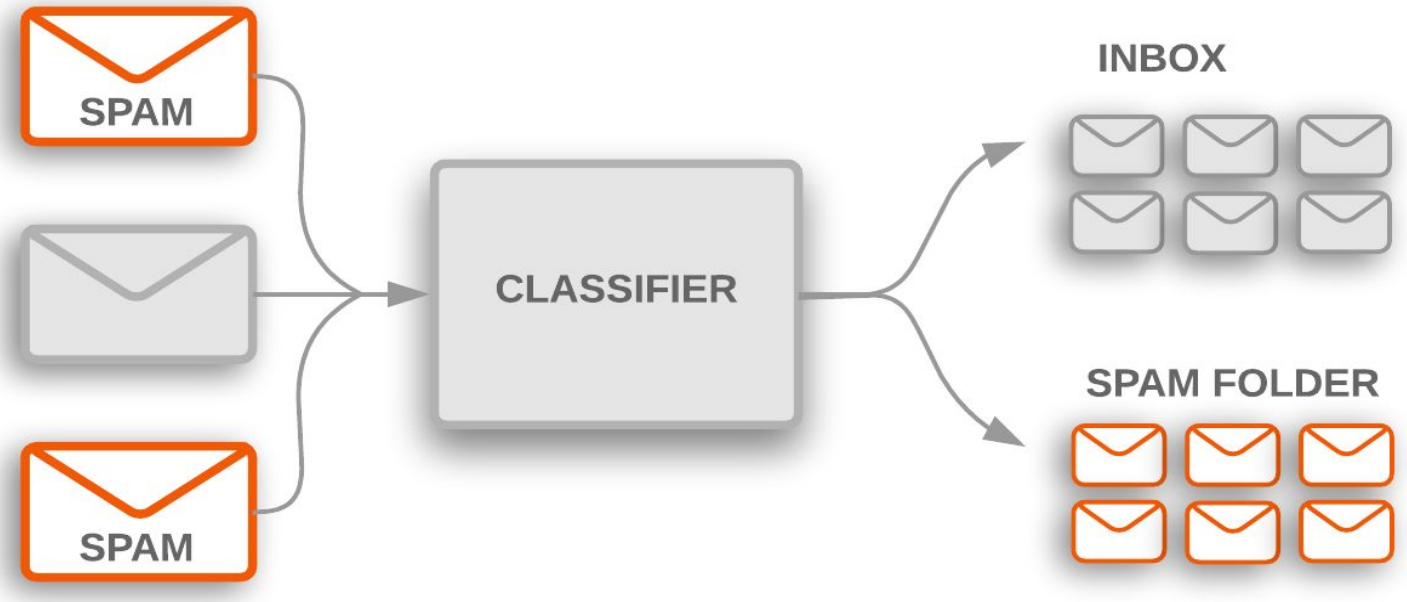


Spam Mail Classification

Regularly we check our emails, not all the emails which came to our mail account will appear into inbox. Many of them go to spam or junk folders.

“Ever wondered how it was happening?”

How the mails are classified and sent to inbox or spam folder based on the email text?

Before any email reaching your inbox, Google is using their own email classifier**,** which will identify whether the received email need to send to inbox or spam.

If you are still thinking about how the email classifier works don't worry.

In this article, we are going to build an email spam classifier in python that classifies the given mail is spam or not.

There are a number of ways to build email classifier using Natural Language Processing different algorithms. But in this article, we are going to use the nltk library to build the email classifier.

Spam Mails :- Email spam, also referred to as junk email, is unsolicited messages sent in bulk by email.

The name comes from a Monty Python sketch in which Spam is ubiquitous, unavoidable, and repetitive. Email spam has steadily grown since the early 1990s, and by 2014 was estimated to account for around 90% of total email traffic.

Since the expense of the spam is borne mostly by the recipient, it is effectively advertising. This makes it an excellent example of a negative externality.

The legal definition and status of spam varies from one jurisdiction to another, but nowhere have laws and lawsuits been particularly successful in stemming spam.

Problem Definition :-

Spam is sent by both reputable organizations and lesser companies. When spam is sent by reputable companies it is sometimes referred to as Mainsleaze. Mainsleaze makes up approximately 3% of the spam sent over the internet. The problem with mainsleaze is that it is generally mixed in with mail that the recipients asked for, and it is difficult to tell the difference using traditional mail filters. As a result, if a mail system filters out all the mail from a mainsleazer, they will get complaints from the people who signed up.

It was estimated in 2009 that spam cost businesses around US$130 billion. As the scale of the spam problem has grown, ISPs and the public have turned to government for relief from spam, which has failed to materialize.

Spam is also a medium for fraudsters to scam users into entering personal information on fake Web sites using emails forged to look like they are from banks or other organizations, such as PayPal. This is known as phishing. Targeted phishing, where known information about the recipient is used to create forged emails, is known as spear-phishing.

Data Analysis : - In this use case, a spam mail dataset is given by our SME which contains various messages for about 2893 and classified as 0 and 1 Label where O means not a spam message and 1 means it is a Spam message.

uploading dataset into the python (Jupyter Notebook):

In [1]:
```
#importing pandas to read the csv file
import pandas as pd
```

In [2]:
```
#creating dataframe to store the csv file
msg_df=pd.read_csv('messages.csv')
```

In [3]:
```
#printing the data frame to see what it consist of
msg_df
```

Out[3]:

	subject	message	label
0	job posting - apple-iss research center	content - length : 3386 apple-iss research cen...	0
1	NaN	lang classification grimes , joseph e . and ba...	0
2	query : letter frequencies for text identifica...	i am posting this inquiry for sergei atamas (...	0
3	risk	a colleague and i are researching the differin...	0
4	request book information	earlier this morning i was on the phone with a...	0
...	...	...	...
2888	love your profile - ysuolvpv	hello thanks for stopping by ! ! we have taken...	1
2889	you have been asked to join kiddin	the list owner of : " kiddin " has invited you...	1
2890	anglicization of composers ' names	judging from the return post , i must have sou...	0
2891	re : 6 . 797 , comparative method : n - ary co...	gotcha ! there are two separate fallacies in t...	0
2892	re : american - english in australia	hello ! i ' m working on a thesis concerning a...	0

2893 rows × 3 columns

we have total 2893 records in our dataset out which the count of spam i.e. “1” and count of not a spam i.e. “0” is

```
In [5]: #counting the number of 0 and 1 labels
        msg_df['label'].value_counts()

Out[5]: 0    2412
        1     481
        Name: label, dtype: int64
```

As the subject column will not play a role in predicting the mail as spam or not a spam we will drop it.

Data Preprocessing :

Importing nltk library to clean the messages and remove stopwords in the dataset

```
In [4]: #data cleaning and pre-processing
        import re
        import nltk
        from nltk.corpus import stopwords
        from nltk.stem.porter import PorterStemmer
```

Code for pre-processing and clean the messages column

```
[8]: #Applying stemming technique inorder to decrese the length of the message in message column
     ps= PorterStemmer()
     Corpus=[]
     for i in range(0,len(msg_df)):
         review=re.sub('[^a-zA-Z]',' ',msg_df['message'][i])
         review=review.lower()
         review=review.split()
         review=[ps.stem(word) for word in review if not word in set(stopwords.words('english'))]
         review=' '.join(review)
         Corpus.append(review)
```

Explanation of above cell;

First we have created an instance of a porter stemmer class,

Secondly A corpus list to store the pre processed messages

Thirdly Initializing the loop from 0 to length of the dataset defining the review list

First line of the loop deletes all the commas, whitespaces full stops etc., it only stores the words starting from small a-z and capital a-z.

Second line of the loop converts all the characters present in the review list to lowercase of english alphabet.

Third line of the loop splits the words to separate them to perform stemming.

Fourth line of the loops checks if the word is present in stop words if not then finding the stem word for it using stemming and saving it

Then finally saving the word in review into the corpus list.

Here I have used bag of words model to change the words into vectors :

```
In [9]: #creating a bag of words to change word into vectors
        from sklearn.feature_extraction.text import CountVectorizer
        cv=CountVectorizer()
        x=cv.fit_transform(Corpus).toarray()
        y=msg_df.iloc[:,-1]
```

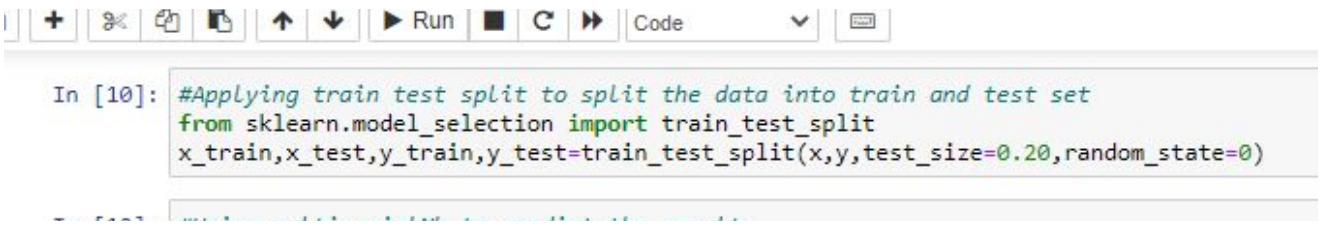
Split train and test datasets

We will split the loaded data into two separate datasets.

- **Train** dataset: For training the text categorization model.
- **Test** dataset: For validating the performance of the model.

To split the data into 2 such datasets we are using scikit learn model selection train test split method, in such a way that the test data will be 20% of the loaded data.

Let's say we are having 100 records in the loaded dataset, if we specify the test as 0.20% then train and test split method split 80 records for training and the remaining 20 records for testing.



Creating training model data

Now we have two datasets one is for training and the other one is for testing our model.

Here we are considering the Naïve baye's and SVC algorithms to predict the accuracy score of our model.

```
In [13]: #Using multinomialNb and support vector machine to predict the result and Confusion matrix to measure the performance
         from sklearn.naive_bayes import MultinomialNB
         from sklearn.svm import SVC
         from sklearn.metrics import confusion_matrix
         from sklearn.metrics import accuracy_score
```

```
In [*]: from sklearn.model_selection import GridSearchCV
        svc=SVC()
        parameters={'kernel':('linear','rbf'), 'C':[0.1,1]}
        gd=GridSearchCV(svc,parameters)
        gd.fit(x_train,y_train)
        gd.best_params_
```

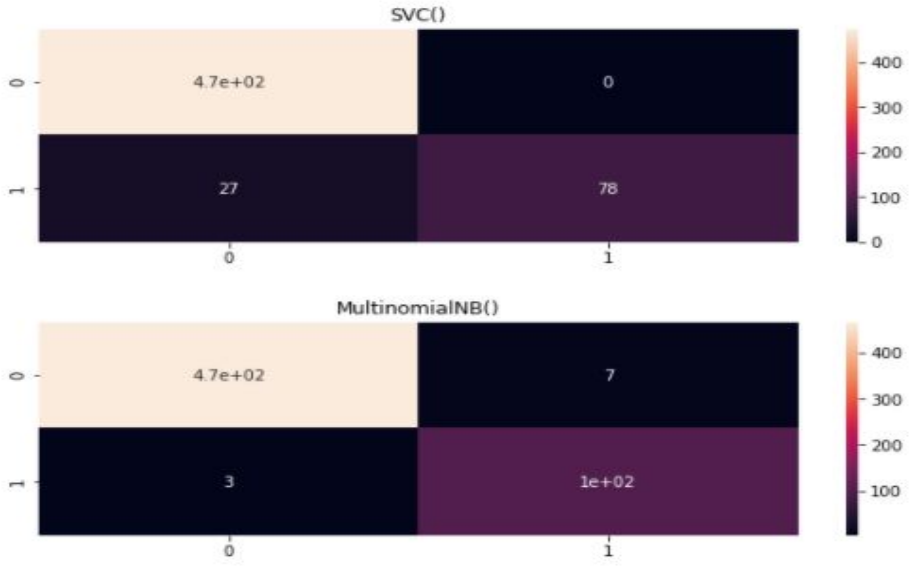
First we will import both the algorithms from sklearn and then finally we predict the result and save the model which is giving best accuracy score.

```
In [13]: #Using multinomialNb and support vector machine to predict the result and Confusion matrix to measure the performance
         from sklearn.naive_bayes import MultinomialNB
         from sklearn.svm import SVC
         from sklearn.metrics import confusion_matrix
         from sklearn.metrics import accuracy_score
```

```
In [25]: Model= []
         score= []
         model=[SVC(),MultinomialNB()]
         for models in model:
             print("\n")
             Model.append(models)
             models.fit(x_train,y_train)
             print(models)
             prediction=models.predict(x_test)
             print("\n")
             acc_score=accuracy_score(y_test,prediction)
             print("ACCURACY_SCORE =",acc_score)
             score.append(acc_score*100)
             print("\n")
             cm=confusion_matrix(y_test,prediction)
             print(cm)
             print("\n")
             plt.figure(figsize=(10,30))
             plt.subplot(911)
             plt.title(models)
             print(sns.heatmap(cm,annot=True))
```

```
SVC()

ACCURACY_SCORE = 0.9533678756476683

[[474   0]
 [ 27  78]]

AxesSubplot(0.125,0.808774;0.62x0.0712264)

MultinomialNB()

ACCURACY_SCORE = 0.9827288428324698

[[467   7]
```

Here we can see that Multinomial NB is giving a best accuracy score. Therefore before confirming our algorithm we first see the confusion matrix for both the algorithm.



From visualizing the confusion matrix we confirm that MultinomialNB is giving good result.

Hence saving the model.

Conclusion :-

For the above email messages, the actual output is not a spam and our model is having high probability which is nearly **99%** for not a spam and 1% for spam. Which means our model is predicting the email text properly.

Now we have build the email spam classifier, using this pipeline you can solve similar kind of problems. All you needs to change is the inputs, outputs and apply few natural language text pre processing techniques before building the model.

In this article we learned how to build the email spam classifier with nltk in few lines of code. Don't stop, using the same code try to build various text classification models.