ection-a-machine-learning-solution

July 16, 2024

0.1 Spam Detection Using Logistic Regression and Machine Learning Techniques -Vignesh Prabhu

Spam emails pose significant challenges, including security threats and productivity loss. This project aims to develop a robust spam detection system using logistic regression, a machine learning technique well-suited for binary classification. By training our model on a dataset of labeled emails, we strive to accurately classify emails as spam or not spam, enhancing email security and efficiency.

Import Dependencies

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer #converting Text to␣
↪numerical Values
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
```

Data Collection And PreProcessing

[2]:
```
#Loading the data into dataframe
mail_data=pd.read_csv('/content/mail_data.csv')
```

[3]:
```
#To print First 5 data's in dataset
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]:
```
#Last 5 datas in dataset
mail_data.tail()
```

[4]:
```
     Category                                            Message
5567     spam  This is the 2nd time we have tried 2 contact u…
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
```

[5]:
```
#To check rows and columns
mail_data.shape
```

[5]:
```
(5572, 2)
```

[6]:
```
#Replace null values with null string
mail_data.where((pd.notnull(mail_data)),'',inplace=True)
```

[7]:
```
#To check Null values
mail_data.isnull().sum()
```

[7]:
```
Category    0
Message     0
dtype: int64
```

Label Encoding

[8]:
```
#label spam mail as 0 , ham mail as 1
mail_data.loc[mail_data['Category']=='spam','Category',]=0
mail_data.loc[mail_data['Category']=='ham','Category',]=1
```

Seperating Feature and Target

[9]:
```
X=mail_data['Message']
Y=mail_data['Category']
```

[10]:
```
print(X)
```

```
0       Go until jurong point, crazy.. Available only …
1                           Ok lar… Joking wif u oni…
2       Free entry in 2 a wkly comp to win FA Cup fina…
3       U dun say so early hor… U c already then say…
4       Nah I don't think he goes to usf, he lives aro…
                              …
5567    This is the 2nd time we have tried 2 contact u…
5568                 Will ü b going to esplanade fr home?
5569    Pity, * was in mood for that. So…any other s…
5570    The guy did some bitching but I acted like i'd…
5571                           Rofl. Its true to its name
Name: Message, Length: 5572, dtype: object
```

[11]:
```
print(Y)
```

```
0       1
1       1
2       0
3       1
4       1
       ..
5567    0
5568    1
5569    1
5570    1
5571    1
Name: Category, Length: 5572, dtype: object
```

Spliting data into Training Data and Testing data

[12]:
```
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.2,random_state=3)
```

[13]:
```
print(X.shape, X_train.shape, X_test.shape)
```

```
(5572,) (4457,) (1115,)
```

Feature Extraction

[14]:
```
#Convert text data to numerical values
vectorizer=TfidfVectorizer(min_df=1,stop_words='english',lowercase=True)
X_train_feature=vectorizer.fit_transform(X_train)
X_test_feature=vectorizer.transform(X_test)

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

[15]:
```
print(X_train_feature)
```

```
  (0, 5413)     0.6198254967574347
  (0, 4456)     0.4168658090846482
  (0, 2224)     0.413103377943378
  (0, 3811)     0.34780165336891333
  (0, 2329)     0.38783870336935383
  (1, 4080)     0.18880584110891163
  (1, 3185)     0.29694482957694585
  (1, 3325)     0.31610586766078863
  (1, 2957)     0.3398297002864083
  (1, 2746)     0.3398297002864083
  (1, 918)      0.22871581159877646
  (1, 1839)     0.2784903590561455
  (1, 2758)     0.3226407885943799
```

```
  (1, 2956)     0.33036995955537024
  (1, 1991)     0.33036995955537024
  (1, 3046)     0.2503712792613518
  (1, 3811)     0.17419952275504033
  (2, 407)      0.509272536051008
  (2, 3156)     0.4107239318312698
  (2, 2404)     0.45287711070606745
  (2, 6601)     0.6056811524587518
  (3, 2870)     0.5864269879324768
  (3, 7414)     0.8100020912469564
  (4, 50)       0.23633754072626942
  (4, 5497)     0.15743785051118356
  :     :
  (4454, 4602)  0.2669765732445391
  (4454, 3142)  0.32014451677763156
  (4455, 2247)  0.37052851863170466
  (4455, 2469)  0.35441545511837946
  (4455, 5646)  0.33545678464631296
  (4455, 6810)  0.29731757715898277
  (4455, 6091)  0.23103841516927642
  (4455, 7113)  0.30536590342067704
  (4455, 3872)  0.3108911491788658
  (4455, 4715)  0.30714144758811196
  (4455, 6916)  0.19636985317119715
  (4455, 3922)  0.31287563163368587
  (4455, 4456)  0.24920025316220423
  (4456, 141)   0.292943737785358
  (4456, 647)   0.30133182431707617
  (4456, 6311)  0.30133182431707617
  (4456, 5569)  0.4619395404299172
  (4456, 6028)  0.21034888000987115
  (4456, 7154)  0.24083218452280053
  (4456, 7150)  0.3677554681447669
  (4456, 6249)  0.17573831794959716
  (4456, 6307)  0.2752760476857975
  (4456, 334)   0.2220077711654938
  (4456, 5778)  0.16243064490100795
  (4456, 2870)  0.31523196273113385
```

[16]:
```
print(X_test_feature)
```

```
  (0, 7271)     0.1940327008179069
  (0, 6920)     0.20571591693537986
  (0, 5373)     0.2365698724638063
  (0, 5213)     0.1988547357502182
  (0, 4386)     0.18353336340308998
  (0, 1549)     0.2646498848307188
  (0, 1405)     0.3176863938914351
```

```
(0, 1361)     0.25132445289897426
(0, 1082)     0.2451068436245027
(0, 1041)     0.28016206931555726
(0, 405)      0.2381316303003606
(0, 306)      0.23975986557206702
(0, 20)       0.30668032384591537
(0, 14)       0.26797874471323896
(0, 9)        0.2852706805264544
(0, 1)        0.2381316303003606
(1, 7368)     0.29957800964520975
(1, 6732)     0.42473488678029325
(1, 6588)     0.3298937975962767
(1, 6507)     0.26731535902873493
(1, 6214)     0.3621564482127515
(1, 4729)     0.22965776503163893
(1, 4418)     0.3457696891316818
(1, 3491)     0.496093956101028
(2, 7205)     0.22341717215670331
:     :
(1110, 3167)  0.5718357066163949
(1111, 7353)  0.4991205841293424
(1111, 6787)  0.40050175714278885
(1111, 6033)  0.4714849709283488
(1111, 3227)  0.44384935772735523
(1111, 2440)  0.4137350055985486
(1112, 7071)  0.33558524648843113
(1112, 6777)  0.32853717524096393
(1112, 6297)  0.3056896872268727
(1112, 5778)  0.22807428098549426
(1112, 5695)  0.3381604952481646
(1112, 5056)  0.2559183043595413
(1112, 4170)  0.3307835623173863
(1112, 2329)  0.241856898377491
(1112, 1683)  0.4017087436272034
(1112, 1109)  0.35334496762883244
(1113, 4080)  0.3045947361955407
(1113, 4038)  0.37023520529413706
(1113, 3811)  0.28103080586555096
(1113, 3281)  0.33232508601719535
(1113, 3113)  0.33840833425155675
(1113, 2852)  0.5956422931588335
(1113, 2224)  0.3337959267435311
(1114, 4557)  0.5196253874825217
(1114, 4033)  0.8543942045002639
```

Training The Model

[17]:
```
Model=LogisticRegression()
```

[18]:
```
Model.fit(X_train_feature,Y_train)
```

[18]:
```
LogisticRegression()
```

Model Evaluation

[19]:
```
#Prediction on Training Data
prediction_on_training_data=Model.predict(X_train_feature)
accuracy_on_training_data=accuracy_score(Y_train,prediction_on_training_data)
print('Accuracy on training data :',accuracy_on_training_data)
```

```
Accuracy on training data : 0.9670181736594121
```

[20]:
```
#Prediction on Test Data
prediction_on_test_data=Model.predict(X_test_feature)
accuracy_on_test_data=accuracy_score(Y_test,prediction_on_test_data)
print('Accuracy on test data :',accuracy_on_test_data)
```

```
Accuracy on test data : 0.9659192825112107
```

Building Predictive System

[21]:
```
input_mail=["Congratulations! You've won a million dollars. Just send us your␣
 ↪credit card details to claim your prize"]
# **Convert text to numerical values**
input_data_feature=vectorizer.transform(input_mail)

# **Making Prediction**
prediction=Model.predict(input_data_feature)
print(prediction)
if(prediction[0]==1):
  print('Ham Mail')
else:
  print('Spam Mail')
```

```
[0]
Spam Mail
```

This project successfully developed a spam detection system using logistic regression. The model effectively classified emails as spam or not spam with high accuracy, enhancing email security and user experience. Future improvements could include exploring additional algorithms and expanding the dataset for even better performance.

0.2 Thank You