FLIP ROBO

EMAIL SPAM CLASSIFIER

Submitted by:

Rijul Kumar

ACKNOWLEDGMENT

Reference that i have used are:

- Data Trained Education online video
- Materials provided by Flip Robo
- Youtube
- Geeks for Geeks
- Stackoverflow

INTRODUCTION

- <u>Business Problem Framing</u>

  1. Spam Detector is used to detect unwanted, malicious and virus infected texts and helps to separate them from the non-spam texts.

  2. It uses a binary type of classification containing the labels such as ‘ham’ (non-spam) and spam.

  3. 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.

- <u>Conceptual Background of the Domain Problem</u>

  The spam mails can be detected with the help of some factors. Such as:

  - Some labeled text data
  - Certain recurring words
  - And so on...

- <u>Motivation for the Problem Undertaken</u>

- Spam Detector model is used to detect unwanted, malicious and virus infected texts and helps to separate them from the non-spam texts.
- The SMS Spam Collection is a set of SMS tagged messages that have been collected for SMS Spam research. It contains one set of SMS messages in English of 5,574 messages, tagged according being ham (legitimate) or spam.
- This model will then be used to predict whether a mail is spam or not.

Analytical Problem Framing

- Mathematical/ Analytical Modeling of the Problem
    - First of all i imported data from csv file to dataframe using pandas.
    - I used .dtypes to know data type of each column of dataframe.
    - After that i used .describe() to know the statistical information (such as max, min value,etc) of continuous data columns in dataframe.
    - Then i used .shape to know shape of dataframe.

- Data Sources and their formats
    - Training data has been extracted from ‘spam.csv’ which has been obtained from flip robo
    - ‘spam detetction project.docx’ from flip robo

- Data Preprocessing Done

- First of all for data preprocessing i checked whether there is a NULL value or not in dataframe using heatmap as well as .isnull()
- Next i encoded ‘v1’ column which tells us whether a mail is spam/ ham.
- After that i converted mail text into lower case alphabets and removed special characters from it.
- Next i tokenized the reviews and removed the stop words from them.
- After that i applied Lemmatization method.
- Afterwards i interpreted them into Bag of words model.
- Next i used Density plots from seaborn library to plot all continuous columns for visualisation.

- ## Data Inputs- Logic- Output Relationships
  - Data Input :

    These are basically the mail text which are to be tested for spam / ham mail.

  - Data Output :

    Our Target variable is ‘v1' which tells us if a mail is spam or not.

- <u>Hardware and Software Requirements and Tools Used</u>
  - <u>Hardware used</u>:
    1. Laptop with intel core i5 7th gen
    2. Internet connection for web scraping

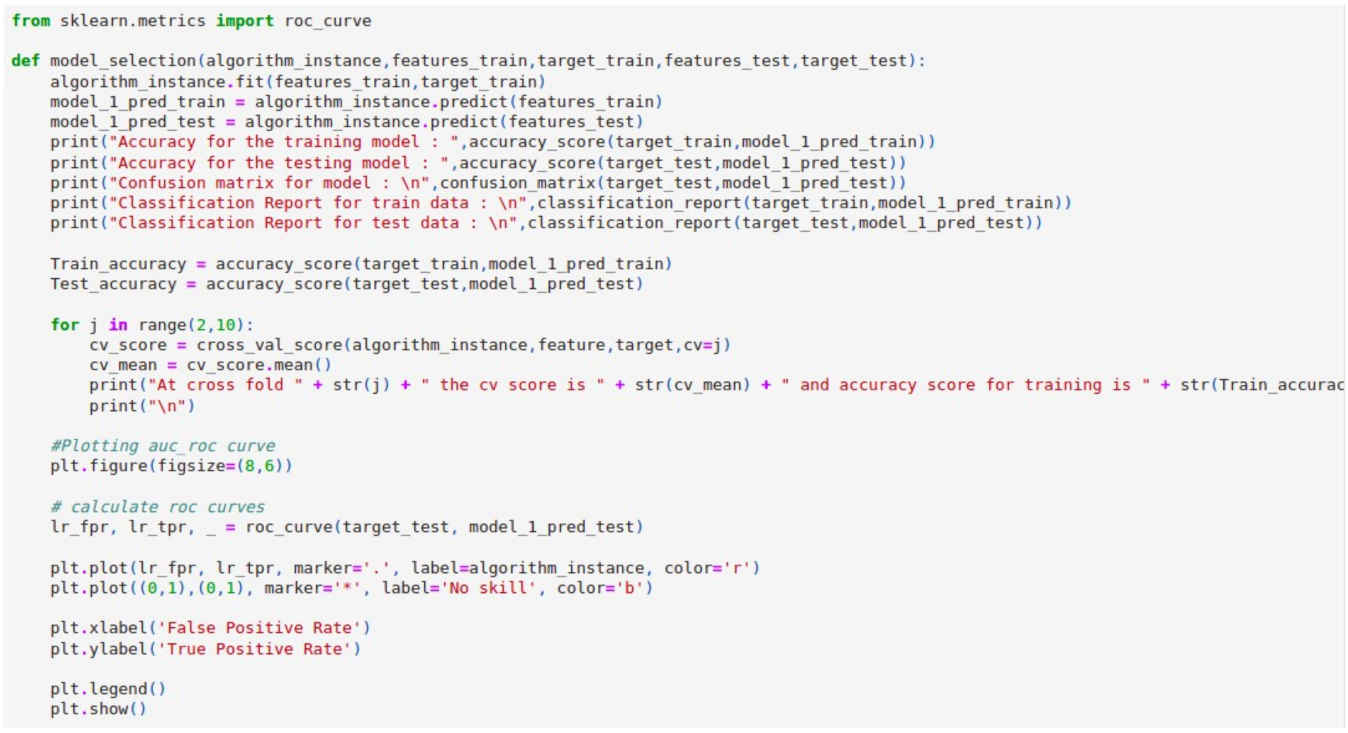
  - <u>Software used</u>:
    1. Jupyter notebook
    2. Required python libraries such as numpy, pandas, seaborn, matplotlib, etc
    3. Required libraries for model such as sklearn, etc

Model/s Development and Evaluation

- <u>Identification of possible problem-solving approaches (methods)</u>
  - After preprocessing data (removing NULL and encoding) i separated columns into features and target.

  - As this is a Binary classification problem of NLP field so i tried Naive Bayes Classifier

- <u>Run and Evaluate selected models</u>
  I defined a function model and then tried Naive Bayes Classification model using it

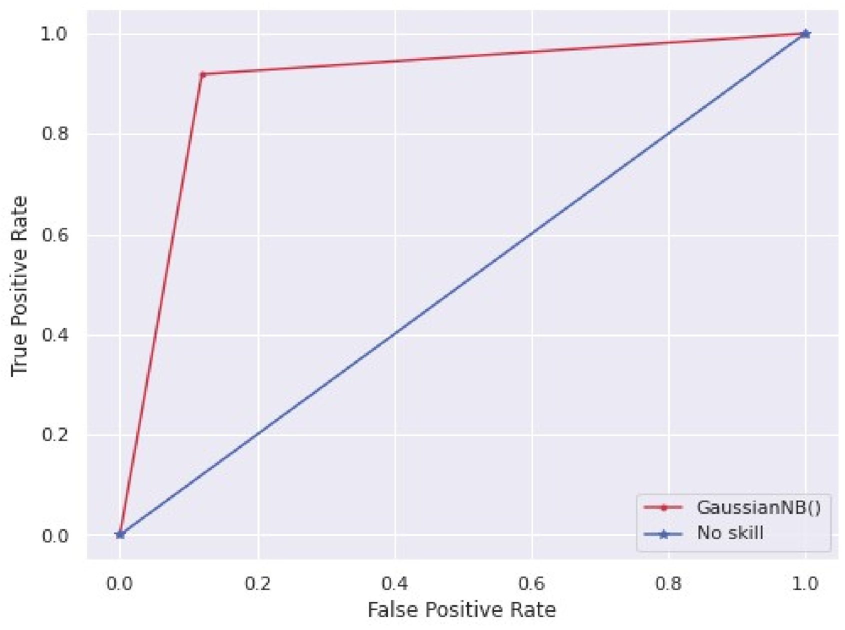


```python
from sklearn.metrics import roc_curve

def model_selection(algorithm_instance,features_train,target_train,features_test,target_test):
    algorithm_instance.fit(features_train,target_train)
    model_1_pred_train = algorithm_instance.predict(features_train)
    model_1_pred_test = algorithm_instance.predict(features_test)
    print("Accuracy for the training model : ",accuracy_score(target_train,model_1_pred_train))
    print("Accuracy for the testing model : ",accuracy_score(target_test,model_1_pred_test))
    print("Confusion matrix for model : \n",confusion_matrix(target_test,model_1_pred_test))
    print("Classification Report for train data : \n",classification_report(target_train,model_1_pred_train))
    print("Classification Report for test data : \n",classification_report(target_test,model_1_pred_test))

    Train_accuracy = accuracy_score(target_train,model_1_pred_train)
    Test_accuracy = accuracy_score(target_test,model_1_pred_test)

    for j in range(2,10):
        cv_score = cross_val_score(algorithm_instance,feature,target,cv=j)
        cv_mean = cv_score.mean()
        print("At cross fold " + str(j) + " the cv score is " + str(cv_mean) + " and accuracy score for training is " + str(Train_accurac
        print("\n")

    #Plotting auc_roc curve
    plt.figure(figsize=(8,6))

    # calculate roc curves
    lr_fpr, lr_tpr, _ = roc_curve(target_test, model_1_pred_test)

    plt.plot(lr_fpr, lr_tpr, marker='.', label=algorithm_instance, color='r')
    plt.plot((0,1),(0,1), marker='*', label='No skill', color='b')

    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')

    plt.legend()
    plt.show()
```

Result that i got for model :

- <u>Naive Bayes Classifier</u> :
  Accuracy for the training model :
  0.9312762973352033
  Accuracy for the testing model :
  0.8856502242152466

  At cross fold 2 the cv score is 0.8902845871259624 and accuracy score for training is 0.9312762973352033 and accuracy score for testing is 0.8856502242152466



```
Confusion matrix for model :
 [[677  92]
 [ 10 113]]
Classification Report for train data :
               precision    recall  f1-score   support

         0.0       1.00      0.92      0.96      3086
         1.0       0.66      1.00      0.80       479

    accuracy                           0.93      3565
   macro avg       0.83      0.96      0.88      3565
weighted avg       0.95      0.93      0.94      3565

Classification Report for test data :
               precision    recall  f1-score   support

         0.0       0.99      0.88      0.93       769
         1.0       0.55      0.92      0.69       123

    accuracy                           0.89       892
   macro avg       0.77      0.90      0.81       892
weighted avg       0.93      0.89      0.90       892
```

- ## Interpretation of the Results

  So the results which i got were:

  - We got Training Accuracy as 93.13% and Testing Accuracy as 88.57%.

  - Since we have imbalanced data (unequal number of ham and spam mails), the precision is low for minority data.

CONCLUSION

- <u>Key Findings and Conclusions of the Study</u>
  - From this study i learnt that many users / bots use certain words a lot while writing spam mails.
  - More number of frequent words we take, higher the accuracy will be in the model till a threshold point.

- <u>Learning Outcomes of the Study in respect of Data Science</u>

  Some problems faced and their solution (using visualisation and algorithm) used were:
  - When i used 100 most frequent words in bag of words the testing accuracy was around 4% but when i took 5000 most frequent words in bag of words the testing accuracy became around 88%.

- <u>Limitations of this work and Scope for Future Work</u>

  Some limitations are :

- There are many other factors which are not in the data which may play major role in spam mails such as sender email address, etc.
- With evolving technology, there may be increase in intelligently hidden spam mails which are also required to be taken care of.

Scope for future work :

- This can be made further accurate by taking more and more factors as well as authenticity of sender email address.