IDS Project Report

Phishing Websites Dataset

Team Members :
Arthak (17ucs036)
Dev Pancham Purohit (17ucs050)
Pulkit Mathur (17ucs116)
Shubham Bansal (17ucs156)

Dataset

Phishing Websites (link)

Source

UCI Machine Learning Repository

Aim

This dataset sheds light on the important features that have proved to be sound and effective in predicting phishing websites and we intend to classify the websites as Phishing Websites or Safe websites based on these 31 attributes.

Dataset Specification

Data Set Characteristics:	Multivariate	**Number of Instances:**	11055	**Area:**	Computer Security
Attribute Characteristics:	Integer	**Number of Attributes:**	32	**Date Donated**	2015-03-26
Associated Tasks:	Classification	**Missing Values?**	N/A	**Number of Web Hits:**	127767

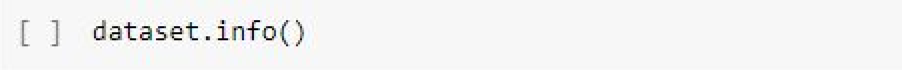
Understanding Various Features of the dataset :

S.No.	Feature	Represented by -1	Represented by 0	Represented by 1
1	IP Address	Domain Part has an IP Address	NA	Otherwise
2	Long URL	Length<54	54-75	>75
3	Tiny URL	Otherwise	NA	Tiny URL
4	having “@” Symbol	Otherwise	NA	having “@” Symbol
5	Redirecting using “//”	Otherwise	NA	Last occurrence position>7
6	Adding Prefix or Suffix Separated by (-)	Otherwise	NA	(-)included
7	Sub Domain and Multi-Sub Domains	Dots in domain part=1	Dots in domain part=2	Otherwise
8	HTTPS	Trusted Certificated with age>=1	Not Trusted	Otherwise
9	Domain Registration Length	Otherwise	NA	Expiry in <1 year
10	Favicon	Otherwise	NA	Loaded from ext domain
11	Using Non-Standard Port	Otherwise	NA	Port # preferred
12	The Existence of “HTTPS” Token	Otherwise	NA	HTTP
13	Request URL	<22%	22-61%	Otherwise
14	URL of Anchor	<31%	31-67%	Otherwise
15	Links in <Meta>, <Script> and <Link> tags	<17%	17-81%	Otherwise
16	Server Form Handler (SFH)	Otherwise	Different Domain	Blank
17	Submitting Information to Email	Otherwise	NA	Using mail to
18	Abnormal URL	Otherwise	NA	Hostname not included
19	Website Forwarding	<1	2-4	Otherwise

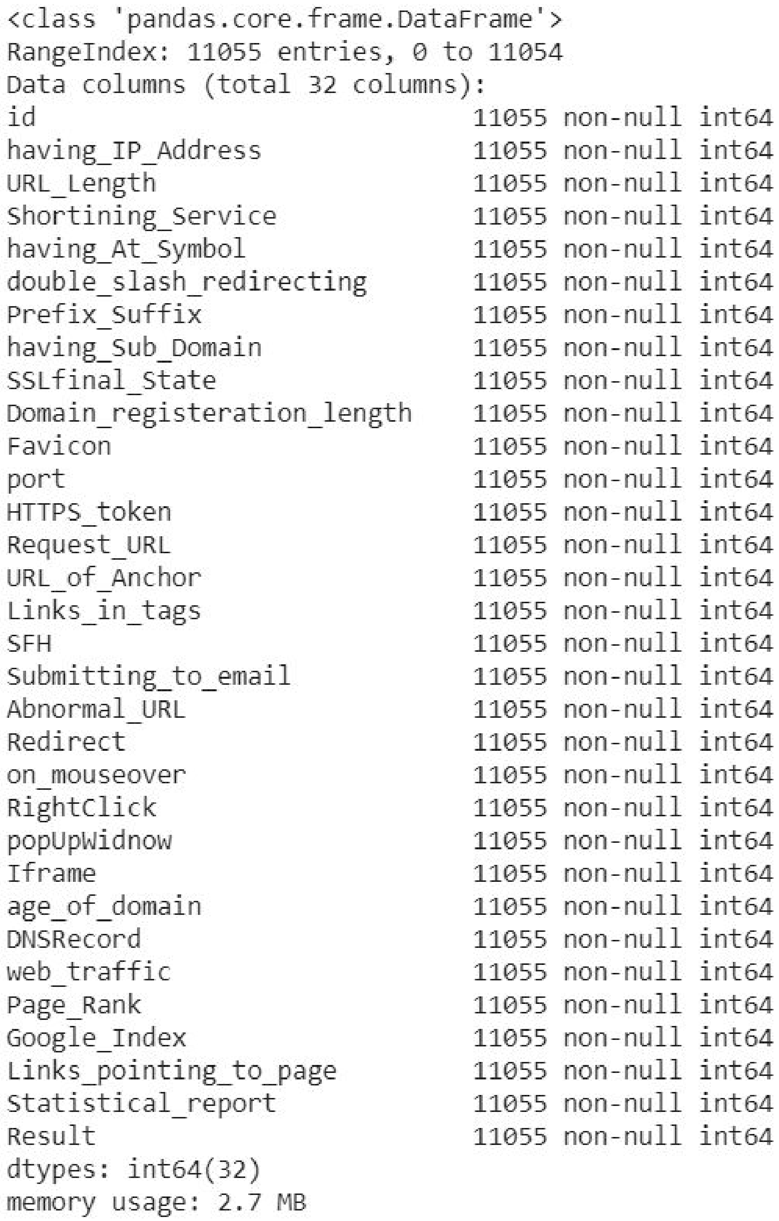
20	Status Bar Customization	Doesn’t	NA	Change occurred
21	Disabling Right Click	Otherwise	NA	Right-click disabled
22	Using Pop-up Window	Otherwise	NA	Popup window with text field
23	IFrame Redirection	Otherwise	NA	Using it
24	Age of Domain	>6 mon	NA	Otherwise
25	DNS Record	Otherwise	NA	None
26	Website Traffic	<1lac	>1lac	Otherwise
27	PageRank	Otherwise	NA	rank<0.2
28	Google Index	Indexed by google	NA	Otherwise
29	Number of Links Pointing to Page	Otherwise	0-2	0
30	Statistical-Reports Based Feature	Otherwise	NA	Belonging to top phishing IPs
31	Suspicious links	Absent	NA	Present

Dataset Analysis and Preprocessing

I. Getting Info on the dataset

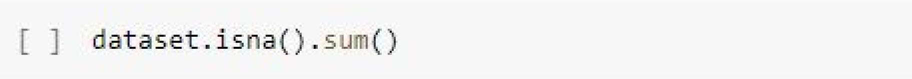


```
[ ]  dataset.info()
```

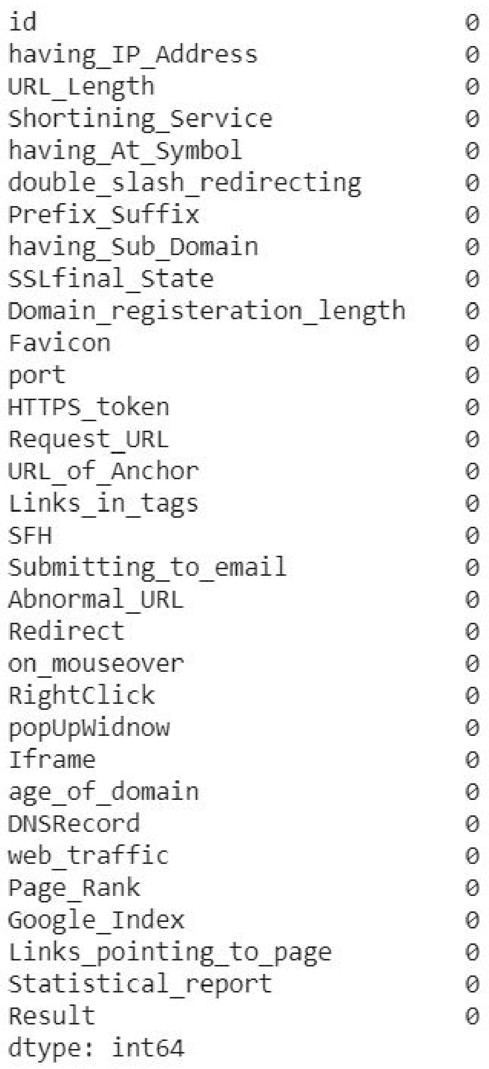


```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11055 entries, 0 to 11054
Data columns (total 32 columns):
id                             11055 non-null int64
having_IP_Address              11055 non-null int64
URL_Length                     11055 non-null int64
Shortining_Service             11055 non-null int64
having_At_Symbol               11055 non-null int64
double_slash_redirecting       11055 non-null int64
Prefix_Suffix                  11055 non-null int64
having_Sub_Domain              11055 non-null int64
SSLfinal_State                 11055 non-null int64
Domain_registeration_length    11055 non-null int64
Favicon                        11055 non-null int64
port                           11055 non-null int64
HTTPS_token                    11055 non-null int64
Request_URL                    11055 non-null int64
URL_of_Anchor                  11055 non-null int64
Links_in_tags                  11055 non-null int64
SFH                            11055 non-null int64
Submitting_to_email            11055 non-null int64
Abnormal_URL                   11055 non-null int64
Redirect                       11055 non-null int64
on_mouseover                   11055 non-null int64
RightClick                     11055 non-null int64
popUpWidnow                    11055 non-null int64
Iframe                         11055 non-null int64
age_of_domain                  11055 non-null int64
DNSRecord                      11055 non-null int64
web_traffic                    11055 non-null int64
Page_Rank                      11055 non-null int64
Google_Index                   11055 non-null int64
Links_pointing_to_page         11055 non-null int64
Statistical_report             11055 non-null int64
Result                         11055 non-null int64
dtypes: int64(32)
memory usage: 2.7 MB
```

II. Checking for Null Values

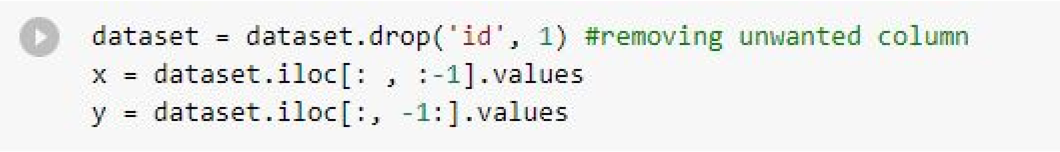


```
[ ]  dataset.isna().sum()
```



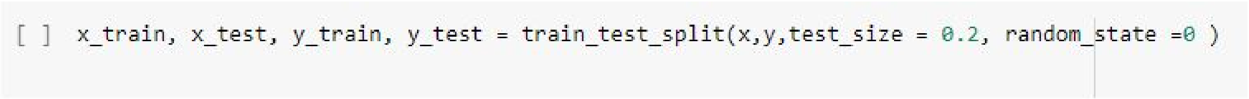
```
id                            0
having_IP_Address             0
URL_Length                    0
Shortining_Service            0
having_At_Symbol              0
double_slash_redirecting      0
Prefix_Suffix                 0
having_Sub_Domain             0
SSLfinal_State                0
Domain_registeration_length   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
```

III. Removing unused column



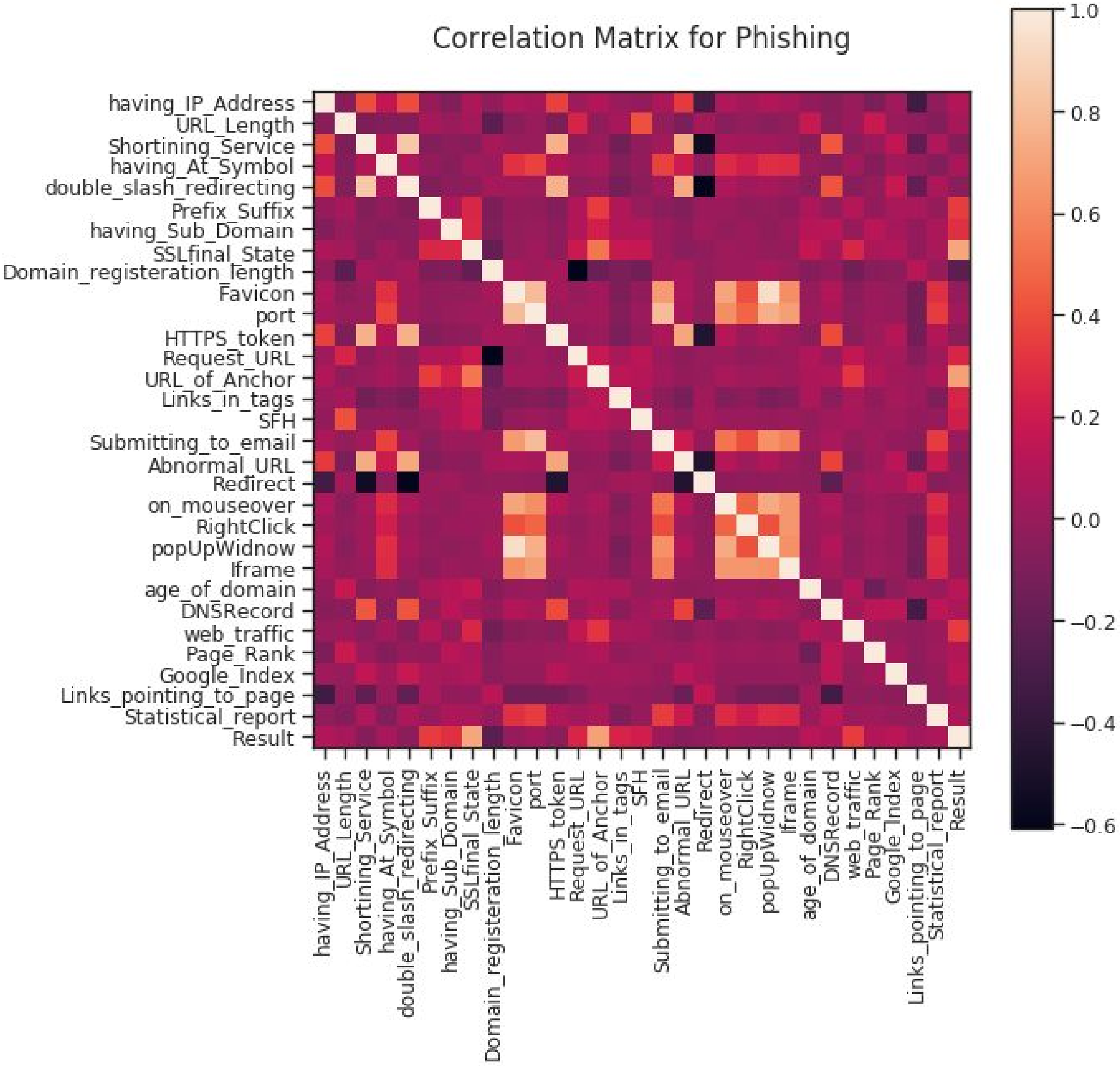
```
dataset = dataset.drop('id', 1) #removing unwanted column
x = dataset.iloc[: , :-1].values
y = dataset.iloc[:, -1:].values
```

IV. Splitting into Test and Train

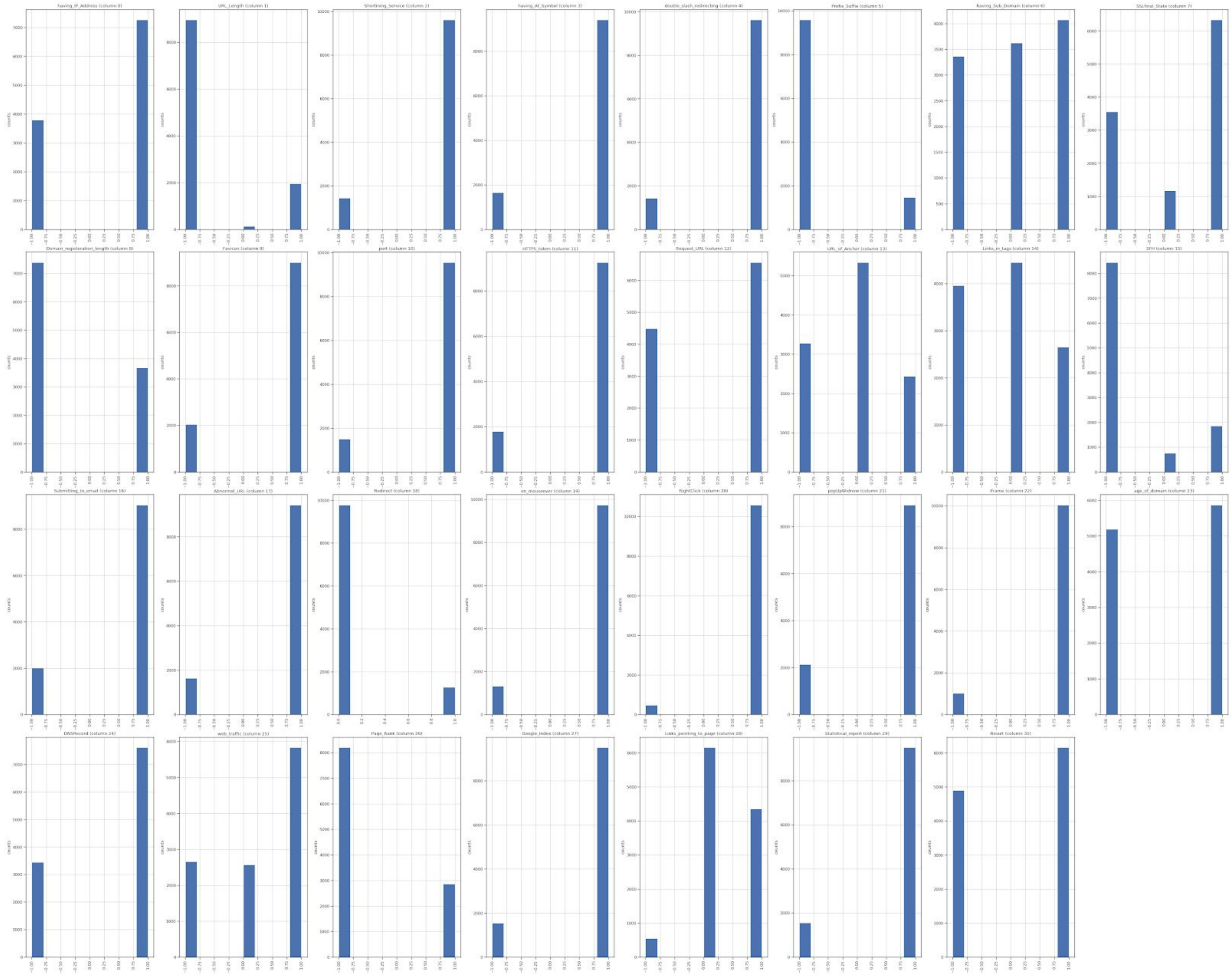


```
[ ]  x_train, x_test, y_train, y_test = train_test_split(x,y,test_size = 0.2, random_state =0 )
```

V. Correlation Matrix for the Dataset



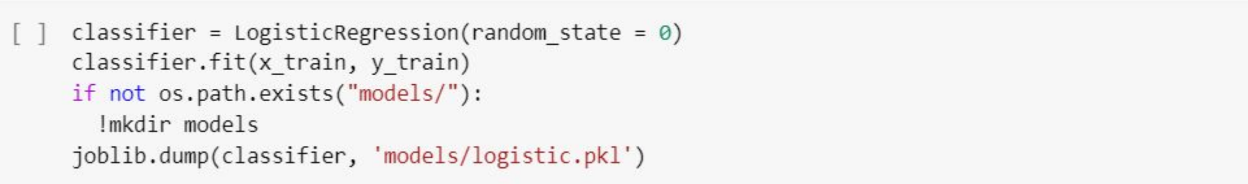
VI. Distribution graph of sampled Columns



Classification

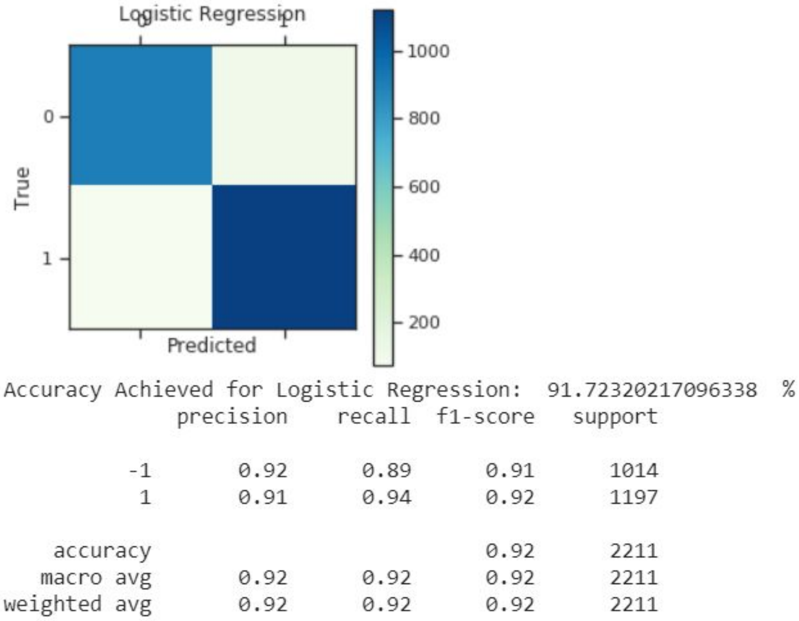
1. Logistic Regression :

Logistic regression is a statistical model that in its basic form uses a logistic function to model a binary dependent variable, although many more complex extensions exist.



```
[ ]  classifier = LogisticRegression(random_state = 0)
     classifier.fit(x_train, y_train)
     if not os.path.exists("models/"):
       !mkdir models
     joblib.dump(classifier, 'models/logistic.pkl')
```

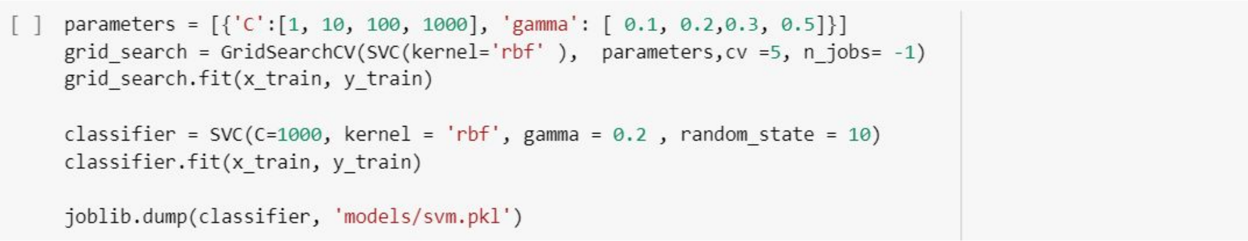
Classification Report:



```
Accuracy Achieved for Logistic Regression:  91.72320217096338  %
              precision    recall  f1-score   support

          -1       0.92      0.89      0.91      1014
           1       0.91      0.94      0.92      1197

    accuracy                           0.92      2211
   macro avg       0.92      0.92      0.92      2211
weighted avg       0.92      0.92      0.92      2211
```

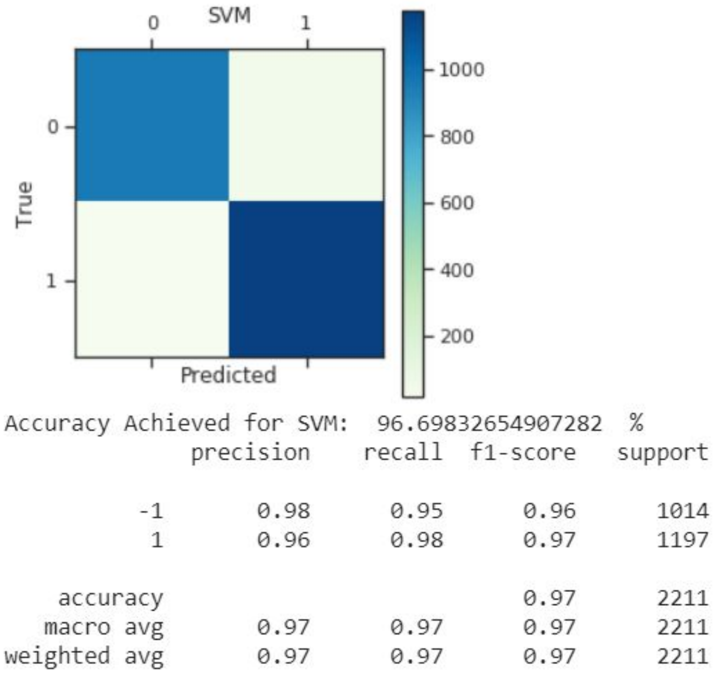
2. SVM (support vector machine) :

A Support Vector Machine (**SVM**) is a discriminative classifier formally defined by a separating hyperplane



```
parameters = [{'C':[1, 10, 100, 1000], 'gamma': [ 0.1, 0.2,0.3, 0.5]}]
grid_search = GridSearchCV(SVC(kernel='rbf' ),  parameters,cv =5, n_jobs= -1)
grid_search.fit(x_train, y_train)

classifier = SVC(C=1000, kernel = 'rbf', gamma = 0.2 , random_state = 10)
classifier.fit(x_train, y_train)

joblib.dump(classifier, 'models/svm.pkl')
```

Classification Report:

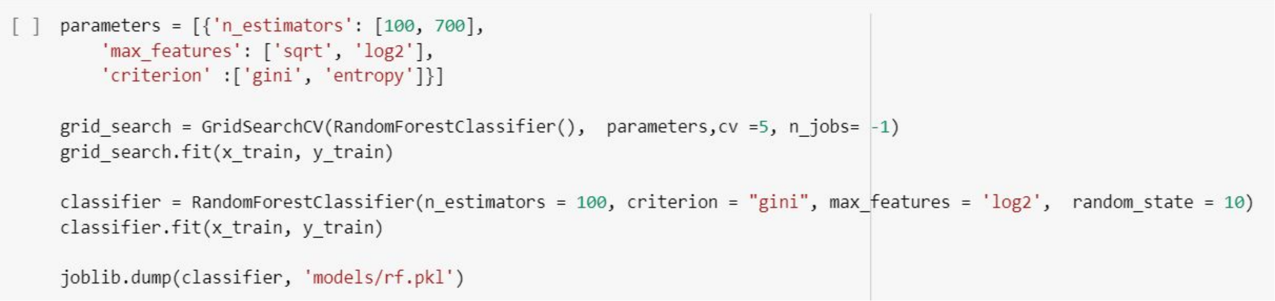


```
Accuracy Achieved for SVM:  96.69832654907282  %
              precision    recall  f1-score   support

          -1       0.98      0.95      0.96      1014
           1       0.96      0.98      0.97      1197

    accuracy                           0.97      2211
   macro avg       0.97      0.97      0.97      2211
weighted avg       0.97      0.97      0.97      2211
```

.

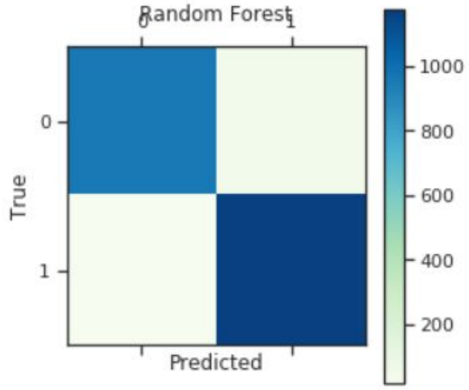
3. Random Forest :

Random forest, like its name implies, consists of a large number of individual decision trees that operate as an ensemble.



```
[ ]  parameters = [{'n_estimators': [100, 700],
         'max_features': ['sqrt', 'log2'],
         'criterion' :['gini', 'entropy']}]

     grid_search = GridSearchCV(RandomForestClassifier(),  parameters,cv =5, n_jobs= -1)
     grid_search.fit(x_train, y_train)

     classifier = RandomForestClassifier(n_estimators = 100, criterion = "gini", max_features = 'log2',  random_state = 10)
     classifier.fit(x_train, y_train)

     joblib.dump(classifier, 'models/rf.pkl')
```

Classification Report:

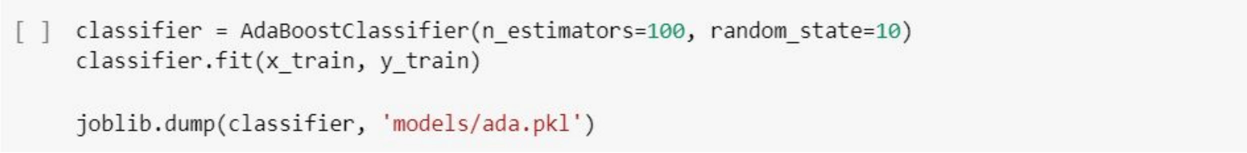


```
Accuracy Achieved for Random Forest:  96.92446856625962  %
              precision    recall  f1-score   support

          -1       0.98      0.95      0.97      1014
           1       0.96      0.98      0.97      1197

    accuracy                           0.97      2211
   macro avg       0.97      0.97      0.97      2211
weighted avg       0.97      0.97      0.97      2211
```

4. Adaboost :

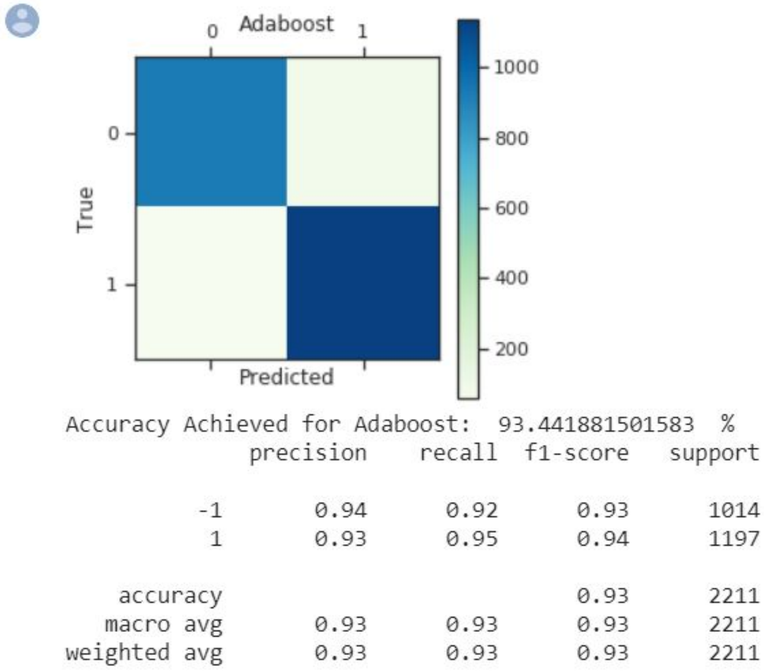
Boosting is a general ensemble method that creates a strong classifier from a number of weak classifiers.

AdaBoost was the first really successful boosting algorithm developed for binary classification. It is the best starting point for understanding boosting.



```
[ ]  classifier = AdaBoostClassifier(n_estimators=100, random_state=10)
     classifier.fit(x_train, y_train)

     joblib.dump(classifier, 'models/ada.pkl')
```

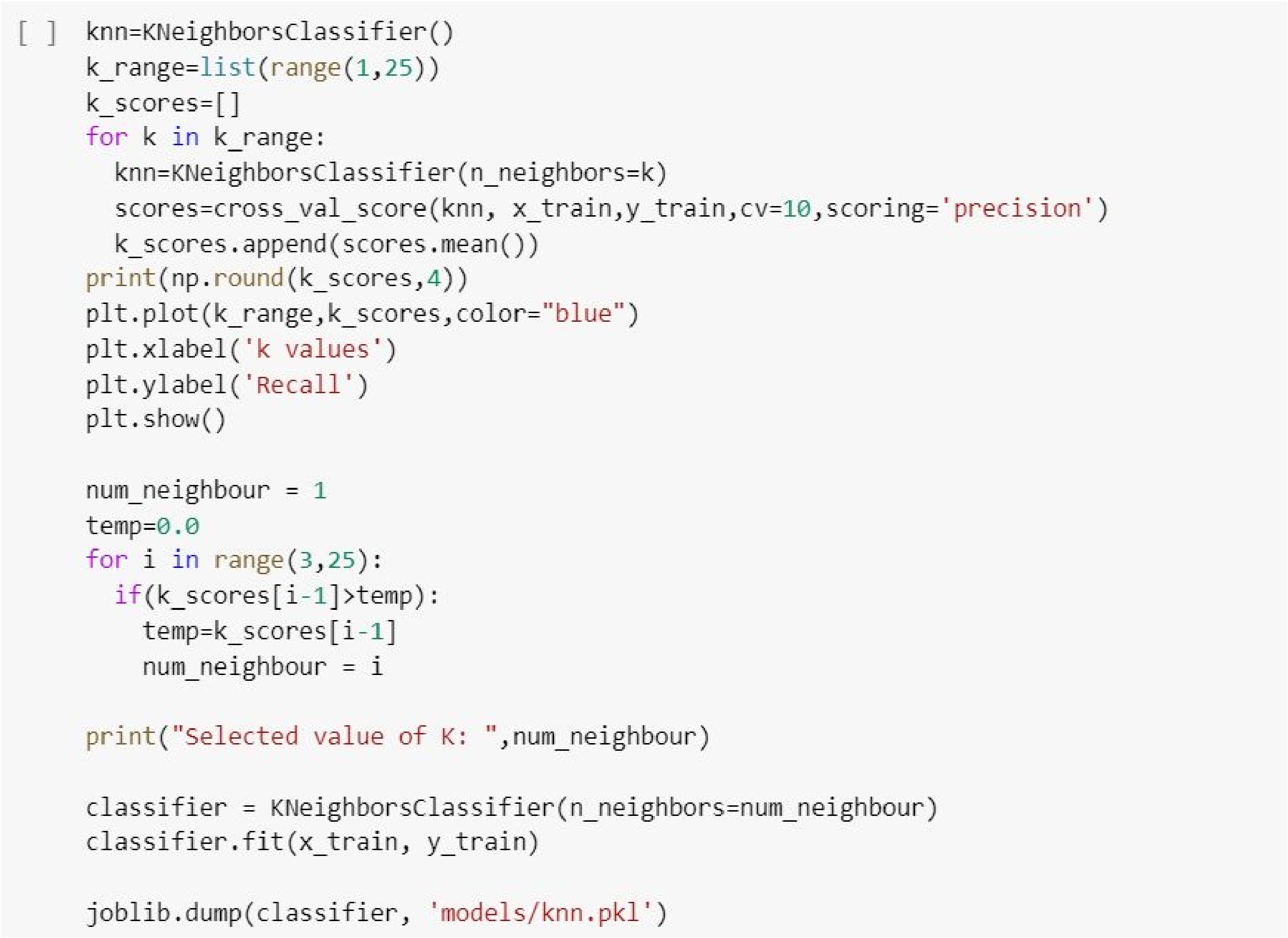
Classification Report:



```
Accuracy Achieved for Adaboost:  93.441881501583  %
              precision    recall  f1-score   support

          -1       0.94      0.92      0.93      1014
           1       0.93      0.95      0.94      1197

    accuracy                           0.93      2211
   macro avg       0.93      0.93      0.93      2211
weighted avg       0.93      0.93      0.93      2211
```

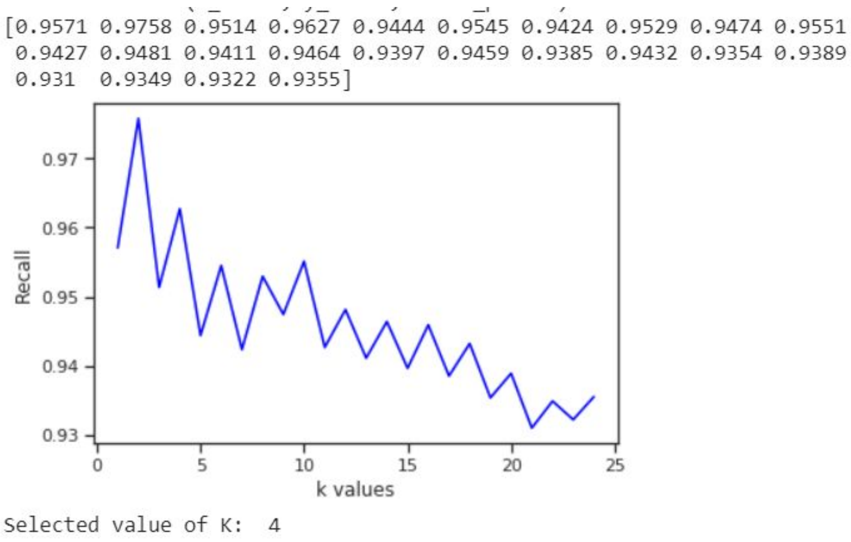
5. KNN :

The k-nearest neighbors (KNN) algorithm is a simple, easy-to-implement supervised machine learning algorithm that can be used to solve both classification and regression problems.



```
knn=KNeighborsClassifier()
k_range=list(range(1,25))
k_scores=[]
for k in k_range:
  knn=KNeighborsClassifier(n_neighbors=k)
  scores=cross_val_score(knn, x_train,y_train,cv=10,scoring='precision')
  k_scores.append(scores.mean())
print(np.round(k_scores,4))
plt.plot(k_range,k_scores,color="blue")
plt.xlabel('k values')
plt.ylabel('Recall')
plt.show()

num_neighbour = 1
temp=0.0
for i in range(3,25):
  if(k_scores[i-1]>temp):
    temp=k_scores[i-1]
    num_neighbour = i

print("Selected value of K: ",num_neighbour)

classifier = KNeighborsClassifier(n_neighbors=num_neighbour)
classifier.fit(x_train, y_train)

joblib.dump(classifier, 'models/knn.pkl')
```

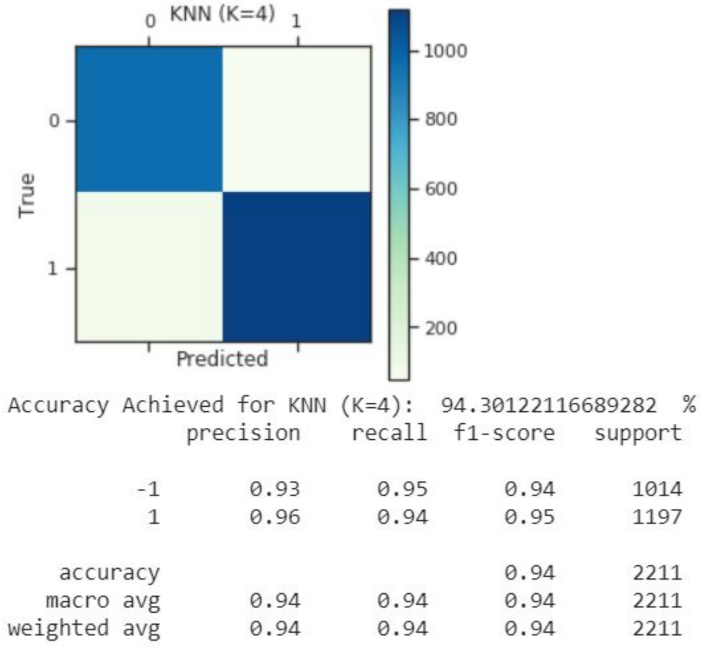
Selecting the best value of k:



```
[0.9571 0.9758 0.9514 0.9627 0.9444 0.9545 0.9424 0.9529 0.9474 0.9551
 0.9427 0.9481 0.9411 0.9464 0.9397 0.9459 0.9385 0.9432 0.9354 0.9389
 0.931  0.9349 0.9322 0.9355]
```

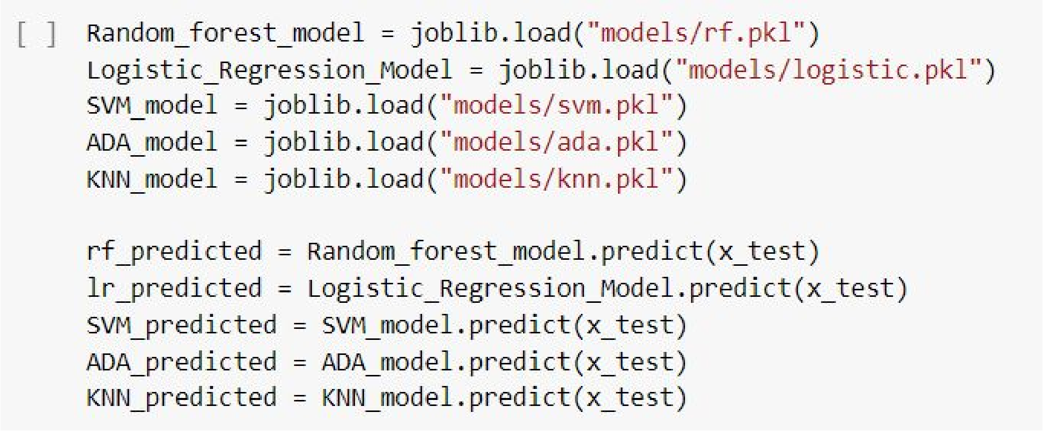
```
Selected value of K:  4
```

Classification Report:



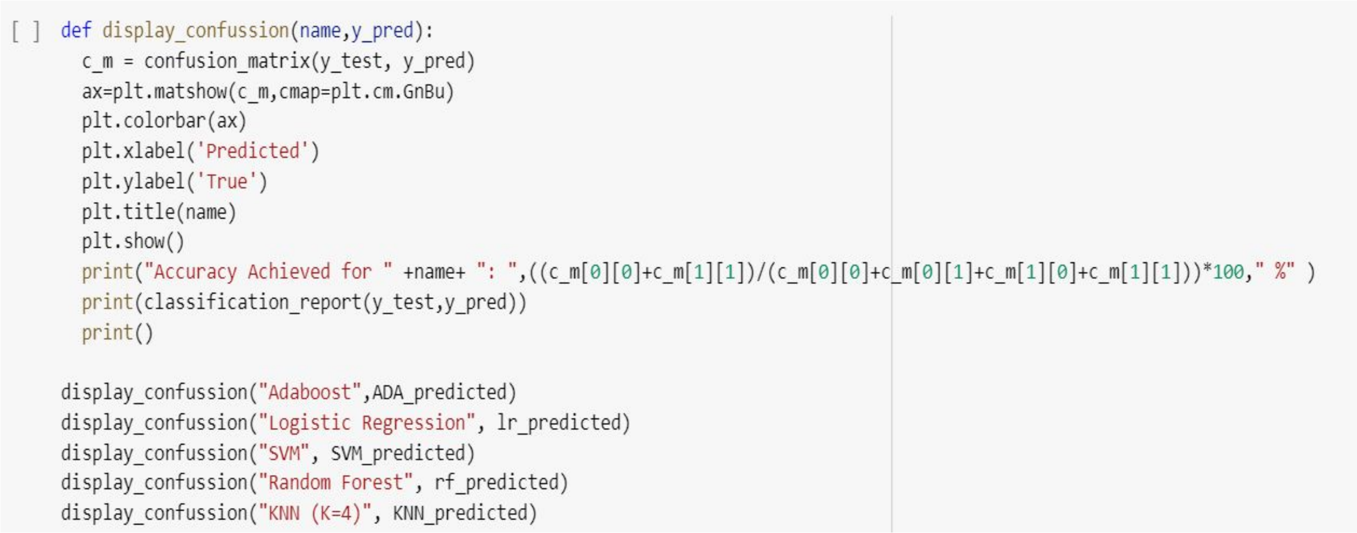
```
Accuracy Achieved for KNN (K=4):  94.30122116689282  %
              precision    recall  f1-score   support

          -1       0.93      0.95      0.94      1014
           1       0.96      0.94      0.95      1197

    accuracy                           0.94      2211
   macro avg       0.94      0.94      0.94      2211
weighted avg       0.94      0.94      0.94      2211
```

Importing all models and Predictions



```
[ ]  Random_forest_model = joblib.load("models/rf.pkl")
     Logistic_Regression_Model = joblib.load("models/logistic.pkl")
     SVM_model = joblib.load("models/svm.pkl")
     ADA_model = joblib.load("models/ada.pkl")
     KNN_model = joblib.load("models/knn.pkl")

     rf_predicted = Random_forest_model.predict(x_test)
     lr_predicted = Logistic_Regression_Model.predict(x_test)
     SVM_predicted = SVM_model.predict(x_test)
     ADA_predicted = ADA_model.predict(x_test)
     KNN_predicted = KNN_model.predict(x_test)
```

Displaying Confusion Matrix code:



```
[ ] def display_confussion(name,y_pred):
      c_m = confusion_matrix(y_test, y_pred)
      ax=plt.matshow(c_m,cmap=plt.cm.GnBu)
      plt.colorbar(ax)
      plt.xlabel('Predicted')
      plt.ylabel('True')
      plt.title(name)
      plt.show()
      print("Accuracy Achieved for " +name+ ": ",((c_m[0][0]+c_m[1][1])/(c_m[0][0]+c_m[0][1]+c_m[1][0]+c_m[1][1]))*100," %" )
      print(classification_report(y_test,y_pred))
      print()

    display_confussion("Adaboost",ADA_predicted)
    display_confussion("Logistic Regression", lr_predicted)
    display_confussion("SVM", SVM_predicted)
    display_confussion("Random Forest", rf_predicted)
    display_confussion("KNN (K=4)", KNN_predicted)
```

Accuracy of the Models

S. No	Model	Accuracy
1	Adaboost	93.44 %
2	Logistic Regression	91.72 %
3	SVM	96.69 %
4	Random Forest	96.92 %
5	KNN (K=4)	94.30 %

Inference

- ➔ Random Forest gives the most accurate classification of the given dataset.
- ➔ SVM and Random Forest are almost equally good
- ➔ Logistic Regression gives the least accurate accuracy.
- ➔ Although we used k=4 for KNN Classification, k=1 gave the best accuracy of 96.95% which shows how even 1-NN can be effective with data having minimal noise.

References

- Link to the live working of Jupyter Notebook: https://colab.research.google.com/drive/1T_NHgIZCloFVZQ1xCdQtkXfCN9xkEPkF
- UCL Machine Learning Repository https://archive.ics.uci.edu/ml/datasets/Phishing+Websites
- Tom Michell, Machine Learning, McGraw Hill.
- Guide to Google Colab https://towardsdatascience.com/getting-started-with-google-colab-f2fff97f594c
- Matplotlib Documentation https://matplotlib.org/3.1.1/users/index.html