Email Spam Naïve Bayer's Classification Project

Submitted by:

Mekhala Misra

ACKNOWLEDGMENT

I took help from following websites:

1)Geek for geeks

2)researchgate.net

3)Kaggle.com

INTRODUCTION

- Business Problem Framing

Spam Detector is used to detect unwanted, malicious and virus infected texts and helps to separate them from the no spam texts. It uses a binary type of classification containing the labels such as '**ham'** (no spam) 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.

It is a NLP problem and we are using naïve bayes classifier for this segregation.

Analytical Problem Framing

- Mathematical/ Analytical Modelling of the Problem

  This is a NLP problem as we have to analyse the email text and depending on that we have to predict whether it is a spam or ham.

- Data Sources and their formats

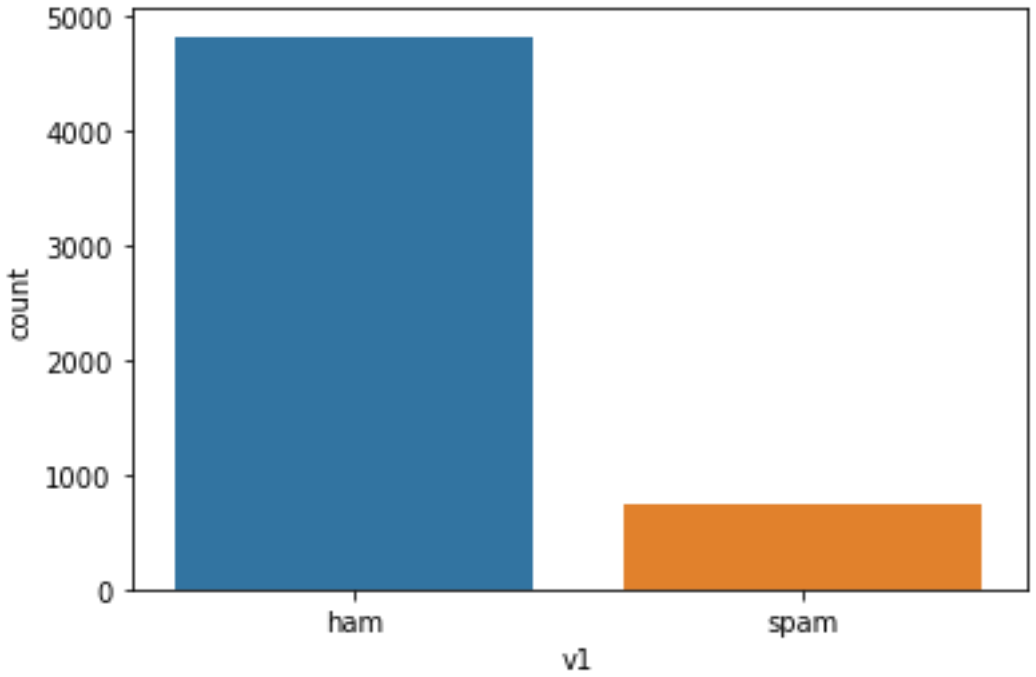
The dataset contain one message per line. Each line is composed by two columns: v1 contains the label (ham or spam) and v2 contains the raw text.This corpus has been collected from free or free for research sources at the Internet.

Out[3]:

	v1	v2	Unnamed: 2	Unnamed: 3	Unnamed: 4
0	ham	Go until jurong point, crazy.. Available only ...	NaN	NaN	NaN
1	ham	Ok lar... Joking wif u oni...	NaN	NaN	NaN
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	NaN	NaN	NaN
3	ham	U dun say so early hor... U c already then say...	NaN	NaN	NaN
4	ham	Nah I don't think he goes to usf, he lives aro...	NaN	NaN	NaN
5	spam	FreeMsg Hey there darling it's been 3 week's n...	NaN	NaN	NaN
6	ham	Even my brother is not like to speak with me. ...	NaN	NaN	NaN
7	ham	As per your request 'Melle Melle (Oru Minnamin...	NaN	NaN	NaN
8	spam	WINNER!! As a valued network customer you have...	NaN	NaN	NaN
9	spam	Had your mobile 11 months or more? U R entitle...	NaN	NaN	NaN

- # Data Pre-processing

- The dataset had no null values
- The dataset had 3 unnamed column with NaN values so we just dropped them.
- We plotted a count plot for checking the number of spam and ham mails



- Further we replaced label column values spam and ham with 1 and 0 respectively.
- Feature Engineering-We have to fetch spam or fraud words from column v2 so in order to do that we first converted entire column v2 feature to lower case then replaced all the phone numbers, email ids, URL's, any sort of number and currency by phno,emailed,link,numbr and currency respectively.

In [6]:
```
#converting all the comments to lower case so thats its easy to analyse them.
df['v2']=df['v2'].str.lower()
```

In [7]:
```
#Replacing email address,links, phone numbers, any sort of numbers and currency as they are not abusive
df['v2']=df['v2'].str.replace(r'^.+@[^\.].*\.[a-z]{2,}$','emailid')
df['v2']=df['v2'].str.replace(r'^http\://[a-zA-Z0-9\-\.]+\.[a-zA-Z]{2,3}(/\S*)?$','link')
df['v2']=df['v2'].str.replace(r'£|\$', 'currency')
df['v2']=df['v2'].str.replace(r'^\(?[\d]{3}\)?[\s-]?[\d]{3}[\s-]?[\d]{4}$','phno')
df['v2']=df['v2'].str.replace(r'\d+(\.\d+)?', 'numbr')
```

- Further we removed all kinds of punctuations from the feature v2.

In [8]:
```
#Removing punctuations
df['v2']=df['v2'].apply(lambda x: ' '.join(
    term for term in x.split() if term not in string.punctuation))

```

- In order to fetch just the fraud words we also have to remove all kind of stop words.

In [9]:
```
# Removing stop words
sw = set(stopwords.words('english') + ['u', 'ur', '4', '2', 'im', 'dont', 'doin', 'ure'])
df['v2']=df['v2'].apply(lambda x: ' '.join(
    term for term in x.split() if term not in sw))
```

- In order to further analyse the text of column v2 using morphological analysis of the words called lemmatization.

In [10]:
```
#word Lemmatizer
lm=WordNetLemmatizer()
df['v2']=df['v2'].apply(lambda x: ' '.join(
 lm.lemmatize(t) for t in x.split()))
```

- We split the data into training set and testing set using train test split method.
- We further found the number of occurrences of each word using count vectorizer.

In [15]:
```
from sklearn.feature_extraction.text import CountVectorizer
```

In [16]:
```
Vectorizer = CountVectorizer()
```

In [20]:
```
count= Vectorizer.fit_transform(X_train.values)
```

- Then we applied naïve bayes classifier on the number of occurrences of each word and the feature v1 values.

In [21]:
```
#Creating a object for this called clf
clf=MultinomialNB()
clf.fit(count,y_train.values)
y_pred=clf.predict(Vectorizer.transform(X_test.values))
print(accuracy_score(y_pred,y_test.values))
```

```
0.9901345291479821
```

- This classifier gave an accuracy of 99%.

• Hardware and Software Requirements and Tools Used

We imported following packages:

from nltk.stem import WordNetLemmatizer
import nltk
from nltk.corpus import stopwords
import string
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

```
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.metrics import roc_curve,auc,classification_report,confusion_matrix
from sklearn.feature_extraction.text import TfidfVectorizer
import warnings
warnings.filterwarnings('ignore')
```

Model/s Development and Evaluation

- Identification of possible problem-solving approaches (methods)
- Since this is a NLP classification problem so I will use Naïve Bayes Classifier.

- Run and Evaluate selected models

1) Naïve Bayes Classifier

We applied this algorithm and found the test accuracy as 99%.

In [21]:
```
#Creating a object for this called clf
clf=MultinomialNB()
clf.fit(count,y_train.values)
y_pred=clf.predict(Vectorizer.transform(X_test.values))
print(accuracy_score(y_pred,y_test.values))
```

```
0.9901345291479821
```

We also tested this model on other metrics-

1) classification report:

Classification Report

In [31]:
```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.99      1.00      0.99       953
           1       0.98      0.95      0.97       162

    accuracy                           0.99      1115
   macro avg       0.99      0.97      0.98      1115
weighted avg       0.99      0.99      0.99      1115
```

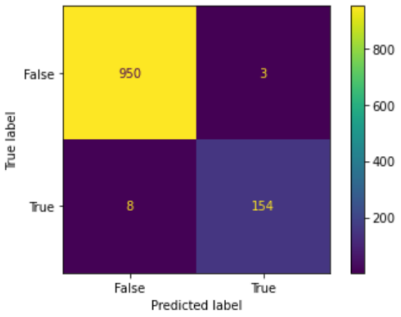
2) confusion matrix:

Confusion matrix

In [27]:
```
confusion_matrix (y_test , y_pred)
```

Out[27]:
```
array([[950,   3],
       [  8, 154]], dtype=int64)
```

In [29]:
```
confusion_matrix = metrics.confusion_matrix(y_test, y_pred)
cm = metrics.ConfusionMatrixDisplay(confusion_matrix = confusion_matrix, display_labels = [False, True])
cm.plot()
plt.show()
```



3) AUC-ROC Curve: