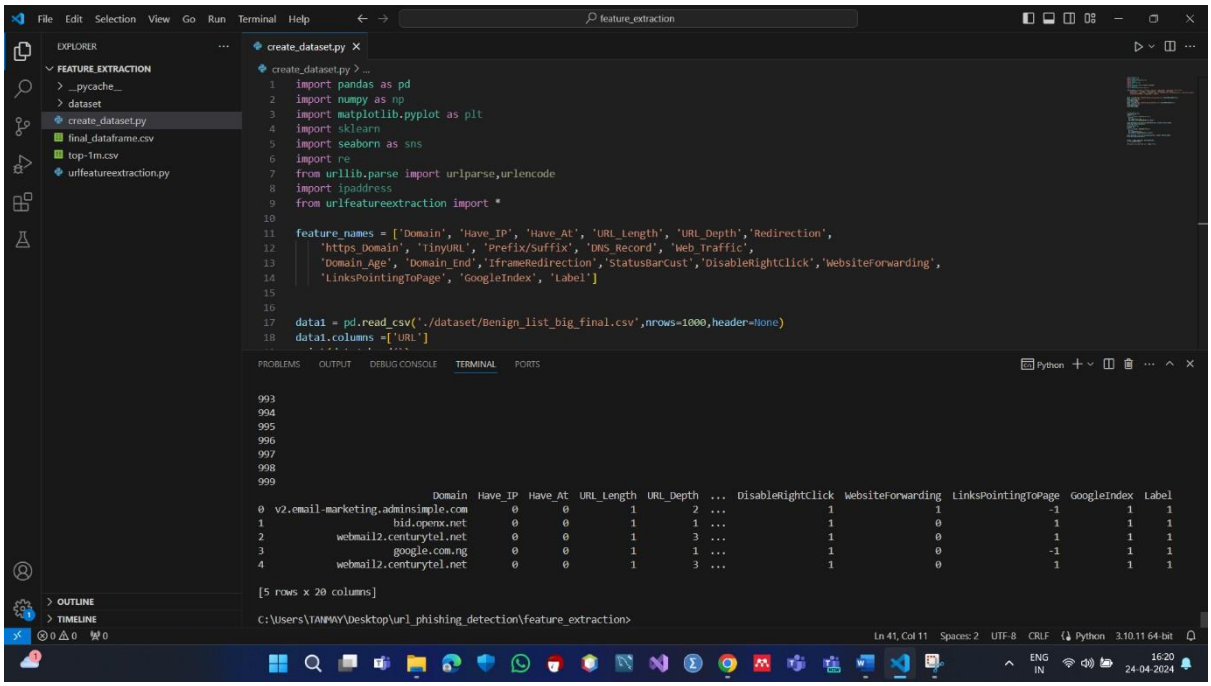

Code.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
[31]
plt.figure()
#Train and Validation Loss
plt.plot(history.history['loss'], 'b', marker='*', label='Train Loss')
plt.plot(history.history['val_loss'], 'g', marker='*', label='Val Loss')
plt.title('Train and Val Loss')
plt.legend()
plt.show()
Train and Val Accuracy
Train Accuracy
Val Accuracy
Connected to Python 3 Google Compute Engine backend

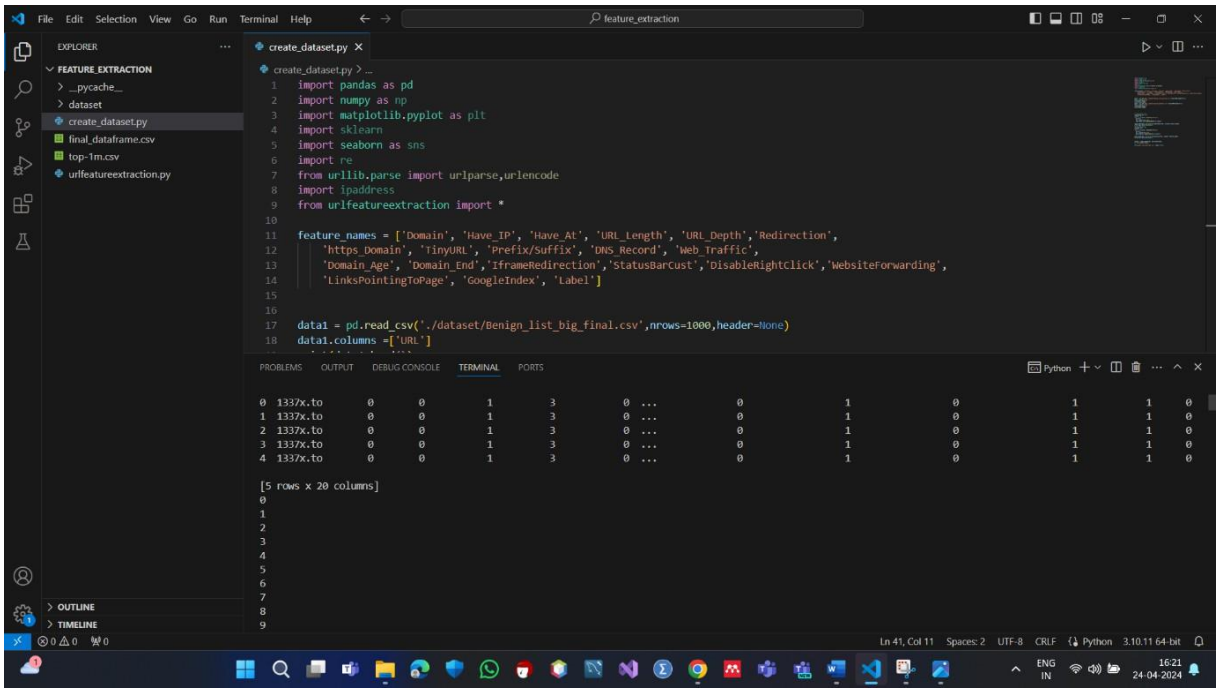

Code.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
[31]
Train and Val Loss
Train Loss
Val Loss
[32] print("Accuracy Score: ", accuracy_score(yorg, ypred))
Accuracy Score: 0.91
matrix =confusion_matrix(yorg, ypred)
sns.heatmap(pd.DataFrame(matrix), annot=True, cmap="crest" ,fmt='.1f')
Connected to Python 3 Google Compute Engine backend


Code.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
sns.heatmap(pd.DataFrame(matrix), annot=True, cmap="crest" ,fmt='.1f')
plt.title('Confusion matrix', y=1.1)
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
plt.show()
Confusion matrix
88.0
12.0
6.0
94.0
Actual label
Predicted label
Connected to Python 3 Google Compute Engine backend


Code.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
print(classification_report(yorg, ypred))
precision recall f1-score support
0 0.94 0.88 0.91 100
1 0.89 0.94 0.91 100
accuracy 0.91 200
macro avg 0.91 0.91 0.91 200
weighted avg 0.91 0.91 0.91 200
[35] """model_json = model.to_json()
with open("/content/drive/MyDrive/url_phishing_detection/models/bestmodel.json", "w") as json_file:
json_file.write(model_json)
# serialize weights to HDF5
model.save_weights("/content/drive/MyDrive/url_phishing_detection/models/bestmodel.h5")
print("Saved model to disk")"""
'model_json = model.to_json()\nwith open("/content/drive/MyDrive/url_phishing_detection/models/bestmodel.json", "w") as json_file:\n json_file.write(model_json)\n# serialize weights to HDF5\nmodel.save_weights("/content/drive/MyDrive/url_phishing_detection/models/bestmodel.h5")\nprint("Saved model to disk")'
Connected to Python 3 Google Compute Engine backend


File Edit Selection View Go Run Terminal Help
feature_extraction
EXPLORER
FEATURE_EXTRACTION
__pycache__
dataset
create_dataset.py
final_dataframe.csv
top-1m.csv
urlfeatureextraction.py
create_dataset.py
phish_dataframe = pd.DataFrame(phishing_feat, columns= feature_names)
print(phish_dataframe.head())
frames = [legit_dataframe, phish_dataframe]
df = pd.concat(frames)
df.to_csv('final_dataframe.csv', index= False)
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
OUTLINE
TIMELINE
Ln 40, Col 19 Spaces: 2 UTF-8 CRLF Python 3.10.11 64-bit
ENG IN
14:24 24-04-2024


File Edit Selection View Go Run Terminal Help
feature_extraction
EXPLORER
FEATURE_EXTRACTION
__pycache__
dataset
create_dataset.py
final_dataframe.csv
top-1m.csv
urlfeatureextraction.py
create_dataset.py
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import sklearn
import seaborn as sns
import re
from urllib.parse import urlparse,urlencode
import ipaddress
from urlfeatureextraction import *
feature_names = ['Domain', 'Have_IP', 'Have_At', 'URL_Length', 'URL_Depth','Redirection',
'https_Domain', 'TinyURL', 'Prefix/Suffix', 'DNS_Record', 'Web_Traffic',
'Domain_Age', 'Domain_End','IframeRedirection','StatusBarCust','DisableRightClick','WebsiteForwarding',
'LinksPointingToPage', 'GoogleIndex', 'Label']
data1 = pd.read_csv('./dataset/Benign_list_big_final.csv',nrows=1000,header=None)
data1.columns =['URL']
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Domain Have_IP Have_At URL_Length URL_Depth ... DisableRightClick WebsiteForwarding LinksPointingToPage GoogleIndex Label
0 v2.email-marketing.adminsimple.com 0 0 1 2 ... 1 1 -1 1 1
1 bid.openx.net 0 0 1 1 ... 1 0 1 1 1
2 webmail2.centurytel.net 0 0 1 3 ... 1 0 1 1 1
3 google.com.ng 0 0 1 1 ... 1 0 -1 1 1
4 webmail2.centurytel.net 0 0 1 3 ... 1 0 1 1 1
[5 rows x 20 columns]
C:\Users\TANMAY\Desktop\url_phishing_detection\feature_extraction>
OUTLINE
TIMELINE
Ln 41, Col 11 Spaces: 2 UTF-8 CRLF Python 3.10.11 64-bit
ENG IN
16:20 24-04-2024



References

Best Python libraries for Machine Learning - GeeksforGeeks (2023). Available at: https://www.geeksforgeeks.org/best-python-libraries-for-machine-learning/ (Accessed: 25 April 2024).
colab.google (n.d). Available at: https://colab.google/ (Accessed: 25 April 2024).
URL 2016 | Datasets | Research | Canadian Institute for Cybersecurity | UNB (n.d). Available at: https://www.unb.ca/cic/datasets/url-2016.html (Accessed: 25 April 2024).
Visual Studio Code - Code Editing. Redefined (n.d). Available at: https://code.visualstudio.com/ (Accessed: 25 April 2024).
Welcome to Flask — Flask Documentation (3.0.x) (no date). Available at: https://flask.palletsprojects.com/en/3.0.x/ (Accessed: 25 April 2024).