In this Data Science Project I will show you how to detect email spam using Machine Learning technique called Natural Language Processing and Python.

So this program will detect if an email is spam (1) or not (0)

Import the libraries :

```
import numpy as np
import pandas as pd
import nltk
from nltk.corpus import stopwords
import string
```

```
# installing tensorflow_text
!pip install tensorflow-text
```

```
Collecting keras<2.12,>=2.11.0
  Downloading keras-2.11.0-py2.py3-none-any.whl (1.7 MB)
     |████████████████████████████████| 1.7 MB 36.1 MB/s
Collecting tensorflow-estimator<2.12,>=2.11.0
  Downloading tensorflow_estimator-2.11.0-py2.py3-none-any.whl (439 kB)
     |████████████████████████████████| 439 kB 21.4 MB/s
Requirement already satisfied: typing-extensions>=3.6.6 in /usr/local/lib/python3.7/d
Collecting flatbuffers>=2.0
  Downloading flatbuffers-22.10.26-py2.py3-none-any.whl (26 kB)
Collecting tensorboard<2.12,>=2.11
  Downloading tensorboard-2.11.0-py3-none-any.whl (6.0 MB)
     |████████████████████████████████| 6.0 MB 40.1 MB/s
Requirement already satisfied: absl-py>=1.0.0 in /usr/local/lib/python3.7/dist-packag
Requirement already satisfied: opt-einsum>=2.3.2 in /usr/local/lib/python3.7/dist-pac
Requirement already satisfied: h5py>=2.9.0 in /usr/local/lib/python3.7/dist-packages

Requirement already satisfied: setuptools in /usr/local/lib/python3.7/dist-packages (
Requirement already satisfied: astunparse>=1.6.0 in /usr/local/lib/python3.7/dist-pac
Requirement already satisfied: six>=1.12.0 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: protobuf<3.20,>=3.9.2 in /usr/local/lib/python3.7/dist
Requirement already satisfied: wheel<1.0,>=0.23.0 in /usr/local/lib/python3.7/dist-pa
Requirement already satisfied: cached-property in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: google-auth-oauthlib<0.5,>=0.4.1 in /usr/local/lib/pyt
Requirement already satisfied: tensorboard-plugin-wit>=1.6.0 in /usr/local/lib/python
Requirement already satisfied: werkzeug>=1.0.1 in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: requests<3,>=2.21.0 in /usr/local/lib/python3.7/dist-p
Requirement already satisfied: google-auth<3,>=1.6.3 in /usr/local/lib/python3.7/dist
Requirement already satisfied: tensorboard-data-server<0.7.0,>=0.6.0 in /usr/local/li
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: pyasn1-modules>=0.2.1 in /usr/local/lib/python3.7/dist
Requirement already satisfied: cachetools<6.0,>=2.0.0 in /usr/local/lib/python3.7/dis
Requirement already satisfied: rsa<5,>=3.1.4 in /usr/local/lib/python3.7/dist-package
Requirement already satisfied: requests-oauthlib>=0.7.0 in /usr/local/lib/python3.7/d
Requirement already satisfied: importlib-metadata>=4.4 in /usr/local/lib/python3.7/di
```

```
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.7/dist-packages (f
Requirement already satisfied: pyasn1<0.5.0,>=0.4.6 in /usr/local/lib/python3.7/dist-
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/local/
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/dist-pa
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-pac
Requirement already satisfied: oauthlib>=3.0.0 in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in /usr/local/lib/python3.7/d
Installing collected packages: tensorflow-estimator, tensorboard, keras, flatbuffers,
  Attempting uninstall: tensorflow-estimator
    Found existing installation: tensorflow-estimator 2.9.0
    Uninstalling tensorflow-estimator-2.9.0:
      Successfully uninstalled tensorflow-estimator-2.9.0
  Attempting uninstall: tensorboard
    Found existing installation: tensorboard 2.9.1
    Uninstalling tensorboard-2.9.1:
      Successfully uninstalled tensorboard-2.9.1
  Attempting uninstall: keras
    Found existing installation: keras 2.9.0
    Uninstalling keras-2.9.0:
      Successfully uninstalled keras-2.9.0
  Attempting uninstall: flatbuffers
    Found existing installation: flatbuffers 1.12
    Uninstalling flatbuffers-1.12:
```

```
import tensorflow_hub as hub
import pandas as pd
import tensorflow_text as text
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
import numpy as np
```

Load the data and print the first 5 rows :¶

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mour
```

```
# Reading the data file into a DATAFRAME and checking the shape
df=pd.read_csv("/content/drive/My Drive/Colab Notebooks/spam_update.csv")
print(df.shape)
```

```
(5572, 2)
```

```
df
```

	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...
...	...	...
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

5572 rows × 2 columns

```
# check count and unique and top values and their frequency
df['Category'].value_counts()
```

```
ham     4825
spam     747
Name: Category, dtype: int64
```

```
# check percentange of data - states how much data needs to be balanced
str(round(747/4825,2))+'%'
```

```
'0.15%'
```

```
# creating 2 new dataframe as df_ham , df_spam

df_spam = df[df['Category']=='spam']
print("Spam Dataset Shape:", df_spam.shape)

df_ham = df[df['Category']=='ham']
print("Ham Dataset Shape:", df_ham.shape)
```

```
Spam Dataset Shape: (747, 2)
Ham Dataset Shape: (4825, 2)
```

```
# downsampling ham dataset - take only random 747 example
# will use df_spam.shape[0] - 747
```

```
df_ham_downsampled = df_ham.sample(df_spam.shape[0])
df_ham_downsampled.shape
```

```
(747, 2)
```

```
# concating both dataset - df_spam and df_ham_balanced to create df_balanced dataset
df_balanced = pd.concat([df_spam , df_ham_downsampled])
df_balanced.head()
```

	Category	Message
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...
5	spam	FreeMsg Hey there darling it's been 3 week's n...
8	spam	WINNER!! As a valued network customer you have...
9	spam	Had your mobile 11 months or more? U R entitle...
11	spam	SIX chances to win CASH! From 100 to 20,000 po...

```
df.columns
```

```
Index(['Category', 'Message'], dtype='object')
```

```
df.drop_duplicates(inplace=True)
print(df.shape)
```

```
(5157, 2)
```

To check for duplicates and remove them :¶

```
df_balanced['Category'].value_counts()
```

```
spam    747
ham     747
Name: Category, dtype: int64
```

To see the number of missing data for each column : ¶

```
print(df.isnull().sum())
```

```
Category    0
```

```
Message     0
dtype: int64
```

Now Download the stop words¶

Stop words in natural language processing, are useless words (data)#

```
# download the stopwords package
nltk.download("stopwords")
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
True
```

```
def process(text):
    nopunc = [char for char in text if char not in string.punctuation]
    nopunc = ''.join(nopunc)

    clean = [word for word in nopunc.split() if word.lower() not in stopwords.words('english'
    return clean
# to show the tokenization
df['Message'].head().apply(process)
```

```
0    [Go, jurong, point, crazy, Available, bugis, n...
1                      [Ok, lar, Joking, wif, u, oni]
2    [Free, entry, 2, wkly, comp, win, FA, Cup, fin...
3        [U, dun, say, early, hor, U, c, already, say]
4    [Nah, dont, think, goes, usf, lives, around, t...
Name: Message, dtype: object
```

Now convert the text into a matrix of token counts :

```
from sklearn.feature_extraction.text import CountVectorizer
message = CountVectorizer(analyzer=process).fit_transform(df['Message'])
```

```
message
```

```
<5157x11422 sparse matrix of type '<class 'numpy.int64'>'
        with 45972 stored elements in Compressed Sparse Row format>
```

Now we need to split the data into training and testing sets, and then we will use this one row of data for testing to make our prediction later on and test to see if the prediction matches with the actual value.

```
#split the data into 80% training and 20% testing
from sklearn.model_selection import train_test_split
xtrain, xtest, ytrain, ytest = train_test_split(message, df['Category'], test_size=0.20, rand
# To see the shape of the data
print(message.shape)
```

```
(5157, 11422)
```

Now we need to create and train the Multinomial Naive Bayes classifier which is suitable for classification with discrete features.

```
# create and train the Naive Bayes Classifier
from sklearn.naive_bayes import MultinomialNB
classifier = MultinomialNB().fit(xtrain, ytrain)
```

To see the classifiers prediction and actual values on the data set :

```
print(classifier.predict(xtrain))
print(ytrain.values)
```

```
['ham' 'spam' 'ham' ... 'ham' 'ham' 'ham']
['ham' 'spam' 'ham' ... 'ham' 'ham' 'ham']
```

Now let’s see how well our model performed by evaluating the Naive Bayes classifier and the report, confusion matrix &

accuracy score.

```
# Evaluating the model on the training data set
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
pred = classifier.predict(xtrain)
print(classification_report(ytrain, pred))
print()
print("Confusion Matrix: \n", confusion_matrix(ytrain, pred))
print("Accuracy: \n", accuracy_score(ytrain, pred))
```

```
              precision    recall  f1-score   support

         ham       1.00      1.00      1.00      3619
        spam       0.98      0.97      0.98       506

    accuracy                           0.99      4125
   macro avg       0.99      0.99      0.99      4125
weighted avg       0.99      0.99      0.99      4125


Confusion Matrix:
 [[3611    8]
 [  13  493]]
Accuracy:
 0.9949090909090909
```

It looks like the model used is 99.71% accurate. Let’s test the model on the test data set (xtest & ytest) by printing the predicted value, and the actual value to see if the model can accurately classify the email text. ¶

```
#print the predictions
print(classifier.predict(xtest))
#print the actual values
print(ytest.values)
```

```
['ham' 'ham' 'ham' ... 'ham' 'ham' 'ham']
['ham' 'ham' 'ham' ... 'ham' 'ham' 'ham']
```

Now let’s evaluate the model on the test data set : ¶

```
# Evaluating the model on the training data set
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
pred = classifier.predict(xtest)
print(classification_report(ytest, pred))
print()
print("Confusion Matrix: \n", confusion_matrix(ytest, pred))
print("Accuracy: \n", accuracy_score(ytest, pred))
```

```
              precision    recall  f1-score   support

         ham       0.99      0.97      0.98       897
        spam       0.81      0.93      0.86       135

    accuracy                           0.96      1032
   macro avg       0.90      0.95      0.92      1032
weighted avg       0.96      0.96      0.96      1032


Confusion Matrix:
 [[867  30]
 [ 10 125]]
Accuracy:
 0.9612403100775194
```

```
df_balanced.sample(10)
```

	Category	Message
5467	spam	Get your garden ready for summer with a FREE s...
3274	ham	Just finished eating. Got u a plate. NOT lefto...
3954	spam	Refused a loan? Secured or Unsecured? Can't ge...
4069	spam	TBS/PERSOLVO. been chasing us since Sept for£3...
1030	ham	Its good, we'll find a way
2615	ham	Sir, hope your day is going smoothly. i really...
5501	spam	PRIVATE! Your 2003 Account Statement for 07808...
3323	ham	Ok darlin i supose it was ok i just worry too ...
515	spam	You are guaranteed the latest Nokia Phone, a 4...
340	ham	U calling me right? Call my hand phone...

Data Prepration

Create Numerical Repersentation Of Category - One hot encoding

1. Create a new column
2. Use df[col].apply(lambda function)
3. Lambda Function - if spam return 1, else return 0 (for ham) - ternary operators : [lambda x : value expression else value]

```
# creating numerical repersentation of category - one hot encoding
df_balanced['spam'] = df_balanced['Category'].apply(lambda x:1 if x=='spam' else 0)
```

```
# displaying data - spam -1 , ham-0
df_balanced.sample(4)
```

	Category	Message	spam
4565	ham	Tell me again what your address is	0
4215	ham	Ard 530 like dat lor. We juz meet in mrt stati...	0
120	spam	PRIVATE! Your 2004 Account Statement for 07742...	1
4278	ham	I'm glad. You are following your dreams.	0

2. Do train-test split

- split dataset into 80-20 ratio with 80% train and remaing as test
- for eveness of data we will use `stratify` agrument which ensures same ratio of both category is loaded for each case, even if one categoy has more training samples - prevents overfitting

Store our data in:

- `X_train, y_train` - traininge set(training_data and labels respectively)
- `X_test,, y_test` - testing set(testing_data and labels)

```
# loading train test split
from sklearn.model_selection import train_test_split
```

```
X_train, X_test, y_train, y_test = train_test_split(df_balanced['Message'], df_balanced['spam
                                                    stratify = df_balanced['spam'])
```

```
# check for startification
y_train.value_counts()
```

```
0    560
1    560
Name: spam, dtype: int64
```

```
560/560
```

```
1.0
```

```
y_test.value_counts()
```

```
0    187
1    187
Name: spam, dtype: int64
```

```
187/187
```

```
1.0
```

```
#We use Support Vector classifier as a classifier
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix
```

```
#training the classifier using X_Train and y_train
clf = SVC(kernel = 'linear').fit(X_train,y_train)
clf.predict(X_train)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-45-6cc4f6b1b835> in <module>
      1 #training the classifier using X_Train and y_train
----> 2 clf = SVC(kernel = 'linear').fit(X_train,y_train)
      3 clf.predict(X_train)

                              4 frames
/usr/local/lib/python3.7/dist-packages/pandas/core/series.py in __array__(self, dtype)
    855                dtype='datetime64[ns]')
    856         """
--> 857         return np.asarray(self._values, dtype)
    858
    859     # ------------------------------------------------------------------

ValueError: could not convert string to float: 'You have registered Sinco as Payee. Log
in at icicibank.com and enter URN  &lt;#&gt;  to confirm. Beware of frauds. Do NOT
share or disclose URN to anyone.'
```

***-> Almost similar, means data is downsampled now ***