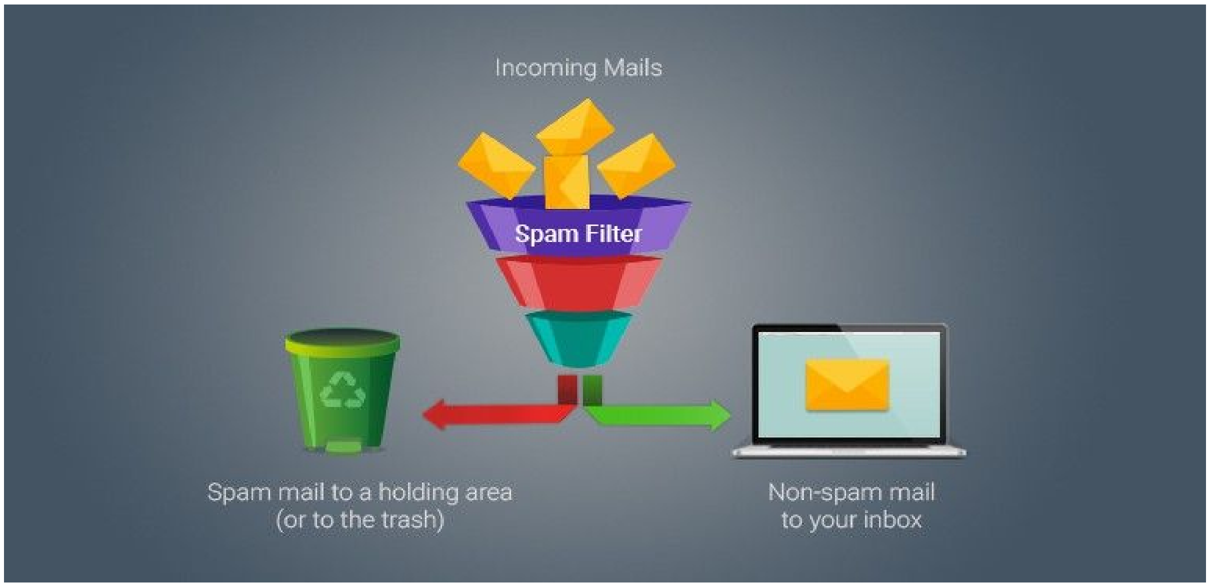
PROJECT REPORT
TOPIC : SPAM DETECTION USING CLASSIFICATION

By- Anushka Jadhav

Introduction

We receive countless spam emails and sms everyday that are dangerous to a great extent. Many people fall prey to these spam messages and lose a huge amount of money. Americans have lost $703,000 to these types of scams in 2018 alone according to a report by ADT Security Services. One way to reduce the number of people falling for these types of scams is spam filtering.


Incoming Mails
Spam Filter
Spam mail to a holding area (or to the trash)
Non-spam mail to your inbox

One way spam emails are sorted is by using a Naive Bayes classifier. The Naive Bayes algorithm relies on Bayes Rule. This algorithm will classify each object by looking at all of it's features individually. Bayes Rule below shows us how to calculate the posterior probability for just one feature.

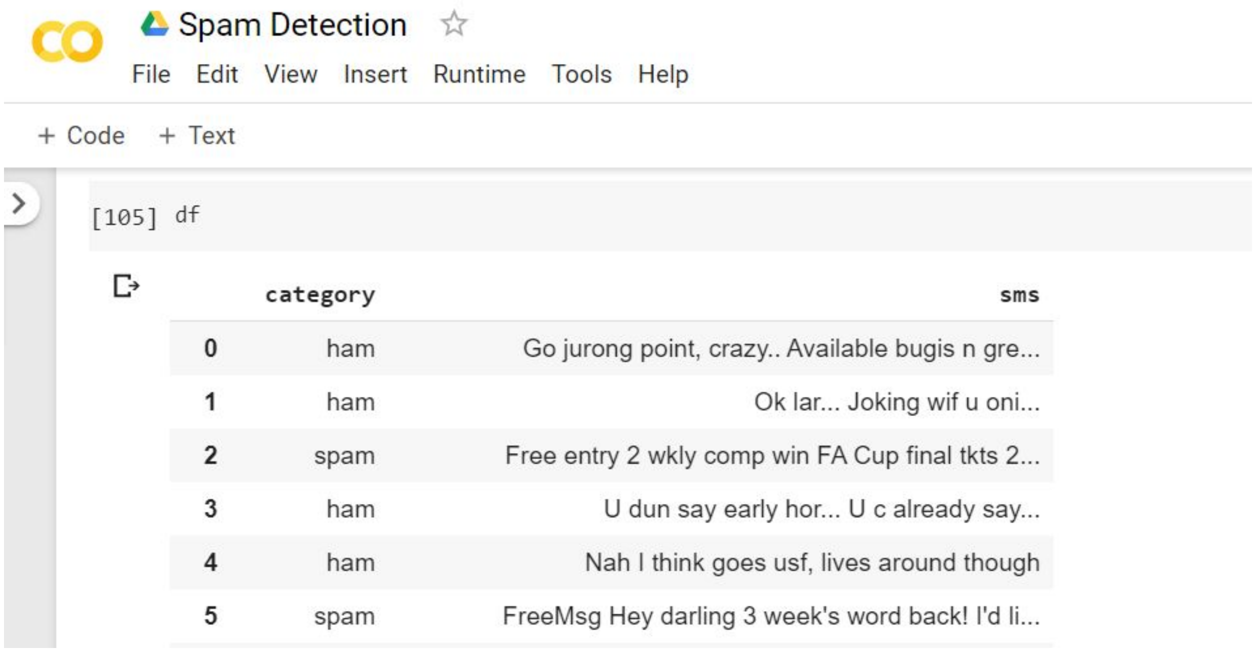
The posterior probability of the object is calculated for each feature and then these probabilities are multiplied together to get a final probability. This probability is calculated for the other class as well. Whichever has the greater probability that ultimately determines what class the object is in.

$$\overset{\text{Posterior}}{P(A|B)} = \frac{\overset{\text{Likelihood}}{P(B|A)}\,\overset{\text{Prior}}{P(A)}}{\underset{\text{Normalizing constant}}{P(B)}}$$

$$P(B) = \sum_{Y} P(B|A)P(A)$$

Creating A Spam Filter Using Naive Bayes Algorithm

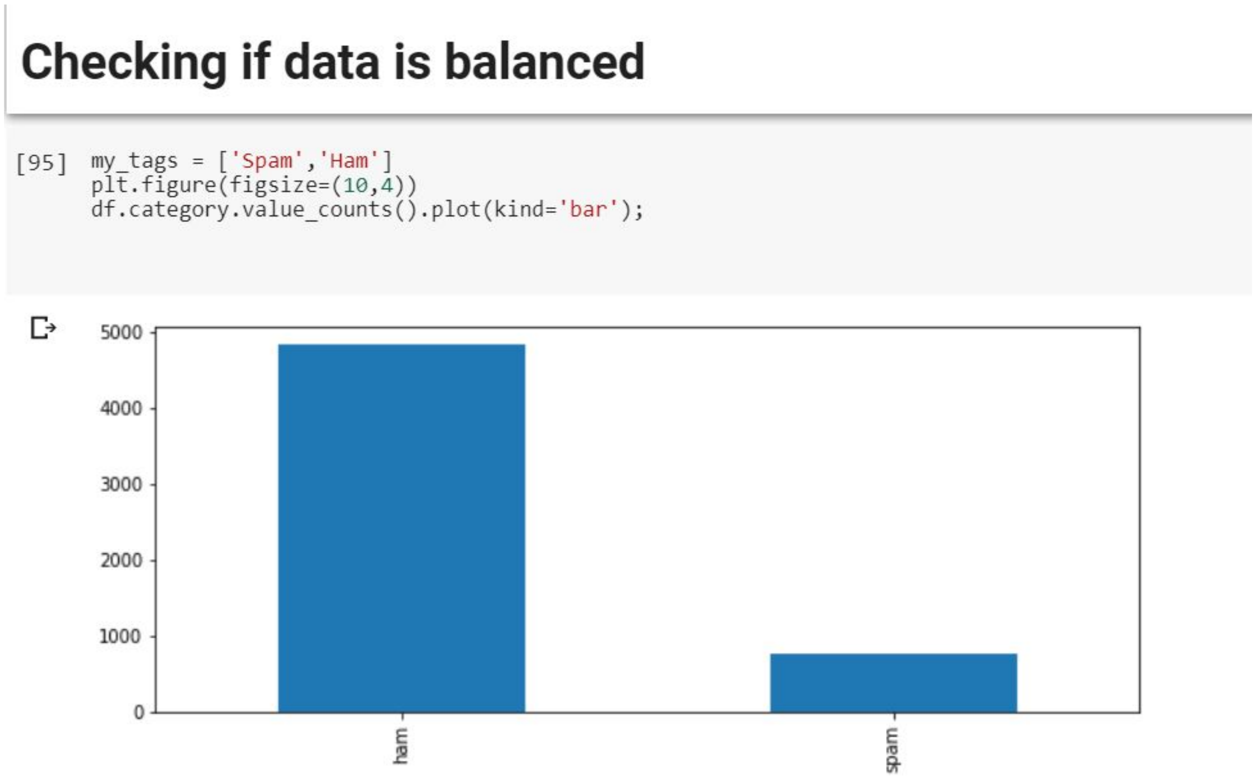
The first step in making a filter is to get a data set of sms/emails. I have used the sms dataset provided and read it into a pandas dataframe. Your dataframe should look something like this:



```
[105] df
```

	category	sms
0	ham	Go jurong point, crazy.. Available bugis n gre...
1	ham	Ok lar... Joking wif u oni...
2	spam	Free entry 2 wkly comp win FA Cup final tkts 2...
3	ham	U dun say early hor... U c already say...
4	ham	Nah I think goes usf, lives around though
5	spam	FreeMsg Hey darling 3 week's word back! I'd li...

I have renamed v1 as category and v2 as sms.

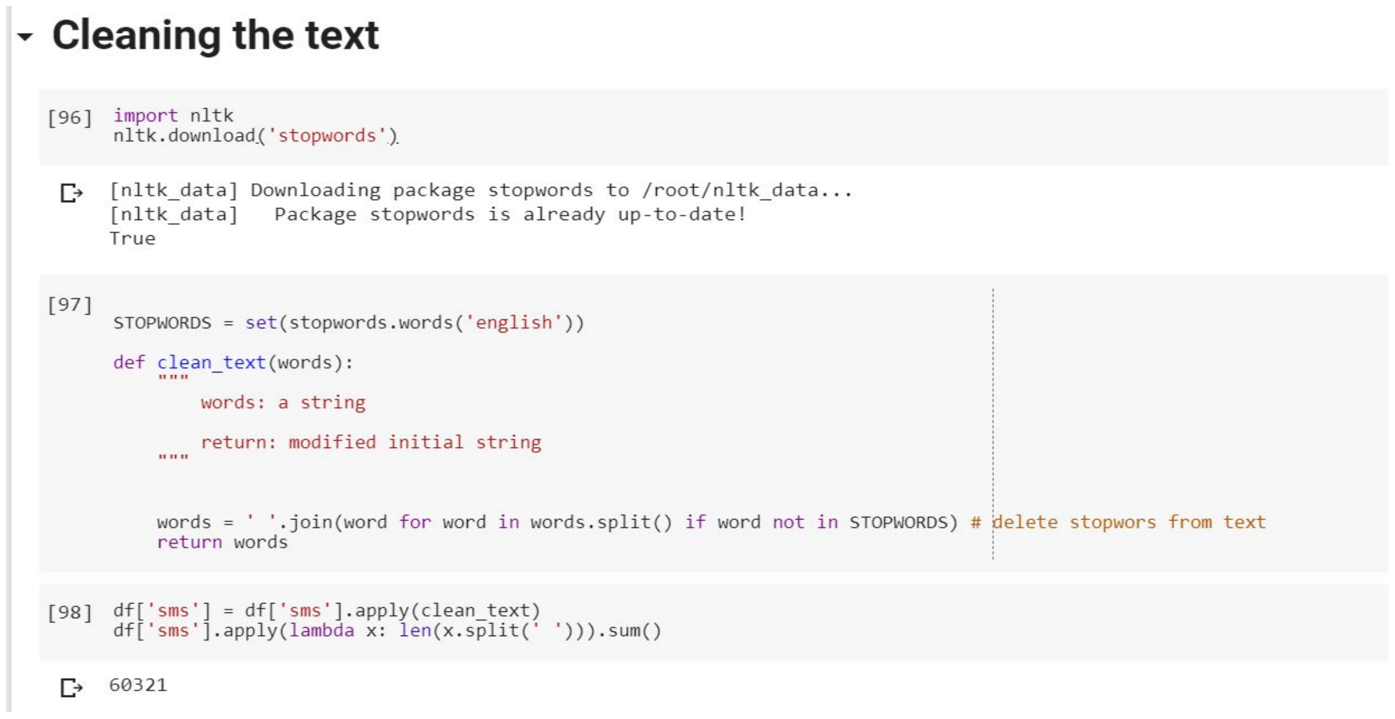


Checking if data is balanced

```
[95] my_tags = ['Spam','Ham']
     plt.figure(figsize=(10,4))
     df.category.value_counts().plot(kind='bar');
```

Cleaning the data

The next step is to clean the data to remove jargon and repetitive words such as “my, a, the, is” etc.



Cleaning the text

```
[96] import nltk
     nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
True
```

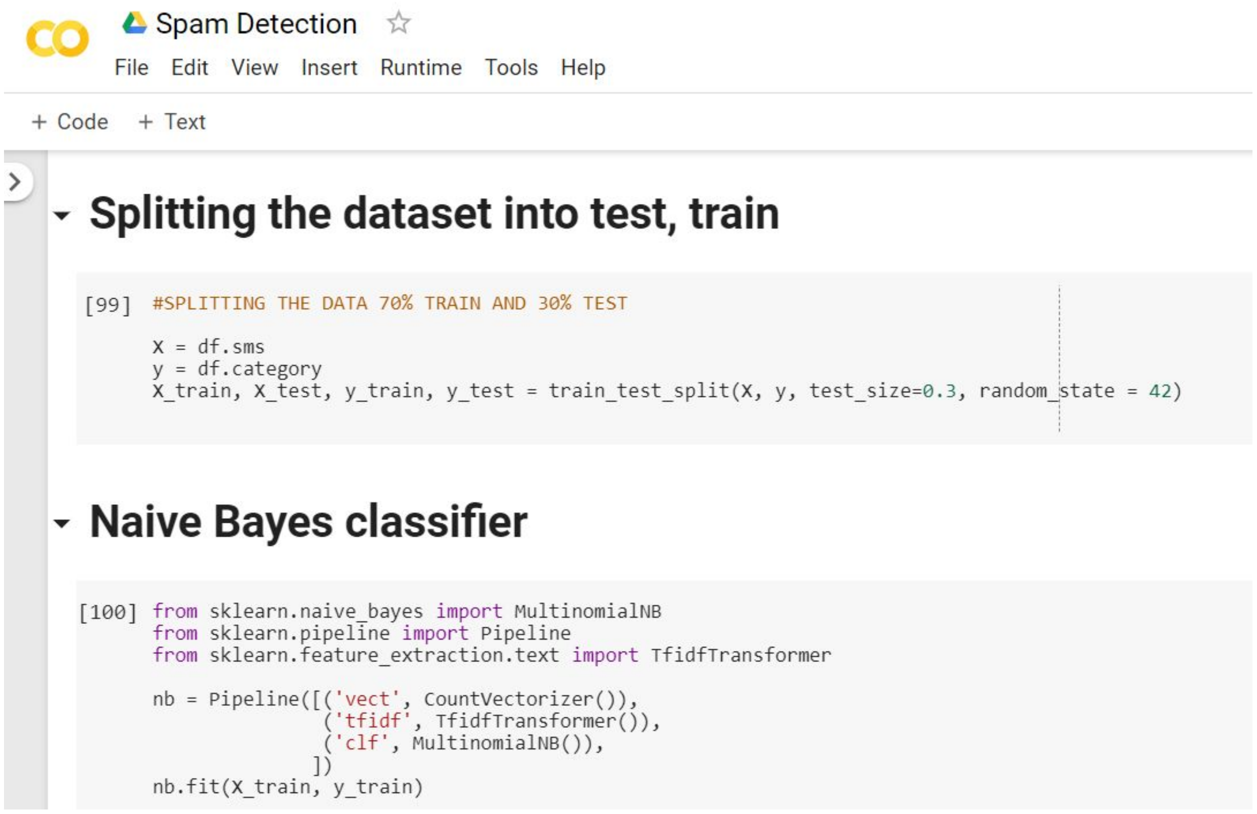
```
[97]
     STOPWORDS = set(stopwords.words('english'))

     def clean_text(words):
         """
             words: a string

             return: modified initial string
         """

         words = ' '.join(word for word in words.split() if word not in STOPWORDS) # delete stopwors from text
         return words
```

```
[98] df['sms'] = df['sms'].apply(clean_text)
     df['sms'].apply(lambda x: len(x.split(' '))).sum()
```

```
60321
```

Classifying the data using Naive Bayes

The next step is to split the data and form the model. I have split the data in such a way that 70% is TRAIN data i.e. to train the model and 30% is TEST data that will test the accuracy of our prediction. We then train the model and calculate the accuracy of prediction on test data in the next step.

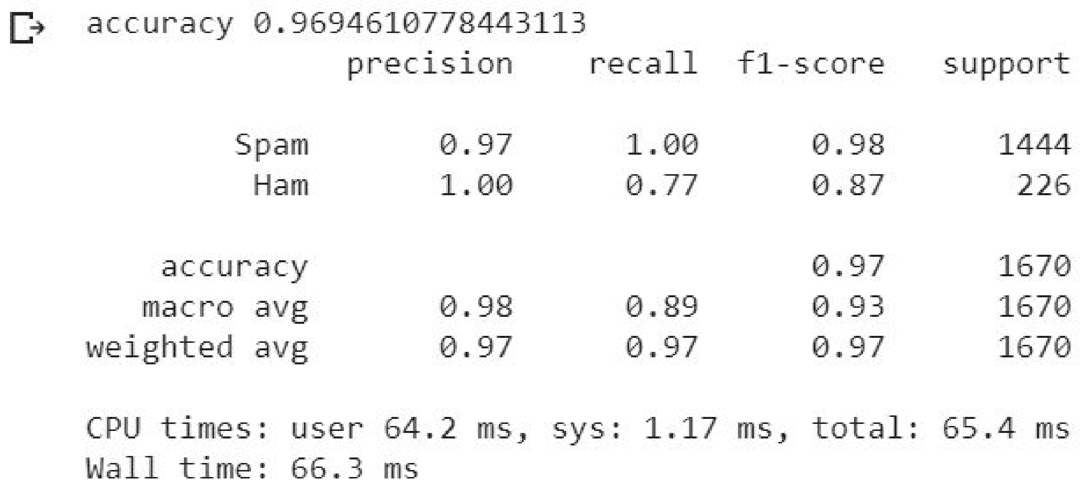


Splitting the dataset into test, train

```
[99] #SPLITTING THE DATA 70% TRAIN AND 30% TEST

     X = df.sms
     y = df.category
     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state = 42)
```

Naive Bayes classifier

```
[100] from sklearn.naive_bayes import MultinomialNB
      from sklearn.pipeline import Pipeline
      from sklearn.feature_extraction.text import TfidfTransformer

      nb = Pipeline([('vect', CountVectorizer()),
                     ('tfidf', TfidfTransformer()),
                     ('clf', MultinomialNB()),
                    ])
      nb.fit(X_train, y_train)
```



```
accuracy 0.9694610778443113
              precision    recall  f1-score   support

        Spam       0.97      1.00      0.98      1444
         Ham       1.00      0.77      0.87       226

    accuracy                           0.97      1670
   macro avg       0.98      0.89      0.93      1670
weighted avg       0.97      0.97      0.97      1670

CPU times: user 64.2 ms, sys: 1.17 ms, total: 65.4 ms
Wall time: 66.3 ms
```

Accuracy of the model is found to be 97%.

Conclusion

Naive Bayes is powerful yet simple algorithm that is especially useful when filtering out spam sms/emails.

Resources

- https://www.analyticsvidhya.com/blog/2017/09/naive-bayes-explained/
- https://www.cnbc.com/2019/04/18/nigerian-prince-scams-still-rake-in-over-700000-dollars-a-year.html
- https://www.youtube.com/watch?v=M59h7CFUwPU