FLIP ROBO

Email Spam Detection Project

Submitted by:

NARA LOHITH

ACKNOWLEDGMENT

The project entitled "Email Spam Detection" is done by me during my internship with Flip Robo Technologies. I am grateful to Data Trained and Flip Robo Technologies for their guidance during this project.

INTRODUCTION

- Spam email is unsolicited and unwanted junk email sent out in bulk to an indiscriminate recipient list. Typically, spam is sent for commercial purposes. We are recently hired in a start-up company and are asked to build a system to identify spam emails.

- This task can be done through Natural Language Processing (NLP), which processes text into useful insights.

- Firstly, we have analysed the data, removed nan values, done text cleaning such as removal of unnecessary stop words, punctuations, checked length of each columns before and after cleaning, done feature extraction and finally did training and testing of our model.

- By detecting unsolicited and unwanted emails, we can prevent spam messages from creeping into the user's inbox, thereby improving user experience.

Analytical Problem Framing

- The sample data is provided to us in csv format and hence we import it using pandas. Then we further checked more about data using info, shapes using .shape, value counts using value_counts(), null values using .isnull. .sum().sum(), and further visualize it through heatmap as follows:

```
In [1]: import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt

        import warnings
        warnings.filterwarnings('ignore')
```

```
In [2]: msg=pd.read_csv('messages.csv')
        msg
```

Out[2]:

	subject	message	label
0	job posting - apple-iss research center	content - length : 3386 apple-iss research cen...	0
1	NaN	lang classification grimes , joseph e . and ba...	0
2	query : letter frequencies for text identifica...	i am posting this inquiry for sergei atamas (...	0
3	risk	a colleague and i are researching the differin...	0
4	request book information	earlier this morning i was on the phone with a...	0
...	...	...	...
2888	love your profile - ysuolvpv	hello thanks for stopping by ! ! we have taken...	1
2889	you have been asked to join kiddin	the list owner of : " kiddin " has invited you...	1
2890	anglicization of composers ' names	judging from the return post , i must have sou...	0
2891	re : 6 . 797 , comparative method : n - ary co...	gotcha ! there are two separate fallacies in t...	0
2892	re : american - english in australia	hello ! i ' m working on a thesis concerning a...	0

2893 rows × 3 columns

```
In [3]: print("Shape-" , msg.shape)

        Shape- (2893, 3)
```

```
In [4]: msg.label.value_counts()

Out[4]: 0    2412
        1     481
        Name: label, dtype: int64
```

```
In [5]: msg.info()

        <class 'pandas.core.frame.DataFrame'>
        RangeIndex: 2893 entries, 0 to 2892
        Data columns (total 3 columns):
         #   Column   Non-Null Count  Dtype
        ---  ------   --------------  -----
         0   subject  2831 non-null   object
         1   message  2893 non-null   object
         2   label    2893 non-null   int64
        dtypes: int64(1), object(2)
        memory usage: 67.9+ KB
```

```
In [6]: msg.isnull().sum().sum()

Out[6]: 62
```

```
In [7]: msg.isnull().values.any()

Out[7]: True
```

- For preparing the data, first we found ratio of each category i.e, spam and non-spam in labels where 83% messages were non-spam and rest 17% were spam. Then we added two more columns to the dataframe where we calculate lengths of string in each column that can be compared with the lengths after cleaning.

```
In [10]: #Ratio
print ('spam ratio = ', round(len(msg[msg['label']==1]) / len(msg.label),2)*100,'%')
print ('non-spam ratio  = ', round(len(msg[msg['label']==0]) / len(msg.label),2)*100,'%')

spam ratio =  17.0 %
non-spam ratio  =  83.0 %
```

```
In [11]: msg['l1']=msg.subject.str.len()
msg['l2']=msg.message.str.len()
msg
```

Out[11]:

	subject	message	label	l1	l2
0	job posting - apple-iss research center	content - length : 3386 apple-iss research cen...	0	39.0	2856
1	NaN	lang classification grimes , joseph e . and ba...	0	NaN	1800
2	query : letter frequencies for text identifica...	i am posting this inquiry for sergei atamas (...	0	50.0	1435
3	risk	a colleague and i are researching the differin...	0	4.0	324
4	request book information	earlier this morning i was on the phone with a...	0	24.0	1046
...	...	...	...	...	...
2888	love your profile - ysuolvpv	hello thanks for stopping by ! ! we have taken...	1	28.0	262
2889	you have been asked to join kiddin	the list owner of : " kiddin " has invited you...	1	34.0	2163
2890	anglicization of composers ' names	judging from the return post , i must have sou...	0	34.0	1039
2891	re : 6 . 797 , comparative method : n - ary co...	gotcha ! there are two separate fallacies in t...	0	54.0	2949
2892	re : american - english in australia	hello ! i ' m working on a thesis concerning a...	0	36.0	700

2893 rows × 5 columns

- Then we dropped all the NAN values and converted all the strings from the columns i.e, subject and message to lower case. Further, we have replaced email addresses with 'email', URLs with 'webaddress', money symbols with 'moneysymb' or ‘dollars’, 10 digit phone numbers (formats include paranthesis, spaces, no spaces, dashes) with 'phonenumber' and numbers with 'numbr' for all the strings in both columns. Punctuations were also removed from both columns.

```
In [17]: # Replace email addresses with 'email'
msg['subject'] = msg['subject'].str.replace(r'^.+@[^\.].*\.[a-z]{2,}$',
                                 'emailaddress')

# Replace URLs with 'webaddress'
msg['subject'] = msg['subject'].str.replace(r'^http\://[a-zA-Z0-9\-\.]+\.[a-zA-Z]{2,3}(/\S*)?$',
                                  'webaddress')

# Replace money symbols with 'moneysymb' (£ can by typed with ALT key + 156)
msg['subject'] = msg['subject'].str.replace(r'£|\$', 'dollers')

# Replace 10 digit phone numbers (formats include paranthesis, spaces, no spaces, dashes) with 'phonenumber'
msg['subject'] = msg['subject'].str.replace(r'^\(?[\d]{3}\)?[\s-]?[\d]{3}[\s-]?[\d]{4}$',
                                  'phonenumber')


# Replace numbers with 'numbr'
msg['subject'] = msg['subject'].str.replace(r'\d+(\.\d+)?', 'numbr')
```

```
In [18]: # Replace email addresses with 'email'
msg['message'] = msg['message'].str.replace(r'^.+@[^\.].*\.[a-z]{2,}$',
                                 'emailaddress')

# Replace URLs with 'webaddress'
msg['message'] = msg['message'].str.replace(r'^http\://[a-zA-Z0-9\-\.]+\.[a-zA-Z]{2,3}(/\S*)?$',
                                  'webaddress')

# Replace money symbols with 'moneysymb' (£ can by typed with ALT key + 156)
msg['message'] = msg['message'].str.replace(r'£|\$', 'dollers')

# Replace 10 digit phone numbers (formats include paranthesis, spaces, no spaces, dashes) with 'phonenumber'
msg['message'] = msg['message'].str.replace(r'^\(?[\d]{3}\)?[\s-]?[\d]{3}[\s-]?[\d]{4}$',
                                  'phonenumber')


# Replace numbers with 'numbr'
msg['message'] = msg['message'].str.replace(r'\d+(\.\d+)?', 'numbr')
```

```
In [20]: # Remove punctuation from subject column
# Remove punctuation
msg['subject'] = msg['subject'].str.replace(r'[^\w\d\s]', ' ')

# Replace whitespace between terms with a single space
msg['subject'] = msg['subject'].str.replace(r'\s+', ' ')

# Remove leading and trailing whitespace
msg['subject'] = msg['subject'].str.replace(r'^\s+|\s+?$', '')
```

```
In [21]: # Remove punctuation from message column
# Remove punctuation
msg['message'] = msg['message'].str.replace(r'[^\w\d\s]', ' ')

# Replace whitespace between terms with a single space
msg['message'] = msg['message'].str.replace(r'\s+', ' ')

# Remove leading and trailing whitespace
msg['message'] = msg['message'].str.replace(r'^\s+|\s+?$', '')
```

- In the second part we built word dictionary in which all the stop words from English present in the dataset were removed and rest of the words were appended and then the new length of the strings were calculated as follows:

```
In [23]: import string
         import nltk
         from nltk.corpus import  stopwords

In [24]: stop_words = set(stopwords.words('english'))

         msg['subject'] = msg['subject'].apply(lambda x: ' '.join(
             term for term in x.split() if term not in stop_words))

         msg['message'] = msg['message'].apply(lambda y: ' '.join(
             term for term in y.split() if term not in stop_words))

In [25]: # New columns (clean_lengths) after puncuations,stopwords removal
         msg['clean_length_l1'] = msg.subject.str.len()
         msg['clean_length_l2'] = msg.message.str.len()

In [26]: msg
Out[26]:
```

	subject	message	label	l1	l2	clean_length_l1	clean_length_l2
0	job posting apple iss research center	content length numbr apple iss research center...	0	39.0	2856	37	2179
2	query letter frequencies text identification	posting inquiry sergei atamas satamas umabnet ...	0	50.0	1435	44	1064
3	risk	colleague researching differing degrees risk p...	0	4.0	324	4	210
4	request book information	earlier morning phone friend mine living south...	0	24.0	1046	24	629
5	call abstracts optimality syntactic theory	content length numbr call papers best good eno...	0	51.0	4492	42	3370
...	...	...	...	...	...	...	...
2888	love profile ysuolvpv	hello thanks stopping taken many new pics made...	1	28.0	262	21	132
2889	asked join kiddin	list owner kiddin invited join mailing list li...	1	34.0	2163	17	1217
2890	anglicization composers names	judging return post must sounded like kind sel...	0	34.0	1039	29	643
2891	numbr numbr comparative method n ary comparison	gotcha two separate fallacies argument n ary c...	0	54.0	2949	47	1873
2892	american english australia	hello working thesis concerning attitudes towa...	0	36.0	700	26	434

2831 rows × 7 columns

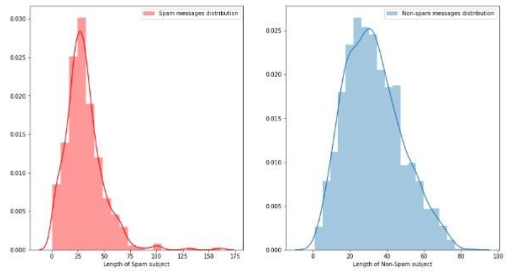
- In the next step we did feature extraction where we first compared lengths of strings in both columns before and after cleaning.

3. Feature Extraction

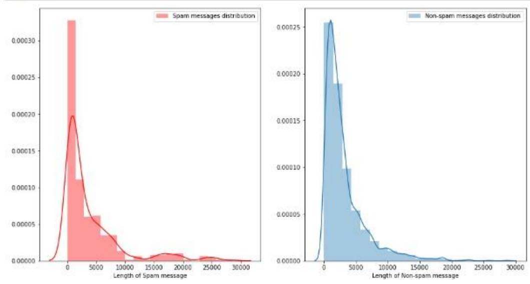
```
In [27]: # Total lengths of subject and message columns before and after removal

         print ('Original Length of subject column', msg.l1.sum())
         print ('New/clean Length of subject column', msg.clean_length_l1.sum())
         print ('Original Length of message column', msg.l2.sum())
         print ('New/clean Length of message column', msg.clean_length_l2.sum())

Original Length of subject column 91663.0
New/clean Length of subject column 78357
Original Length of message column 9186422
New/clean Length of message column 6646683
```

Then, using distplot we have have visualized the word count distribution BEFORE cleaning for subject and message column in both categories i.e, spam and non-spam as follows:

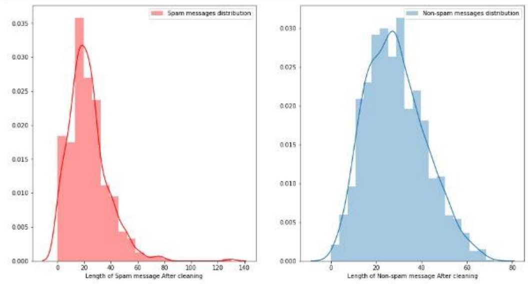
```
In [28]: # Word count distribution BEFORE cleaning for subject column

         f,ax = plt.subplots(1,2,figsize = (15,8))

         sns.distplot(msg[msg['label']==1]['l1'],bins=20,ax=ax[0],label='Spam messages distribution',color='r')

         ax[0].set_xlabel('Length of Spam subject')
         ax[0].legend()

         sns.distplot(msg[msg['label']==0]['l1'],bins=20,ax=ax[1],label='Non-spam messages distribution')
         ax[1].set_xlabel('Length of Non-Spam subject')
         ax[1].legend()

         plt.show()
```



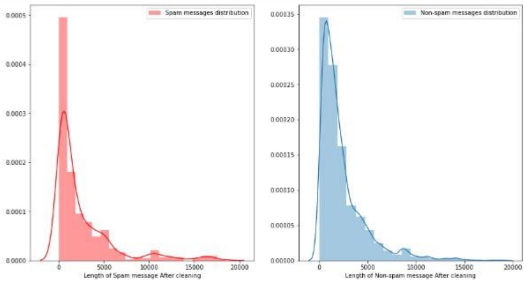

```
In [29]: # Word count distribution BEFORE cleaning for message column
         f,ax = plt.subplots(1,2,figsize = (15,8))

         sns.distplot(msg[msg['label']==1]['l2'],bins=20,ax=ax[0],label='Spam messages distribution',color='r')

         ax[0].set_xlabel('Length of Spam message')
         ax[0].legend()

         sns.distplot(msg[msg['label']==0]['l2'],bins=20,ax=ax[1],label='Non-spam messages distribution')
         ax[1].set_xlabel('Length of Non-spam message')
         ax[1].legend()

         plt.show()
```



```
In [30]: # Word count distribution AFTER cleaning for subject column
f,ax = plt.subplots(1,2,figsize = (15,8))

sns.distplot(msg[msg['label']==1]['clean_length_l1'],bins=20,ax=ax[0],label='Spam messages distribution',color='r')
ax[0].set_xlabel('Length of Spam message After cleaning')
ax[0].legend()

sns.distplot(msg[msg['label']==0]['clean_length_l1'],bins=20,ax=ax[1],label='Non-spam messages distribution')
ax[1].set_xlabel('Length of Non-spam message After cleaning')
ax[1].legend()

plt.show()
```

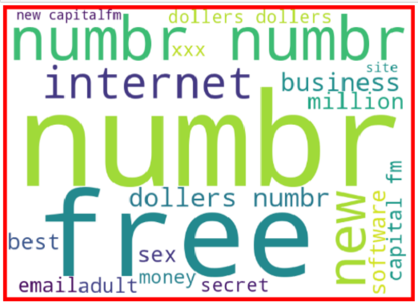



```
In [31]: # Word count distribution AFTER cleaning for message column
f,ax = plt.subplots(1,2,figsize = (15,8))

sns.distplot(msg[msg['label']==1]['clean_length_l2'],bins=20,ax=ax[0],label='Spam messages distribution',color='r')
ax[0].set_xlabel('Length of Spam message After cleaning')
ax[0].legend()

sns.distplot(msg[msg['label']==0]['clean_length_l2'],bins=20,ax=ax[1],label='Non-spam messages distribution')
ax[1].set_xlabel('Length of Non-spam message After cleaning')
ax[1].legend()

plt.show()
```

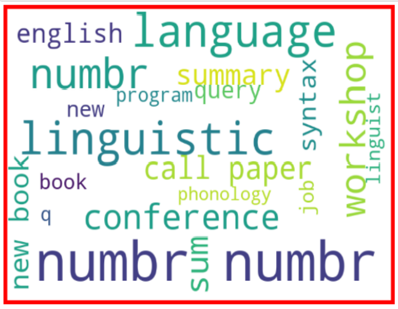


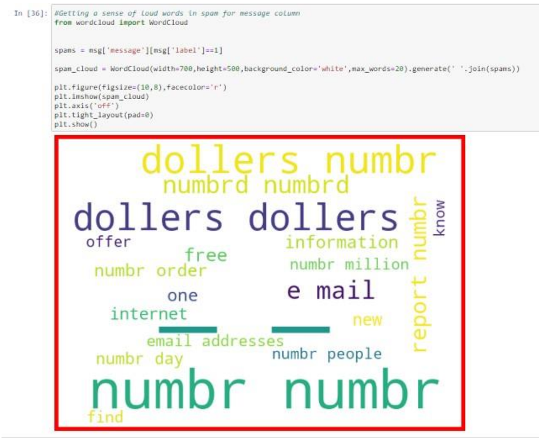
To get sense of loud words in spam and non-spam messages for subject and message columns we have used wordcloud and visualized it as follows:

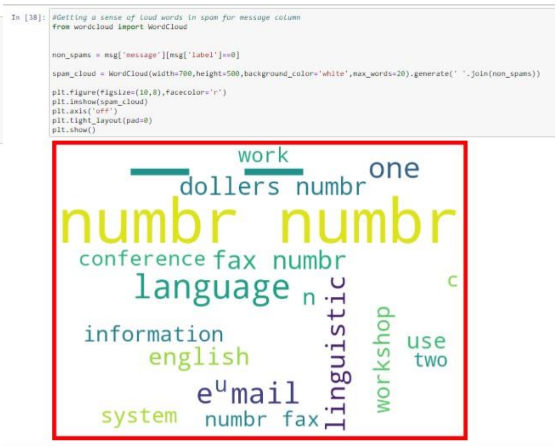
```
In [34]: #Getting a sense of loud words in spam for subject column
from wordcloud import WordCloud


spams = msg['subject'][msg['label']==1]

spam_cloud = WordCloud(width=700,height=500,background_color='white',max_words=20).generate(' '.join(spams))

plt.figure(figsize=(10,8),facecolor='r')
plt.imshow(spam_cloud)
plt.axis('off')
plt.tight_layout(pad=0)
plt.show()
```




```
In [37]: #Getting a sense of loud words in spam for subject column
from wordcloud import WordCloud


non_spams = msg['subject'][msg['label']==0]

spam_cloud = WordCloud(width=700,height=500,background_color='white',max_words=20).generate(' '.join(non_spams))

plt.figure(figsize=(10,8),facecolor='r')
plt.imshow(spam_cloud)
plt.axis('off')
plt.tight_layout(pad=0)
plt.show()
```







- In this project we have used HP Pavilion PC with 64-bit operating system and have Windows 10 pro. We have used python to develop this project in which we have used various libraries such as numpy, pandas, matplotlib, seaborn, wordcloud for handling data or arrays and their visualization. To build dictionary we have imported string, NLTK and from NLTK we have imported stopwords. To convert text into vectors we have used TfidfVectorizer. Lastly, to develop the model we have used various libraries and metrics from sklearn such as train_test_split, Logistic Regression, SVC, Decision Tree Classifier, KNeighbors Classifier, MultinomialNB, accuracy_score, confusion_matrix, classification_report, roc_curve, auc, roc_auc_score.

Model/s Development and Evaluation

- Message and subject columns have been converted to tokens using TFidfVectorizer. Then using train_test_split we split the data into training and testing dataset.

4.Training Classifiers

```
In [39]: # Convert text into vectors using TF-IDF
         # Split feature and label
         from sklearn.feature_extraction.text import TfidfVectorizer
         from sklearn.model_selection import train_test_split

         tf_vec = TfidfVectorizer()

         features = msg["subject"] + msg["message"]

         x = tf_vec.fit_transform(features)

         y = msg['label']

In [40]: # Train and predict
         x_train,x_test,y_train,y_test = train_test_split(x,y,random_state=42)
```

- We have used following algorithms such as: Logistic Regression, SVC, Decision Tree Classifier, KNeighbors Classifier, MultinomialNB.

- We have formed a loop where all the algorithms will be used one by one and their corresponding accuracy_score, cross_val_score, roc_auc_score and classification report will be evaluated. Finally corresponding to each algorithm roc curve will be printed.

5. Testing

```
In [46]: #importing all the libraries required for carrying out machine learning process
         from sklearn.model_selection import cross_val_score
         from sklearn.linear_model import LogisticRegression
         from sklearn.naive_bayes import MultinomialNB

         from sklearn.svm import SVC
         from sklearn.tree import DecisionTreeClassifier
         from sklearn.neighbors import KNeighborsClassifier
         from sklearn.metrics import accuracy_score,confusion_matrix,classification_report,roc_curve,auc,roc_auc_score

In [47]: #using a range of base algorithms
         lg=LogisticRegression()
         KNN=KNeighborsClassifier()
         SV=SVC()
         DT=DecisionTreeClassifier()
         MNB=MultinomialNB()

In [48]: #creating a list model where each of the models will be appended
         models=[]
         models.append(('LogisticRegression',lg))
         models.append(('KNeighborsClassifier',KNN))
         models.append(('SVC',SV))
         models.append(('DecisionTreeClassifier',DT))
         models.append(('MultinomialNB',MNB))
```

```
In [49]: Model=[]
score=[]
cvs=[]
rocscore=[]
for name,model in models:
    print('**************',name,'***************')
    print('\n')
    Model.append(name)
    model.fit(x_train,y_train)
    print(model)
    pre=model.predict(x_test)
    print('\n')
    AS=accuracy_score(y_test,pre)
    print('Accuracy_score=',AS)
    score.append(AS*100)
    print('\n')
    sc=cross_val_score(model,x,y,cv=10,scoring='accuracy').mean()
    print('Cross_Val_Score=',sc)
    cvs.append(sc*100)
    print('\n')
    false_positive_rate,true_positive_rate,thresholds=roc_curve(y_test,pre)
    roc_auc=auc(false_positive_rate,true_positive_rate)
    print('roc_auc_score=',roc_auc)
    rocscore.append(roc_auc*100)
    print('\n')
    print('classification report\n',classification_report(y_test,pre))
    print('\n')
    cm=confusion_matrix(y_test,pre)
    print(cm)
    print('\n')
    plt.figure(figsize=(10,40))
    plt.subplot(911)
    plt.title(name)
    print(sns.heatmap(cm,annot=True))
    plt.subplot(912)
    plt.title(name)
    plt.plot(false_positive_rate,true_positive_rate,label='AUC= %0.2f'% roc_auc)
    plt.plot([0,1],[0,1],'r--')
    plt.legend(loc='lower right')
    plt.ylabel('True_Positive_Rate')
    plt.xlabel('False_Positive_Rate')
    print('\n\n')
```

- Key metrics used for finalising the model was accuracy_score, cross_val_score and roc_auc_score. Since in case of SVC it's giving us good score among all other models and it's performing well. It's cross_val_score is also satisfactory and it shows that our model is neither underfitting/overfitting.

```
************** LogisticRegression ***************


LogisticRegression()


Accuracy_score= 0.9576271186440678


Cross_Val_Score= 0.9530172199273379


roc_auc_score= 0.8790322580645161


classification report
               precision    recall  f1-score   support

           0       0.95      1.00      0.97       584
           1       1.00      0.76      0.86       124

    accuracy                           0.96       708
   macro avg       0.98      0.88      0.92       708
weighted avg       0.96      0.96      0.96       708



[[584   0]
 [ 30  94]]


AxesSubplot(0.125,0.808774;0.62x0.0712264)
```

^"*"**'"*"'"* Klye1ghbors€Lassys-1en "*"*"*"*"*"*"

rNel ghborscl ass1fier ()

Accuracy_sc ore= e. 9646192655Z 67232

cross_Val_Score= 0.9597322450604688

roc_auc_score= 8. 9436588599284595

classification report

	precision	recall	{1-score	support
0	B. 98	0.98	0.9B	584
1	0. 89	4.91	0.9B	124
accuracy			0.96	708
wcxavg	0.94	0.94	0.94	708
weighted avg	0.97	‹5.96	0.96	7d I

[IN 113]]

Axessubplot(0.125,0.B08774;0.62x0.07I226Q)

************** SVC ***************

sVC()

ccc uracy_sc ore= e. 9717 514124293786

cross_Val_Score= 0.974917B818494004

roc_auc_score=g.9I935483B7B96775

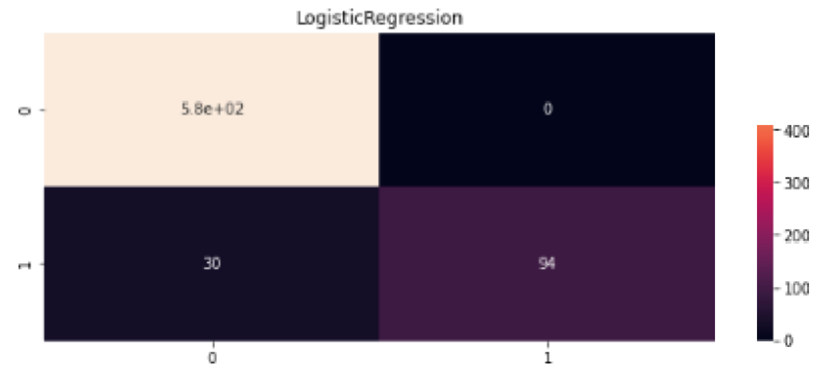
classification report

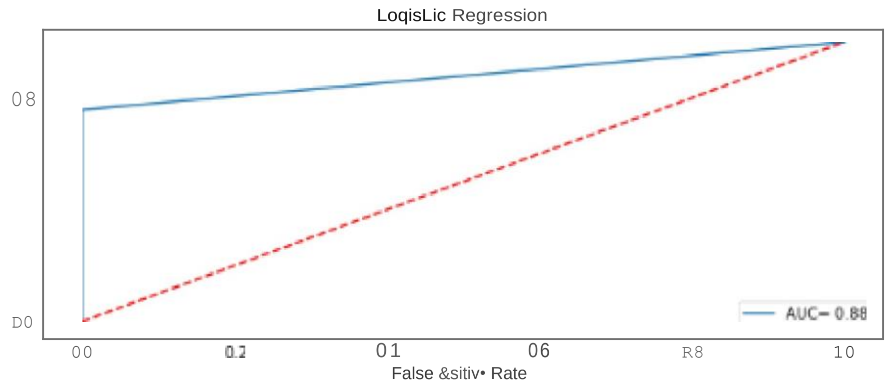
	precision	recall	/r-score	support
0	0.97	1.00	0.PB	5B4-
1	1.00	0.84	0.91	124
accuracy			0.97	708
macro avg	e.98	0.92	0.95	70 I
weighted avg	0.97	0.97	0.97	708

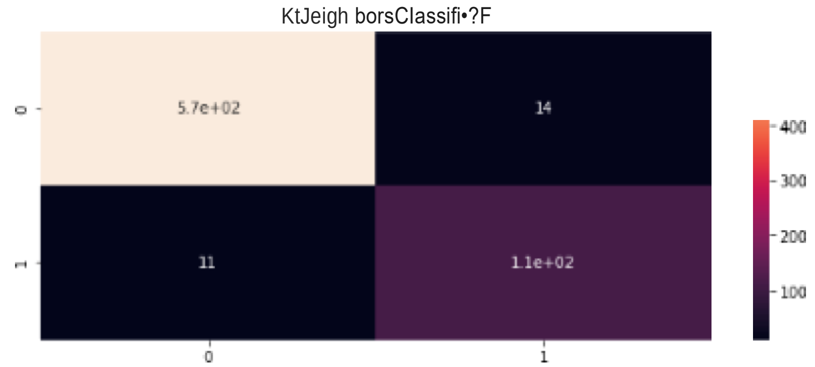
[[5B4 0]

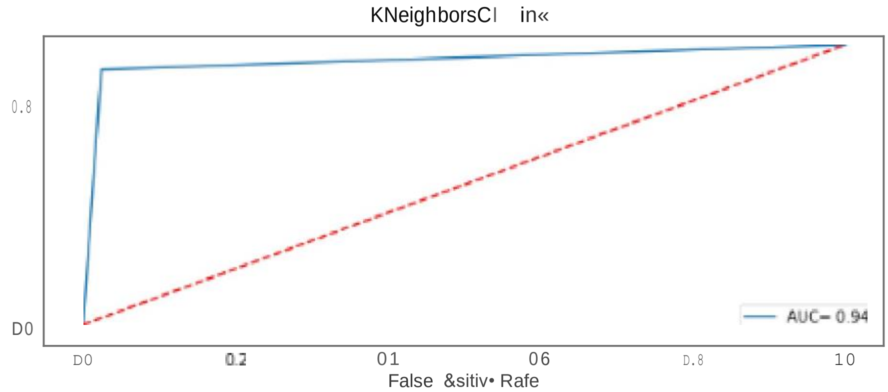
AxesSubplot(O.125,0.B08774;0.62x0.O7I2Z64)

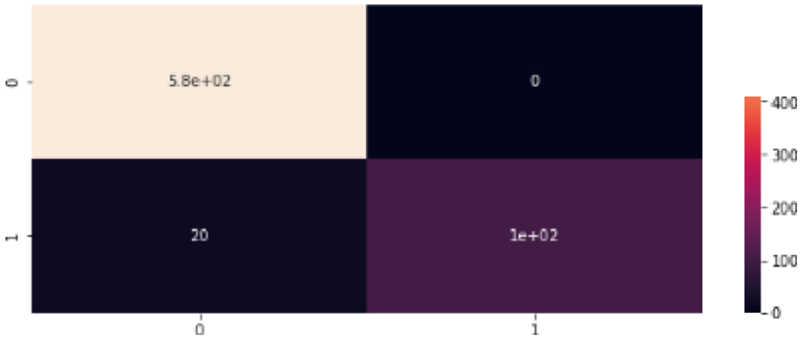
```
***«***«**** * DecisionTreeclassifger ***«***«***«*

Dec is1onTreeC1as si f 1er ()

Accuracy_score= e.sss039   OzzsssB

cro5s_Val_5core= 0.947724331B568656

roc_auc_score= e. sMws3592576227

classification report
              precision    xcall  fl-score   support

           e       e.s7      0.98     0.98       5-84
           i       e.s2      0.84     0.B8       124

    accuracy                          0.96       708
   racno avg       e.so      e.91     0.93       708
we   tea awg       e.ss      e.s6     8.96       zm




Axessubplot(e.izs,e.80877B;e.see.e7izze4)

« »«+ »e »^e NultinOmialNB «+ ^e ^e ^e«

MultincaialNB()

Accuracy_score=e.  s7BB5X 71752

cro5s_Val_5core= 0.B6047B7737022B4Q

roc_auc_score= e.sees+a3B70967742

classification neport
              precision    xcall fl-score   support

           e       eds       iB6      0.92       584
           1       1.BO      0. 14    0. 24      124

    accuracy                          0.B5       708
   macro avg       e.s2      0.57     0.58       708
weighted avg       es7       0S5      0.B0       708



 [1B7  17]]

Axessubplot(e.izs,e.80877B;e.see.e7izze4)
```

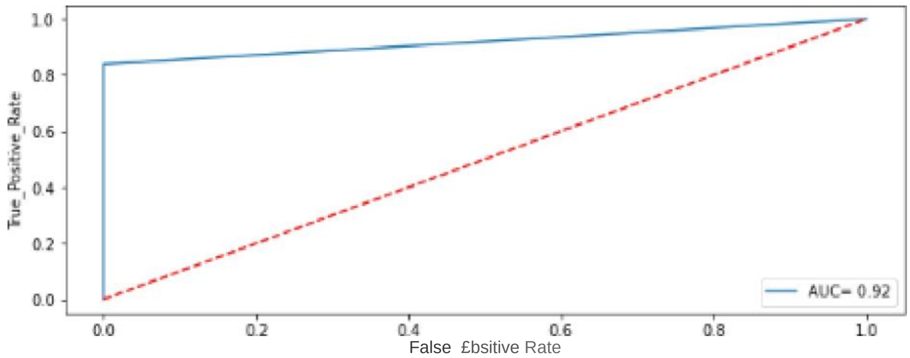

LogisticRegression
5.8e+02
0
30
94
400
300
200
100
0
0
1
0
1


LoqisLic Regression
08
D0
AUC= 0.88
00
0.2
01
06
R8
10
False &sitiv• Rate

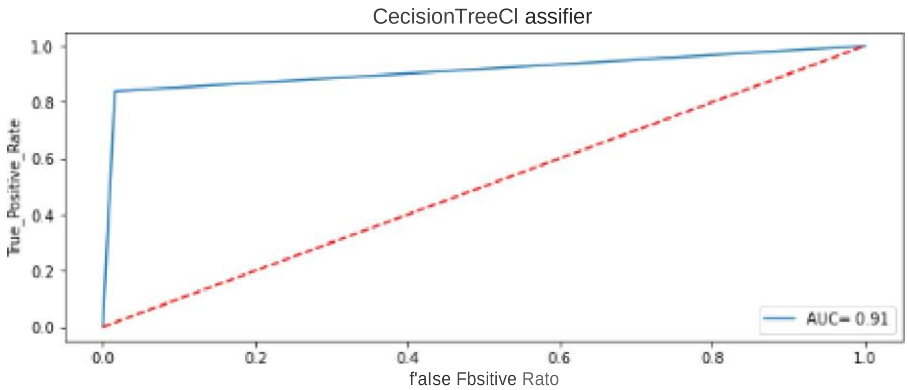

KtJeigh borsClassifi•?F
5.7e+02
14
11
1.1e+02
400
300
200
100
0
1
0
1

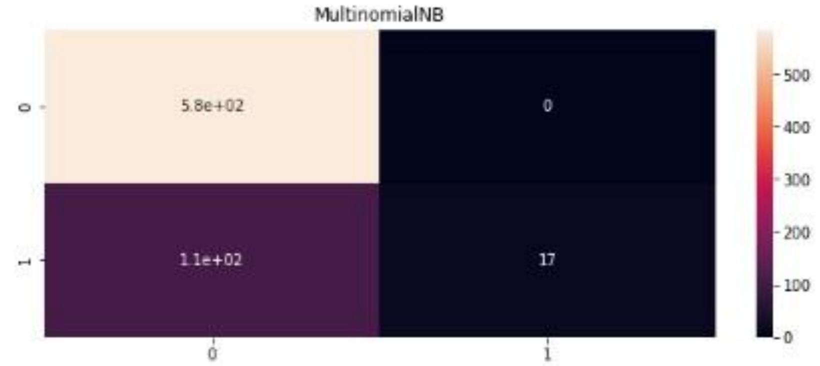

KNeighborsCl in«
0.8
D0
AUC= 0.94
D0
0.2
01
06
D.8
10
False &sitiv• Rafe

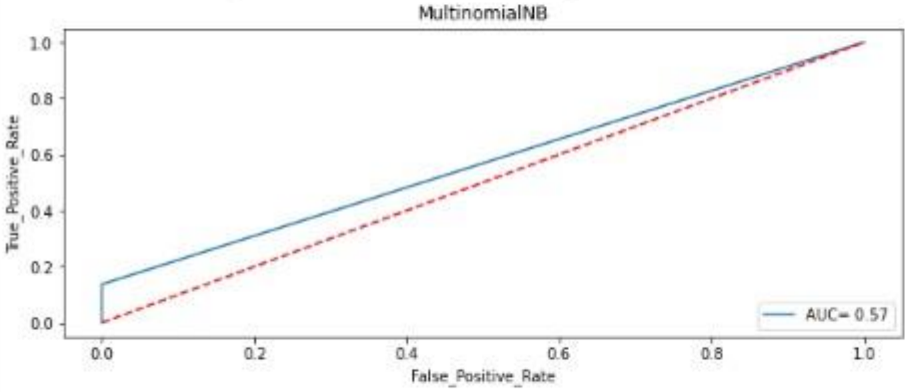

5.8e+02
0
20
1e+02
400
300
200
100
0


True_Positive_Rate
AUC= 0.92
False £bsitive Rate


DecisionTreeClassifier
5.8e+02
9
20
1e+02
400
300
200
100


CecisionTreeCl assifier
True_Positive_Rate
AUC= 0.91
f'alse Fbsitive Rato






```
In [50]: result=pd.DataFrame({'Model':Model,'Accuracy_score':score,'Cross_val_score':cvs,'Roc_auc_curve':rocscore})
         result
```

Out[50]:

	Model	Accuracy_score	Cross_val_score	Roc_auc_curve
0	LogisticRegression	95.762712	95.301722	87.903226
1	KNeighborsClassifier	96.468927	95.973225	94.365886
2	SVC	97.175141	97.491788	91.935484
3	DecisionTreeClassifier	95.903955	94.772433	91.164936
4	MultinomialNB	84.887006	86.047877	56.854839

SO we choose SVC as our best model because it has the highest scores i.e., 97% as we can see from the table above.