Codes:

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

%matplotlib inline

import seaborn as sns

import time

from sklearn.metrics import classification\_report

from sklearn.metrics import confusion\_matrix

from nltk.tokenize import RegexpTokenizer

from nltk.stem.snowball import SnowballStemmer

from sklearn.feature\_extraction.text import CountVectorizer

from sklearn.pipeline import make\_pipeline

import joblib

# Loading the dataset

df= pd.read\_csv("dataset\_phishing.csv")

df.head()

df.info()

df.describe()

df.shape

df.isnull().sum()

plt.figure(figsize=(5, 5)) # Adjust the figure size if needed

status\_counts = df['status'].value\_counts()

colors = ['skyblue', 'lightcoral']

explode = (0.5, 0)

plt.pie(status\_counts, labels=status\_counts.index, autopct='%1.1f%%', startangle=90, wedgeprops=dict(width=0.4), explode=explode, colors=colors)

plt.title("Distribution of Status")

# Add legend

plt.legend(status\_counts.index, loc="best")

plt.show()

# checking unique values and counts from the collected object features

df['status'].value\_counts()

tokenizer = RegexpTokenizer(r'[A-Za-z]+')

tokenizer.tokenize(df.url[0]) # this will fetch all the words from the first URL

# Tokenizing all the rows

print('Getting words tokenized ...')

t0= time.perf\_counter()

df['text\_tokenized'] = df.url.map(lambda t: tokenizer.tokenize(t))

t1 = time.perf\_counter() - t0

print('Time taken',t1 ,'sec')

stemmer = SnowballStemmer("english") # choose a language

# Getting all the stemmed words

print('Getting words stemmed ...')

t0= time.perf\_counter()

df['text\_stemmed'] = df['text\_tokenized'].map(lambda l: [stemmer.stem(word) for word in l])

t1= time.perf\_counter() - t0

print('Time taken',t1 ,'sec')

# Joining all the stemmmed words.

print('Get joiningwords ...')

t0= time.perf\_counter()

df['text\_sent'] = df['text\_stemmed'].map(lambda l: ' '.join(l))

t1= time.perf\_counter() - t0

print('Time taken',t1 ,'sec')

bad\_sites = df[df.status == 'phishing']

good\_sites = df[df.status == 'legitimate']

cv = CountVectorizer()

feature = cv.fit\_transform(df.text\_sent) #transform all text which we tokenize and stemed

feature[:5].toarray() # convert sparse matrix into array to print transformed features

joblib.dump(cv, 'count\_vectorizer.pkl')

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

trainX, testX, trainY, testY = train\_test\_split(feature, df.status)

from sklearn.metrics import accuracy\_score,confusion\_matrix,classification\_report

from sklearn.svm import SVC

svm\_clf =SVC(C=100, gamma=0.0001, kernel='rbf')

svm\_clf.fit(trainX,trainY)

pred=svm\_clf.predict(testX)

accuracy\_score(pred,testY)

print('Training Accuracy :',svm\_clf.score(trainX,trainY))

print('Testing Accuracy :',svm\_clf.score(testX,testY))

con\_mat = pd.DataFrame(confusion\_matrix(svm\_clf.predict(testX), testY),

columns = ['Predicted:Bad', 'Predicted:Good'],

index = ['Actual:Bad', 'Actual:Good'])

print('\nCLASSIFICATION REPORT\n')

print(classification\_report(svm\_clf.predict(testX), testY,

target\_names =['Bad','Good']))

print('\nCONFUSION MATRIX')

plt.figure(figsize= (6,4))

sns.heatmap(con\_mat, annot = True,fmt='d',cmap="YlGnBu")

from sklearn.ensemble import RandomForestClassifier

forest\_clf = RandomForestClassifier(max\_depth=10, random\_state=0)

forest\_clf.fit(trainX,trainY)

ran\_pred=forest\_clf.predict(testX)

accuracy\_score(ran\_pred,testY)

print('Training Accuracy :',forest\_clf.score(trainX,trainY))

print('Testing Accuracy :',forest\_clf.score(testX,testY))

con\_mat = pd.DataFrame(confusion\_matrix(forest\_clf.predict(testX), testY),

columns = ['Predicted:Bad', 'Predicted:Good'],

index = ['Actual:Bad', 'Actual:Good'])

print('\nCLASSIFICATION REPORT\n')

print(classification\_report(forest\_clf.predict(testX), testY,

target\_names =['Bad','Good']))

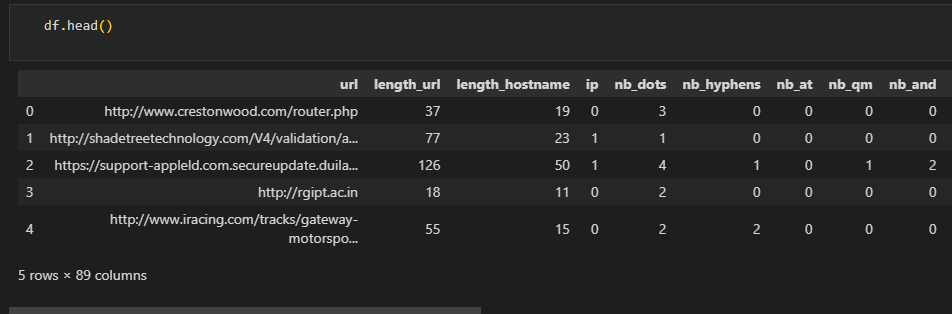
print('\nCONFUSION MATRIX')

plt.figure(figsize= (6,4))

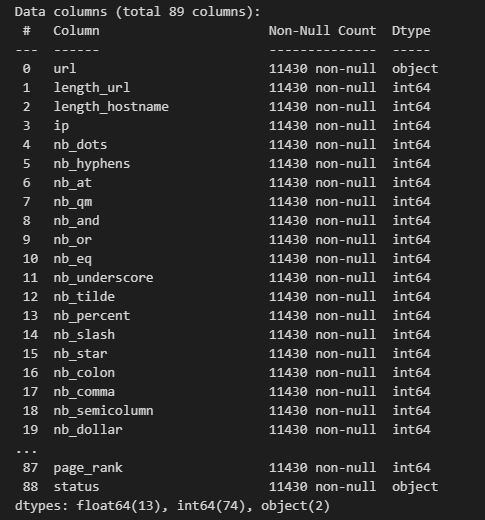
sns.heatmap(con\_mat, annot = True,fmt='d',cmap="YlGnBu")

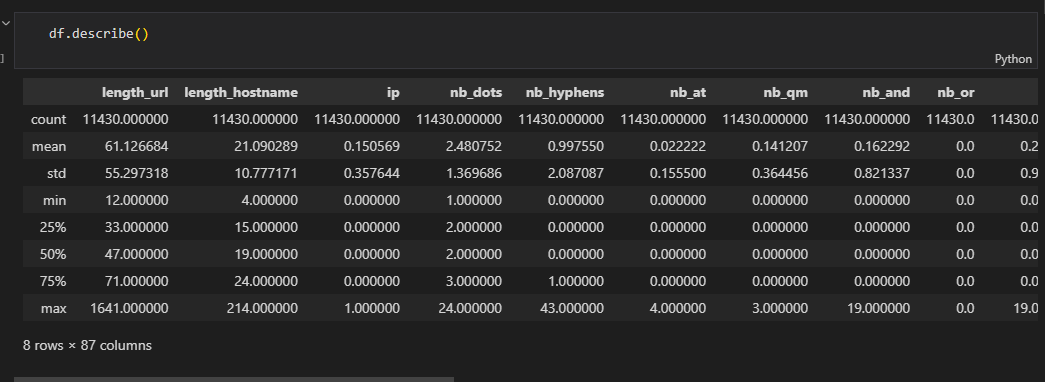
**Screenshots**

Dataset Sample Values

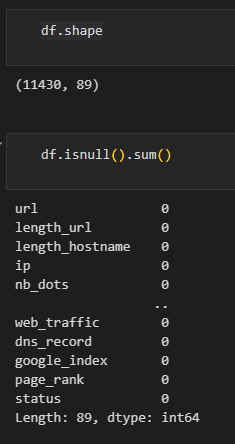


Columns names and Datatypes

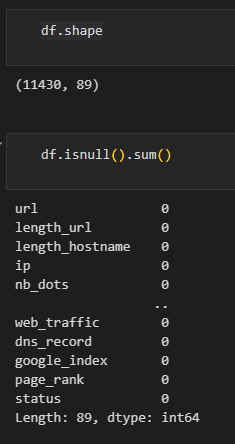


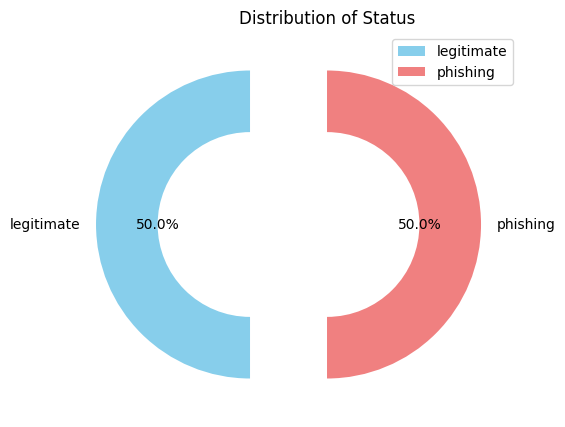
Statistical Analysis 

Dataset Rows and Columns



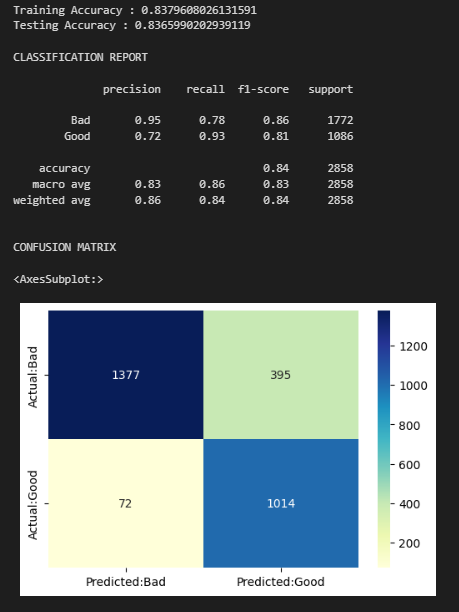
Missing Values Count



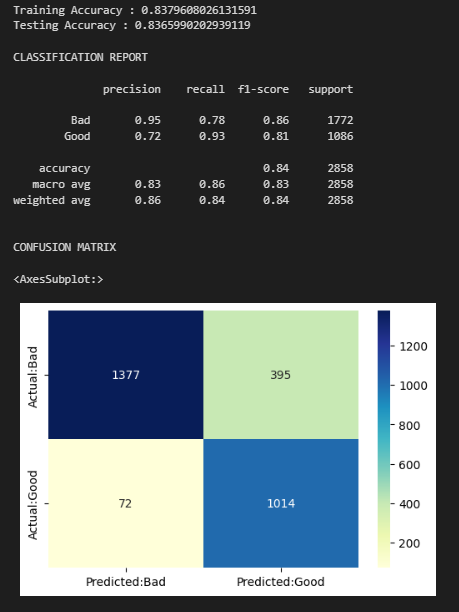


Random Forest

Accuracy and Classification Report

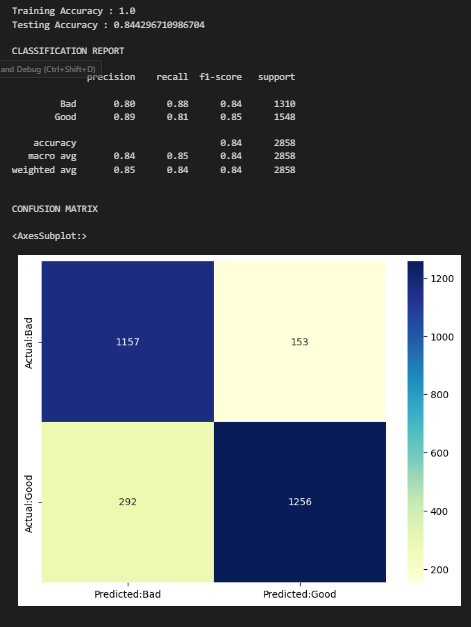


Confusion Matrix

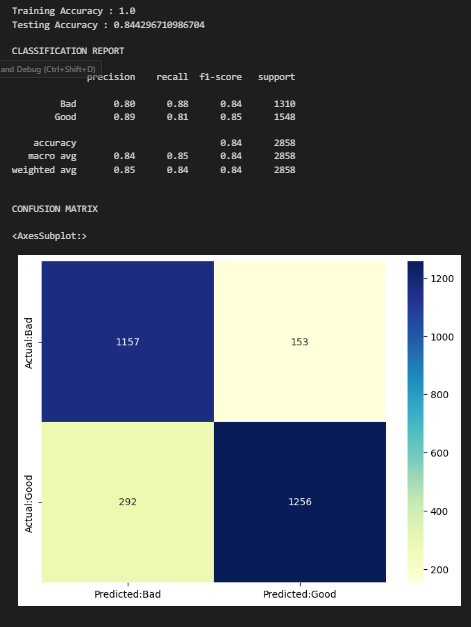


Decision Tree

Accuracy and Classification Report



Confusion Matrix



Comparison Chart

