EMAIL SPAM CLASSIFIER PROJECT – PREDICTING WHETHER AM EMAIL IS SPAM OR NOT

Submitted by:

NATASHA PODDAR

ACKNOWLEDGMENT

The data used in the project was provided in spam.csv file

INTRODUCTION

- Business Problem Framing

  Spam Detector is used to detect unwanted, malicious and virus infected texts and helps to separate them from the nonspam texts. It uses a binary type of classification containing the labels such as '**ham'** (nonspam) and **spam**. Application of this can be seen in Google Mail (GMAIL) where it segregates the spam emails in order to prevent them from getting into the user's inbox

Conceptual Background of the Domain Problem

A basic understanding on mail background is needed

- Review of Literature

  Each and everyday individuals are flooded with infinite mails and most of them are spam, if we have a system designed which can differentiate between a spam and a ham mail, it would be extremely user friendly, the user wouldn't have to waste a lot of time segregating his mails and the job can be completed sooner

Motivation for the Problem Undertaken

This issue is very realistic and common in today's world and one should know to deal with such situations in the future

Analytical Problem Framing

- Mathematical/ Analytical Modeling of the Problem

  Firstly missing values were checked

  Correlation with all independent variables and wrt target were checked

  Feature extraction was done through count vectorizer & Tfidvectorizer

  Models were applied to train and test the model

  .

- Data Sources and their formats

  The complete data was provided In spam.csv file

- Data Preprocessing Done:
  1. Duplicate values check
  2. Unique & Count of all columns were checked
  3. Missing values were checked
  4. Catagorical data was Encoded
  5. Correlation check
  6. Graphical Univariate, Bivariate & Multivariate Analysis
  7. Feature extraction was done - count vectorizer & Tfidvectorizer

- Hardware and Software Requirements and Tools Used
  1. Pandas – For Data Reading and understanding
  2. Label Encoder –(SK LEARN) – For Encoding the categorical data into numerical ones
  3. Duplicate- To check for duplicate Values
  4. CORR-To check Correlation
  5. VIF -To check for multicollinearity
  6. Numpy- For mathematical operations
  7. LOGITSIC REGRESSION (SKLEARN) – Training & Testing the model

8. SVC (SKLEARN) – Training & Testing the model
9. NAUSSIAN NB (SKLEARN) – Training & Testing the model
10. RANDOM FOREST CLASSIFIER (SKLEARN) – Training & Testing the model
11. CROSS VAL SCORE – Regularizing the model
12. GRID SEARCH CV- Hyper Tuning the Model for higher accuracy
13. SEABORN- VISUALIZATION LIBRARY – HISTPLOTS, DISTPLOTS, SCATTERPLOTS, COUNTPLOTS, BOXPLOTS and other graphs
14. MATPLOTLIB.PY PLOT -Visualization tool

Model/s Development and Evaluation

- Identification of possible problem-solving approaches (methods)
    1. Firstly missing values were checked .
    2. Correlation with all independent variables and wrt target were checked
    3. Feature extraction was performed
    4. Models were applied to train and test the model

- Testing of Identified Approaches (Algorithms)
    1. LINEAR REGRESSION
    2. SVC
    3. NAUSSIAN NB
    4. RANDOM FOREST REGRESSOR

- Key Metrics for success in solving problem under consideration
    1. ACCURACY SCORE
    2. CONFUSION MATRIX
    3. CLASSIFICATION REPORT
    4. AUC-ROC CURVE
- Visualizations

    Seaborn Library was used along with matplotlib Library for visualizations

    Histplots, bar plots, count plots, swarmplots, boxplots etc were made and analysed

- Interpretation of the Results

    RANDOM FOREST REGRESSOR had the second highest model accuracy and the difference between CV MEAN SCORE & MODEL

ACCURACY SCORE was the least hence we had hyper tuned the said model and saved the same

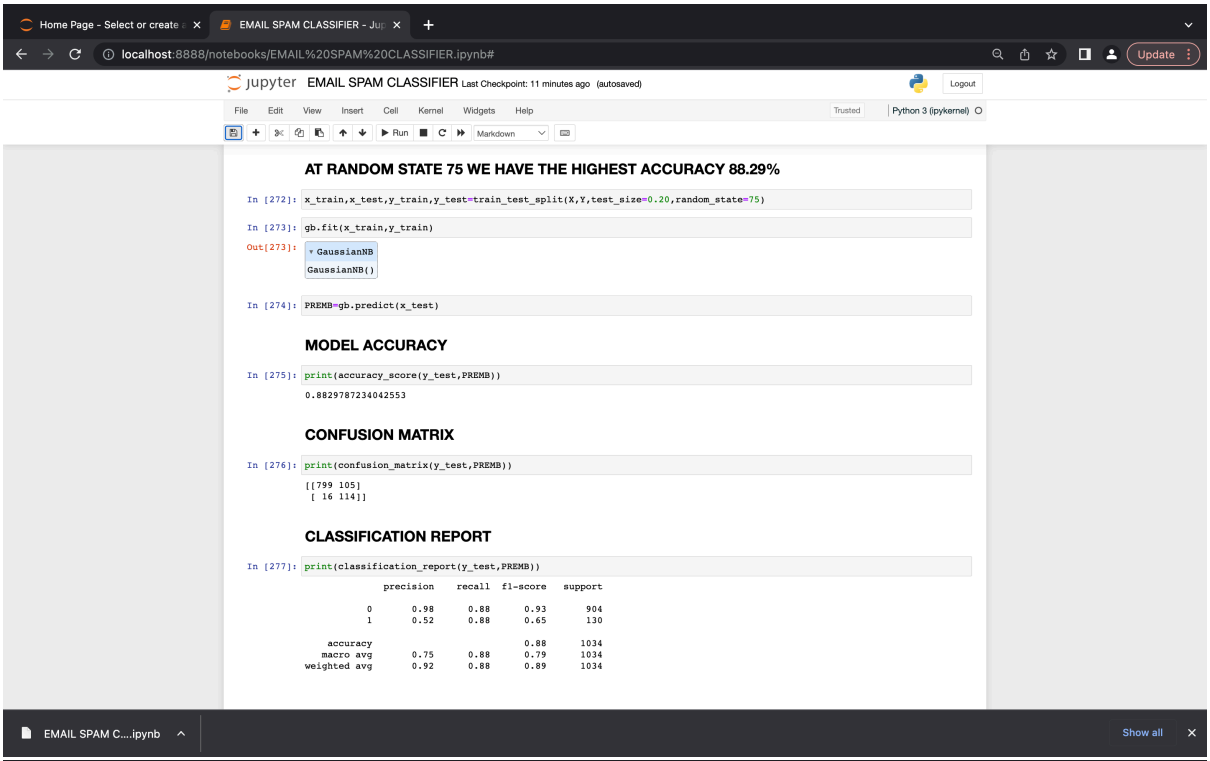
CONCLUSION

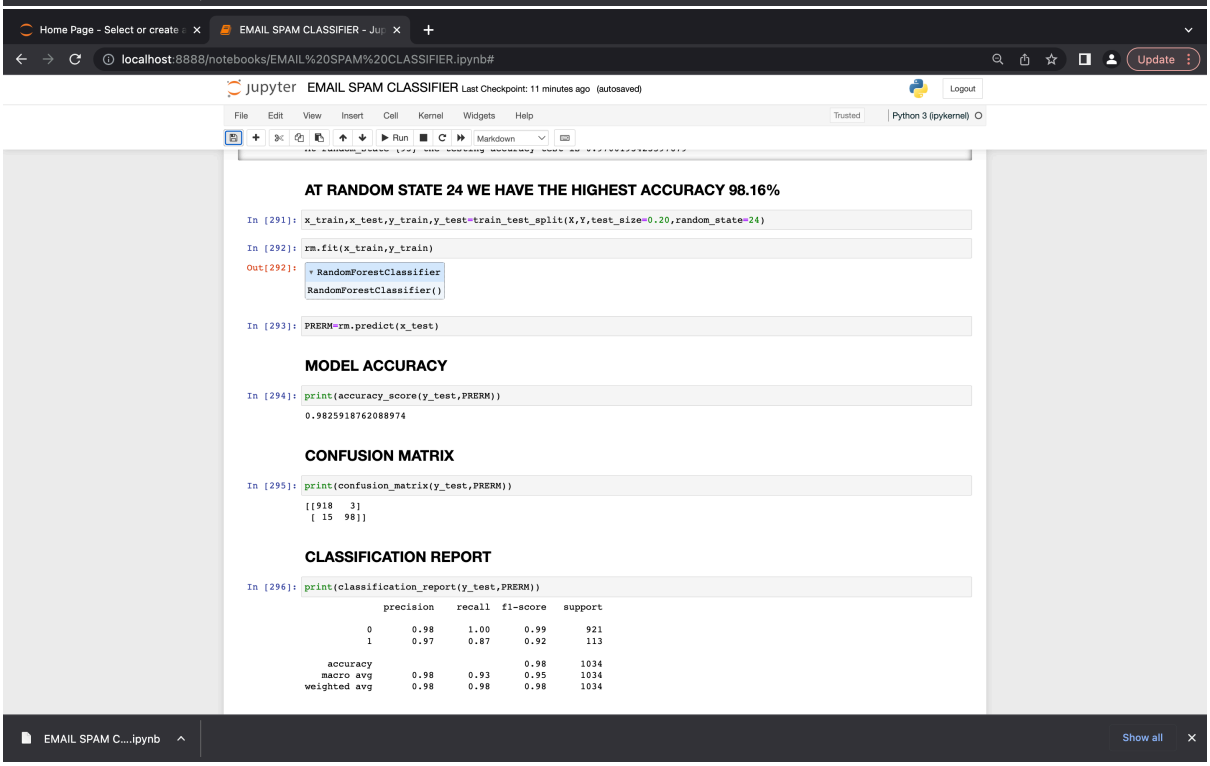
- Key Findings and Conclusions of the Study

  Random Forest Regressor without hyper tuning had a higher accuracy and the same model was selected and saved

- Learning Outcomes of the Study in respect of Data Science

  Strong insights were derived from the various visualization tools which helped in understanding the various relationships between the target and other variables


Home Page - Select or create
EMAIL SPAM CLASSIFIER - Jup
localhost:8888/notebooks/EMAIL%20SPAM%20CLASSIFIER.ipynb#
Update
jupyter EMAIL SPAM CLASSIFIER Last Checkpoint: 11 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Markdown
AT RANDOM STATE 75 WE HAVE THE HIGHEST ACCURACY 88.29%
In [272]: x_train,x_test,y_train,y_test=train_test_split(X,Y,test_size=0.20,random_state=75)
In [273]: gb.fit(x_train,y_train)
Out[273]: GaussianNB GaussianNB()
In [274]: PREMB=gb.predict(x_test)
MODEL ACCURACY
In [275]: print(accuracy_score(y_test,PREMB))
0.8829787234042553
CONFUSION MATRIX
In [276]: print(confusion_matrix(y_test,PREMB))
[[799 105]
[16 114]]
CLASSIFICATION REPORT
In [277]: print(classification_report(y_test,PREMB))
precision recall f1-score support
0 0.98 0.88 0.93 904
1 0.52 0.88 0.65 130
accuracy 0.88 1034
macro avg 0.75 0.88 0.79 1034
weighted avg 0.92 0.88 0.89 1034
EMAIL SPAM C....ipynb
Show all


Home Page - Select or create
EMAIL SPAM CLASSIFIER - Jup
localhost:8888/notebooks/EMAIL%20SPAM%20CLASSIFIER.ipynb#
Update
jupyter EMAIL SPAM CLASSIFIER Last Checkpoint: 11 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Markdown
AT RANDOM STATE 24 WE HAVE THE HIGHEST ACCURACY 98.16%
In [291]: x_train,x_test,y_train,y_test=train_test_split(X,Y,test_size=0.20,random_state=24)
In [292]: rm.fit(x_train,y_train)
Out[292]: RandomForestClassifier RandomForestClassifier()
In [293]: PRERM=rm.predict(x_test)
MODEL ACCURACY
In [294]: print(accuracy_score(y_test,PRERM))
0.9825918762088974
CONFUSION MATRIX
In [295]: print(confusion_matrix(y_test,PRERM))
[[918 3]
[15 98]]
CLASSIFICATION REPORT
In [296]: print(classification_report(y_test,PRERM))
precision recall f1-score support
0 0.98 1.00 0.99 921
1 0.97 0.87 0.92 113
accuracy 0.98 1034
macro avg 0.98 0.93 0.95 1034
weighted avg 0.98 0.98 0.98 1034
EMAIL SPAM C....ipynb
Show all

Home Page - Select or create | EMAIL SPAM CLASSIFIER - Jup

localhost:8888/notebooks/EMAIL%20SPAM%20CLASSIFIER.ipynb#

jupyter EMAIL SPAM CLASSIFIER Last Checkpoint: 11 minutes ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3 (ipykernel)

Run | Markdown

In [242]:
```
svc=SVC(probability=True)
```

In [244]:
```
x_train,x_test,y_train,y_test=train_test_split(X,Y,test_size=0.20,random_state=14)
```

In [245]:
```
svc.fit(x_train,y_train)
```

Out[245]:
```
SVC
SVC(probability=True)
```

In [246]:
```
PRESVC=svc.predict(x_test)
```

MODEL ACCURACY

In [247]:
```
print(accuracy_score(y_test,PRESVC))
```

```
0.9845261121856866
```

CONFUSION MATRIX

In [248]:
```
print(confusion_matrix(y_test,PRESVC))
```

```
[[913   0]
 [ 16 105]]
```

CLASSIFICATION REPORT

In [249]:
```
print(classification_report(y_test,PRESVC))
```

```
              precision    recall  f1-score   support

           0       0.98      1.00      0.99       913
           1       1.00      0.87      0.93       121

    accuracy                           0.98      1034
   macro avg       0.99      0.93      0.96      1034
weighted avg       0.98      0.98      0.98      1034
```

AUC ROC CURVE

EMAIL SPAM C....ipynb | Show all

Home Page - Select or create | EMAIL SPAM CLASSIFIER - Jup

localhost:8888/notebooks/EMAIL%20SPAM%20CLASSIFIER.ipynb#

jupyter EMAIL SPAM CLASSIFIER Last Checkpoint: 11 minutes ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3 (ipykernel)

Run | Markdown

In [220]:
```
lr=LogisticRegression()
```

In [221]:
```
lr.fit(x_train,y_train)
```

Out[221]:
```
LogisticRegression
LogisticRegression()
```

In [222]:
```
PRELR=lr.predict(x_test)
```

MODEL ACCURACY

In [223]:
```
print(accuracy_score(y_test,PRELR))
```

```
0.9671179883945842
```

CONFUSION MATRIX

In [224]:
```
print(confusion_matrix(y_test,PRELR))
```

```
[[911   0]
 [ 34  89]]
```

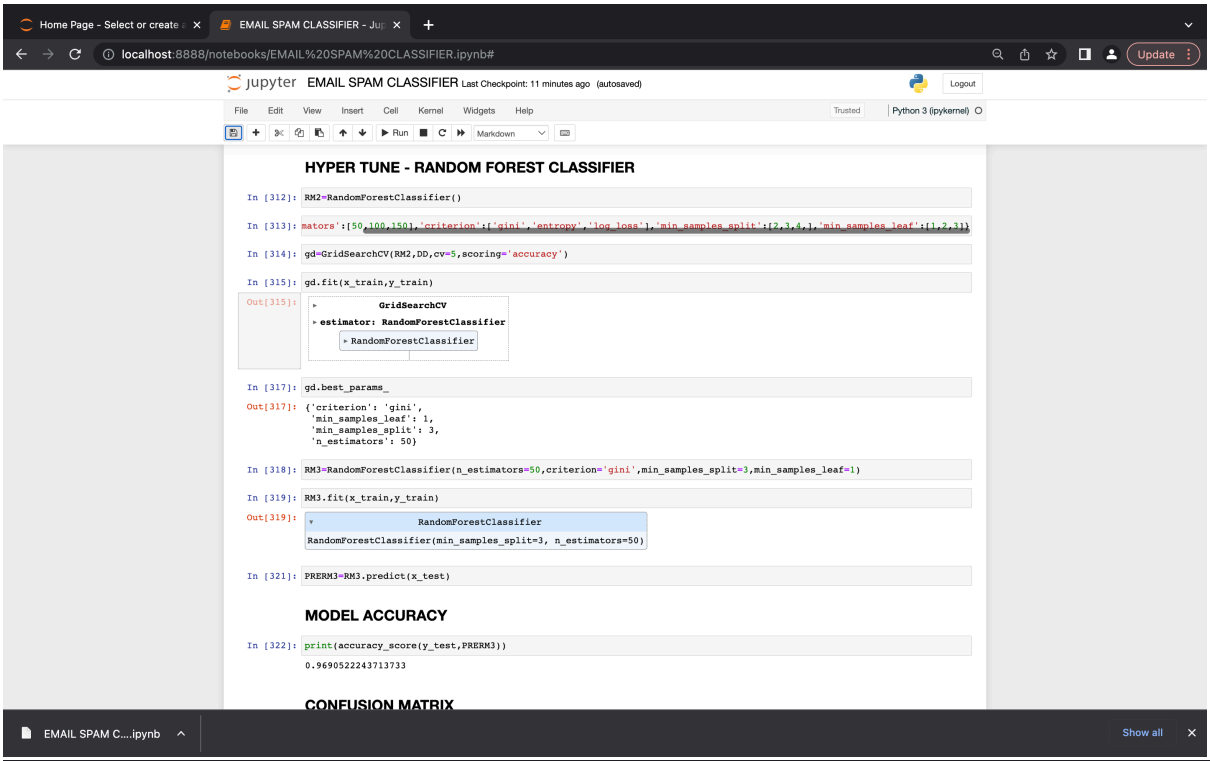
CLASSIFICATION REPORT

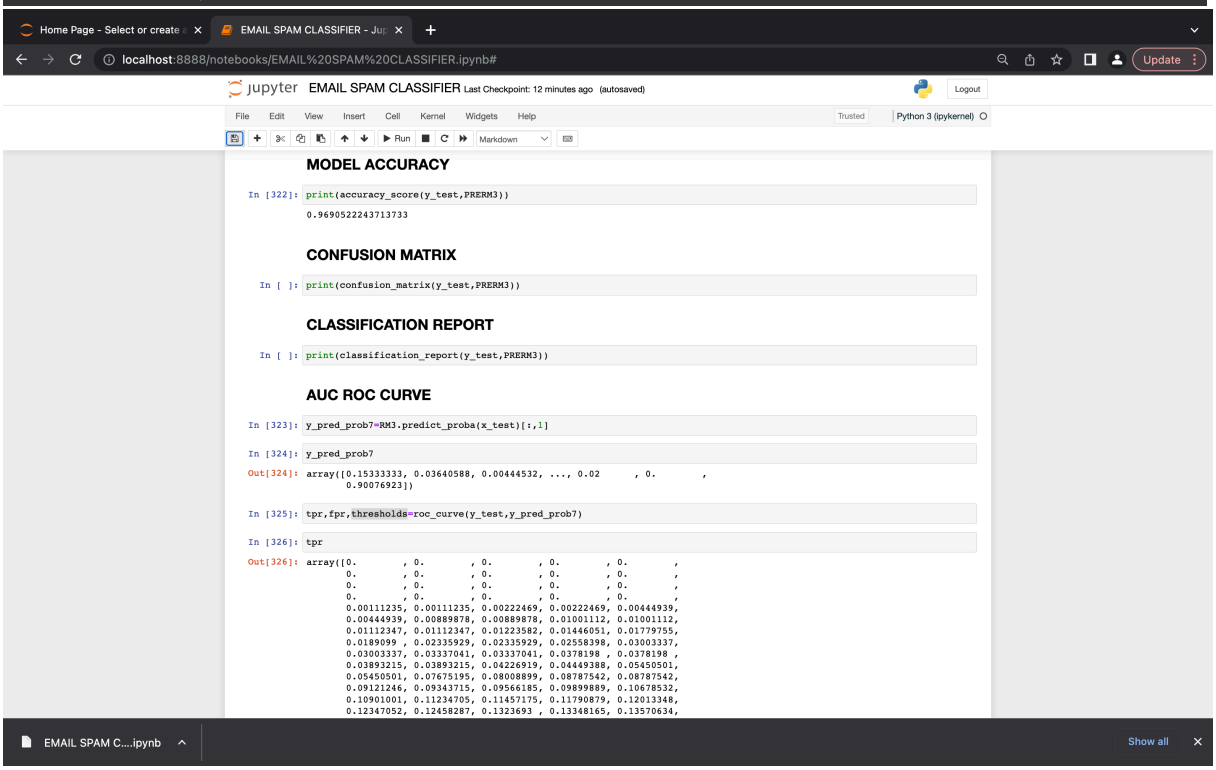
In [225]:
```
print(classification_report(y_test,PRELR))
```

```
              precision    recall  f1-score   support

           0       0.96      1.00      0.98       911
           1       1.00      0.72      0.84       123

    accuracy                           0.97      1034
   macro avg       0.98      0.86      0.91      1034
weighted avg       0.97      0.97      0.96      1034
```

AUC ROC CURVE

EMAIL SPAM C....ipynb | Show all


Home Page - Select or create
EMAIL SPAM CLASSIFIER - Jup
localhost:8888/notebooks/EMAIL%20SPAM%20CLASSIFIER.ipynb#
Update
jupyter EMAIL SPAM CLASSIFIER Last Checkpoint: 11 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Markdown
HYPER TUNE - RANDOM FOREST CLASSIFIER
In [312]: RM2=RandomForestClassifier()
In [313]: mators':[50,100,150],'criterion':['gini','entropy','log_loss'],'min_samples_split':[2,3,4,],'min_samples_leaf':[1,2,3]}
In [314]: gd=GridSearchCV(RM2,DD,cv=5,scoring='accuracy')
In [315]: gd.fit(x_train,y_train)
Out[315]: GridSearchCV
estimator: RandomForestClassifier
RandomForestClassifier
In [317]: gd.best_params_
Out[317]: {'criterion': 'gini',
'min_samples_leaf': 1,
'min_samples_split': 3,
'n_estimators': 50}
In [318]: RM3=RandomForestClassifier(n_estimators=50,criterion='gini',min_samples_split=3,min_samples_leaf=1)
In [319]: RM3.fit(x_train,y_train)
Out[319]: RandomForestClassifier
RandomForestClassifier(min_samples_split=3, n_estimators=50)
In [321]: PRERM3=RM3.predict(x_test)
MODEL ACCURACY
In [322]: print(accuracy_score(y_test,PRERM3))
0.9690522243713733
CONFUSION MATRIX
EMAIL SPAM C....ipynb
Show all


Home Page - Select or create
EMAIL SPAM CLASSIFIER - Jup
localhost:8888/notebooks/EMAIL%20SPAM%20CLASSIFIER.ipynb#
Update
jupyter EMAIL SPAM CLASSIFIER Last Checkpoint: 12 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Markdown
MODEL ACCURACY
In [322]: print(accuracy_score(y_test,PRERM3))
0.9690522243713733
CONFUSION MATRIX
In [ ]: print(confusion_matrix(y_test,PRERM3))
CLASSIFICATION REPORT
In [ ]: print(classification_report(y_test,PRERM3))
AUC ROC CURVE
In [323]: y_pred_prob7=RM3.predict_proba(x_test)[:,1]
In [324]: y_pred_prob7
Out[324]: array([0.15333333, 0.03640588, 0.00444532, ..., 0.02 , 0. ,
0.90076923])
In [325]: tpr,fpr,thresholds=roc_curve(y_test,y_pred_prob7)
In [326]: tpr
Out[326]: array([0. , 0. , 0. , 0. , 0. ,
0. , 0. , 0. , 0. , 0. ,
0. , 0. , 0. , 0. , 0. ,
0. , 0. , 0. , 0. , 0. ,
0.00111235, 0.00111235, 0.00222469, 0.00222469, 0.00444939,
0.00444939, 0.00889878, 0.00889878, 0.01001112, 0.01001112,
0.01112347, 0.01112347, 0.01223582, 0.01446051, 0.01779755,
0.0189099 , 0.02335929, 0.02335929, 0.02558398, 0.03003337,
0.03003337, 0.03337041, 0.03337041, 0.0378198 , 0.0378198 ,
0.03893215, 0.03893215, 0.04226919, 0.04449388, 0.05450501,
0.05450501, 0.07675195, 0.08008899, 0.08787542, 0.08787542,
0.09121246, 0.09343715, 0.09566185, 0.09899889, 0.10678532,
0.10901001, 0.11234705, 0.11457175, 0.11790879, 0.12013348,
0.12347052, 0.12458287, 0.1323693 , 0.13348165, 0.13570634,
EMAIL SPAM C....ipynb
Show all

Home Page - Select or create | EMAIL SPAM CLASSIFIER - Jup

localhost:8888/notebooks/EMAIL%20SPAM%20CLASSIFIER.ipynb#

Update

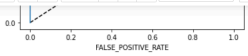
jupyter **EMAIL SPAM CLASSIFIER** Last Checkpoint: 11 minutes ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3 (ipykernel)

Run | Markdown



In [330]:
```
auc_score=roc_auc_score(y_test,(RM3.predict(x_test)))
auc_score
```

Out[330]: 0.8909240720141721

SINCE THE ACCURACY SCORE HASNT IMPROVED, WE WILL SELECT rm MODEL

SAVING THE BEST MODEL

In [331]:
```
import pickle
```

In [332]:
```
filename='churn.pkl'
pickle.dump(rm,open(filename,'wb'))
```

In [333]:
```
loaded_model=pickle.load(open('churn.pkl','rb'))
```

In [334]:
```
conclusion=pd.DataFrame([loaded_model.predict(x_test)[:],y_test[:]],index=['PREDICTED','ORIGINAL'])
```

In [335]:
```
conclusion
```

Out[335]:

	0	1	2	3	4	5	6	7	8	9	...	1024	1025	1026	1027	1028	1029	1030	1031	1032	1033
PREDICTED	0	0	0	1	0	0	0	0	0	0	...	0	0	1	0	0	0	0	0	0	1
ORIGINAL	0	0	0	1	0	0	0	0	0	0	...	0	0	1	0	0	0	0	0	0	1

2 rows × 1034 columns

In [ ]:

EMAIL SPAM C....ipynb

Show all