Spam Mail Prediction

April 18, 2025

1 Spam Mail Prediction System

[1]:
```python
# Importing dependencies
import pandas as pd
import numpy as np
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
```

Data Collection and Pre-processing

[2]:
```python
# Loading the data from csv file to a pandas dataframe
raw_mail_data = pd.read_csv('dataset/mail_data.csv')
```

[3]:
```python
# Printing the first 5 rows
raw_mail_data.head()
```

[3]:
```
  Category                                           Message
0      ham  Go until jurong point, crazy.. Available only …
1      ham                      Ok lar… Joking wif u oni…
2     spam  Free entry in 2 a wkly comp to win FA Cup fina…
3      ham  U dun say so early hor… U c already then say…
4      ham  Nah I don't think he goes to usf, he lives aro…
```

[4]:
```python
# Checking the number of rows and cols inside the data
raw_mail_data.shape
```

[4]:
```
(5572, 2)
```

[5]:
```python
# checking for any null values
raw_mail_data.isna().sum()
```

[5]:
```
Category    0
Message     0
dtype: int64
```

[6]:

```
# Checking total number of labels
raw_mail_data['Category'].value_counts()
```

[6]:

```
Category
ham     4825
spam     747
Name: count, dtype: int64
```

Here : Ham represents legit data and Spam represents spam data or fake data

[7]:

```
# replacing and removing spaces inside the data
mail_data = raw_mail_data.where((pd.notnull(raw_mail_data)),'')
```

[8]:

```
mail_data.head()
```

[8]:

```
  Category                                            Message
0      ham  Go until jurong point, crazy.. Available only …
1      ham                      Ok lar… Joking wif u oni…
2     spam  Free entry in 2 a wkly comp to win FA Cup fina…
3      ham  U dun say so early hor… U c already then say…
4      ham  Nah I don't think he goes to usf, he lives aro…
```

Balancing the data

[9]:

```
# Seprating the data based on labels
ham_data = mail_data[mail_data.Category == 'ham']
spam_data = mail_data[mail_data.Category == 'spam']
```

[10]:

```
# Checking the datasets
print(ham_data)
print(spam_data)
```

```
     Category                                            Message
0         ham  Go until jurong point, crazy.. Available only …
1         ham                      Ok lar… Joking wif u oni…
3         ham  U dun say so early hor… U c already then say…
4         ham  Nah I don't think he goes to usf, he lives aro…
6         ham  Even my brother is not like to speak with me. …
…        …                                                …
5565      ham                                       Huh y lei…
5568      ham                Will ü b going to esplanade fr home?
5569      ham  Pity, * was in mood for that. So…any other s…
5570      ham  The guy did some bitching but I acted like i'd…
5571      ham                             Rofl. Its true to its name

[4825 rows x 2 columns]
     Category                                            Message
2        spam  Free entry in 2 a wkly comp to win FA Cup fina…
5        spam  FreeMsg Hey there darling it's been 3 week's n…
```

```
8        spam  WINNER!! As a valued network customer you have…
9        spam  Had your mobile 11 months or more? U R entitle…
11       spam  SIX chances to win CASH! From 100 to 20,000 po…
…       …                                                …
5537     spam  Want explicit SEX in 30 secs? Ring 02073162414…
5540     spam  ASKED 3MOBILE IF 0870 CHATLINES INCLU IN FREE …
5547     spam  Had your contract mobile 11 Mnths? Latest Moto…
5566     spam  REMINDER FROM O2: To get 2.50 pounds free call…
5567     spam  This is the 2nd time we have tried 2 contact u…

[747 rows x 2 columns]
```

[11]:
```
# Reducing the Ham labels to equal to spam labels
ham_sample = ham_data.sample(n=747)
ham_sample.shape
```

[11]:
```
(747, 2)
```

Now both contains same number of labels and we can continue forward

[12]:
```
# Joining both spam and ham reduced or balanced data into one
new_mail_data = pd.concat([ham_sample,spam_data],axis = 0)
```

[13]:
```
# printing the first 5 data
new_mail_data.head()
```

[13]:
```
     Category                                            Message
1641      ham          Alright, we're all set here, text the man
861       ham                  In work now. Going have in few min.
4547      ham  Never try alone to take the weight of a tear t…
2477      ham                                 i dnt wnt to tlk wid u
3584      ham          I sent your maga that money yesterday oh.
```

[14]:
```
# printing the last 5 data
new_mail_data.tail()
```

[14]:
```
     Category                                            Message
5537     spam  Want explicit SEX in 30 secs? Ring 02073162414…
5540     spam  ASKED 3MOBILE IF 0870 CHATLINES INCLU IN FREE …
5547     spam  Had your contract mobile 11 Mnths? Latest Moto…
5566     spam  REMINDER FROM O2: To get 2.50 pounds free call…
5567     spam  This is the 2nd time we have tried 2 contact u…
```

[15]:
```
# Checking total no of spam and ham data inside the new data created
new_mail_data['Category'].value_counts()
```

[15]:
```
Category
ham     747
spam    747
```

```
Name: count, dtype: int64
```

Label Encoding the category to numerical values

[16]:

```
# now label encoding all this values where spam mail as 0 and ham mail as 1
# Loc is used to locate few values
new_mail_data.loc[new_mail_data['Category'] == 'spam','Category',] = 0
new_mail_data.loc[new_mail_data['Category'] == 'ham','Category',] = 1
```

Spam - 0 Ham - 1

[17]:

```
# Rechecking the data is encoded or not
new_mail_data['Category'].value_counts()
```

[17]:

```
Category
1    747
0    747
Name: count, dtype: int64
```

From the value counts we can see that spam is replaced by 0 and ham is replaced by 1

[18]:

```
# Splitting the data into features and target
X = new_mail_data.drop(columns='Category',axis = 1)
y = new_mail_data['Category']
```

[19]:

```
print(X)
print(y)
```

```
                                                Message
1641          Alright, we're all set here, text the man
861                 In work now. Going have in few min.
4547  Never try alone to take the weight of a tear t…
2477                             i dnt wnt to tlk wid u
3584          I sent your maga that money yesterday oh.
…                                                   …
5537  Want explicit SEX in 30 secs? Ring 02073162414…
5540  ASKED 3MOBILE IF 0870 CHATLINES INCLU IN FREE …
5547  Had your contract mobile 11 Mnths? Latest Moto…
5566  REMINDER FROM O2: To get 2.50 pounds free call…
5567  This is the 2nd time we have tried 2 contact u…

[1494 rows x 1 columns]
1641    1
861     1
4547    1
2477    1
3584    1
       ..
5537    0
```

```
5540    0
5547    0
5566    0
5567    0
Name: Category, Length: 1494, dtype: object
```

Splitting the data into train and test

[20]:
```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,stratify =␣
  ↪y, random_state=3)
```

[21]:
```
print(X.shape,X_train.shape,X_test.shape)
```

```
(1494, 1) (1195, 1) (299, 1)
```

Feature Extraction

[25]:
```
# Trasnform the text data into numerical/feature vectors that can be used as␣
  ↪input to trian the model.
# min_df decides the minimum score values to take , lowercase = convert the␣
  ↪words to lowercase
feature_extraction =␣
  ↪TfidfVectorizer(min_df=1,stop_words='english',lowercase=True)
```

[26]:
```
# Converting X_train and X_test to feature vectors
X_train_features = feature_extraction.fit_transform(X_train['Message'])
X_test_features = feature_extraction.transform(X_test['Message'])

# Convert Y_train and Y_test values as integer
y_train = y_train.astype('int')
y_test = y_test.astype('int')
```

[30]:
```
# Checking if the featured is vectorized or not
print(X_train_features,X_test_features)
```

```
<Compressed Sparse Row sparse matrix of dtype 'float64'
        with 13158 stored elements and shape (1195, 3828)>
  Coords        Values
  (0, 1814)     0.3830827586878181
  (0, 2353)     0.5870194963816389
  (0, 2063)     0.46384362759041964
  (0, 2182)     0.3830827586878181
  (0, 1750)     0.3830827586878181
  (1, 1378)     0.2721065180770504
  (1, 2025)     0.2721065180770504
  (1, 1152)     0.15364214508490576
  (1, 846)      0.25718417486917206
  (1, 1732)     0.2721065180770504
  (1, 1954)     0.25718417486917206
  (1, 3228)     0.25718417486917206
  (1, 998)      0.20616434645459256
  (1, 1498)     0.2721065180770504
  (1, 2616)     0.21287433158097807
  (1, 2284)     0.25718417486917206
  (1, 386)      0.23838424578826778
  (1, 2418)     0.2721065180770504
  (1, 3293)     0.2721065180770504
  (1, 1056)     0.2721065180770504
  (1, 0)        0.22108668966247091
  (2, 3370)     0.18994422055631405
  (2, 3553)     0.1250078380415494
  (2, 3099)     0.27585312733823314
  (2, 3712)     0.1557576294465119
  :     :
  (1191, 3163)  0.4010485479982434
  (1192, 2128)  0.35181281593053393
  (1192, 1374)  0.3748263440950111
  (1192, 3797)  0.4550868940244342
  (1192, 874)   0.4912440736725648
  (1192, 2300)  0.5360113182463349
  (1193, 1798)  0.5084602995276144
  (1193, 3612)  0.5914781337012311
  (1193, 1758)  0.6257968849056355
  (1194, 2432)  0.13463460213309028
  (1194, 1719)  0.1278353650515705
  (1194, 1376)  0.13921465726598733
  (1194, 2165)  0.27842931453197467
  (1194, 2714)  0.21967501746420334
  (1194, 868)   0.21967501746420334
  (1194, 2318)  0.4393500349284067
  (1194, 1974)  0.21967501746420334
  (1194, 1947)  0.21967501746420334
  (1194, 2434)  0.21967501746420334
  (1194, 2698)  0.21967501746420334
  (1194, 3118)  0.21967501746420334
  (1194, 2697)  0.21967501746420334
  (1194, 1459)  0.21967501746420334
  (1194, 2695)  0.4393500349284067
  (1194, 2332)  0.21967501746420334 <Compressed Sparse Row sparse matrix of
dtype 'float64'
        with 2633 stored elements and shape (299, 3828)>
  Coords        Values
  (0, 1254)     0.5111113459541785
  (0, 2644)     0.5611761756979092
  (0, 3147)     0.6510349390516437
  (1, 3532)     1.0
  (2, 272)      0.3121916456445628
  (2, 627)      0.3034041467394656
```

```
(2, 1112)     0.35356920277873455
(2, 1139)     0.250453445497332
(2, 1254)     0.21196712698615724
(2, 1857)     0.3121916456445628
(2, 2440)     0.24537529987229245
(2, 2728)     0.18639200982931925
(2, 2890)     0.3121916456445628
(2, 2999)     0.2561303293231936
(2, 3573)     0.2561303293231936
(2, 3576)     0.2895385022093288
(2, 3719)     0.2699959738533304
(3, 524)      0.3806690095113308
(3, 627)      0.3241054714847226
(3, 1483)     0.3597930905929803
(3, 1811)     0.28429674542550076
(3, 2135)     0.3806690095113308
(3, 2455)     0.3806690095113308
(3, 2691)     0.3806690095113308
(3, 3706)     0.3241054714847226
:     :
(297, 596)    0.2677778435119213
(297, 1614)   0.12106821224731018
(297, 1625)   0.20610582510533537
(297, 1819)   0.24121715089513393
(297, 1999)   0.134184270383929
(297, 2336)   0.1914299528345027
(297, 2462)   0.28331485268592316
(297, 2463)   0.17255875648755728
(297, 2863)   0.14443380669874945
(297, 3328)   0.179545132488548
(297, 3426)   0.5176762694626718
(297, 3492)   0.12516727470572792
(297, 3496)   0.2567541600691358
(297, 3553)   0.2567785079043299
(297, 3679)   0.16074496300071636
(298, 650)    0.31678643134632195
(298, 1622)   0.3516681426878865
(298, 1708)   0.24299496569862364
(298, 1842)   0.5230004538179247
(298, 2510)   0.23254579041742351
(298, 2810)   0.27787667704018815
(298, 3135)   0.2643766481773806
(298, 3337)   0.3516681426878865
(298, 3743)   0.19766407907585898
(298, 3816)   0.28190472000475747
```

[29]:

```
# Checking the shape of the fearures
print(X_train_features.shape,X_test_features.shape)
```

```
(1195, 3828) (299, 3828)
```

Training the Machine Learning model

[31]:

```
model = LogisticRegression()
```

[32]:

```
# training the regression model with training data
model.fit(X_train_features,y_train)
```

[32]:

```
LogisticRegression()
```

Evaluation of model

[33]:

```
# Prediction on trained model
X_train_prediction = model.predict(X_train_features)

# Storing accuracy of the model
X_train_accuracy = accuracy_score(X_train_prediction,y_train)

# printing the accuracy value
print("Accuracy on training data is : ",X_train_accuracy)
```

```
Accuracy on training data is :  0.9866108786610879
```

[34]:

```
# Prediction on trained model
X_test_prediction = model.predict(X_test_features)

# Storing accuracy of the model
X_test_accuracy = accuracy_score(X_test_prediction,y_test)

# printing the accuracy value
print("Accuracy on testing data is : ",X_test_accuracy)
```

```
Accuracy on testing data is :  0.9531772575250836
```

Making a predictive system

[37]:

```
# Insert the data
input_mail = ["U dun say so early hor... U c already then say..."]

# Converting it into feature vector
input_data_featured = feature_extraction.transform(input_mail)

# Making prediction
prediction = model.predict(input_data_featured)

# Printing the predicted value
```

```
print(prediction)

# Output in the form of text
if(prediction[0]== 1 ):
    print("The given mail or input is a ham mail")
else:
   print("The given mail or input is a spam mail")
```

```
[1]
The given mail or input is a ham mail
```

As we have seen previously that we have converted the ham and spam in a labelled data so for 1 it means its ham and if 0 its spam