Import the dataset

import pandas as pd

import numpy as np

data=pd.read\_csv("dataset\_website.csv")

data

**index having\_IPhaving\_IP\_Address URLURL\_Length Shortining\_Service having\_At\_Symbol double\_slash\_redirecting Prefix\_Suffix having\_Sub\_Domain SSLfinal\_State Domain\_registeration\_length ... popUpWidnow Iframe age\_of\_domain DNSRecord web\_traffic Page\_Rank Google\_Index Links\_pointing\_to\_page Statistical\_report Result**

0 1 -1 1 1 1 -1 -1 -1 -1 -1 ... 1 1 -1 -1 -1 -1 1 1 -1 -1

1 2 1 1 1 1 1 -1 0 1 -1 ... 1 1 -1 -1 0 -1 1 1 1 -1

2 3 1 0 1 1 1 -1 -1 -1 -1 ... 1 1 1 -1 1 -1 1 0 -1 -1

3 4 1 0 1 1 1 -1 -1 -1 1 ... 1 1 -1 -1 1 -1 1 -1 1 -1

4 5 1 0 -1 1 1 -1 1 1 -1 ... -1 1 -1 -1 0 -1 1 1 1 1

... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ...

11050 11051 1 -1 1 -1 1 1 1 1 -1 ... -1 -1 1 1 -1 -1 1 1 1 1

11051 11052 -1 1 1 -1 -1 -1 1 -1 -1 ... -1 1 1 1 1 1 1 -1 1 -1

11052 11053 1 -1 1 1 1 -1 1 -1 -1 ... 1 1 1 1 1 -1 1 0 1 -1

11053 11054 -1 -1 1

**11055 rows × 32 columns**

data.head()

index having\_IPhaving\_IP\_Address URLURL\_Length Shortining\_Service having\_At\_Symbol double\_slash\_redirecting Prefix\_Suffix having\_Sub\_Domain SSLfinal\_State Domain\_registeration\_length ... popUpWidnow Iframe age\_of\_domain DNSRecord web\_traffic Page\_Rank Google\_Index Links\_pointing\_to\_page Statistical\_report Result

0 1 -1 1 1 1 -1 -1 -1 -1 -1 ... 1 1 -1 -1 -1 -1 1 1 -1 -1

1 2 1 1 1 1 1 -1 0 1 -1 ... 1 1 -1 -1 0 -1 1 1 1 -1

2 3 1 0 1 1 1 -1 -1 -1 -1 ... 1 1 1 -1 1 -1 1 0 -1 -1

3 4 1 0 1 1 1 -1 -1 -1 1 ... 1 1 -1 -1 1 -1 1 -1 1 -1

4 5 1 0 -1 1 1 -1 1 1 -1 ... -1 1 -1 -1 0 -1 1 1 1 1

**5 rows × 32 columns**

**Numerical analysis:**

data**.**shape

(11055, 32)

data**.**size

353760

data**.**info()

RangeIndex: 11055 entries, 0 to 11054

Data columns (total 32 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 index 11055 non-null int64

1 having\_IPhaving\_IP\_Address 11055 non-null int64

2 URLURL\_Length 11055 non-null int64

3 Shortining\_Service 11055 non-null int64

4 having\_At\_Symbol 11055 non-null int64

5 double\_slash\_redirecting 11055 non-null int64

6 Prefix\_Suffix 11055 non-null int64

7 having\_Sub\_Domain 11055 non-null int64

8 SSLfinal\_State 11055 non-null int64

9 Domain\_registeration\_length 11055 non-null int64

10 Favicon 11055 non-null int64

11 port 11055 non-null int64

12 HTTPS\_token 11055 non-null int64

13 Request\_URL 11055 non-null int64

14 URL\_of\_Anchor 11055 non-null int64

15 Links\_in\_tags 11055 non-null int64

16 SFH 11055 non-null int64

17 Submitting\_to\_email 11055 non-null int64

18 Abnormal\_URL 11055 non-null int64

19 Redirect 11055 non-null int64

20 on\_mouseover 11055 non-null int64

21 RightClick 11055 non-null int64

22 popUpWidnow 11055 non-null int64

23 Iframe 11055 non-null int64

24 age\_of\_domain 11055 non-null int64

25 DNSRecord 11055 non-null int64

26 web\_traffic 11055 non-null int64

27 Page\_Rank 11055 non-null int64

28 Google\_Index 11055 non-null int64

29 Links\_pointing\_to\_page 11055 non-null int64

30 Statistical\_report 11055 non-null int64

31 Result 11055 non-null int64

dtypes: int64(32)

memory usage: 2.7 MB

data**.**describe()

**index having\_IPhaving\_IP\_Address URLURL\_Length Shortining\_Service having\_At\_Symbol double\_slash\_redirecting Prefix\_Suffix having\_Sub\_Domain SSLfinal\_State Domain\_registeration\_length ... popUpWidnow Iframe age\_of\_domain DNSRecord web\_traffic Page\_Rank Google\_Index Links\_pointing\_to\_page Statistical\_report Result**

count 11055.000000 11055.000000 11055.000000 11055.000000 11055.000000 11055.000000 11055.000000 11055.000000 11055.000000 11055.000000 ... 11055.000000 11055.000000 11055.000000 11055.000000 11055.000000 11055.000000 11055.000000 11055.000000 11055.000000 11055.000000

mean 5528.000000 0.313795 -0.633198 0.738761 0.700588 0.741474 -0.734962 0.063953 0.250927 -0.336771 ... 0.613388 0.816915 0.061239 0.377114 0.287291 -0.483673 0.721574 0.344007 0.719584 0.113885

std 3191.447947 0.949534 0.766095 0.673998 0.713598 0.671011 0.678139 0.817518 0.911892 0.941629 ... 0.789818 0.576784 0.998168 0.926209 0.827733 0.875289 0.692369 0.569944 0.694437 0.993539

min 1.000000 -1.000000 -1.000000 -1.000000 -1.000000 -1.000000 -1.000000 -1.000000 -1.000000 -1.000000 ... -1.000000 -1.000000 -1.000000 -1.000000 -1.000000 -1.000000 -1.000000 -1.000000 -1.000000 -1.000000

25% 2764.500000 -1.000000 -1.000000 1.000000 1.000000 1.000000 -1.000000 -1.000000 -1.000000 -1.000000 ... 1.000000 1.000000 -1.000000 -1.000000 0.000000 -1.000000 1.000000 0.000000 1.000000 -1.000000

50% 5528.000000 1.000000 -1.000000 1.000000 1.000000 1.000000 -1.000000 0.000000 1.000000 -1.000000 ... 1.000000 1.000000 1.000000 1.000000 1.000000 -1.000000 1.000000 0.000000 1.000000 1.000000

75% 8291.500000 1.000000 -1.000000 1.000000 1.000000 1.000000 -1.000000 1.000000 1.000000 1.000000 ... 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000

max 11055.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 ... 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000

8 rows × 32 columns

**Handling Null Values**

**Checking for Null values in a dataset and handling if any**

ata**.**isnull()**.**any()

index False

having\_IPhaving\_IP\_Address False

URLURL\_Length False

Shortining\_Service False

having\_At\_Symbol False

double\_slash\_redirecting False

Prefix\_Suffix False

having\_Sub\_Domain False

SSLfinal\_State False

Domain\_registeration\_length False

Favicon False

port False

HTTPS\_token False

Request\_URL False

URL\_of\_Anchor False

Links\_in\_tags False

SFH False

Submitting\_to\_email False

Abnormal\_URL False

Redirect False

on\_mouseover False

RightClick False

popUpWidnow False

Iframe False

age\_of\_domain False

DNSRecord False

web\_traffic False

Page\_Rank False

Google\_Index False

Links\_pointing\_to\_page False

Statistical\_report False

Result False

dtype: bool

In [11]:

data**.**isnull()**.**sum()

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

**Data Visualization**

**import** matplotlib.pyplot **as** plt

**import** seaborn **as** sns

**def** plot\_corr(df,size**=**10):

corr**=**df**.**corr()

fig,ax**=**plt**.**subplots(figsize**=**(size,size))

ax**.**legend()

cax**=**ax**.**matshow(corr)

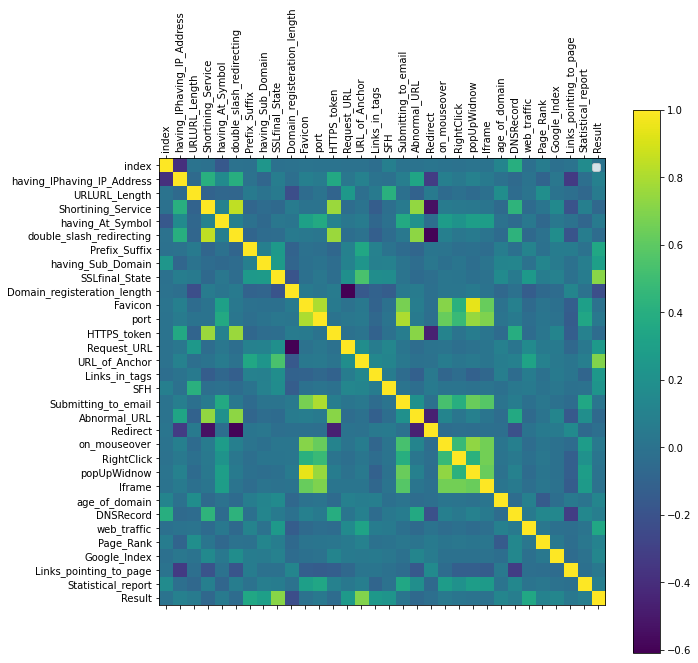
fig**.**colorbar(cax)

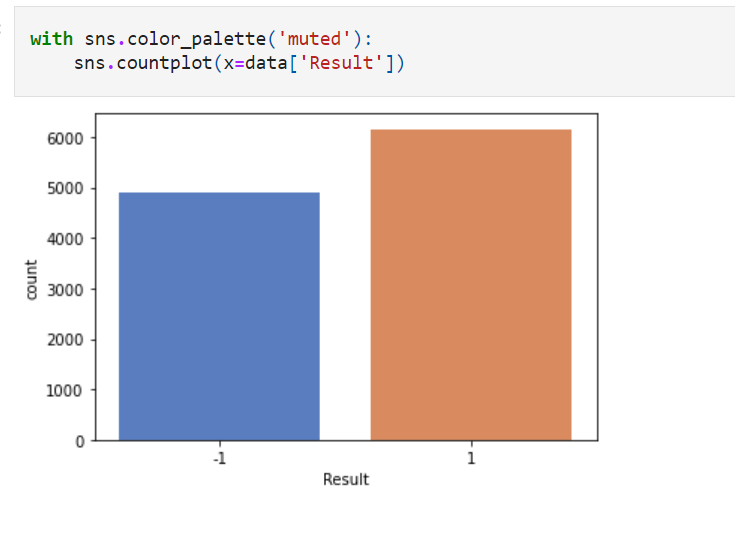
plt**.**xticks(range(len(corr**.**columns)), corr**.**columns, rotation**=**'vertical')

plt**.**yticks(range(len(corr**.**columns)), corr**.**columns)

plot\_corr(data)

No handles with labels found to put in legend.





**Splitting The Data:**

x**=**data**.**iloc[:,1:31]**.**values

y**=**data**.**iloc[:,**-**1]**.**values

print(x)

[[-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]]

print(y)

[-1 -1 -1 ... -1 -1 -1]

**Splitting data into train and test:**

**from** sklearn.model\_selection **import** train\_test\_split

x\_train,x\_test,y\_train,y\_test**=**train\_test\_split(x,y,test\_size**=**0.2,random\_state**=**0)

x\_train**.**shape

(8844, 30)

y\_train**.**shape

(8844,)

x\_test**.**shape

(2211, 30)

y\_test**.**shape

(2211,)

*# Creating holders to store the model performance results*

ML\_Model **=** []

acc\_train **=** []

acc\_test **=** []

*#function to call for storing the results*

**def** storeResults(model, a,b):

ML\_Model**.**append(model)

acc\_train**.**append(round(a, 3))

acc\_test**.**append(round(b, 3))

**Model Building**

**from** sklearn.metrics **import** accuracy\_score, classification\_report

**Logistic Regression**

**from** sklearn.linear\_model **import** LogisticRegression

lr**=**LogisticRegression()

lr**.**fit(x\_train,y\_train)

LogisticRegression()

y\_pred**=**lr**.**predict(x\_test)

**from** sklearn.metrics **import** accuracy\_score

y\_test\_lr **=** lr**.**predict(x\_test)

y\_train\_lr **=** lr**.**predict(x\_train)

acc\_train\_lr **=** accuracy\_score(y\_train,y\_train\_lr)**\***100

acc\_test\_lr **=** accuracy\_score(y\_test,y\_test\_lr)**\***100

storeResults('Logistic Regression', acc\_train\_lr, acc\_test\_lr)

log\_reg**=**accuracy\_score(y\_test,y\_pred)**\***100

print("Logistic Regression: Accuracy: {:.3f}"**.**format(log\_reg))

Logistic Regression: Accuracy: 91.678

**Random Forest**

*# Random Forest model*

**from** sklearn.ensemble **import** RandomForestClassifier

*# instantiate the model*

forest **=** RandomForestClassifier(max\_depth**=**5)

*# fit the model*

forest**.**fit(x\_train, y\_train)

RandomForestClassifier(max\_depth=5)

*#predicting the target value from the model for the samples*

y\_test\_forest **=** forest**.**predict(x\_test)

y\_train\_forest **=** forest**.**predict(x\_train)

*#computing the accuracy of the model performance*

acc\_train\_forest **=** accuracy\_score(y\_train,y\_train\_forest)**\***100

acc\_test\_forest **=** accuracy\_score(y\_test,y\_test\_forest)**\***100

storeResults('Random Forest', acc\_train\_forest, acc\_test\_forest)

print("Random forest: Accuracy on training Data: {:.3f}"**.**format(acc\_train\_forest))

print("Random forest: Accuracy on test Data: {:.3f}"**.**format(acc\_test\_forest))

Random forest: Accuracy on training Data: 93.397

Random forest: Accuracy on test Data: 93.171

**Support Vector Machine**

*#Support vector machine model*

**from** sklearn.svm **import** SVC

*# instantiate the model*

svm **=** SVC(kernel**=**'linear', C**=**1.0, random\_state**=**12)

*#fit the model*

svm**.**fit(x\_train, y\_train)

SVC(kernel='linear', random\_state=12)

*#predicting the target value from the model for the samples*

y\_test\_svm **=** svm**.**predict(x\_test)

y\_train\_svm **=** svm**.**predict(x\_train)

*#computing the accuracy of the model performance*

acc\_train\_svm **=** accuracy\_score(y\_train,y\_train\_svm)**\***100

acc\_test\_svm **=** accuracy\_score(y\_test,y\_test\_svm)**\***100

print("SVM: Accuracy on training Data: {:.3f}"**.**format(acc\_train\_svm))

print("SVM : Accuracy on test Data: {:.3f}"**.**format(acc\_test\_svm))

storeResults('SVM', acc\_train\_svm, acc\_test\_svm)

SVM: Accuracy on training Data: 93.069

SVM : Accuracy on test Data: 91.814

results **=** pd**.**DataFrame({ 'ML Model': ML\_Model,

Train Accuracy': acc\_train,

'Test Accuracy': acc\_test})

results

|  | **ML Model** | **Train Accuracy** | **Test Accuracy** |
| --- | --- | --- | --- |
| **0** | Logistic Regression | 93.193 | 91.678 |
| **1** | Random Forest | 93.397 | 93.171 |
| **2** | SVM | 93.069 | 91.814 |

**import** matplotlib.pyplot **as** plt

plt**.**bar(ML\_Model,acc\_test,width**=**0.3,color**=**['yellow','grey','red'])

## 

## Conclusion:

**Based on accuracy we are going with random forest**