Project name

Spam Mail Classifier

By:-- Ankita Srivastava

Data Description:--

The dataset contains two columns. The total corpus of 5728 documents. The descriptive feature consists of text. The target feature consists of two classes ham and spam, the column name is spam. The classes are labeled for each document in the data set and represent our target feature with a binary string-type alphabet of {ham; spam}. Classes are further mapped to integer 0 (ham) and 1 (spam).

Related to real word:--

Google and other email services are providing utility for flagging email spam but are still in the infancy stage and need regular feedback from the end-user. Also, popular email services such as Gmail, Yandex, yahoo mail, etc provide basic services as free to the end-user and that of course comes with EULA. There is a great scope in building email spam classifiers, as the private companies run their own email servers and want them to be more secure because of the confidential data, in such cases email spam classifier solutions can be provided to such companies.

Data preparation

The following steps we used for data preparation.

- Identifying Missing values.
- Converting all text to lower case.
- Performing tokenization.
- Removing Stop words.
- Labelling classes: ham/spam: {0;1}
- Splitting Train and Test Data: 75% and 25%.

Modeling

Feature representation: Using word embedding technique CountVectorizer.

Models used: **Naive Bayes**.

Email spam classification done using traditional machine learning techniques comprise Baive Bayes , due to not having sufficient hardware resources, takes less time to train. Also, not opting for neural algorithms due to less data and computing resources.

Feature representation:

Word embeddings can be broadly classified into two categories: Frequency and prediction-based. I have chosen a count vector that shows the count of occurrence of a feature in the given document, thus it is a matrix of document vs vocabulary (containing all the features as a column).

In our case, the size of the Count vector matrix is 5728 x 20114, where 5728 represents the number of documents in the corpus and 20114 represents the number of features in the vocabulary.

Splitting Training and Testing Data:

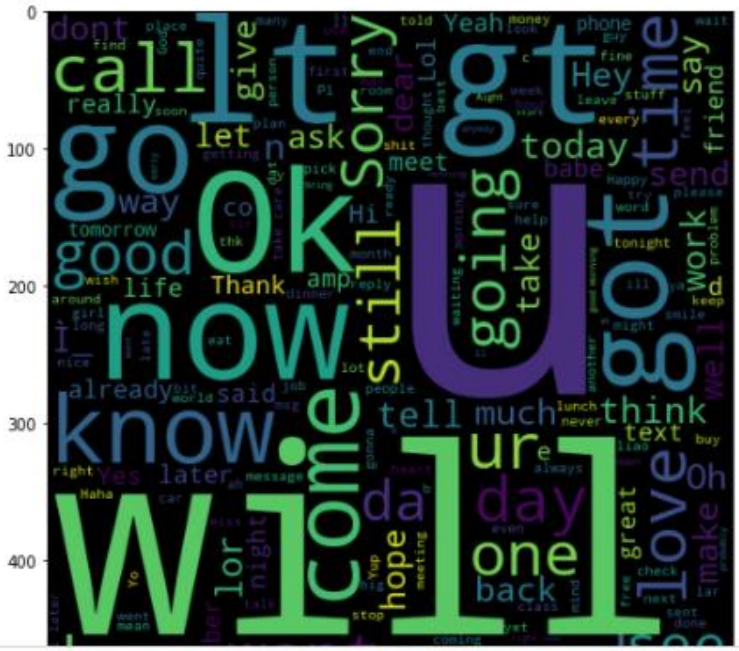
Splitting the data into training and test datasets, where training data contains 75 percent and test data contains 25 percent.

Visualization:--

```
spam_words= ' '.join(list(df[df['labels']=='spam']['messages']))
spam_wc= WordCloud(width=500,height=500).generate(spam_words)
plt.figure(figsize=(10,8))
plt.imshow(spam_wc)
plt.show()
```

✓ 0s completed at 11:19 PM

```
spam_words= ' '.join(list(df[df['labels']=='ham']['messages']))
spam_wc= WordCloud(width=500,height=500).generate(spam_words)
plt.figure(figsize=(10,8))
plt.imshow(spam_wc)
plt.show()
```



✓ 0s completed at 11:19 PM

```
#encoding our labels
df['labels']=df['labels'].apply(lambda x: 1 if x=='spam' else 0)
```

```
[ ]  new_df=[]

     for i in range(df.shape[0]):
         lines=df.iloc[i,1]
         # removing non alphabatic characters
         lines = re.sub('[^A-Za-z]',' ',lines)
         #lowering the every word
         lines = lines.lower()
         #tokenization
         tokenized_lines=word_tokenize(lines)
         #removing stop words , steming and spell correction
         processed_lines=[]
         for i in tokenized_lines:
             if i not in set(stopwords.words('english')):
                 processed_lines.append(spell(stemmer().stem(i)))
         final_lines=' '.join(processed_lines)
         new_df.append(final_lines)
```

Splitting the data into Train and test

Splitting the data into Train and test

```
[ ] from sklearn.model_selection import train_test_split
    X_train,X_test,Y_train,Y_test=train_test_split(new_df,Y,test_size=0.25)
```

```
[ ] X_train
```

```
'day kick euro u kept date latest news result daily remove send get txt stop',
'also north carolina text atm would go gre site pay test result sent',
'back amp pack car let know room',
'k delete contact',
'meh that clash really ah dun mind dun seen lost weight gee',
'asked u question hour answer',
'much get',
'tag friend seem count friend',
'free video camera phone half price line rental th cross news min txt call mobileupd call output',
'win shop spree every week start play text store skills ts wink age perweeksub',
'wish family merry x ma happy new year advance',
'sure mean get',
```

Vectarozation:--

```
[ ] from sklearn.feature_extraction.text import CountVectorizer #tf_idf
    matrix=CountVectorizer()
```

```
[ ] X_train_vect=matrix.fit_transform(X_train).toarray()
    X_test_vect=matrix.transform(X_test).toarray()
```

```
[ ] X_train_vect
```

```
array([[0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       ...,
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0]])
```

Confusion matrix:--

```
[ ] from sklearn.metrics import accuracy_score, confusion_matrix
```

```
[ ] accuracy_score(Y_test,Y_pred) * 100
```

```
85.1399856424982
```

```
[ ] confusion_matrix(Y_test,Y_pred)
```

```
array([[1018,  185],
       [  22,  168]])
```