Configuration Manual

MSc Research Project
MSc Cyber Security

Udhaya Thirunavukarasu
21215898

School of Computing
National College of Ireland

Supervisor: Vikas Sahni

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: Udhaya Thirunavukarasu

Student ID: 21215898

Programme: MSc Cyber Security **Year:** 2023-2024

Module: Msc Academic Internship

Final: Submission Due Date: 31 Jan 2024 …………………………………………………………………………………………………….………

Project Title: A Combinational Approach of Hybrid Model BiLSTM-CNN-GRU to Improve the Detection rate of Click jacking in Websites

Word Count: ……………13…………………… **Page Count:** …………1382………..…….………

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: …………………Udhaya Thirunavukarasu……………………………

Date: ………………………28-01-2024………………………………………

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

Udhaya Thirunavukarasu
21215898

1 Introduction

This document contains all the information and details about the software technology, tools that are used during this research project that is used in the idea of improving the detection rate using CNN-BiLSTM hybrid model in the data preparation , feature extraction, implementation and evaluation phases.

2 System specification

Code Editor –
VS code studio 1.85.0
Google Colab
Program Language
python – version >3.7
Web Brower – Google chrome

Hardware Specification
Ram – 4GB
Disk Space – Minimum 2GB
OS – Windows 10 and above
NVIDIA GPU driver version: Windows 461.33 or higher

3 Package Details

Python libraries used are pandas, matplotlib,scikit learn, seaborn, numpy, tensorflow, keras, beautifulsoup, flask and etc.

Numpy – Version 1.21
Used for numerical operations in python. Supports handling large number dimensional arrays and matrices.
Pandas – Version 1.3.5
Used for data manipulation and offer structured dataframes.
Matplotlib – Version – 3.4
Used for 2D ploting visuals in python.
Seaborn – Version 0.12.0
Used for statistical data visualization in python like graphs.
Scikit learn – Version – 0.22
This library contribute to various machine learning workflows.
Keras – Version 3
Used for high level neural network API.
Tenserflow – Version 2.11.0
It is a opensource machine learning library used for deep-learning frameworks so that keras lib can run above it.
Beautiful soup – Version 2.12.2
It is used for pulling out from htlm files.

Flask – Version 3.0
 Used for micro web framework for web application in python.

4 Dataset

The CIC-IDS 2016 URL dataset, curated by the Canadian Institute for Cybersecurity[1], is a comprehensive collection designed to address the challenges posed by malicious URLs on the web. The dataset encompasses two distinct URL categories: Benign and Phishing. Over 35,300 benign URLs from Alexa's top websites were gathered, and 10,000 phishing URLs were taken from OpenPhish. Alexa-ranked websites hosting concealed malicious content.

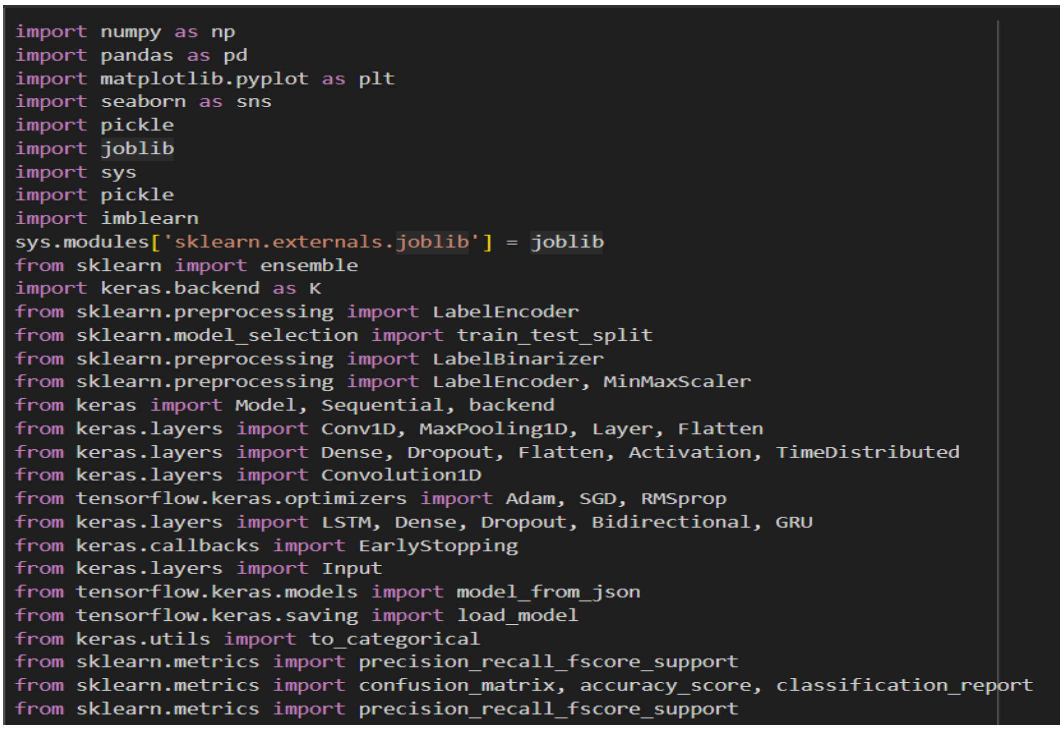
Created a single dataset from taking considerable amount of 800 legitimate urls and 850 phishing urls and created single csv file.

5 Implementation

This sections describes about the implementation process step and step procedure.

Step 1: Open Google Colab and mount the drive for the desired code location

Step 2: Import all the necessary libraries that required like numpy, pandas, matplotlib, pickle, imblearn, seaborn, keras and tensorflow.

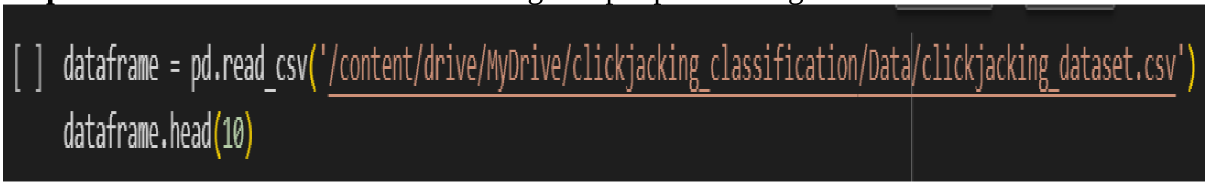


```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import pickle
import joblib
import sys
import pickle
import imblearn
sys.modules['sklearn.externals.joblib'] = joblib
from sklearn import ensemble
import keras.backend as K
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelBinarizer
from sklearn.preprocessing import LabelEncoder, MinMaxScaler
from keras import Model, Sequential, backend
from keras.layers import Conv1D, MaxPooling1D, Layer, Flatten
from keras.layers import Dense, Dropout, Flatten, Activation, TimeDistributed
from keras.layers import Convolution1D
from tensorflow.keras.optimizers import Adam, SGD, RMSprop
from keras.layers import LSTM, Dense, Dropout, Bidirectional, GRU
from keras.callbacks import EarlyStopping
from keras.layers import Input
from tensorflow.keras.models import model_from_json
from tensorflow.keras.saving import load_model
from keras.utils import to_categorical
from sklearn.metrics import precision_recall_fscore_support
from sklearn.metrics import confusion_matrix, accuracy_score, classification_report
from sklearn.metrics import precision_recall_fscore_support
```

Figure 1- Importing libraries

[1] https://www.unb.ca/cic/datasets/url-2016.html

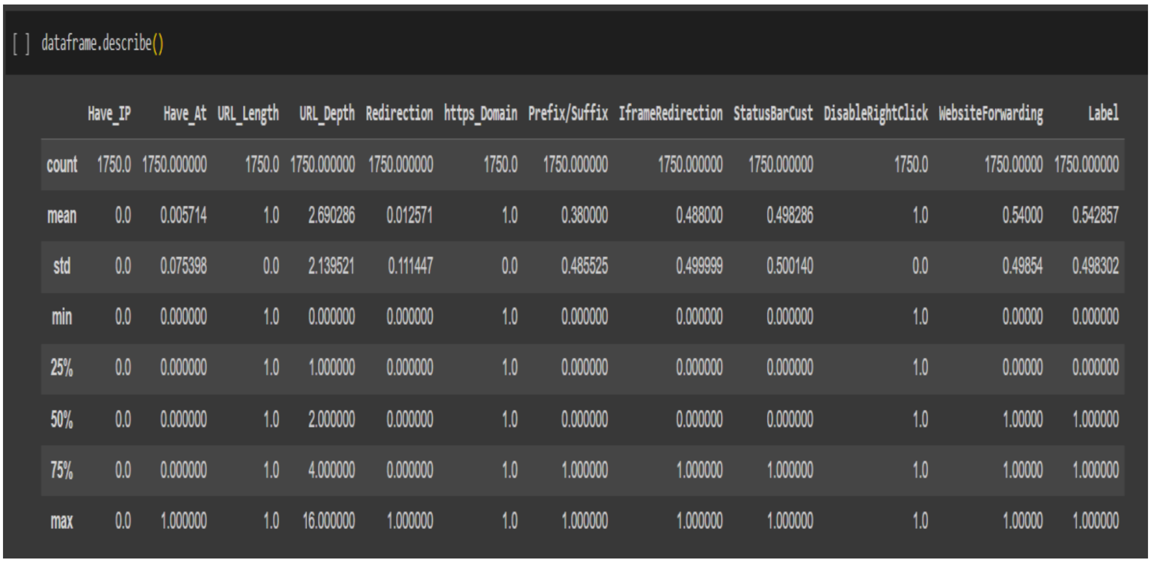
Step 3- Dataset is loaded for data clearing and pre proccessing



```
[ ] dataframe = pd.read_csv('/content/drive/MyDrive/clickjacking_classification/Data/clickjacking_dataset.csv')
    dataframe.head(10)
```

Figure 2- dataset loaded

Step 4- Loaded dataset summary is viewed for shape of the distribution of each column's values

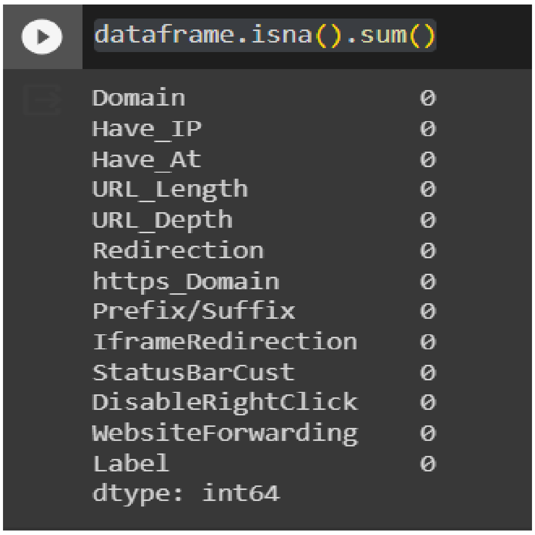


```
[ ] dataframe.describe()
```

	Have_IP	Have_At	URL_Length	URL_Depth	Redirection	https_Domain	Prefix/Suffix	IframeRedirection	StatusBarCust	DisableRightClick	WebsiteForwarding	Label
count	1750.0	1750.000000	1750.0	1750.000000	1750.000000	1750.0	1750.000000	1750.000000	1750.000000	1750.0	1750.00000	1750.000000
mean	0.0	0.005714	1.0	2.690286	0.012571	1.0	0.380000	0.488000	0.498286	1.0	0.54000	0.542857
std	0.0	0.075398	0.0	2.139521	0.111447	0.0	0.485525	0.499999	0.500140	0.0	0.49854	0.498302
min	0.0	0.000000	1.0	0.000000	0.000000	1.0	0.000000	0.000000	0.000000	1.0	0.00000	0.000000
25%	0.0	0.000000	1.0	1.000000	0.000000	1.0	0.000000	0.000000	0.000000	1.0	0.00000	0.000000
50%	0.0	0.000000	1.0	2.000000	0.000000	1.0	0.000000	0.000000	0.000000	1.0	1.00000	1.000000
75%	0.0	0.000000	1.0	4.000000	0.000000	1.0	1.000000	1.000000	1.000000	1.0	1.00000	1.000000
max	0.0	1.000000	1.0	16.000000	1.000000	1.0	1.000000	1.000000	1.000000	1.0	1.00000	1.000000

Figure 3 - Description of dataframe

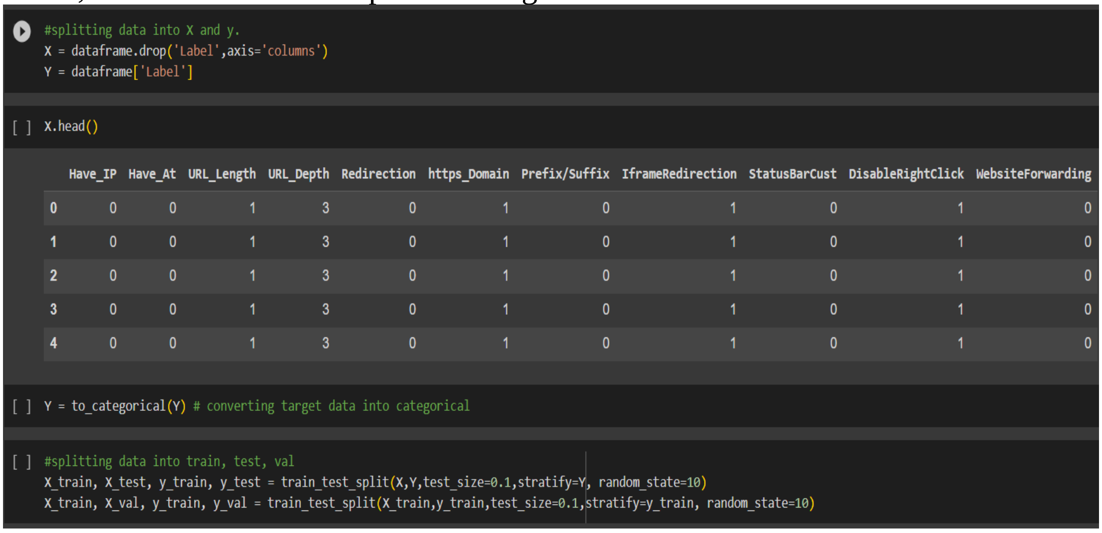
Once data clear is done, Normalized data which is with no null values and missing values present in dataframe.



```
dataframe.isna().sum()

Domain               0
Have_IP              0
Have_At              0
URL_Length           0
URL_Depth            0
Redirection          0
https_Domain         0
Prefix/Suffix        0
IframeRedirection    0
StatusBarCust        0
DisableRightClick    0
WebsiteForwarding    0
Label                0
dtype: int64
```

Figure 4 - Data Normalization

Step 6 – Data pre processing is carried out to split the available dataset into training, validation, and test sets. This helps in training the model



```
#splitting data into X and y.
X = dataframe.drop('Label',axis='columns')
Y = dataframe['Label']
```

```
[ ] X.head()
```

```
   Have_IP  Have_At  URL_Length  URL_Depth  Redirection  https_Domain  Prefix/Suffix  IframeRedirection  StatusBarCust  DisableRightClick  WebsiteForwarding
0        0        0           1          3            0             1              0                  1              0                  1                  0
1        0        0           1          3            0             1              0                  1              0                  1                  0
2        0        0           1          3            0             1              0                  1              0                  1                  0
3        0        0           1          3            0             1              0                  1              0                  1                  0
4        0        0           1          3            0             1              0                  1              0                  1                  0
```

```
[ ] Y = to_categorical(Y) # converting target data into categorical
```

```
[ ] #splitting data into train, test, val
    X_train, X_test, y_train, y_test = train_test_split(X,Y,test_size=0.1,stratify=Y, random_state=10)
    X_train, X_val, y_train, y_val = train_test_split(X_train,y_train,test_size=0.1,stratify=y_train, random_state=10)
```

Figure 5- Splitting the data

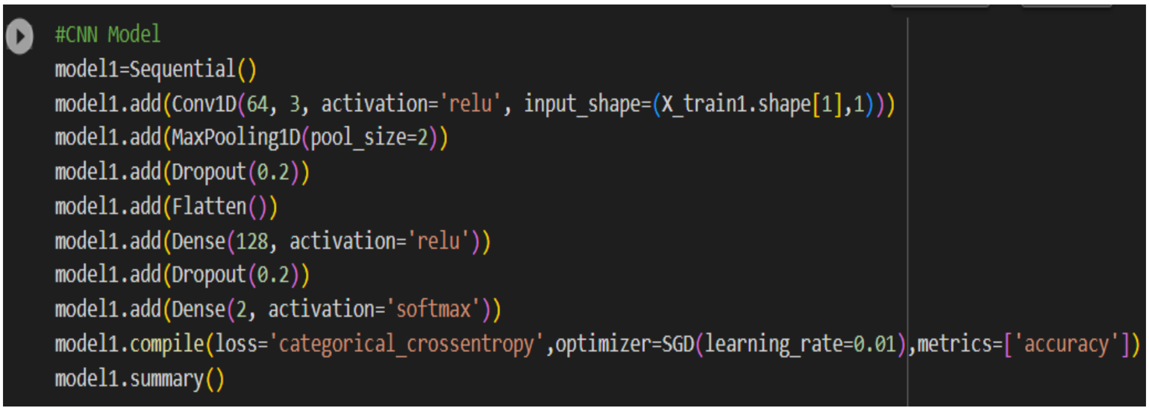
10% of the original dataset was allocated to the testing set, and an additional 10% of the training set was designated for the validation set. The remaining data was utilized for training the machine learning model.

6 Training Models

For this project, CNN and Hybrid model (CNN+BiLSTM+GRU) training models are deployed to see better balance between sensitivity and specificity, making it a robust choice for this classification.

6.1 CNN Model-

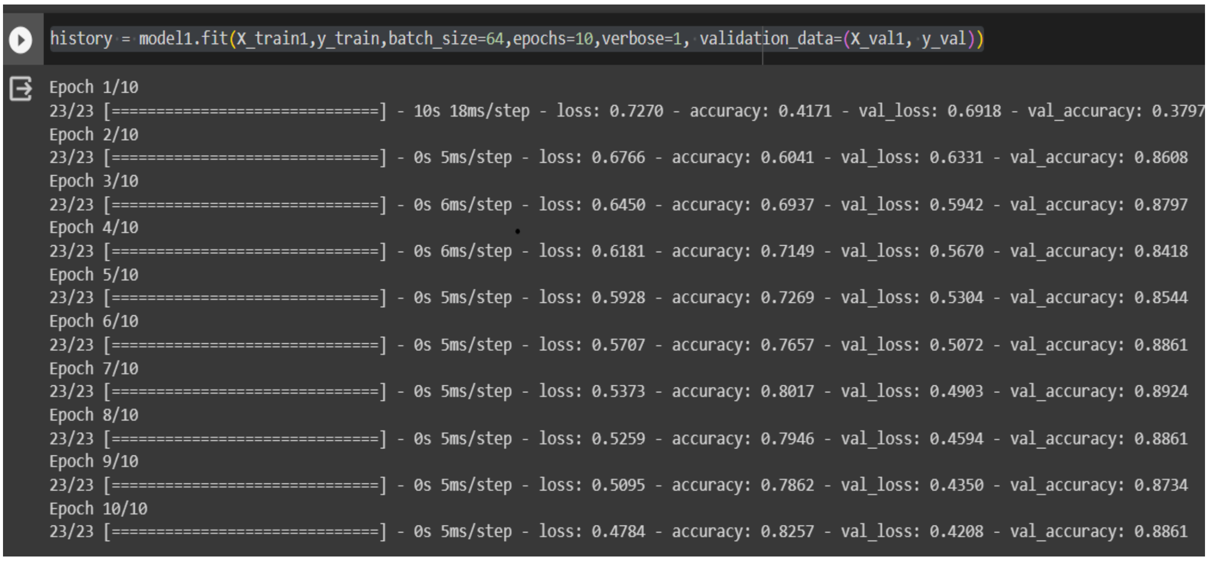
- The necessary libraries are imported for the CNN train model



```
#CNN Model
model1=Sequential()
model1.add(Conv1D(64, 3, activation='relu', input_shape=(X_train1.shape[1],1)))
model1.add(MaxPooling1D(pool_size=2))
model1.add(Dropout(0.2))
model1.add(Flatten())
model1.add(Dense(128, activation='relu'))
model1.add(Dropout(0.2))
model1.add(Dense(2, activation='softmax'))
model1.compile(loss='categorical_crossentropy',optimizer=SGD(learning_rate=0.01),metrics=['accuracy'])
model1.summary()
```

Figure 6- Importing the libraries

- The training data is trained in CNN model

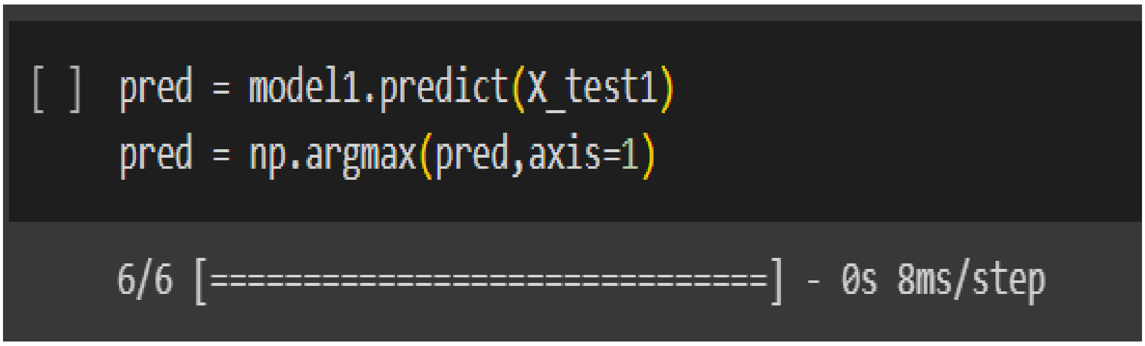


```
history = model1.fit(X_train1,y_train,batch_size=64,epochs=10,verbose=1, validation_data=(X_val1, y_val))
```

```
Epoch 1/10
23/23 [==============================] - 10s 18ms/step - loss: 0.7270 - accuracy: 0.4171 - val_loss: 0.6918 - val_accuracy: 0.3797
Epoch 2/10
23/23 [==============================] - 0s 5ms/step - loss: 0.6766 - accuracy: 0.6041 - val_loss: 0.6331 - val_accuracy: 0.8608
Epoch 3/10
23/23 [==============================] - 0s 6ms/step - loss: 0.6450 - accuracy: 0.6937 - val_loss: 0.5942 - val_accuracy: 0.8797
Epoch 4/10
23/23 [==============================] - 0s 6ms/step - loss: 0.6181 - accuracy: 0.7149 - val_loss: 0.5670 - val_accuracy: 0.8418
Epoch 5/10
23/23 [==============================] - 0s 5ms/step - loss: 0.5928 - accuracy: 0.7269 - val_loss: 0.5304 - val_accuracy: 0.8544
Epoch 6/10
23/23 [==============================] - 0s 5ms/step - loss: 0.5707 - accuracy: 0.7657 - val_loss: 0.5072 - val_accuracy: 0.8861
Epoch 7/10
23/23 [==============================] - 0s 5ms/step - loss: 0.5373 - accuracy: 0.8017 - val_loss: 0.4903 - val_accuracy: 0.8924
Epoch 8/10
23/23 [==============================] - 0s 5ms/step - loss: 0.5259 - accuracy: 0.7946 - val_loss: 0.4594 - val_accuracy: 0.8861
Epoch 9/10
23/23 [==============================] - 0s 5ms/step - loss: 0.5095 - accuracy: 0.7862 - val_loss: 0.4350 - val_accuracy: 0.8734
Epoch 10/10
23/23 [==============================] - 0s 5ms/step - loss: 0.4784 - accuracy: 0.8257 - val_loss: 0.4208 - val_accuracy: 0.8861
```

Figure 7 - CNN training the data

- The process of making predictions using the trained model in CNN. It can make predictions on new or unseen data through this step



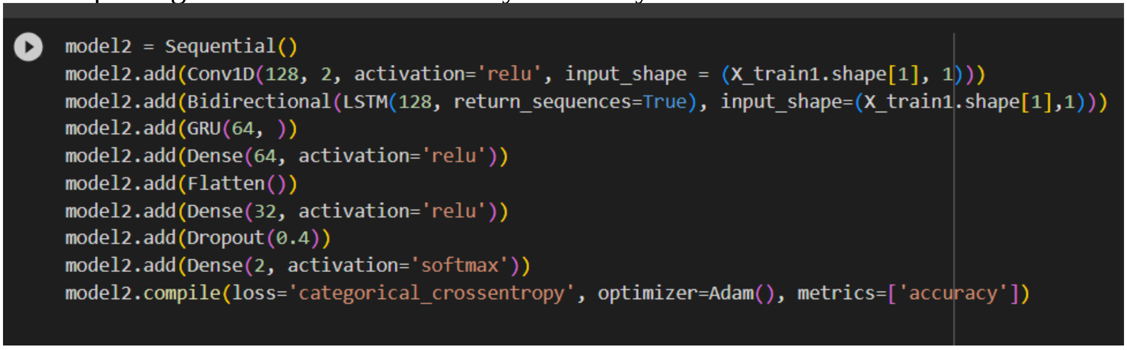
```
[ ]  pred = model1.predict(X_test1)
     pred = np.argmax(pred,axis=1)
```

```
6/6 [==============================] - 0s 8ms/step
```

Figure 8- Prediction

6.2 CNN+BiLSTM+GRU

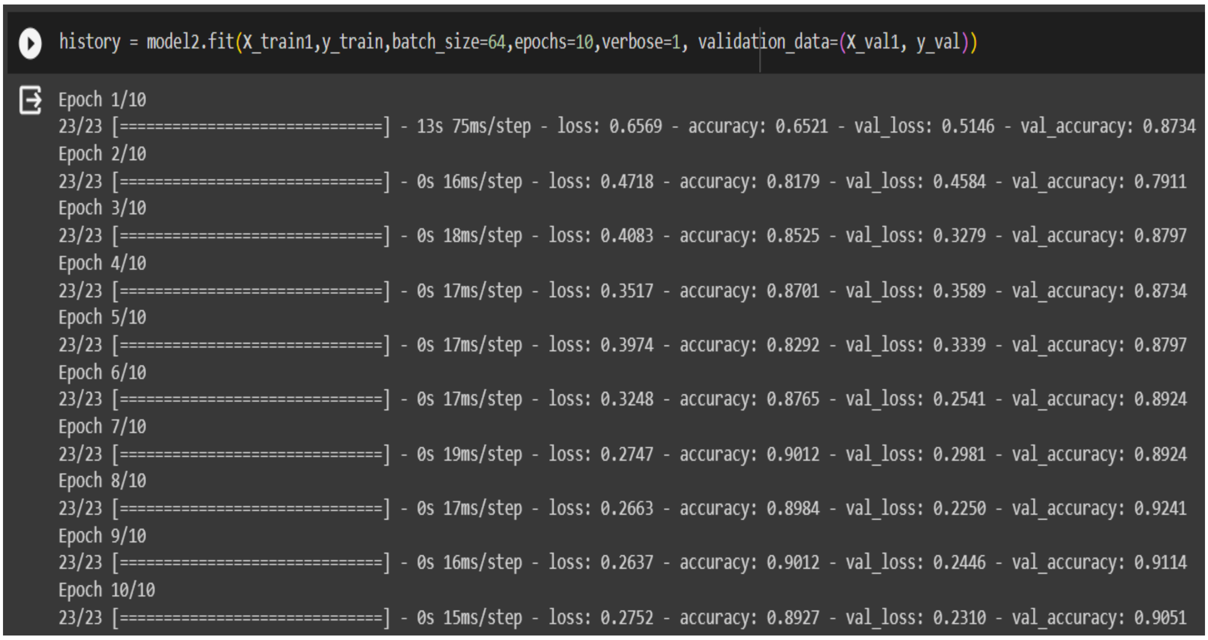
- Importing all the libraries necessary for the hybrid model for CNN+BiLSTM+GRU



```
model2 = Sequential()
model2.add(Conv1D(128, 2, activation='relu', input_shape = (X_train1.shape[1], 1)))
model2.add(Bidirectional(LSTM(128, return_sequences=True), input_shape=(X_train1.shape[1],1)))
model2.add(GRU(64, ))
model2.add(Dense(64, activation='relu'))
model2.add(Flatten())
model2.add(Dense(32, activation='relu'))
model2.add(Dropout(0.4))
model2.add(Dense(2, activation='softmax'))
model2.compile(loss='categorical_crossentropy', optimizer=Adam(), metrics=['accuracy'])
```

Figure 9 - Importing libraries

- Training data is trained in hybrid model and the training history, which contains information about loss and accuracy on both the training and validation sets for each epoch

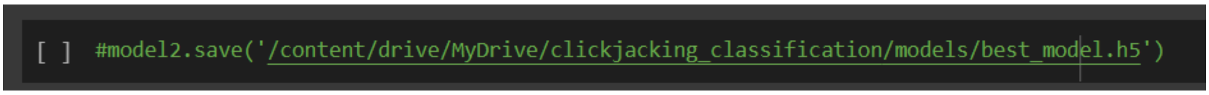


```
history = model2.fit(X_train1,y_train,batch_size=64,epochs=10,verbose=1, validation_data=(X_val1, y_val))
```

```
Epoch 1/10
23/23 [==============================] - 13s 75ms/step - loss: 0.6569 - accuracy: 0.6521 - val_loss: 0.5146 - val_accuracy: 0.8734
Epoch 2/10
23/23 [==============================] - 0s 16ms/step - loss: 0.4718 - accuracy: 0.8179 - val_loss: 0.4584 - val_accuracy: 0.7911
Epoch 3/10
23/23 [==============================] - 0s 18ms/step - loss: 0.4083 - accuracy: 0.8525 - val_loss: 0.3279 - val_accuracy: 0.8797
Epoch 4/10
23/23 [==============================] - 0s 17ms/step - loss: 0.3517 - accuracy: 0.8701 - val_loss: 0.3589 - val_accuracy: 0.8734
Epoch 5/10
23/23 [==============================] - 0s 17ms/step - loss: 0.3974 - accuracy: 0.8292 - val_loss: 0.3339 - val_accuracy: 0.8797
Epoch 6/10
23/23 [==============================] - 0s 17ms/step - loss: 0.3248 - accuracy: 0.8765 - val_loss: 0.2541 - val_accuracy: 0.8924
Epoch 7/10
23/23 [==============================] - 0s 19ms/step - loss: 0.2747 - accuracy: 0.9012 - val_loss: 0.2981 - val_accuracy: 0.8924
Epoch 8/10
23/23 [==============================] - 0s 17ms/step - loss: 0.2663 - accuracy: 0.8984 - val_loss: 0.2250 - val_accuracy: 0.9241
Epoch 9/10
23/23 [==============================] - 0s 16ms/step - loss: 0.2637 - accuracy: 0.9012 - val_loss: 0.2446 - val_accuracy: 0.9114
Epoch 10/10
23/23 [==============================] - 0s 15ms/step - loss: 0.2752 - accuracy: 0.8927 - val_loss: 0.2310 - val_accuracy: 0.9051
```

Figure 10- Hybrid model training on data

- Once the data is trained in model and the model is saved as best_model file.



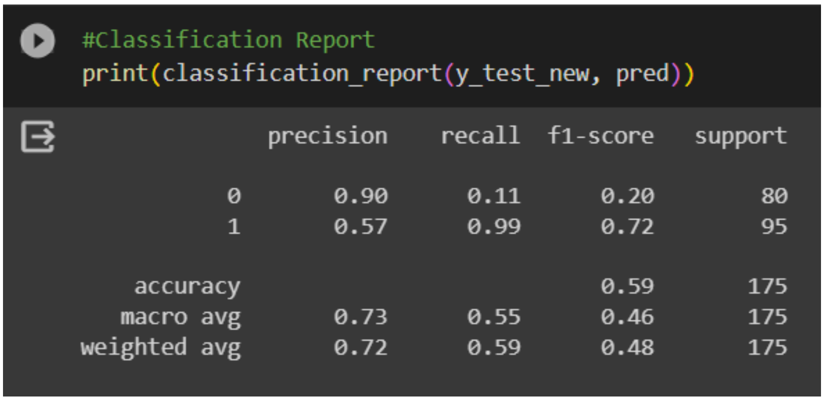
```
[ ]  #model2.save('/content/drive/MyDrive/clickjacking_classification/models/best_model.h5')
```

Figure 11 - Saving the model

7 Evaluation

7.1 CNN Evaluation

- The classification report is a helpful tool for evaluating the overall performance of a CNN model.

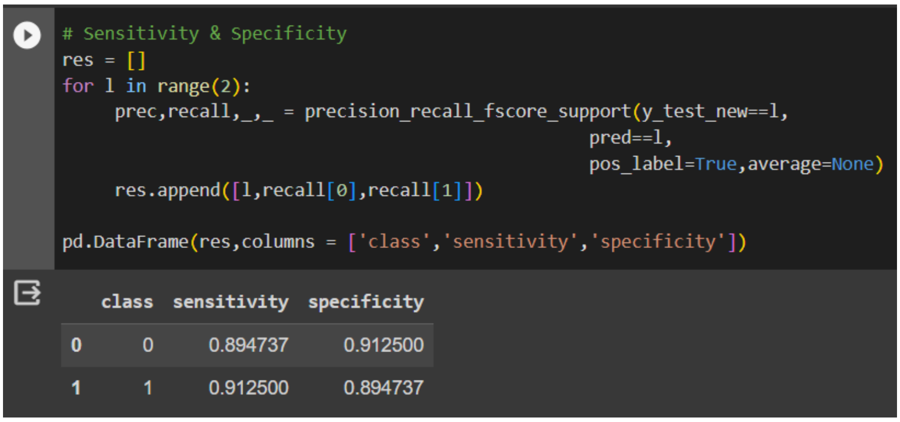


```
#Classification Report
print(classification_report(y_test_new, pred))
```

```
              precision    recall  f1-score   support

           0       0.90      0.11      0.20        80
           1       0.57      0.99      0.72        95

    accuracy                           0.59       175
   macro avg       0.73      0.55      0.46       175
weighted avg       0.72      0.59      0.48       175
```

Figure 12- Classification Report

- sensitivity and specificity values for each class, giving an evaluation of the model's performance on each class individually.



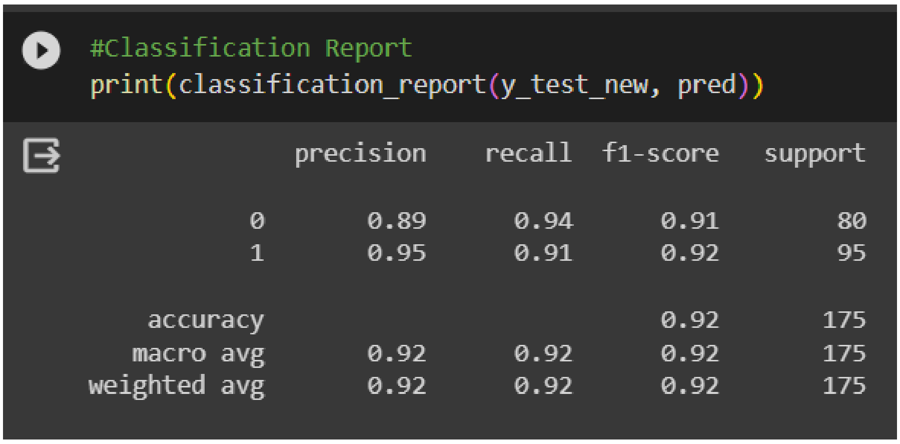
```
# Sensitivity & Specificity
res = []
for l in range(2):
     prec,recall,_,_ = precision_recall_fscore_support(y_test_new==l,
                                                      pred==l,
                                                      pos_label=True,average=None)

     res.append([l,recall[0],recall[1]])

pd.DataFrame(res,columns = ['class','sensitivity','specificity'])
```

```
   class  sensitivity  specificity
0      0     0.894737     0.912500
1      1     0.912500     0.894737
```

Figure 13 - Sensitivity & Specificity

7.2 CNN+BiLSTM+GRU

- The classification is done for evaluating the overall performance of this hybrid model.

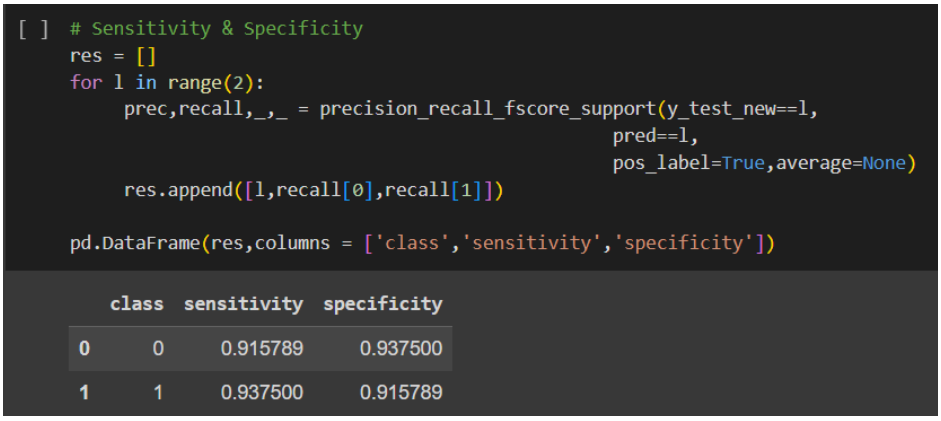


```
#Classification Report
print(classification_report(y_test_new, pred))
```

```
              precision    recall  f1-score   support

           0       0.89      0.94      0.91        80
           1       0.95      0.91      0.92        95

    accuracy                           0.92       175
   macro avg       0.92      0.92      0.92       175
weighted avg       0.92      0.92      0.92       175
```

Figure 14 - Classification report of CNN+BiLSTM+GRU

- Here calculates sensitivity and specificity for each class (0 and 1) and presents the results in a DataFrame of Hybird Model



```
# Sensitivity & Specificity
res = []
for l in range(2):
     prec,recall,_,_ = precision_recall_fscore_support(y_test_new==l,
                                                      pred==l,
                                                      pos_label=True,average=None)

     res.append([l,recall[0],recall[1]])

pd.DataFrame(res,columns = ['class','sensitivity','specificity'])
```

```
   class  sensitivity  specificity
0      0     0.915789     0.937500
1      1     0.937500     0.915789
```

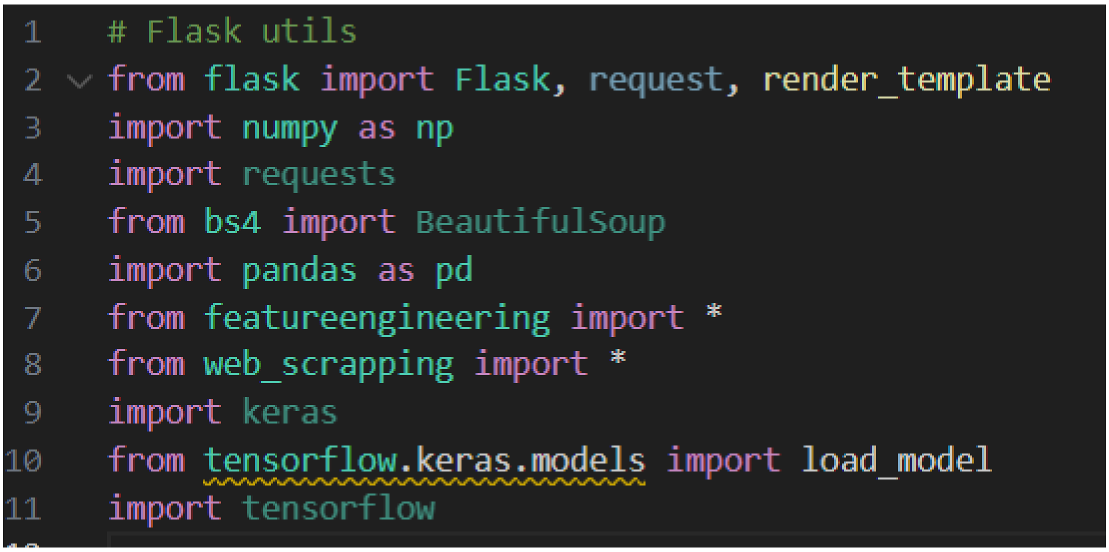
Figure 15 - Sensitive & Specificity of CNN+BiLSTM+GRU

- Both models perform Performed well, however the CNN + GRU + BiLSTM model shows a marginally better overall performance. This model demonstrates a stronger balance between specificity and sensitivity than the CNN Model, underscoring its resilience in the phishing detection job. In order to reduce false positives, it is crucial to achieve higher specificity in phishing detection, and the CNN + GRU + BiLSTM model performs exceptionally well in this area for both classes. Overall, the CNN + GRU + BiLSTM model is a good option for this classification task given on the evaluation metrics and the particular needs of phishing detection.

8 Graphical User Interface- Web Page

The Html webpage(GUI) is designed to interact with the trained model. Web scraper scrapes all url link present in dummy clickjack web page using beautifulsoup and receive all extracted url features to the training model.

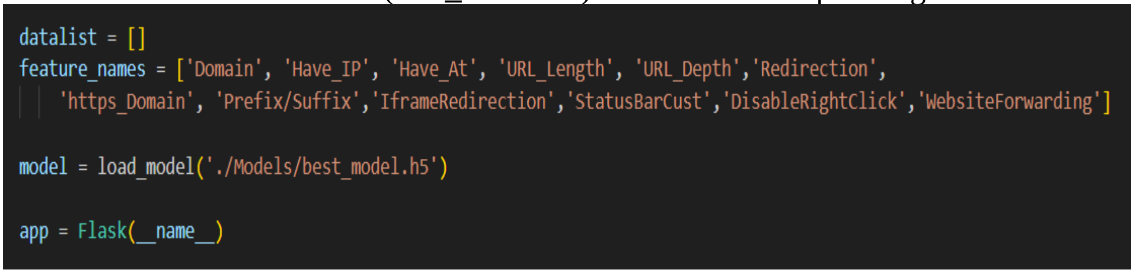
- Importing the necessary libraries and modules, including Flask for web development, NumPy for numerical operations, BeautifulSoup for web scraping, Pandas for data manipulation, and a pre-trained Keras/TensorFlow model for phishing detection.



```python
# Flask utils
from flask import Flask, request, render_template
import numpy as np
import requests
from bs4 import BeautifulSoup
import pandas as pd
from featureengineering import *
from web_scrapping import *
import keras
from tensorflow.keras.models import load_model
import tensorflow
```

Figure 16 - Importing libraries

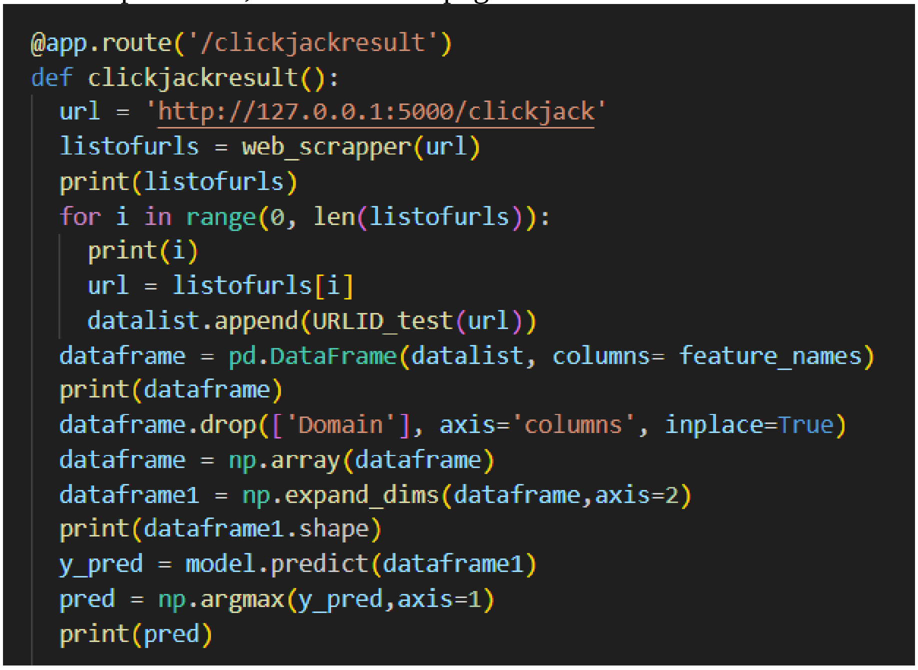
- Creating a an empty list (datalist) to store data extracted from URLs. Feature names is a list of feature names used in the model.
- Pre-trained Keras model (best_model.h5) is loaded for the phishing detection.



```python
datalist = []
feature_names = ['Domain', 'Have_IP', 'Have_At', 'URL_Length', 'URL_Depth','Redirection',
    'https_Domain', 'Prefix/Suffix','IframeRedirection','StatusBarCust','DisableRightClick','WebsiteForwarding']

model = load_model('./Models/best_model.h5')

app = Flask(__name__)
```

Figure 17 - Loading trained model

- Once all setup is done , A HTML webpage is created with iframes embedded in it.

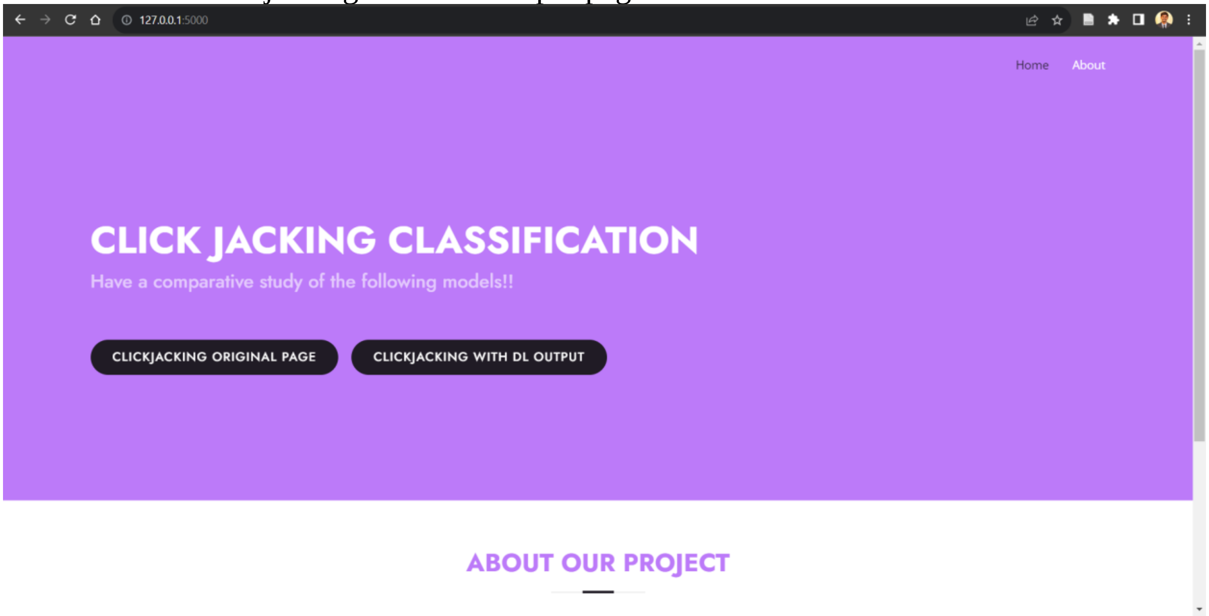


```
@app.route('/clickjackresult')
def clickjackresult():
  url = 'http://127.0.0.1:5000/clickjack'
  listofurls = web_scrapper(url)
  print(listofurls)
  for i in range(0, len(listofurls)):
    print(i)
    url = listofurls[i]
    datalist.append(URLID_test(url))
  dataframe = pd.DataFrame(datalist, columns= feature_names)
  print(dataframe)
  dataframe.drop(['Domain'], axis='columns', inplace=True)
  dataframe = np.array(dataframe)
  dataframe1 = np.expand_dims(dataframe,axis=2)
  print(dataframe1.shape)
  y_pred = model.predict(dataframe1)
  pred = np.argmax(y_pred,axis=1)
  print(pred)
```

Figure 18 - Scrape and process URLs for clickjacking detection using a pre-trained Hybird machine

- This webpage is designed to showcase potential clickjacking websites using iframes and provides a form for testing machine learning models. It includes a home page and two sub pages .

1. Index
2. Clickjacking Original page
3. Clickjacking with ML Output page

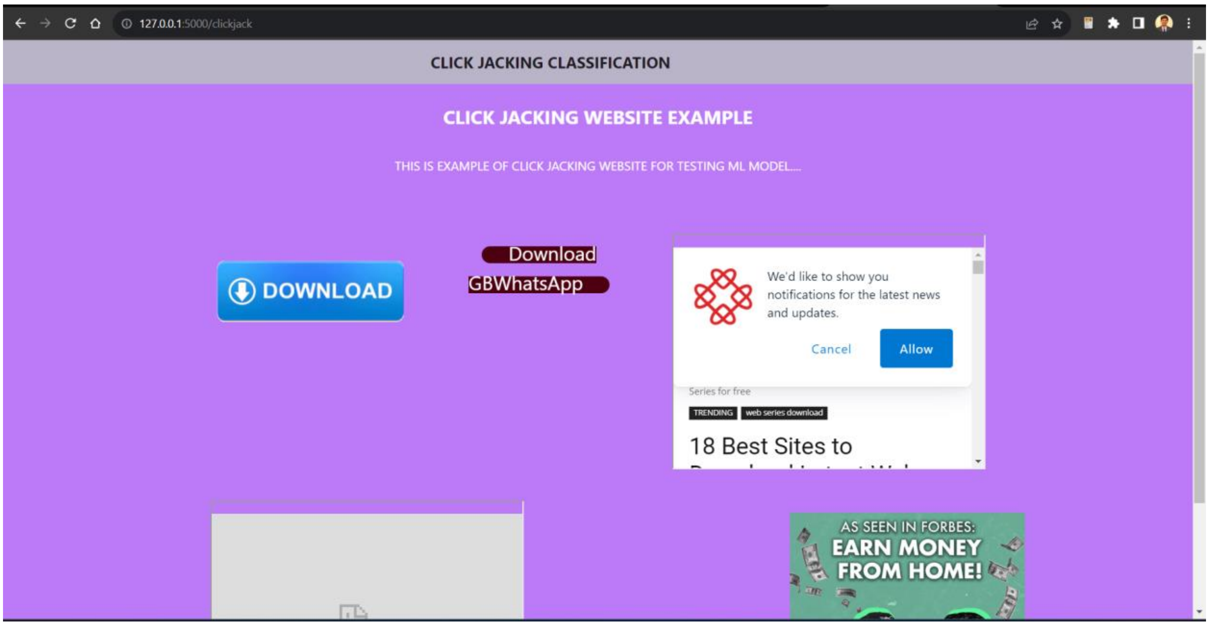


CLICK JACKING CLASSIFICATION

Have a comparative study of the following models!!

ABOUT OUR PROJECT

Figure 19 - Home page


127.0.0.1:5000/clickjack

CLICK JACKING CLASSIFICATION

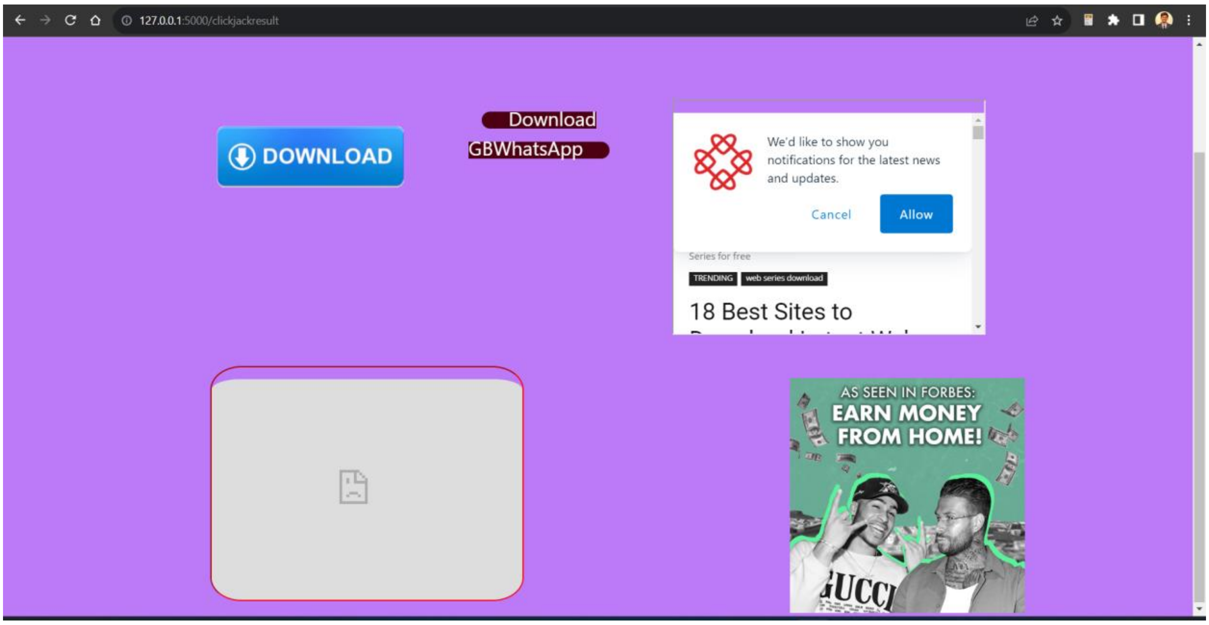
CLICK JACKING WEBSITE EXAMPLE

THIS IS EXAMPLE OF CLICK JACKING WEBSITE FOR TESTING ML MODEL....

We'd like to show you notifications for the latest news and updates.

Figure 20 - Webpage with malicious iframes

- Once the Hybrid CNN+BiLSTM+GRU Prediction code is executed and the Malicious Iframe is hightlighted with red boarders with html elements.


127.0.0.1:5000/clickjackresult

We'd like to show you notifications for the latest news and updates.

Figure 21 - Malicious Clickjacking iframe is highlighted