ARTIFICIAL INTELLIGENCE INTERNSHIP

PROJECT REPORT

PROJECT TITLE:

Web Phishing Detection Using IBM Watson

TEAM MEMBERS:

Ambuj Gupta

Karan Rochlani

Harshal Vijay Ghadge

DVS Hitesh Reddy

TABLE OF CONTENTS

1) INTRODUCTION

1.1 OVERVIEW

There are a number of users who purchase products online and make payments through e-banking. There are e-banking websites that ask users to provide sensitive data such as username, password & credit card details, etc., often for malicious reasons.

This type of e-banking website is known as a phishing website. Web service is one of the key communications software services for the Internet.

Web phishing is one of many security threats to web services on the Internet.
Common threats of web phishing:

- Web phishing aims to steal private information, such as usernames, passwords, and credit card details, by way of impersonating a legitimate entity.

- It will lead to information disclosure and property damage.

- Large organizations may get trapped in different kinds of scams.

2.2 PURPOSE

- In order to detect and predict e-banking phishing websites, we proposed an intelligent, flexible and effective system that is based on using classification algorithms.

- We implemented classification algorithms and techniques to extract the phishing datasets criteria to classify their legitimacy.

- The e-banking phishing website can be detected based on some important characteristics like URL and domain identity, and security and encryption criteria in the final phishing detection rate.
- Once a user makes a transaction online when he makes payment through an e-banking website our system will use a data mining algorithm to detect whether the e-banking website is a phishing website or not.

2) LITERATURE SURVEY:

2.2. PROBLEM STATEMENT:

- You’ll be able to understand the problem to classify if it is a regression or a classification kind of problem.
- You will be able to know how to pre-process/clean the data using different data pre-processing techniques.
- Applying different algorithms according to the dataset
- You will be able to know how to find the accuracy of the model.
- You will be able to build web applications using the Flask framework.

2.3. SOLUTION:

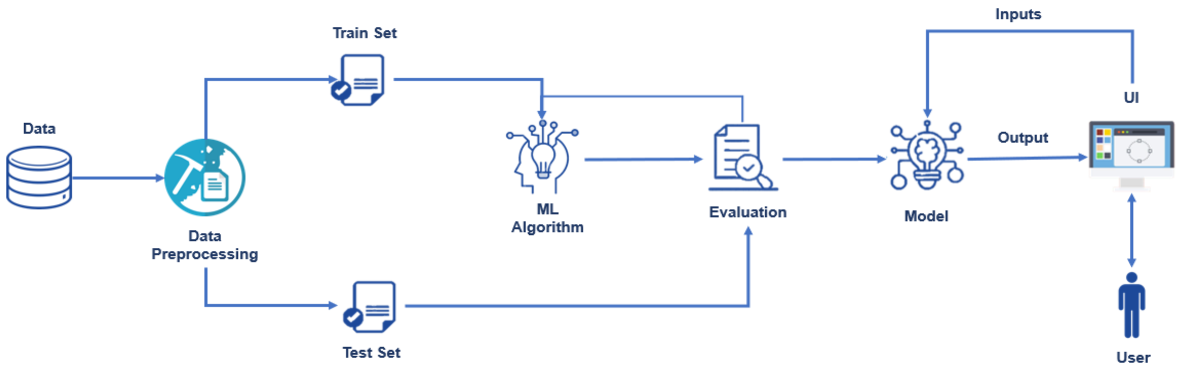
- Download the dataset.
- Preprocess or clean the data.
- Analyze the pre-processed data.
- Train the machine with preprocessed data using an appropriate machine learning algorithm.
-
- Save the model and its dependencies.
- Build a Web application using a flask that integrates with the model built.

- These are the steps for this project:
    - Installing the required packages and libraries.

- Importing the required libraries for the model to run.
- Downloading the dataset, feeding it to the model, and understanding the dataset
- Data Preprocessing – Checking for outliers and null values. If there any null values we use Label Encoding to convert them into binary format.
- Dividing the model into Train and Test data. Fitting the model and predicting.
- Building Flask Web Application.

3) THEORITICAL ANALYSIS:

3.1. BLOCK DIAGRAM:



3.2. SPECIFICATIONS:

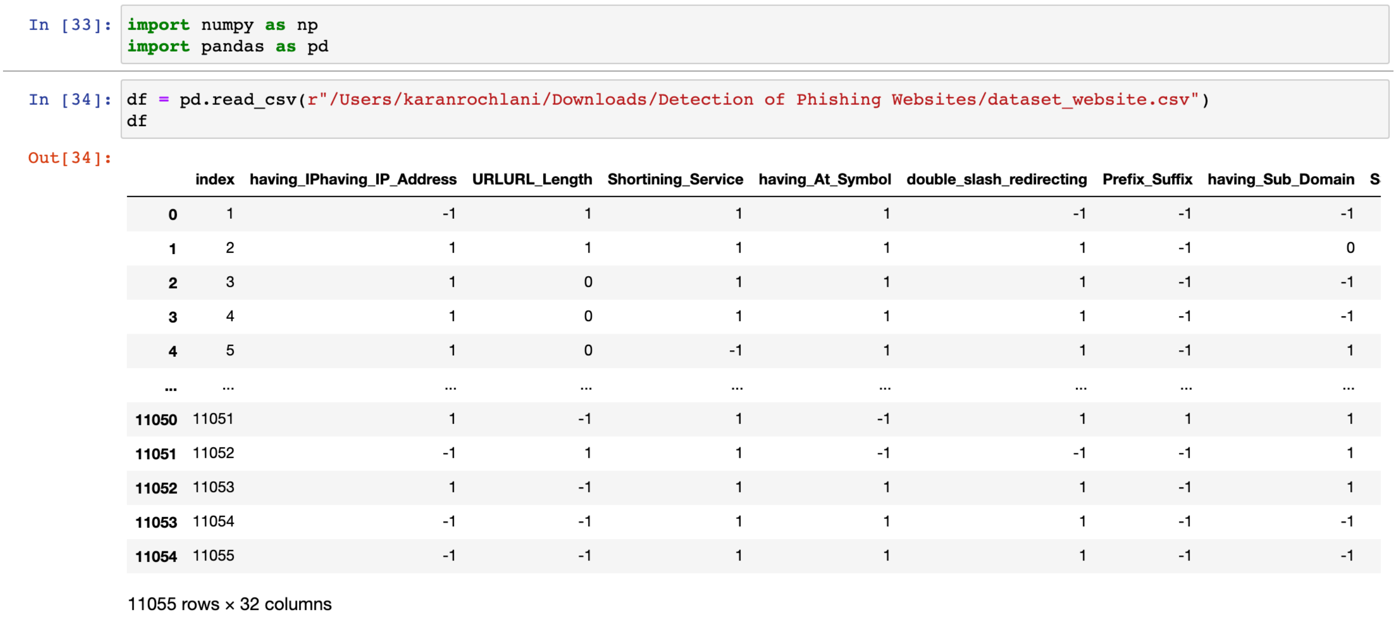
A) HARDWARE: Laptop / Computer

B) SOFTWARE: Python, HTML (Hyper Text Markup Language), CSS (Cascading Style Sheets), TensorFlow, Keras, Spyder, Jupyter Notebook etc.

4) EXPERIMENTAL INVESTIGATIONS:

The following shows the pseudo code for the proposed loan prediction method.

1. Load the data



```
In [33]: import numpy as np
         import pandas as pd
```

```
In [34]: df = pd.read_csv(r"/Users/karanrochlani/Downloads/Detection of Phishing Websites/dataset_website.csv")
         df
```

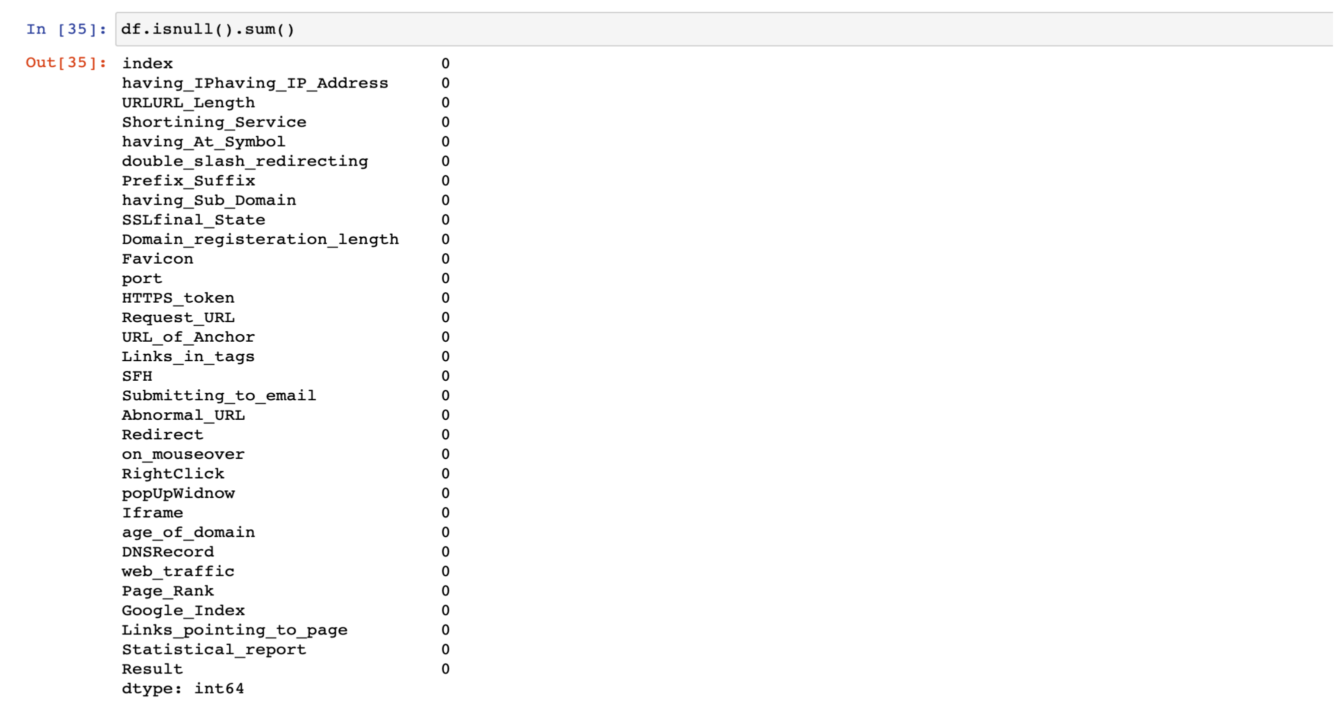
Out[34]:

	index	having_IPhaving_IP_Address	URLURL_Length	Shortining_Service	having_At_Symbol	double_slash_redirecting	Prefix_Suffix	having_Sub_Domain	S
0	1	-1	1	1	1	-1	-1	-1	
1	2	1	1	1	1	1	-1	0	
2	3	1	0	1	1	1	-1	-1	
3	4	1	0	1	1	1	-1	-1	
4	5	1	0	-1	1	1	-1	1	
...	...	...	...	...	...	...	...	...	
11050	11051	1	-1	1	-1	1	1	1	
11051	11052	-1	1	1	-1	-1	-1	1	
11052	11053	1	-1	1	1	1	-1	1	
11053	11054	-1	-1	1	1	1	-1	-1	
11054	11055	-1	-1	1	1	1	-1	-1	

11055 rows × 32 columns

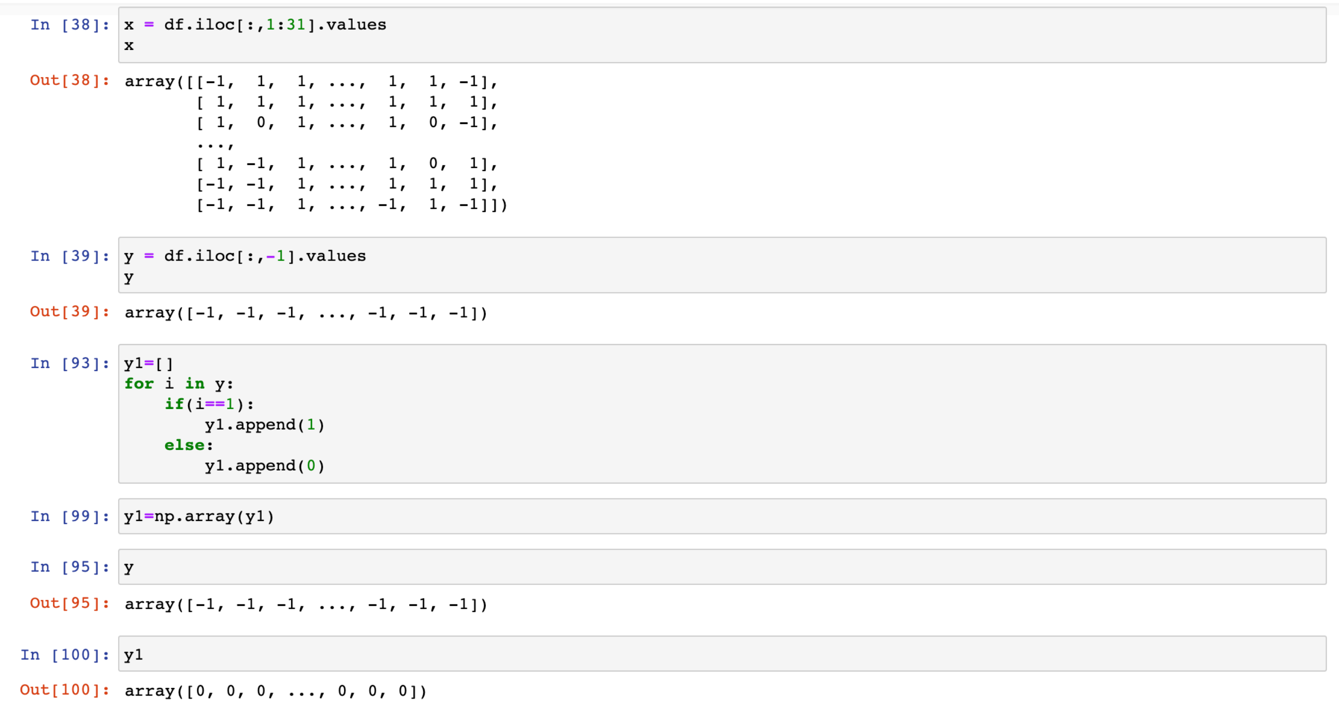
2. Data cleaning and pre-processing.

a) Check for null values in the dataset



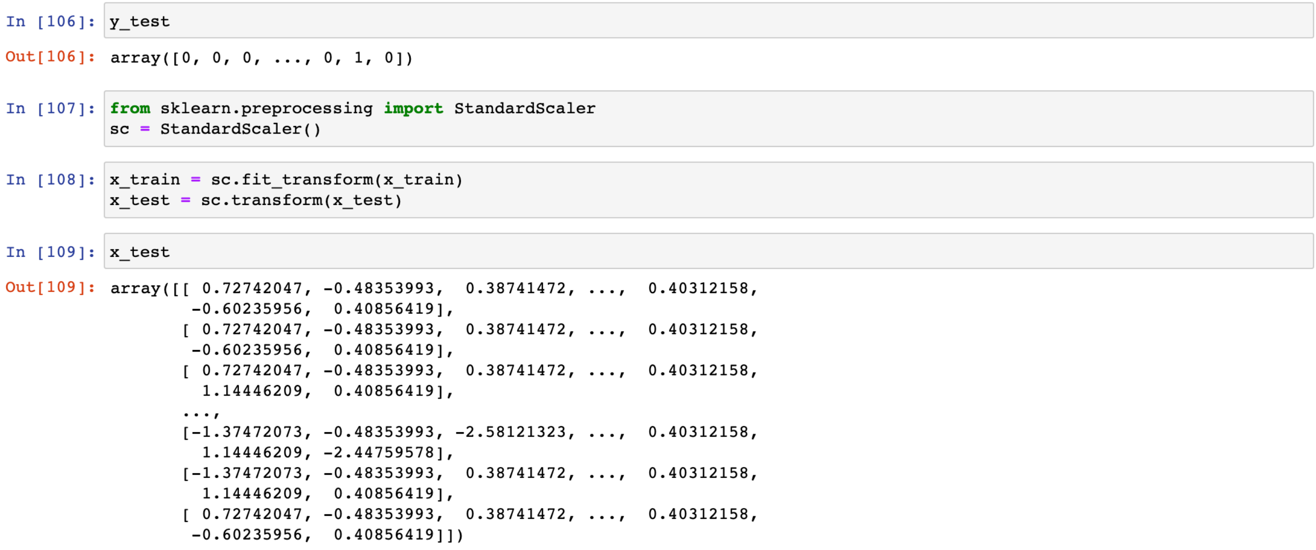
```
In [35]: df.isnull().sum()

Out[35]: index                          0
         having_IPhaving_IP_Address     0
         URLURL_Length                  0
         Shortining_Service             0
         having_At_Symbol               0
         double_slash_redirecting       0
         Prefix_Suffix                  0
         having_Sub_Domain              0
         SSLfinal_State                 0
         Domain_registeration_length    0
         Favicon                        0
         port                           0
         HTTPS_token                    0
         Request_URL                    0
         URL_of_Anchor                  0
         Links_in_tags                  0
         SFH                            0
         Submitting_to_email            0
         Abnormal_URL                   0
         Redirect                       0
         on_mouseover                   0
         RightClick                     0
         popUpWidnow                    0
         Iframe                         0
         age_of_domain                  0
         DNSRecord                      0
         web_traffic                    0
         Page_Rank                      0
         Google_Index                   0
         Links_pointing_to_page         0
         Statistical_report             0
         Result                         0
         dtype: int64
```

b) Find the x and y as per which model will be trained and tested.



```
In [38]:  x = df.iloc[:,1:31].values
          x

Out[38]:  array([[-1,  1,  1, ...,  1,  1, -1],
                 [ 1,  1,  1, ...,  1,  1,  1],
                 [ 1,  0,  1, ...,  1,  0, -1],
                 ...,
                 [ 1, -1,  1, ...,  1,  0,  1],
                 [-1, -1,  1, ...,  1,  1,  1],
                 [-1, -1,  1, ..., -1,  1, -1]])

In [39]:  y = df.iloc[:,-1].values
          y

Out[39]:  array([-1, -1, -1, ..., -1, -1, -1])

In [93]:  y1=[]
          for i in y:
              if(i==1):
                  y1.append(1)
              else:
                  y1.append(0)

In [99]:  y1=np.array(y1)

In [95]:  y

Out[95]:  array([-1, -1, -1, ..., -1, -1, -1])

In [100]: y1

Out[100]: array([0, 0, 0, ..., 0, 0, 0])
```

C) Standardise the data using the Standard Scalar.



```
In [106]: y_test

Out[106]: array([0, 0, 0, ..., 0, 1, 0])

In [107]: from sklearn.preprocessing import StandardScaler
          sc = StandardScaler()

In [108]: x_train = sc.fit_transform(x_train)
          x_test = sc.transform(x_test)

In [109]: x_test

Out[109]: array([[ 0.72742047, -0.48353993,  0.38741472, ...,  0.40312158,
                  -0.60235956,  0.40856419],
                 [ 0.72742047, -0.48353993,  0.38741472, ...,  0.40312158,
                  -0.60235956,  0.40856419],
                 [ 0.72742047, -0.48353993,  0.38741472, ...,  0.40312158,
                   1.14446209,  0.40856419],
                 ...,
                 [-1.37472073, -0.48353993, -2.58121323, ...,  0.40312158,
                   1.14446209, -2.44759578],
                 [-1.37472073, -0.48353993,  0.38741472, ...,  0.40312158,
                   1.14446209,  0.40856419],
                 [ 0.72742047, -0.48353993,  0.38741472, ...,  0.40312158,
                  -0.60235956,  0.40856419]])
```

3. Determine the training and testing data

```
In [101]: from sklearn.model_selection import train_test_split
          x_train,x_test,y_train,y_test = train_test_split(x,y1,test_size=0.25,random_state=0)

In [102]: x_train.shape

Out[102]: (8291, 30)

In [103]: y_train.shape

Out[103]: (8291,)

In [104]: x_test.shape

Out[104]: (2764, 30)

In [105]: y_test.shape

Out[105]: (2764,)

In [106]: y_test

Out[106]: array([0, 0, 0, ..., 0, 1, 0])
```

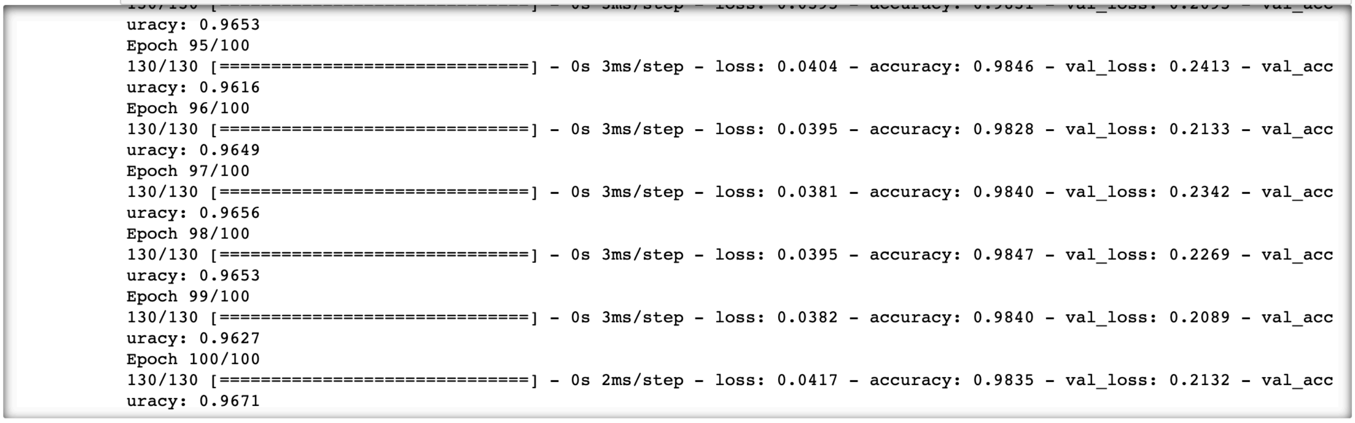
4. Apply the modelling for prediction using Artificial Neural Networks (ANN)

a) Model Building

```
In [121]: from tensorflow.keras.models import Sequential
          from tensorflow.keras.layers import Dense,Dropout

In [122]: Regression_Model = Sequential()

In [123]: Regression_Model.add(Dense(units=30, kernel_initializer="random_uniform", activation="relu"))

In [124]: Regression_Model.add(Dense(units=60, kernel_initializer="random_uniform", activation="relu"))

In [125]: Regression_Model.add(Dropout(0.2))

In [126]: Regression_Model.add(Dense(units=60, kernel_initializer="random_uniform", activation="relu"))

In [127]: Regression_Model.add(Dropout(0.2))

In [128]: Regression_Model.add(Dense(units=1, kernel_initializer="random_uniform", activation="sigmoid"))

In [129]: Regression_Model.compile(optimizer = "rmsprop",loss ="binary_crossentropy", metrics = ["accuracy"] )
```

b) Model Training

```
In [130]: history = Regression_Model.fit(x_train,y_train, batch_size =64,epochs = 100,validation_data=(x_test, y_test))
```



```
130/130 [==============================] - 0s 3ms/step - loss: 0.0393 - accuracy: 0.9851 - val_loss: 0.2093 - val_acc
uracy: 0.9653
Epoch 95/100
130/130 [==============================] - 0s 3ms/step - loss: 0.0404 - accuracy: 0.9846 - val_loss: 0.2413 - val_acc
uracy: 0.9616
Epoch 96/100
130/130 [==============================] - 0s 3ms/step - loss: 0.0395 - accuracy: 0.9828 - val_loss: 0.2133 - val_acc
uracy: 0.9649
Epoch 97/100
130/130 [==============================] - 0s 3ms/step - loss: 0.0381 - accuracy: 0.9840 - val_loss: 0.2342 - val_acc
uracy: 0.9656
Epoch 98/100
130/130 [==============================] - 0s 3ms/step - loss: 0.0395 - accuracy: 0.9847 - val_loss: 0.2269 - val_acc
uracy: 0.9653
Epoch 99/100
130/130 [==============================] - 0s 3ms/step - loss: 0.0382 - accuracy: 0.9840 - val_loss: 0.2089 - val_acc
uracy: 0.9627
Epoch 100/100
130/130 [==============================] - 0s 2ms/step - loss: 0.0417 - accuracy: 0.9835 - val_loss: 0.2132 - val_acc
uracy: 0.9671
```

5. Determine the accuracy

```
In [131]: ypred2 = Regression_Model.predict(x_test)
```

```
In [135]: y_test
```

```
Out[135]: array([0, 0, 0, ..., 0, 1, 0])
```

```
In [148]: (ypred2>0.5).round()
```

```
Out[148]: array([[0.],
                 [0.],
                 [1.],
                 ...,
                 [0.],
                 [1.],
                 [0.]], dtype=float16)
```

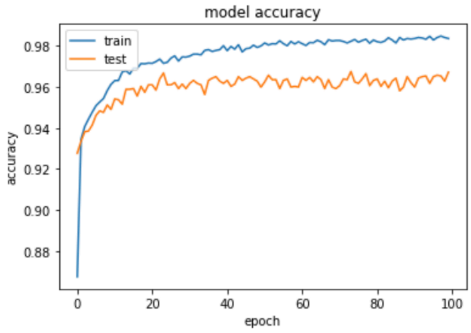
```
In [136]: from sklearn.metrics import accuracy_score
          accuracy=accuracy_score(y_test,ypred2.round())
```

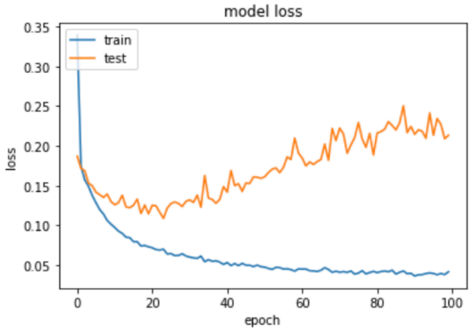
```
In [137]: accuracy
```

```
Out[137]: 0.9670767004341534
```

6) Building the graphs.

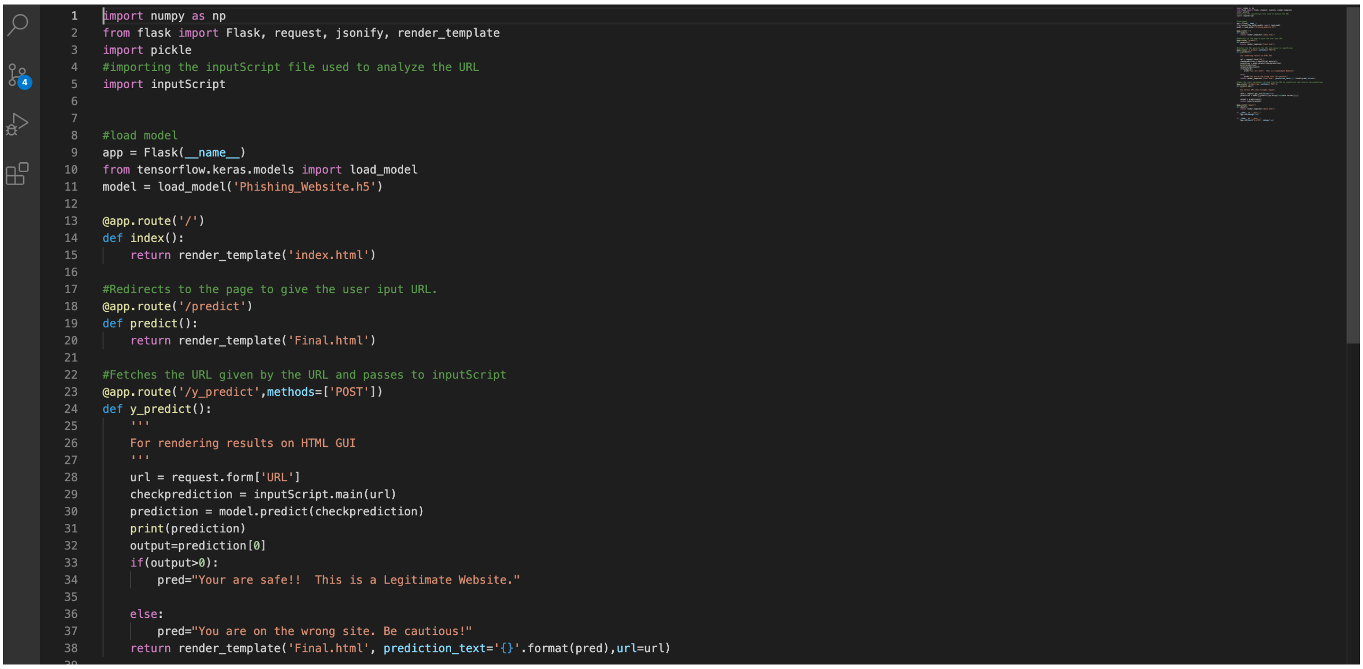
```
In [149]: import matplotlib.pyplot as plt
          plt.plot(history.history['accuracy'])
          plt.plot(history.history['val_accuracy'])
          plt.title('model accuracy')
          plt.ylabel('accuracy')
          plt.xlabel('epoch')
          plt.legend(['train', 'test'], loc='upper left')
          plt.show()
          # summarize history for loss
          plt.plot(history.history['loss'])
          plt.plot(history.history['val_loss'])
          plt.title('model loss')
          plt.ylabel('loss')
          plt.xlabel('epoch')
          plt.legend(['train', 'test'], loc='upper left')
          plt.show()
```



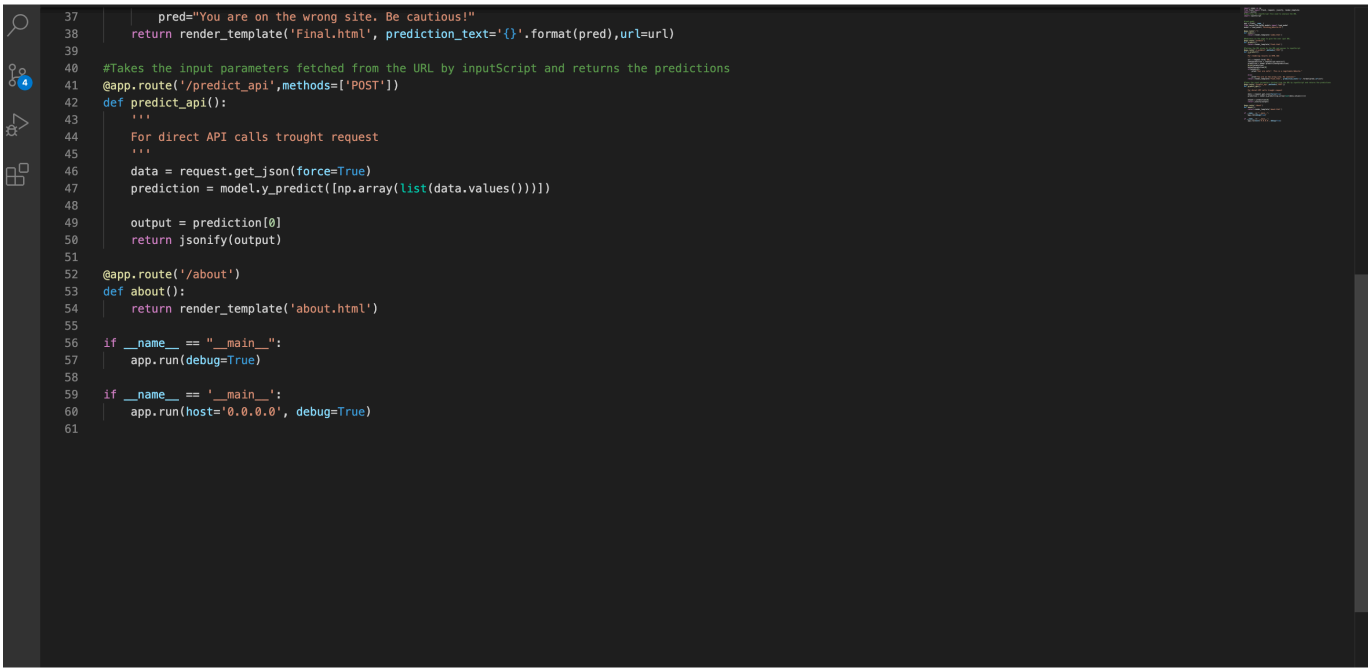


7)Import the model to Flask and link with the Web Applications.

```
In [138]: type(Regression_Model)

Out[138]: tensorflow.python.keras.engine.sequential.Sequential

In [139]: Regression_Model.save('Phishing_Website.h5')

In [140]: Regression_Model.summary()

          Model: "sequential_6"
          _________________________________________________________________
          Layer (type)                 Output Shape              Param #   
          =================================================================
          dense_24 (Dense)             (None, 30)                930       
          _________________________________________________________________
          dense_25 (Dense)             (None, 60)                1860      
          _________________________________________________________________
          dropout_12 (Dropout)         (None, 60)                0         
          _________________________________________________________________
          dense_26 (Dense)             (None, 60)                3660      
          _________________________________________________________________
          dropout_13 (Dropout)         (None, 60)                0         
          _________________________________________________________________
          dense_27 (Dense)             (None, 1)                 61        
          =================================================================
          Total params: 6,511
          Trainable params: 6,511
          Non-trainable params: 0
          _________________________________________________________________

In [141]: from keras.models import load_model
          model = load_model('Phishing_Website.h5')

In [142]: pred=model.predict([[1, 1, 1, 1, 1, -1, 1, -1, -1, -1, -1, 1, -1, 0, 0, -1, -1, -1, 0, -1, -1, 1, -1, -1, -1, -1, 1, -

In [145]: (pred[0]>0.5).round()

Out[145]: array([0.], dtype=float16)
```

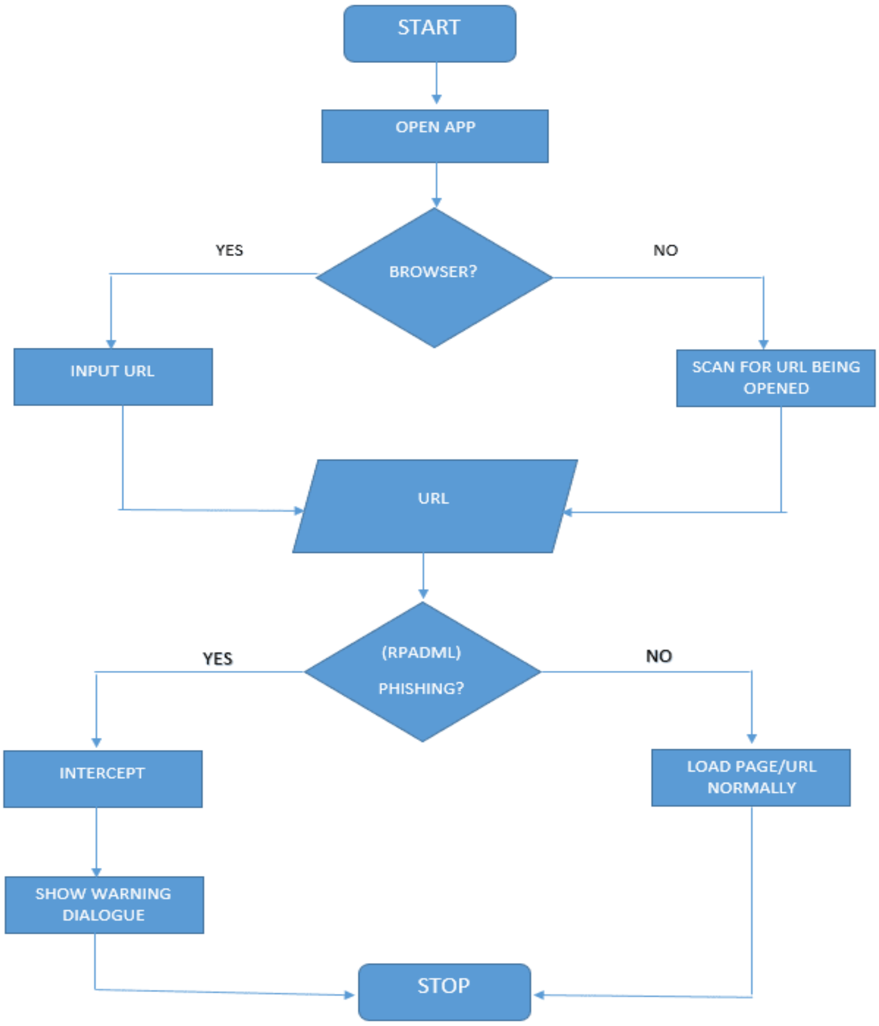


```python
import numpy as np
from flask import Flask, request, jsonify, render_template
import pickle
#importing the inputScript file used to analyze the URL
import inputScript


#load model
app = Flask(__name__)
from tensorflow.keras.models import load_model
model = load_model('Phishing_Website.h5')

@app.route('/')
def index():
    return render_template('index.html')

#Redirects to the page to give the user iput URL.
@app.route('/predict')
def predict():
    return render_template('Final.html')

#Fetches the URL given by the URL and passes to inputScript
@app.route('/y_predict',methods=['POST'])
def y_predict():
    '''
    For rendering results on HTML GUI
    '''
    url = request.form['URL']
    checkprediction = inputScript.main(url)
    prediction = model.predict(checkprediction)
    print(prediction)
    output=prediction[0]
    if(output>0):
        pred="Your are safe!!  This is a Legitimate Website."

    else:
        pred="You are on the wrong site. Be cautious!"
    return render_template('Final.html', prediction_text='{}'.format(pred),url=url)
```



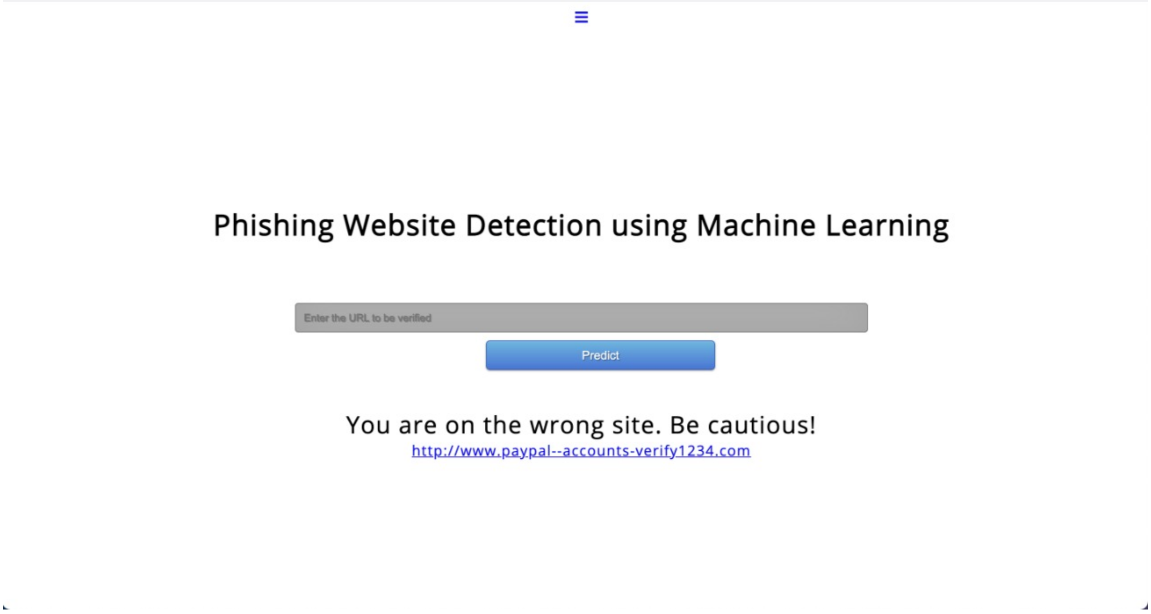
```python
        pred="You are on the wrong site. Be cautious!"
    return render_template('Final.html', prediction_text='{}'.format(pred),url=url)

#Takes the input parameters fetched from the URL by inputScript and returns the predictions
@app.route('/predict_api',methods=['POST'])
def predict_api():
    '''
    For direct API calls trought request
    '''
    data = request.get_json(force=True)
    prediction = model.y_predict([np.array(list(data.values()))])

    output = prediction[0]
    return jsonify(output)

@app.route('/about')
def about():
    return render_template('about.html')

if __name__ == "__main__":
    app.run(debug=True)

if __name__ == '__main__':
    app.run(host='0.0.0.0', debug=True)

```

5) FLOW CHART



6) RESULT:

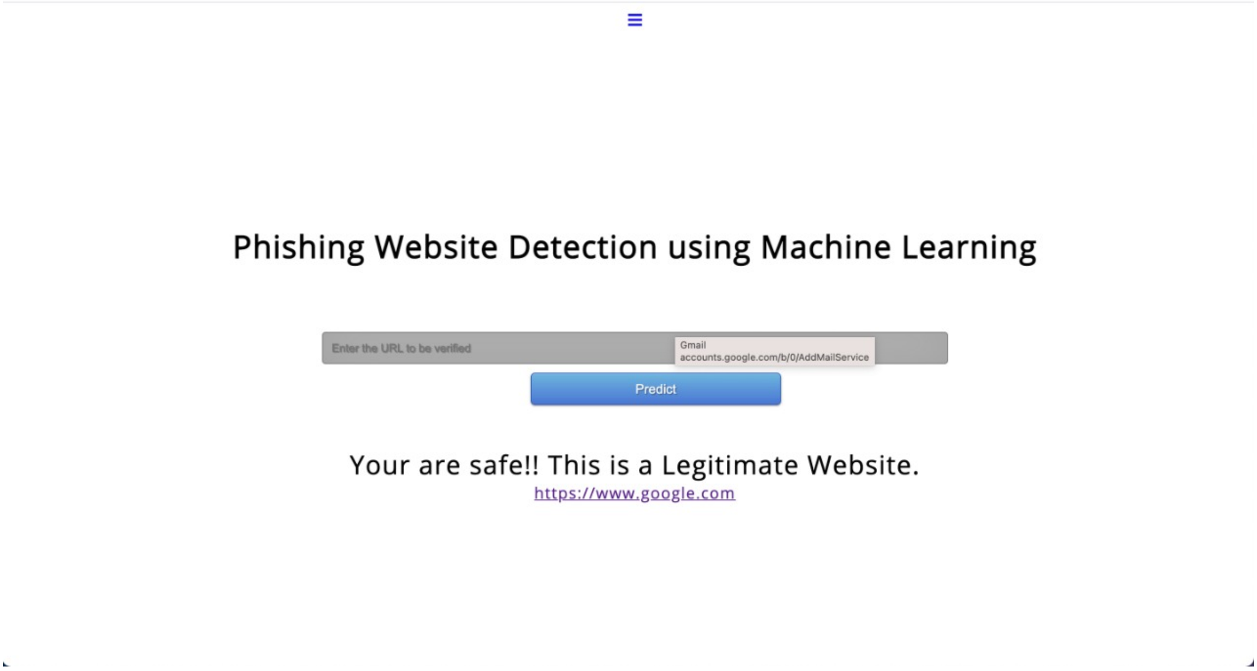
The following images shows the screenshot of our applications of Web Phishing Detection



Phishing Website Detection using Machine Learning

You are on the wrong site. Be cautious!

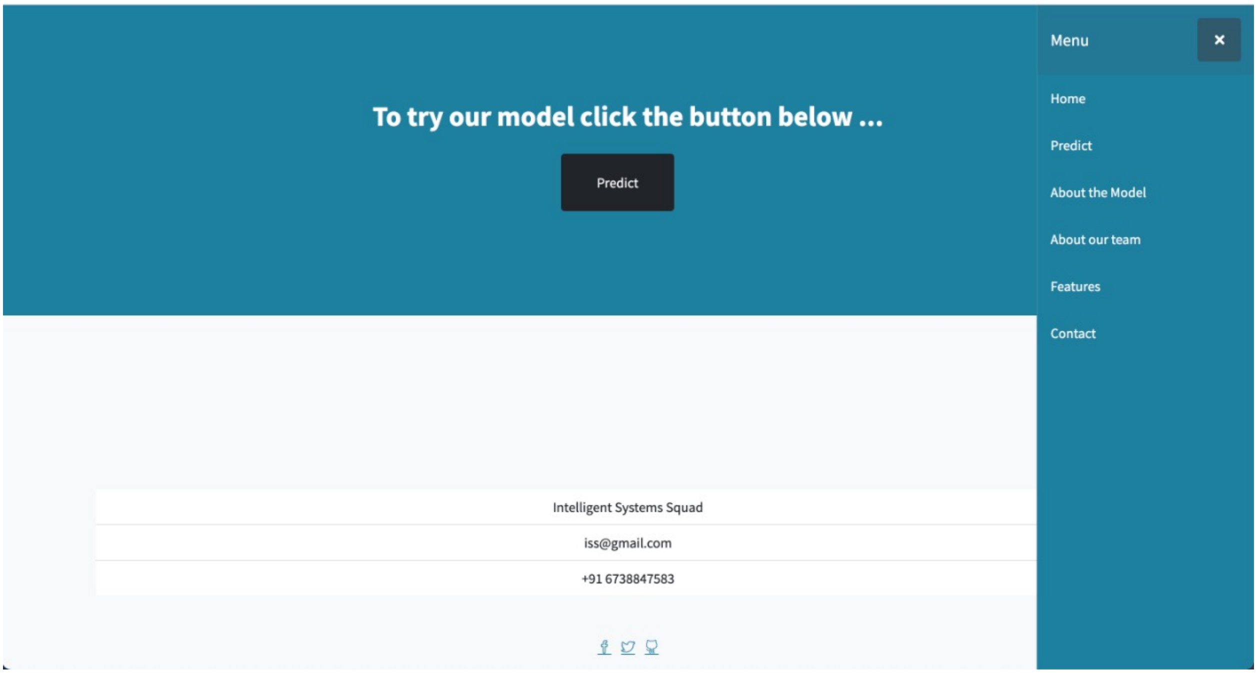
http://www.paypal--accounts-verify1234.com


≡
Enter the URL to be verified
Gmail
accounts.google.com/b/0/AddMailService
Predict

Phishing Website Detection using Machine Learning

Your are safe!! This is a Legitimate Website.

https://www.google.com

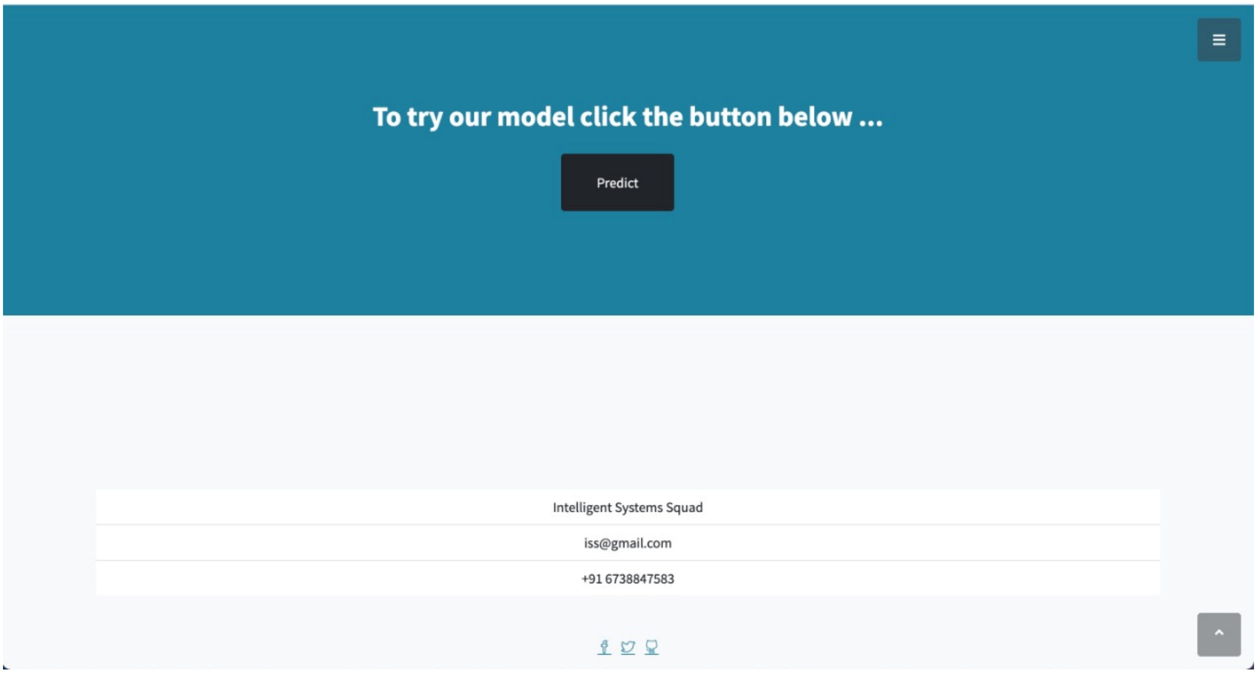


To try our model click the button below ...

Intelligent Systems Squad

iss@gmail.com

+91 6738847583



To try our model click the button below ...

Intelligent Systems Squad

iss@gmail.com

+91 6738847583

7)Advantages and Disadvantages

1) ADVANTAGES:

☺ To analyze the data the python libraries, help a lot.

☺ The accuracy of the existing model is exceptionally good.

☺Statistical and prediction is quite easy comparing to existing technologies.

☺Once a user makes a transaction online when he makes payment through an e-banking website our system will use a data mining algorithm to detect whether the e-banking website is a phishing website or not.

2) DISADVANTAGES:

☹Complexity in analyzing the data.

☹ Prediction is challenging task working in the model

☹More work should be done web UI.

☹ Coding is complex maintaining multiple methods.

8) APPLICATIONS:

➢This project web application can be used at various e-banking website.

➢It can be used for detect whether the e-banking website is a phishing website or not. the customer with data visualization.

➢It can be used at net banking system, so it can automatically predict it self whether the website phishing

9) CONCLUSION:

- ✓ So here, it can be concluded with confidence that the Artificial Neural Network (ANN) model is extremely efficient and gives a better result when compared to other models.
- ✓ It works correctly and fulfills all requirements of e-banking website. This system properly works and accurately calculates the result.
- ✓ It predicts the nature of the Web phising is one of many security threats to web services on the Internet.

10)FUTURE WORK:

- ❖ This application can be inserted into various applications regarding e-banking website
- ❖ The UI of the web Application can be developed in variety of ways to look it more attractive.
- ❖ There have been numbers cases of computer glitches and most important weight of features is fixed in automated prediction system.
- ❖ In near future this module of prediction can be integrated with the module of automated processing system.

11)BIBILOGRAPHY:

- https://towardsdatascience.com/an-introduction-to-exploratory-data-analysis-in-python- 9a76f04628b8
- https://youtu.be/ST1ZYLmYw2U

12)APPENDIX:

Source Code: Github