I

PHISHING WEBSITE DETECTION.

A

MINOR PROJECT REPORT

Submitted in partial fulfilment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

Under the supervision of

Dr. Ekta Gandotra

Associate Professor

(Department of Computer Science & Engineering and Information Technology)

By

ADITYA SINGH [201394]

SUDITI RATHORE [201327]

to

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY

WAKNAGHAT SOLAN – 173234,]

HIMACHAL PRADESH, INDIA

May, 2023

TABLE OF CONTENTS

I

STUDENT'S DECLARATION

We hereby declare that the work presented in the Project report entitled " **PHISHING WEBSITE DETECTION** " Submitted for partial fulfilment of the requirements for the degree of Bachelor of Technology in Computer Science and Engineering at the **Jaypee University of Information Technology, Waknaghat** is an authentic record of our work carried out under the supervision of **Dr. Ekta Gandotra**. This work has not been submitted elsewhere for the reward of any other degree/diploma. We are fully responsible for the contents of our project report.

Signature of Student	Signature of Student
Aditya Singh (201394)	Suditi Rathore (201327)
Department of CSE	Department of CSE

CERTIFICATE

This is to certify that the work presented in the project report titled **" PHISHING WEBSITE DETECTION "** submitted to the Department of Computer Science and Engineering**, Jaypee University of Information Technology, Waknaghat** in partial fulfilment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering is an authentic record of work carried out by **Aditya Singh (201394) and Suditi Rathore(201327)** under the supervision of **Dr. Ekta Gandotra.** To the best of our knowledge, the preceding statement is correct.

Date:

Signature of Supervisor
Dr. Ekta Gandotra
Associate Professor

ACKNOWLEDGEMENT

The success and end of this project necessitated a great deal of direction and assistance from our supervisor, and we are extremely fortunate to have received it all as part of the project's completion. We owe everything we have accomplished to her oversight and help, and we'd like to express our gratitude.

We would like to express our sincere gratitude to ***Dr. Ekta Gandotra*** for her valuable guidance, endless patience, scholarly guidance, continual encouragement, constructive criticism, energetic and kind supervision throughout the project.
We would like to extend our sincere thanks to her. We are highly indebted to her for her guidance and constant support.

ABSTRACT

Phishing attack is a simplest way to obtain sensitive information from innocent users. Aim of the phishers is to acquire critical information like username, password and bank account details. Detecting these websites is a challenging problem, as they often mimic legitimate websites and employ sophisticated techniques to avoid detection. This paper deals with machine learning technology for detection of phishing URLs by extracting and analysing various features of legitimate and phishing URLs. Decision Tree, random forest, KNN, Logistic Regression, Adaboost and Support vector machine algorithms are used to detect phishing websites. Aim of the paper is to detect phishing URLs as well as narrow down to best machine learning algorithm by comparing accuracy rate of each algorithm.

Chapter 01: INTRODUCTION

1.1 Introduction

Phishing URLs are a common tactic used by cybercriminals to steal sensitive information such as usernames, passwords, and credit card details. These URLs often appear to be legitimate websites, but in reality, they are designed to trick unsuspecting users into divulging their confidential data.

To protect against these attacks, the development of automated phishing URL detection models has become a crucial area of research. These models use machine learning algorithms to analyze the characteristics of URLs and identify those that are likely to be malicious.

In this project, we will develop a phishing URL detecting model that uses machine learning techniques to detect and classify phishing URLs with high accuracy. Our goal is to build a model that can help individuals and organizations prevent cyber attacks and protect their valuable data. By leveraging the power of machine learning, we can develop a tool that is capable of quickly and accurately identifying phishing URLs, which will help to mitigate the risk of falling victim to these types of attacks.

1.2 Objective.

- Train and evaluate various machine learning models, such as decision trees, random forests, SVM, KNN, on the dataset to identify the best performing model.
- Create an Algorithm to extract features from URL such as URL length , specific symbols or keywords.
- Develop a web-based interface that allows users to enter a URL and receive a classification of whether it is likely to be part of a phishing attack.

1.3 Motivation

The motivation for the phishing website detecting model is to address the increasing threat of phishing attacks, which can result in severe consequences such as financial loss, identity theft, and reputational damage. The development of automated detection models that use machine learning algorithms to analyze URLs can help prevent these attacks and protect users from their harmful effects. The goal of the project is to develop a robust phishing website detecting model that can accurately identify potential phishing attacks, providing users with the peace of mind that their data is secure.

1.4 Language Used

To train this model we have used Python 3.8 and its various libraries.

1.5 Technical Requirements

- Google colab
- libraries including NumPy, matplotlib, seaborn, sklearn, pandas

1.6 Deliverables/Outcomes

- A phishing website detecting model that uses machine learning algorithms to analyze the features of URLs and identify those that are likely to be malicious. This model will be fine-tuned to achieve high accuracy .

- A web-based interface that allows users to enter a URL and receive a classification of whether it is likely to be part of a phishing attack. This interface will be intuitive and user-friendly, enabling users to quickly and easily determine whether a URL is safe to visit.

- A report detailing the methodology, results, and evaluation of the phishing URL detecting model, including its performance compared to other state-of-the-art approaches.

Chapter 02: Feasibility Study, Requirements Analysis and Design

2.1 Literature Review

Table 2.1: Literature review for the project

S. No	**TITLE**	**AUTHOR**	**TOOLS**	**OBJECTIVE**	**ACCURACY**	**FUTURE WORK**
1.	An intelligent cyber security phishing detection system using deep learning techniques.	Ala Mughaid,SalahTaamneh, AsmaAlnajjar & Esraa Abu Elsoud	Locally-deep supportvector machine, Support vectormachine, Boosted decision tree,Logistic regression,Averagedperceptron, Neural network, Decision forest	classifying Phishing Email Using Machine Learning	88.81% 100% 97.7%	Feature selection techniques need more improvement to cope with the continuous development of new techniques by the phishers over the time.
2.	Detecting phishing websites using machine learning technique	Ashit Kumar Dutta	Convolutional Neural Network (CNN)	Detection of phishing websites using machine learning	97.4%	develop an unsupervised deep learning method to generate insight from a URL
3.	Comparative study of machine learning algorithms for phishing website detection	Anuraag Velamati	XGBoost Classifier ,Multilayer Perceptrons ,Random ,Decision Tree ,Support Vector Machines	Test and compare machine learning algorithms on the dataset in order to detect the phishing websites .	86.88%	Creating a GUI or web extension which would helpour user if he accesses any phishing websites by any chance
4.	Fighting Phishing with Discriminative Keypoint Features	Kuan-Ta Chen; Jau-Yuan Chen; Chun-Rong Huang; Chu-Song Chen	Contrast Context Histogram (CCH	computes the similarity degree between suspicious and authentic pages	97%	Creating a GUI or web extension

2.1.1 Problem Definition

The aim of this project is to develop an automated phishing website detecting model that can accurately identify potential phishing URLs and enable users to take appropriate action to prevent further damage. The model must be capable of analyzing the features of URLs and detecting those that are likely to be part of a phishing attack, with a high degree of accuracy.

The results will be very helpful to protect individuals and organizations from the harmful effects of phishing attacks and ensure the security of their confidential data.

Firstly, model will be trained to detect phished websites. Secondly, a web-based user friendly application will be made which will allow user to easily identify phished websites.

2.1.2 Problem Analysis

Step 1: Data Collection: This step involves collecting data that can be used to train the model so that later when it comes across an unknown URL, it can identify the URL based on the knowledge acquired during the training phase.

Step 2: Data Preprocessing: Here the data collected in the previous step is checked for any errors or null values. The dataset will be split into 70% of the data for training, and 30% for validation.

Step 3: Training Phase: Here the actual training of the model takes place. In this, the model learns from the training data.

Step 4: Validation phase: Once the model completes its training from the training set it is then tested with the test data.

Step 5: Output prediction: Once the validation phase is over, the model is ready to take an unknown URL from user and predict weather it is phished or legitimate from the knowledge it gained during training and validation phases.

2.1.3 Solution

Our project is implemented by using SVM as it is giving the highest accuracy among all the models practiced. Support Vector Machine (SVM) is a popular machine learning algorithm that can be used to build a phishing URL detecting model. SVM is a binary classification algorithm that finds the best hyperplane that separates the two classes of data. In the case of phishing URL detection, the two classes are phishing URLs and legitimate URLs.

Using SVM to build a phishing website detecting model has several advantages. SVM is a robust algorithm that can handle high-dimensional data and is less prone to overfitting. SVM can also handle imbalanced datasets, where one class of data is more prevalent than the other, which is often the case with phishing URLs.

2.2 Requirements

We use Google colab for the implementation of the code.

here are the hardware specifications of colab-

➢ 2vCPU @ 2.2GHz

➢ 13GB RAM

➢ 100GB Free Space

➢ idle cut-off 90 minutes

➢ maximum 12 hours

➢ 2020 Update:

GPU instance downgraded to 64GB disk space.

2.3 E-R Diagram / Data-Flow Diagram (DFD)

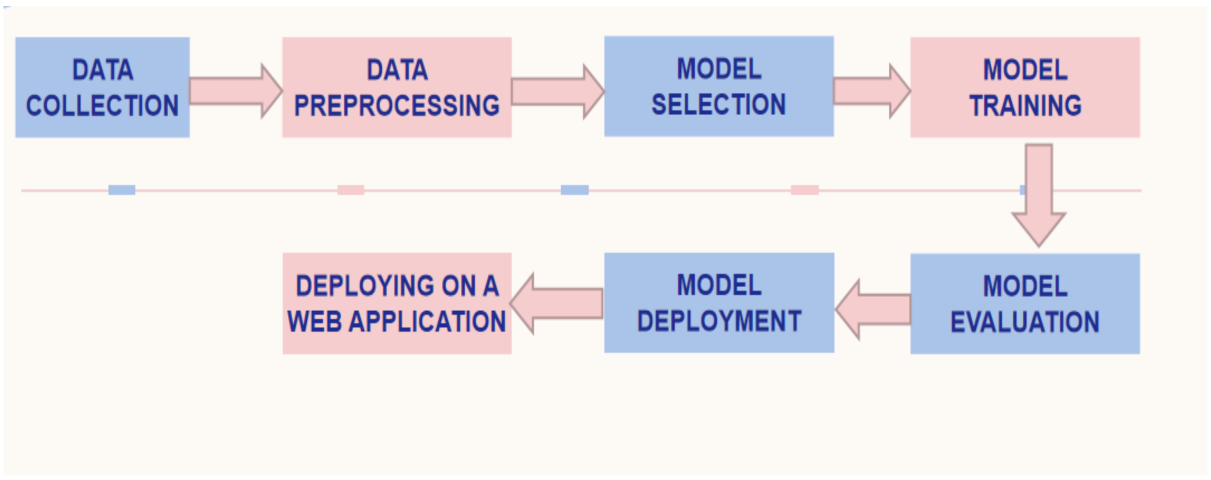


Figure 2.1: Data-flow diagram of machine learning model.

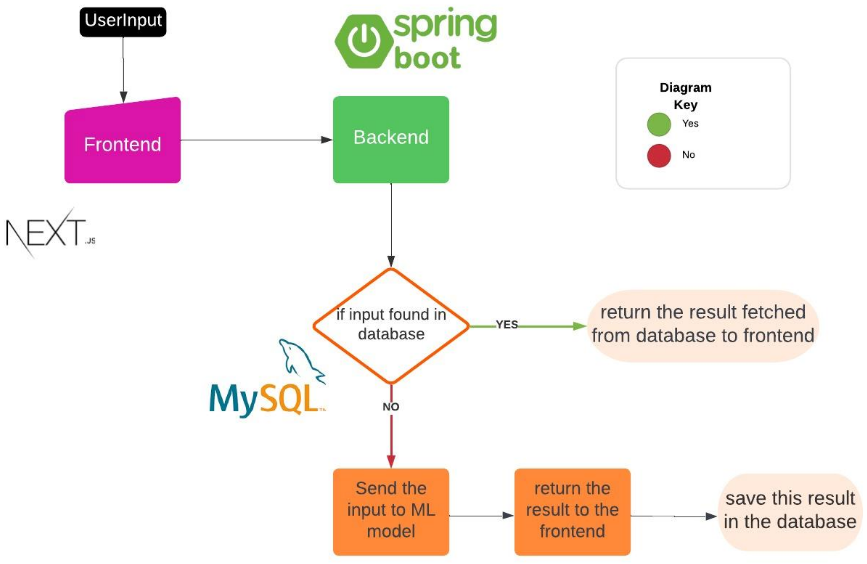

UserInput
Frontend
NEXT.JS
spring boot
Backend
Diagram Key
Yes
No
if input found in database
YES
return the result fetched from database to frontend
MySQL
NO
Send the input to ML model
return the result to the frontend
save this result in the database

Figure 2.2: Flow chart of the complete project.

Chapter 03: IMPLEMENTATION

3.1 Date Set Used in the Minor Project

There are two datasets used in this project. One is used as a database and the other is used for model training.

The datasets used are from Kaggle:

1. Containing 11054 good and bad URLs with 32 extracted features.
2. Containing 507195 good and bad URLs.

You can use this dataset to train model to learn various features and patterns of a URL.

The dataset used is a numeric categorical dataset.

The dataset has 32 features as follow:

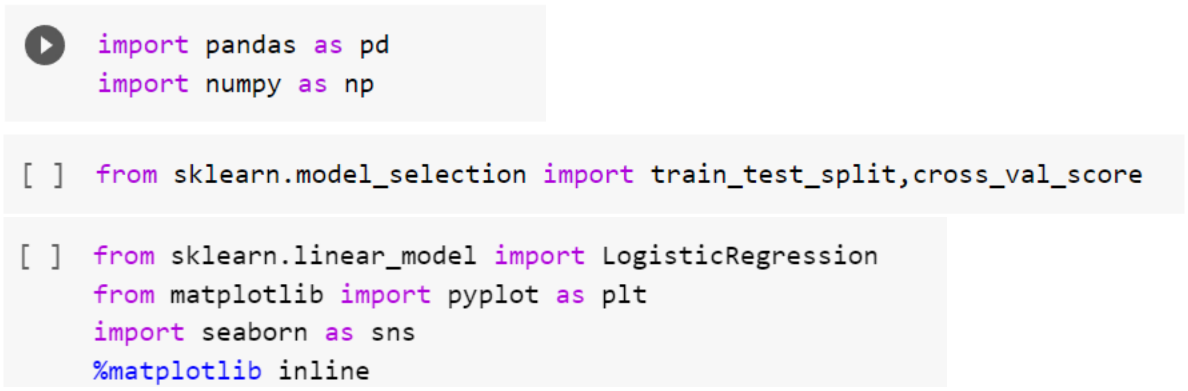
'Index', 'UsingIP', 'LongURL', 'ShortURL', 'Symbol@', 'Redirecting//', 'PrefixSuffix-', 'SubDomains', 'HTTPS', 'DomainRegLen', 'Favicon', 'NonStdPort', 'HTTPSDomainURL', 'RequestURL', 'AnchorURL', 'LinksInScriptTags', 'ServerFormHandler', 'InfoEmail', 'AbnormalURL', 'WebsiteForwarding', 'StatusBarCust', 'DisableRightClick', 'UsingPopupWindow', 'IframeRedirection', 'AgeofDomain', 'DNSRecording', 'WebsiteTraffic', 'PageRank', 'GoogleIndex', 'LinksPointingToPage', 'StatsReport', 'class'

3.2 Algorithm / Pseudo code of the Project Problem

```
IMPORT libraries
IMPORT dataset.
DATA PREPROCESSING
SPLIT dataset in training and testing
        train the model and validate the model using different classifiers
Print accuracy
Predict the result
END
```

3.3 Screenshots of various stages of the Project

<u>Step-1</u> Import the required libraries like Scikit learn, pandas, numpy, matplotlib, seaborn.

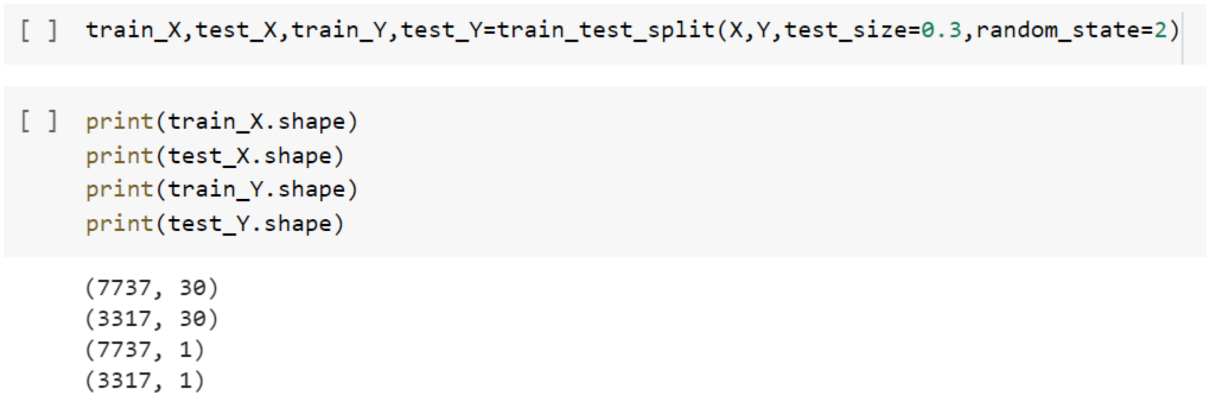


```
import pandas as pd
import numpy as np
```

```
from sklearn.model_selection import train_test_split,cross_val_score
```

```
from sklearn.linear_model import LogisticRegression
from matplotlib import pyplot as plt
import seaborn as sns
%matplotlib inline
```

<u>Step 2</u> Now split the dataset into 70% of the data used for training, and 30% for validation.

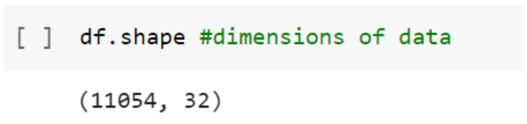


```
train_X,test_X,train_Y,test_Y=train_test_split(X,Y,test_size=0.3,random_state=2)
```

```
print(train_X.shape)
print(test_X.shape)
print(train_Y.shape)
print(test_Y.shape)
```

```
(7737, 30)
(3317, 30)
(7737, 1)
(3317, 1)
```

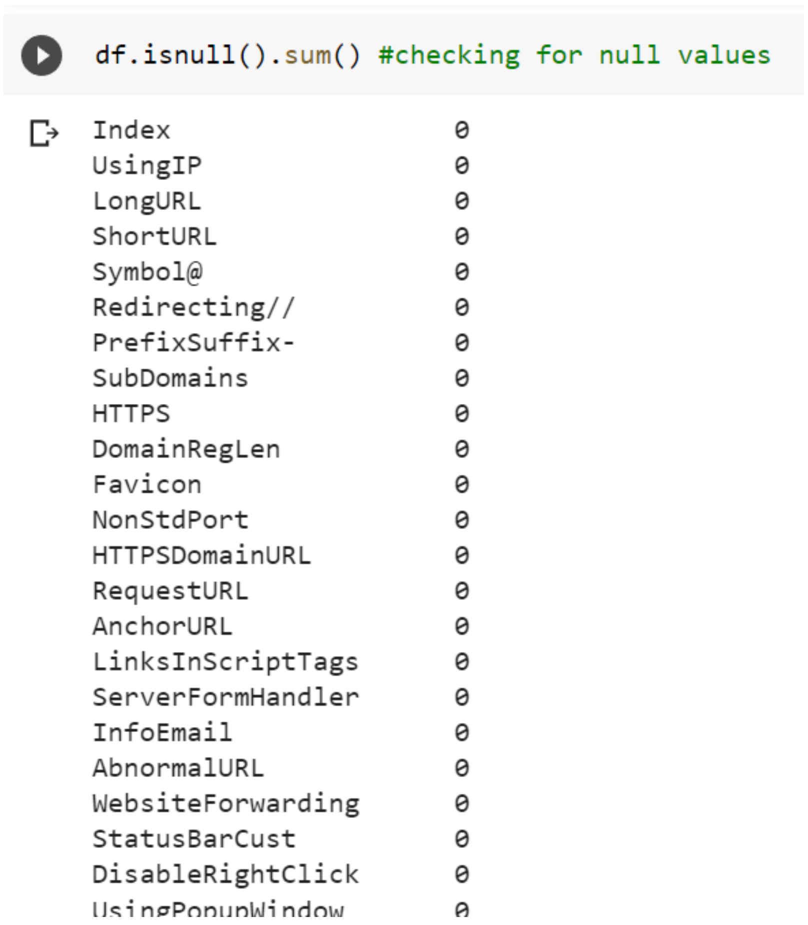
<u>Step-3</u> data visualization, we can see how the data looks like and what kind of correlation is held by the attributes of data.



```
df.shape #dimensions of data
```

```
(11054, 32)
```

Step-4 Data pre-processing, checking for null values



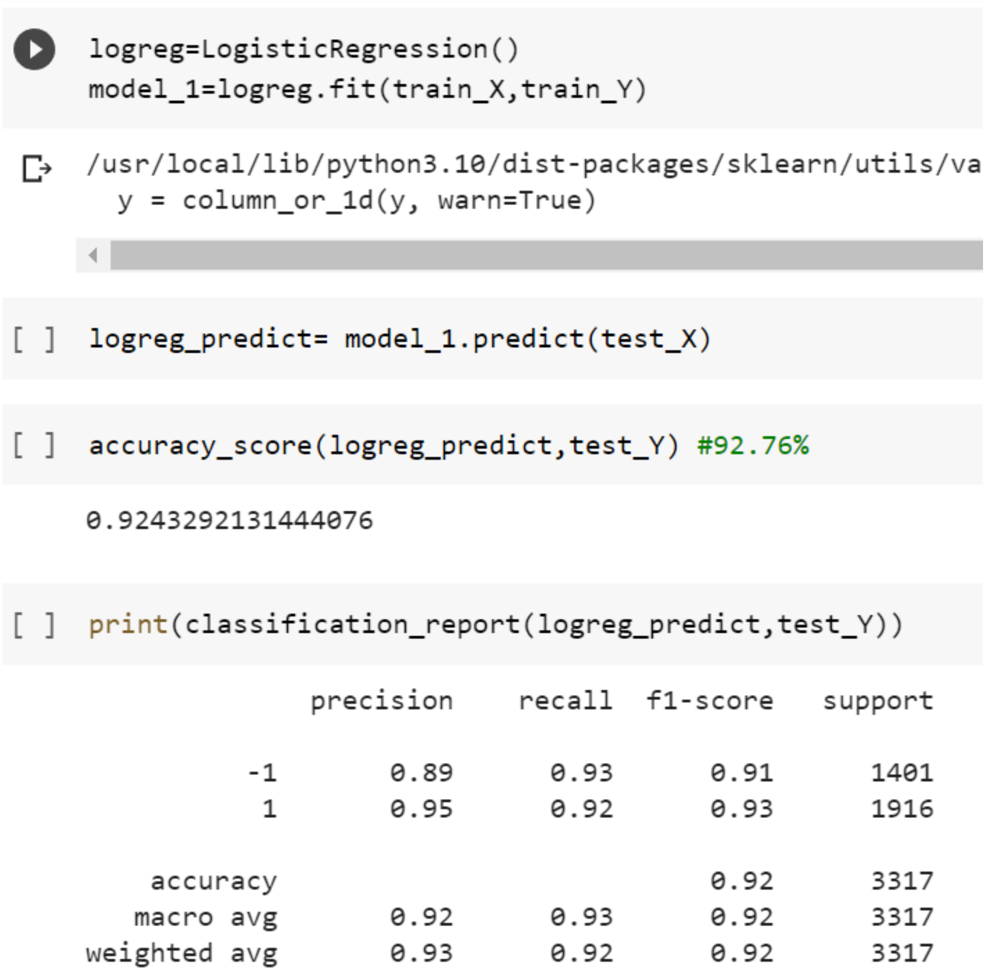
```
df.isnull().sum() #checking for null values
```

```
Index                  0
UsingIP                0
LongURL                0
ShortURL               0
Symbol@                0
Redirecting//          0
PrefixSuffix-          0
SubDomains             0
HTTPS                  0
DomainRegLen           0
Favicon                0
NonStdPort             0
HTTPSDomainURL         0
RequestURL             0
AnchorURL              0
LinksInScriptTags      0
ServerFormHandler      0
InfoEmail              0
AbnormalURL            0
WebsiteForwarding      0
StatusBarCust          0
DisableRightClick      0
UsingPopupWindow       0
```

Step 5 Applying following classifiers to all 32 attributes of the dataset and predicting accuracies.

1. Logistic Regression
2. KNN Classifier
3. Decision Tree Classifier
4. Random Forest Classifier
5. SVM Classifier
6. AdaBoost Classifier

1.Logistic Regression



```
logreg=LogisticRegression()
model_1=logreg.fit(train_X,train_Y)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/utils/va
  y = column_or_1d(y, warn=True)
```

```
[ ]  logreg_predict= model_1.predict(test_X)
```

```
[ ]  accuracy_score(logreg_predict,test_Y) #92.76%
```

```
0.9243292131444076
```

```
[ ]  print(classification_report(logreg_predict,test_Y))
```

```
              precision    recall  f1-score   support

          -1       0.89      0.93      0.91      1401
           1       0.95      0.92      0.93      1916

    accuracy                           0.92      3317
   macro avg       0.92      0.93      0.92      3317
weighted avg       0.93      0.92      0.92      3317
```

```
[ ]  plot_confusion_matrix(test_Y, logreg_predict)
```

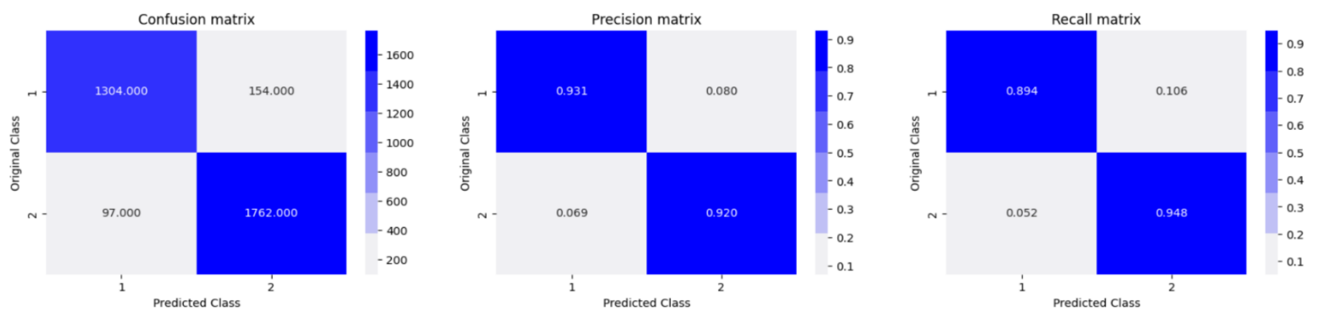


Figure 3.1: Confusion matrix for logistic regression.

2.KNN Classifier

```
knn=KNeighborsClassifier(n_neighbors=2)
model_2= knn.fit(train_X,train_Y)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/neighbors/_c
  return self._fit(X, y)
```

```
[ ]  knn_predict=model_2.predict(test_X)
```

```
[ ]  accuracy_score(knn_predict,test_Y)
```

```
0.9381971661139584
```

```
[ ]  print(classification_report(test_Y,knn_predict))
```

```
              precision    recall  f1-score   support

          -1       0.90      0.97      0.93      1458
           1       0.97      0.91      0.94      1859

    accuracy                           0.94      3317
   macro avg       0.94      0.94      0.94      3317
weighted avg       0.94      0.94      0.94      3317
```

```
[ ]  plot_confusion_matrix(test_Y, knn_predict)
```

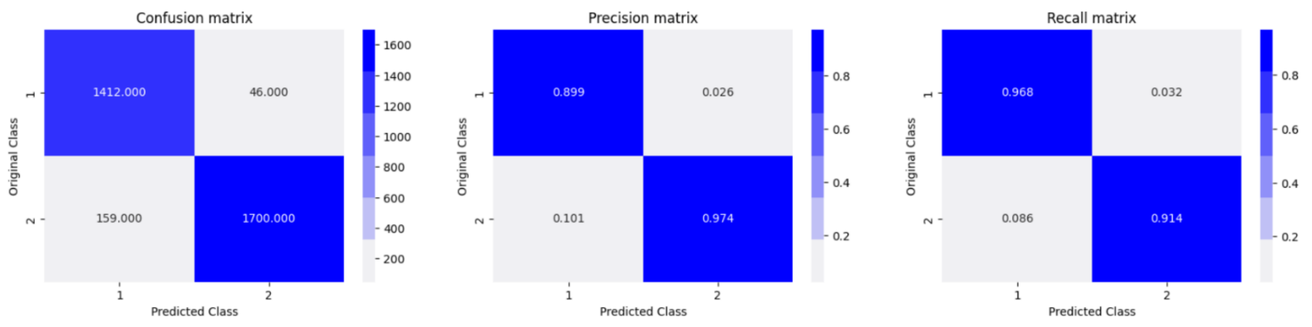
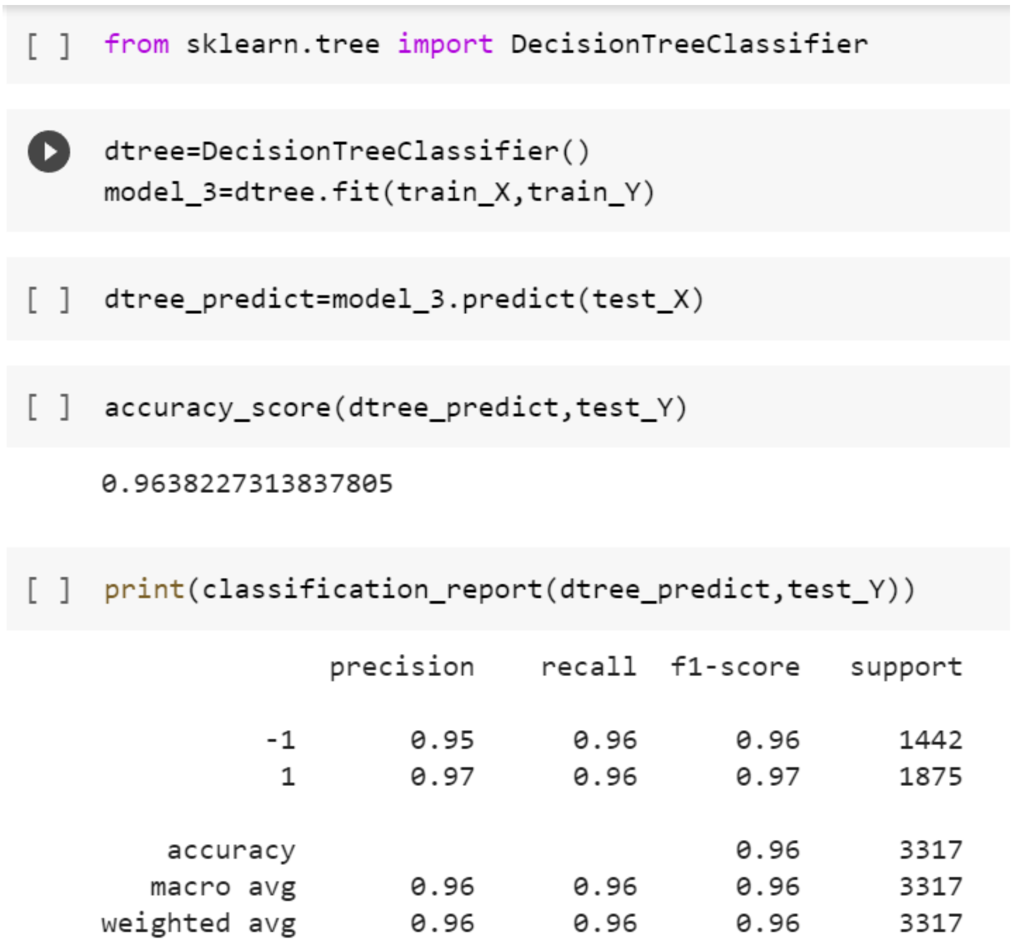


Figure 3.2: Confusion matrix for KNN.

3.Decision Tree Classifier



```
[ ]  from sklearn.tree import DecisionTreeClassifier
```

```
dtree=DecisionTreeClassifier()
model_3=dtree.fit(train_X,train_Y)
```

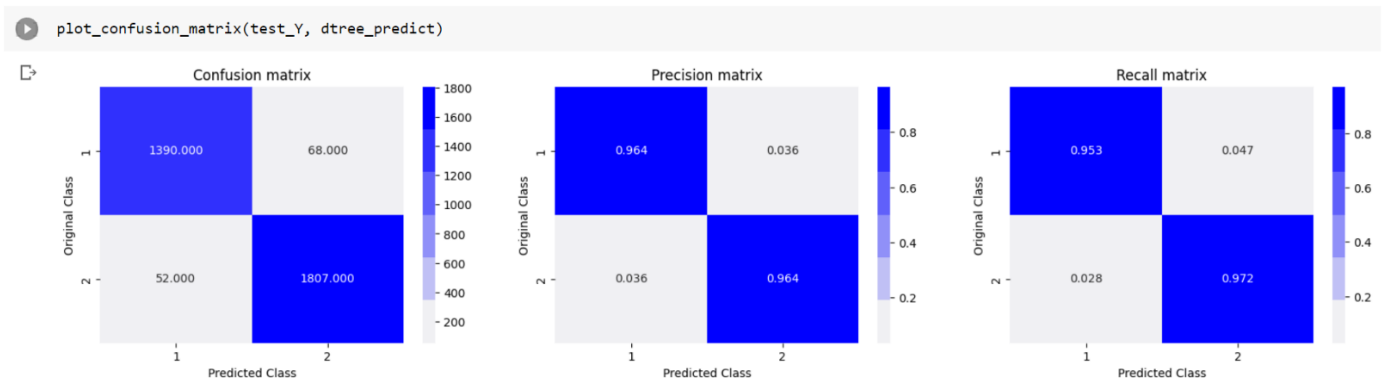
```
[ ]  dtree_predict=model_3.predict(test_X)
```

```
[ ]  accuracy_score(dtree_predict,test_Y)
```

```
0.9638227313837805
```

```
[ ]  print(classification_report(dtree_predict,test_Y))
```

```
              precision    recall  f1-score   support

          -1       0.95      0.96      0.96      1442
           1       0.97      0.96      0.97      1875

    accuracy                           0.96      3317
   macro avg       0.96      0.96      0.96      3317
weighted avg       0.96      0.96      0.96      3317
```



```
plot_confusion_matrix(test_Y, dtree_predict)
```

Figure 3.3: Confusion matrix for decision tree classifier.

4.Random Forest Classifier

```
]  rfc=RandomForestClassifier()
   model_4=rfc.fit(train_X,train_Y)

   <ipython-input-32-749d4f1c8dd2>:2: DataConversionWarning: A columr
     model_4=rfc.fit(train_X,train_Y)
```

```
]  rfc_predict=model_4.predict(test_X)
```

```
]  accuracy_score(rfc_predict,test_Y)

   0.9746759119686463
```

```
]  print(classification_report(rfc_predict,test_Y))

                 precision    recall  f1-score   support

             -1       0.96      0.98      0.97      1438
              1       0.98      0.97      0.98      1879

       accuracy                           0.97      3317
      macro avg       0.97      0.98      0.97      3317
   weighted avg       0.97      0.97      0.97      3317
```

```
[ ]  plot_confusion_matrix(test_Y, rfc_predict)
```

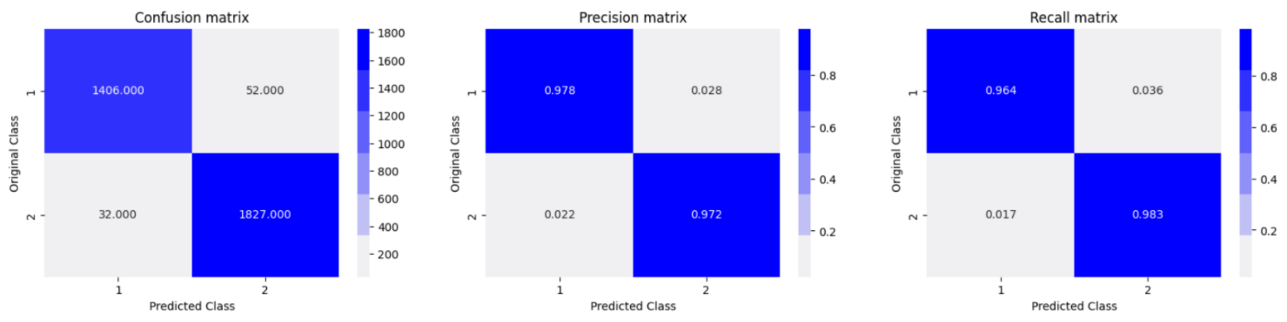
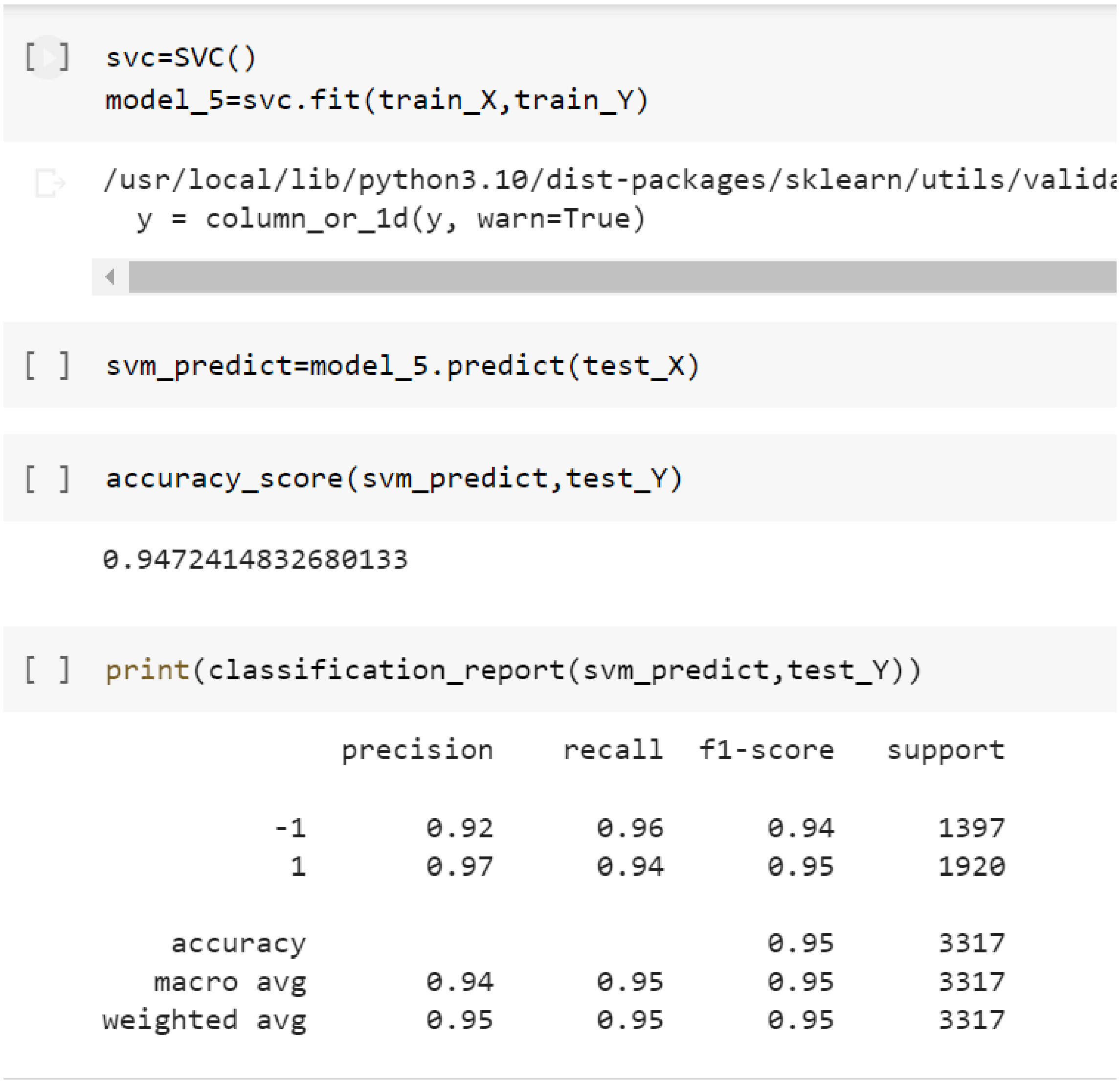


Figure 3.4: Confusion matrix for random forest classifier.

5.SVM Classifier

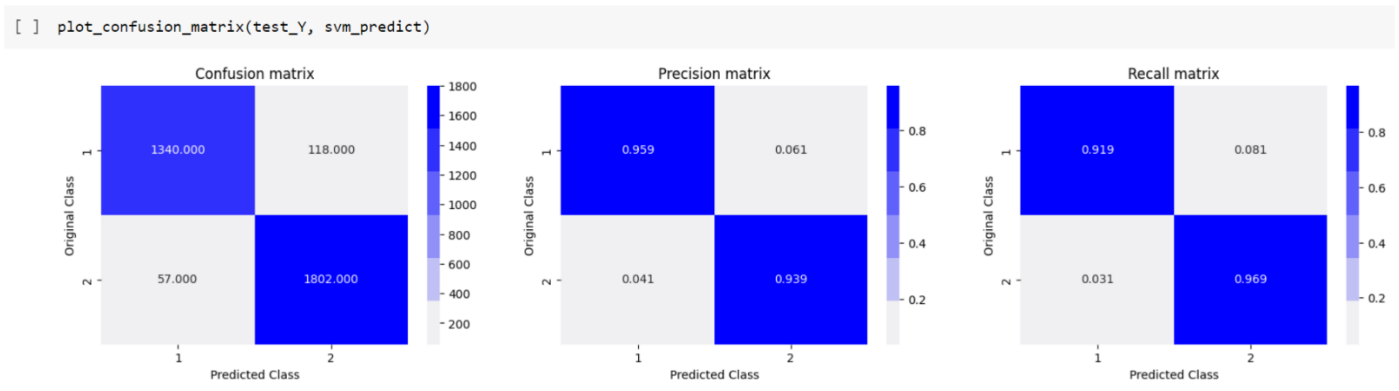


```
[ ]  svc=SVC()
     model_5=svc.fit(train_X,train_Y)

     /usr/local/lib/python3.10/dist-packages/sklearn/utils/valida
       y = column_or_1d(y, warn=True)
```

```
[ ]  svm_predict=model_5.predict(test_X)
```

```
[ ]  accuracy_score(svm_predict,test_Y)

     0.9472414832680133
```

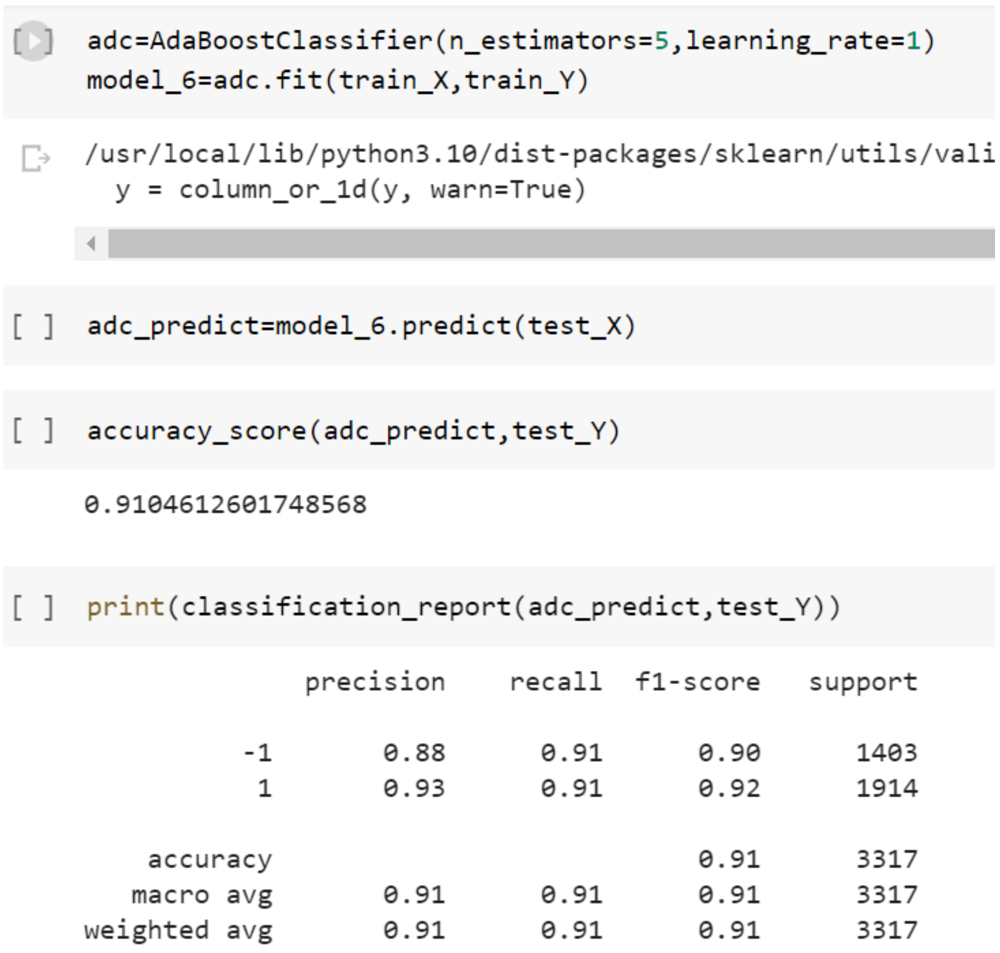
```
[ ]  print(classification_report(svm_predict,test_Y))

                   precision    recall  f1-score   support

              -1       0.92      0.96      0.94      1397
               1       0.97      0.94      0.95      1920

        accuracy                           0.95      3317
       macro avg       0.94      0.95      0.95      3317
    weighted avg       0.95      0.95      0.95      3317
```



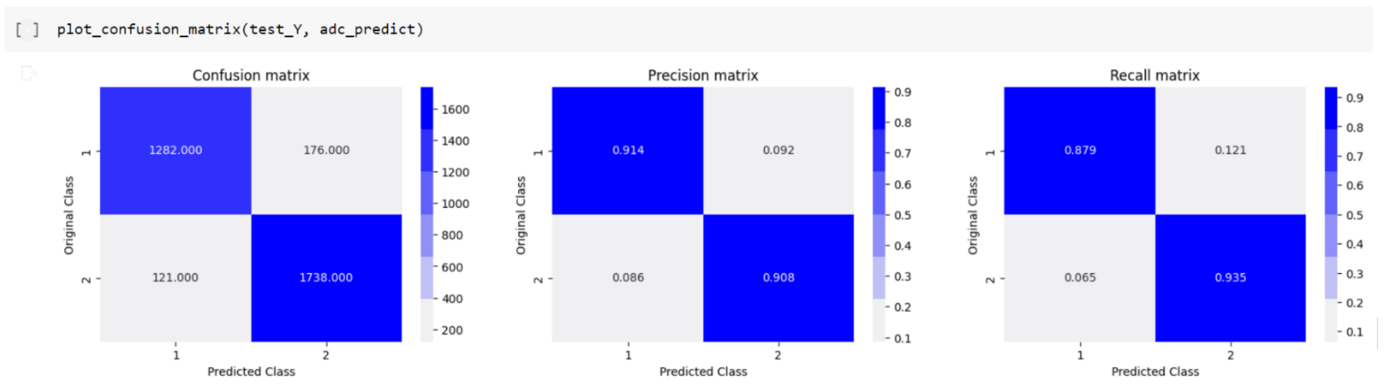
```
[ ]  plot_confusion_matrix(test_Y, svm_predict)
```

Figure 3.5: Confusion matrix for SVM.

6.AdaBoost Classifier



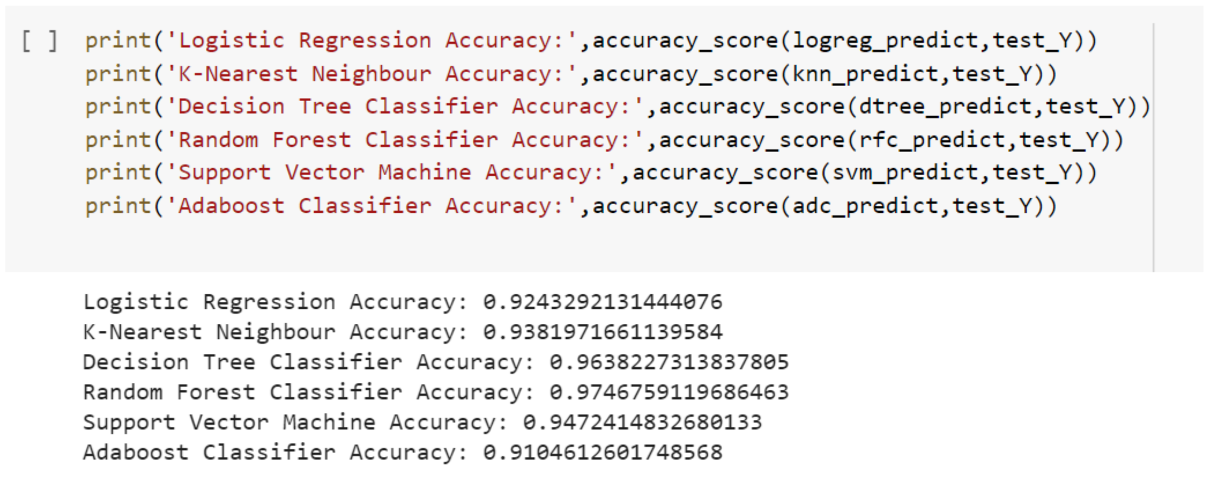
```
[ ] adc=AdaBoostClassifier(n_estimators=5,learning_rate=1)
    model_6=adc.fit(train_X,train_Y)

    /usr/local/lib/python3.10/dist-packages/sklearn/utils/vali
      y = column_or_1d(y, warn=True)

[ ] adc_predict=model_6.predict(test_X)

[ ] accuracy_score(adc_predict,test_Y)

    0.9104612601748568

[ ] print(classification_report(adc_predict,test_Y))

                  precision    recall  f1-score   support

              -1       0.88      0.91      0.90      1403
               1       0.93      0.91      0.92      1914

        accuracy                           0.91      3317
       macro avg       0.91      0.91      0.91      3317
    weighted avg       0.91      0.91      0.91      3317
```



```
[ ] plot_confusion_matrix(test_Y, adc_predict)
```

Figure 3.6: Confusion matrix for AdaBooast classifier.

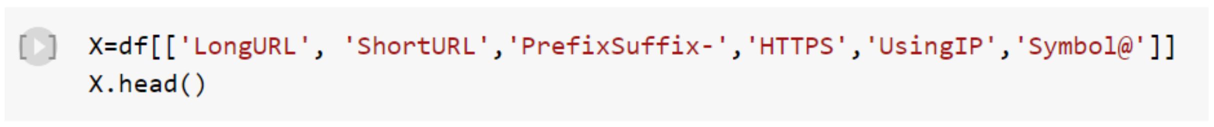
Step 6 Comparing different Accuracies given by the 6 used models.



```
[ ] print('Logistic Regression Accuracy:',accuracy_score(logreg_predict,test_Y))
    print('K-Nearest Neighbour Accuracy:',accuracy_score(knn_predict,test_Y))
    print('Decision Tree Classifier Accuracy:',accuracy_score(dtree_predict,test_Y))
    print('Random Forest Classifier Accuracy:',accuracy_score(rfc_predict,test_Y))
    print('Support Vector Machine Accuracy:',accuracy_score(svm_predict,test_Y))
    print('Adaboost Classifier Accuracy:',accuracy_score(adc_predict,test_Y))
```

```
    Logistic Regression Accuracy: 0.9243292131444076
    K-Nearest Neighbour Accuracy: 0.9381971661139584
    Decision Tree Classifier Accuracy: 0.9638227313837805
    Random Forest Classifier Accuracy: 0.9746759119686463
    Support Vector Machine Accuracy: 0.9472414832680133
    Adaboost Classifier Accuracy: 0.9104612601748568
```

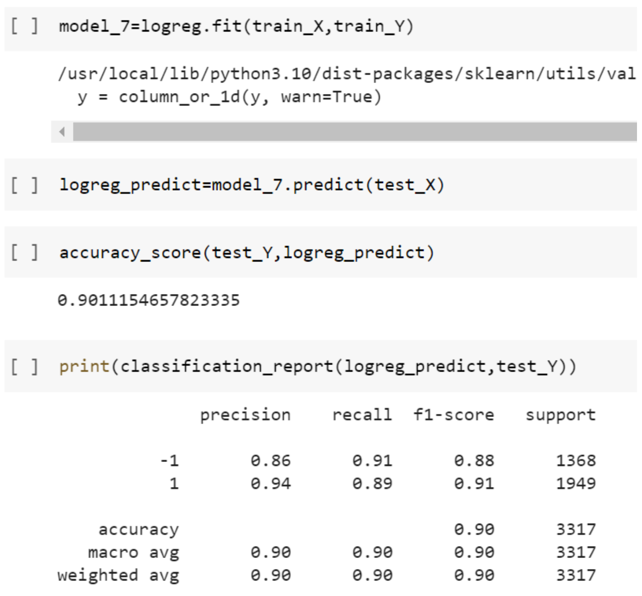
Step 7 Dimensionality Reduction, Reducing the number of attributes to 6.



```
X=df[['LongURL', 'ShortURL','PrefixSuffix-','HTTPS','UsingIP','Symbol@']]
X.head()
```

Step 8 Applying the models to the dataset with 6 attributes.

1.Logistic Regression



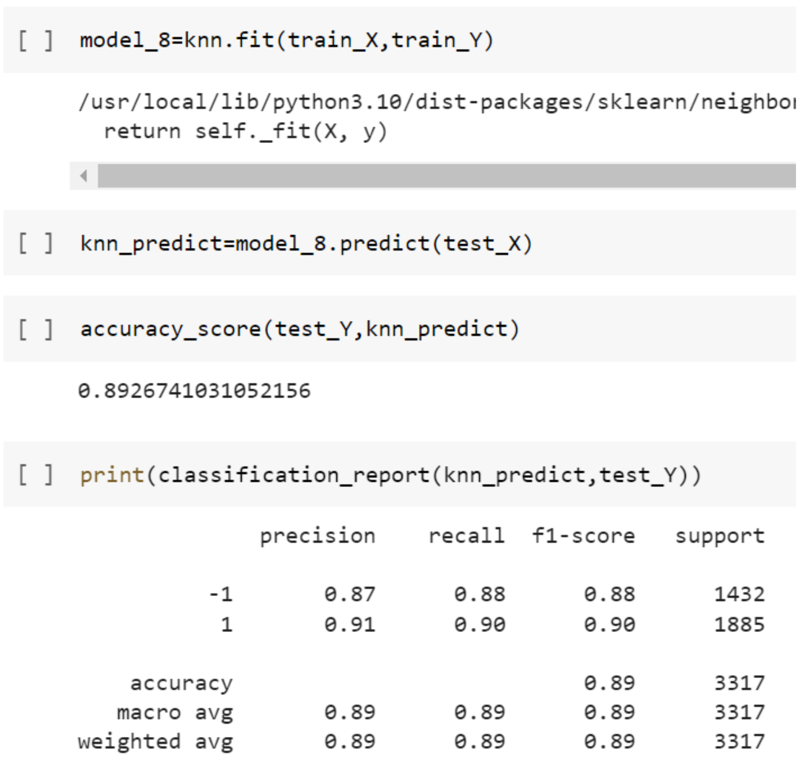
```
[ ] model_7=logreg.fit(train_X,train_Y)

    /usr/local/lib/python3.10/dist-packages/sklearn/utils/val
      y = column_or_1d(y, warn=True)
```

```
[ ] logreg_predict=model_7.predict(test_X)
```

```
[ ] accuracy_score(test_Y,logreg_predict)

    0.9011154657823335
```

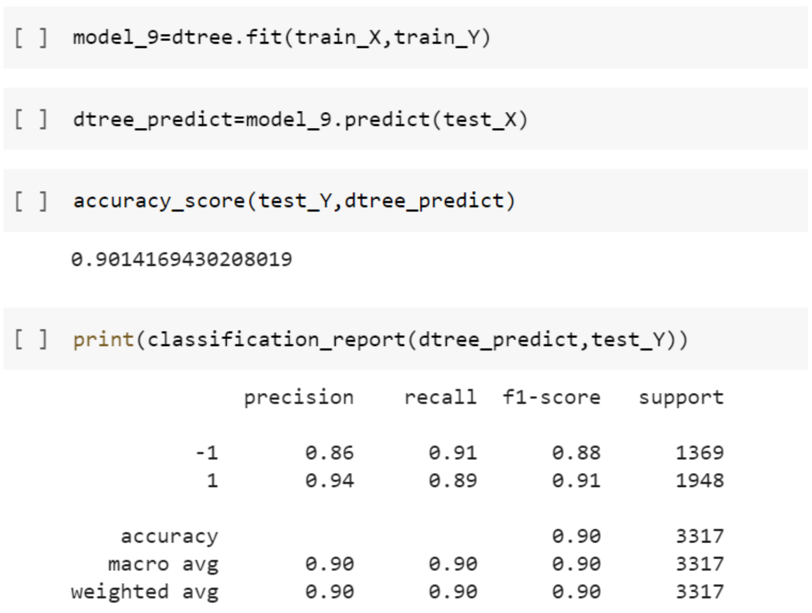
```
[ ] print(classification_report(logreg_predict,test_Y))

                  precision    recall  f1-score   support

              -1       0.86      0.91      0.88      1368
               1       0.94      0.89      0.91      1949

        accuracy                           0.90      3317
       macro avg       0.90      0.90      0.90      3317
    weighted avg       0.90      0.90      0.90      3317
```

2.KNN Classifier



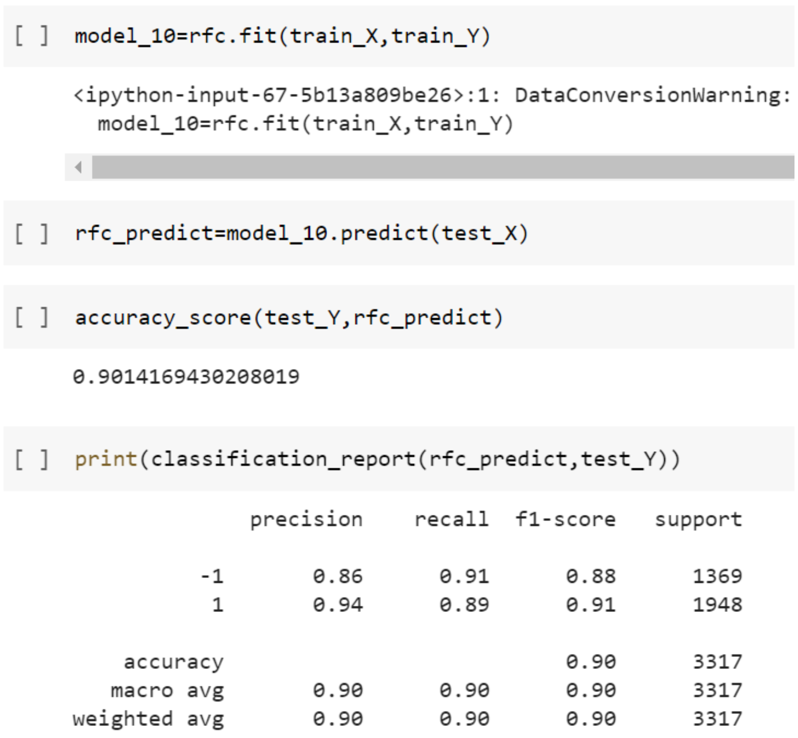
```
[ ]  model_8=knn.fit(train_X,train_Y)

     /usr/local/lib/python3.10/dist-packages/sklearn/neighbo
       return self._fit(X, y)

[ ]  knn_predict=model_8.predict(test_X)

[ ]  accuracy_score(test_Y,knn_predict)

     0.8926741031052156

[ ]  print(classification_report(knn_predict,test_Y))

                   precision    recall  f1-score   support

              -1       0.87      0.88      0.88      1432
               1       0.91      0.90      0.90      1885

        accuracy                           0.89      3317
       macro avg       0.89      0.89      0.89      3317
    weighted avg       0.89      0.89      0.89      3317
```

3.Decision Tree Classifier



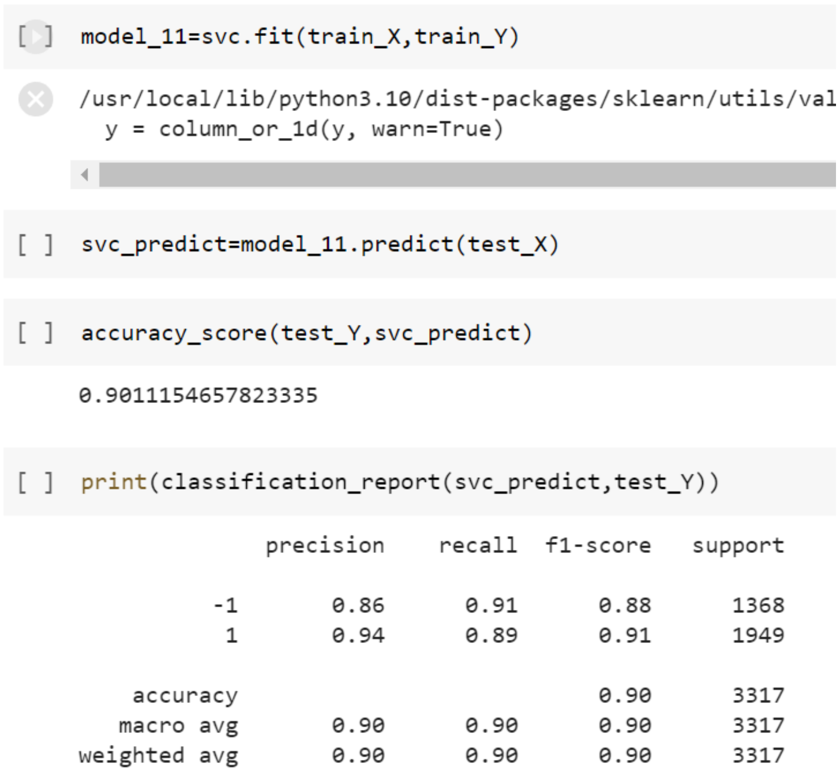
```
[ ]  model_9=dtree.fit(train_X,train_Y)

[ ]  dtree_predict=model_9.predict(test_X)

[ ]  accuracy_score(test_Y,dtree_predict)

     0.9014169430208019

[ ]  print(classification_report(dtree_predict,test_Y))

                   precision    recall  f1-score   support

              -1       0.86      0.91      0.88      1369
               1       0.94      0.89      0.91      1948

        accuracy                           0.90      3317
       macro avg       0.90      0.90      0.90      3317
    weighted avg       0.90      0.90      0.90      3317
```

4.Random Forest Classifier



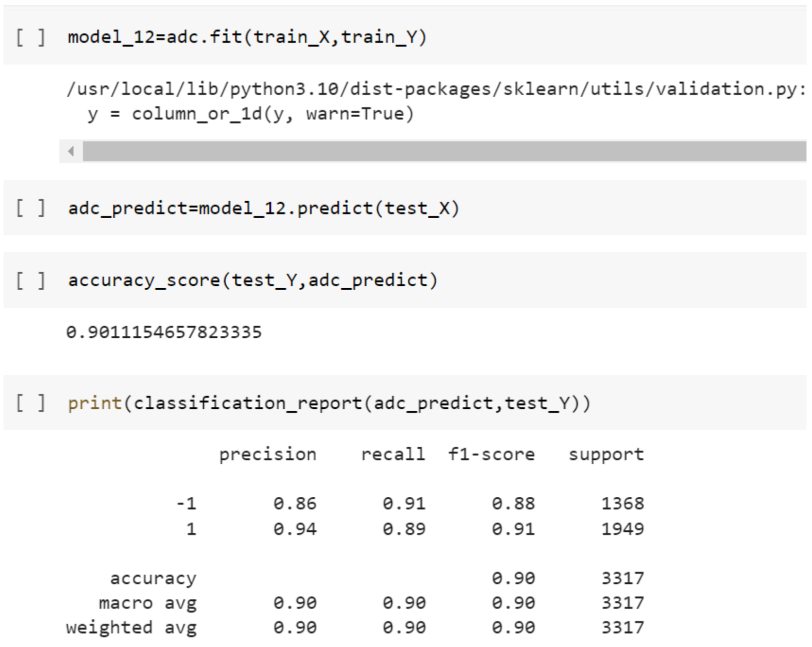
```
[ ]  model_10=rfc.fit(train_X,train_Y)

     <ipython-input-67-5b13a809be26>:1: DataConversionWarning:
       model_10=rfc.fit(train_X,train_Y)

[ ]  rfc_predict=model_10.predict(test_X)

[ ]  accuracy_score(test_Y,rfc_predict)

     0.9014169430208019

[ ]  print(classification_report(rfc_predict,test_Y))

                   precision    recall  f1-score   support

              -1       0.86      0.91      0.88      1369
               1       0.94      0.89      0.91      1948

        accuracy                           0.90      3317
       macro avg       0.90      0.90      0.90      3317
    weighted avg       0.90      0.90      0.90      3317
```

5.SVM Classifier



```
[ ]  model_11=svc.fit(train_X,train_Y)

     /usr/local/lib/python3.10/dist-packages/sklearn/utils/val
       y = column_or_1d(y, warn=True)

[ ]  svc_predict=model_11.predict(test_X)

[ ]  accuracy_score(test_Y,svc_predict)

     0.9011154657823335

[ ]  print(classification_report(svc_predict,test_Y))

                   precision    recall  f1-score   support

              -1       0.86      0.91      0.88      1368
               1       0.94      0.89      0.91      1949

        accuracy                           0.90      3317
       macro avg       0.90      0.90      0.90      3317
    weighted avg       0.90      0.90      0.90      3317
```

6.AdaBooast Classifier



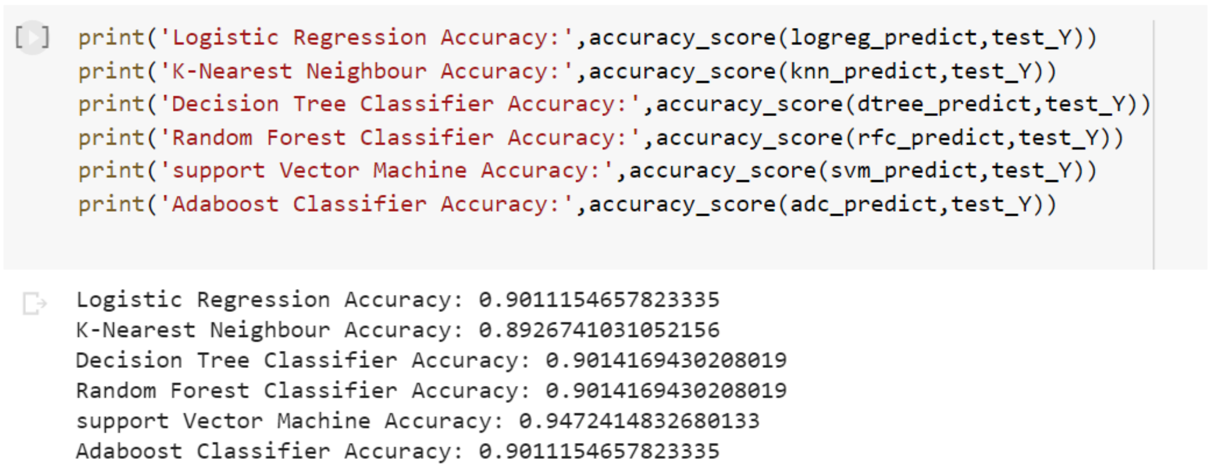
```
[ ]  model_12=adc.fit(train_X,train_Y)

     /usr/local/lib/python3.10/dist-packages/sklearn/utils/validation.py:
       y = column_or_1d(y, warn=True)
```

```
[ ]  adc_predict=model_12.predict(test_X)
```

```
[ ]  accuracy_score(test_Y,adc_predict)

     0.9011154657823335
```

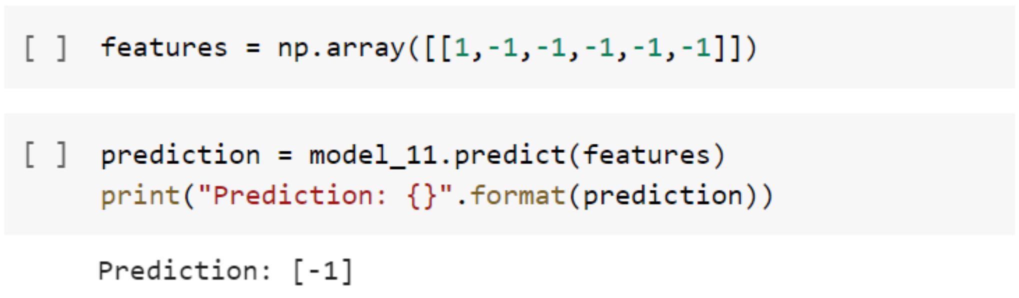
```
[ ]  print(classification_report(adc_predict,test_Y))

                   precision    recall  f1-score   support

              -1       0.86      0.91      0.88      1368
               1       0.94      0.89      0.91      1949

        accuracy                           0.90      3317
       macro avg       0.90      0.90      0.90      3317
    weighted avg       0.90      0.90      0.90      3317
```

Step 9 Comparing the accuracies of the above 6 models.



```
[ ]  print('Logistic Regression Accuracy:',accuracy_score(logreg_predict,test_Y))
     print('K-Nearest Neighbour Accuracy:',accuracy_score(knn_predict,test_Y))
     print('Decision Tree Classifier Accuracy:',accuracy_score(dtree_predict,test_Y))
     print('Random Forest Classifier Accuracy:',accuracy_score(rfc_predict,test_Y))
     print('support Vector Machine Accuracy:',accuracy_score(svm_predict,test_Y))
     print('Adaboost Classifier Accuracy:',accuracy_score(adc_predict,test_Y))

     Logistic Regression Accuracy: 0.9011154657823335
     K-Nearest Neighbour Accuracy: 0.8926741031052156
     Decision Tree Classifier Accuracy: 0.9014169430208019
     Random Forest Classifier Accuracy: 0.9014169430208019
     support Vector Machine Accuracy: 0.9472414832680133
     Adaboost Classifier Accuracy: 0.9011154657823335
```

Step 10 Predicting the status of URLs using SVM model as it is giving the best results.



```
[ ]  features = np.array([[1,-1,-1,-1,-1,-1]])
```

```
[ ]  prediction = model_11.predict(features)
     print("Prediction: {}".format(prediction))

     Prediction: [-1]
```

Chapter 04: RESULTS

4.1 Discussion on the Results Achieved

Our project "**PHISHING WEBSITE DETECTION"** where we use SVM as the method to train our model. During the training process, our model discovers features and patterns and learns them for the identification purpose. Here, numeric categorical data is used to train the model. The model was trained with11054 instances or URLs. The model attained an overall accuracy of 94.72% over which it got best results among the applied classifiers.

4.2 Application of the Minor Project

The application of this project is crucial for protecting users from falling victim to phishing attacks. Phishing detection models can be integrated into web browsers to provide real-time alerts when users visit suspected phishing websites. It can be build into a user friendly application to detect phished websites.

4.3 Limitation of the Minor Project

The challenges in the project was to extract complicated features from URL and to provide them as input to the model. Getting higher accuracy with limited extracted features.

4.4 Future work

We can also train our model to detect phished email and articles. Just like URLs the model will be able to detect suspicious emails**.**

References

[1] Mughaid, A., AlZu’bi, S., Hnaif, A. (2022). “An intelligent cyber security phishing detection system using deep learning techniques”. Cluster Comput 25, 3819–3828

[2] Dutta, A., (2021). “Detecting phishing websites using machine learning technique”. PLoS ONE 16(10): e0258361. https://doi.org/10.1371/journal.pone.0258361 (2021).

[3] Sarma, D., Mittra, T., Bawm, R.M., Sarwar, T., Lima, F.F., Hossain, S. (2021). “Comparative Analysis of Machine Learning Algorithms for Phishing Website Detection”. In: Smys, S., Balas, V.E., Kamel, K.A., Lafata, P. (eds) Inventive Computation and Information Technologies. Lecture Notes in Networks and Systems, vol 173. Springer, Singapore. https://doi.org/10.1007/978-981-33-4305-4_64

[4] Chen, K., Chen, J., Huang, R., (2009) "Fighting Phishing with Discriminative Keypoint Features," in IEEE Internet Computing, vol. 13, no. 3, pp. 56-63, May-June, doi: 10.1109/MIC.2009.59.