Machine Learning Engineer Nanodegree

Capstone project

Bahaa Ibrahim Hassan
October 2nd,2019

I - Definition

---

1 - Project Overview:-

Email has become one of the most important forms of communication. In 2014, there are estimated to be 4.1 billion email accounts worldwide, and about 196 billion emails are sent each day worldwide. Spam is one of the major threats posed to email users. In 2013, 69.6% of all email flows were spam, phishing schemes, which can access and disrupt the receiver’s computer system . Current spam techniques could be paired with content-based spam filtering methods to increase effectiveness. Content-based methods analyze the content of the email to determine if the email is spam. The goal of our project was to analyze machine learning algorithms and determine their effectiveness as content-based spam filters; the traditional programming technique will not suitable here because I do not know all types of words that lead to spam so I use machine learning algorithm to learn the structure of spam

messages and common words used in it and predict if new one become spam or not . Other similar problem of classification is (Predict the Quality of Wines).

https://www.freecodecamp.org/news/using-data-science-to-understand-what-makes-wine-taste-good-669b496c67ee/

2- Problem Statements :-

The machine learning system is given labeled data from a training data set. In our project, the labeled training data are a large set of emails that are labeled spam or ham. During the training process, the classifier (the part of the machine learning system that actually predicts labels of future emails) learns from the training data by determining the connections between the features of an email and its label , after doing training I will test the model to see if it does well or not . I will use the different approach, based on word count and term-frequency inverse document frequency transform to classify the messages

The strategy that I will follow is :-

1-Download and pre-process the SMS Spam Collection v.1 dataset.
2. Test and find best approach (word count or tf-idf vectorizer) to classify the messages.
3. Selection of approach and splitting the dataset into training and testing data.
4. Initialize various classifiers and train it.
5. Evaluate the classifiers and finding best the model for a dataset.

The way to do that is as Follow :--

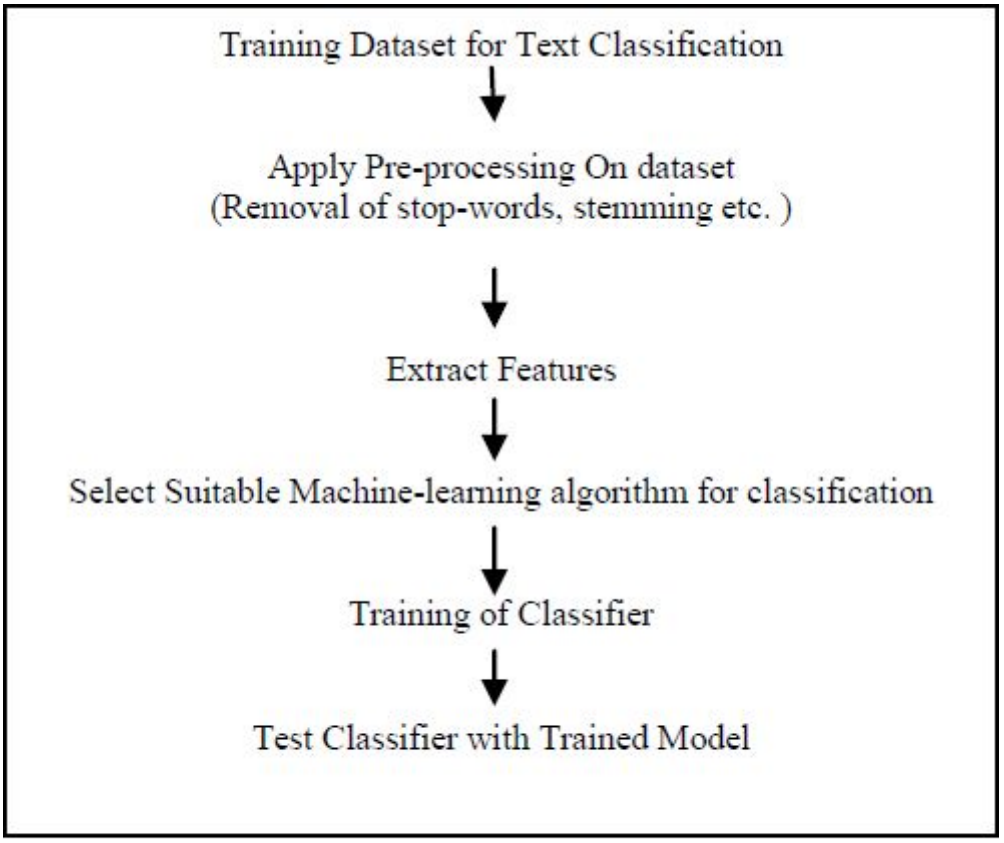


Figure 2: Strategy for Text classification

3- Metrics :-

1-Accuracy:-

Accuracy simply measures how often the classifier makes the correct prediction. It's the ratio between the number of correct predictions and the total number of predictions (the number of data points in the test set) :

Accuracy = *Correct predictions / Total data points*

2- F-Score:

which is defined as the harmonic mean of precision and recall. I found F1_score as appropriate to use as report metric in order to have a good idea of how the algorithm is :

F1 = (2 P * R) /(P + R)

I will use F-Score because F1 Score might be a better measure to use if we need to seek a balance between Precision and Recall AND there is an uneven class distribution (large number of Actual Negatives).

3- ROC and AUROC

Another interesting metric is the ROC curve (standing for Receiver Operating Characteristic), defined with respect to a given class (that we will denote C in the following).

Suppose that for a given point x, we have a model that outputs the probability that this point belongs to C: P(C | x). Based on this probability, we can define a decision rule that consists in saying that x belongs to class C if and only if $P(C \mid x) \geq T$, where T is a given threshold defining our decision rule. If T=1, a point is labelled as belonging to C only if the model is 100% confident it does. If T=0, every points are labelled as belonging to C.Each value of the threshold T generates a point (false positive, true positive) and, then, the ROC curve is the curve described by the ensemble of points generated when T varies from 1 to 0. This curve starts at point (0,0), ends at point (1,1)

and is increasing. A good model will have a curve that increases quickly from 0 to 1 (meaning that only a little precision has to be sacrificed to get a high recall).

II- Analysis

Below describes how the data was gathered, which features were selected, and which algorithms were explored. Finally, I outline the benchmark used to evaluate the performance of the trading strategy .

4.1- Data exploration :

The dataset used for this project is SMS Spam Collection dataset originates from the UCI Machine Learning Repository. This dataset has been collected from free or free for research sources at the Internet. The collection is composed of just one text file, where each line has the correct class followed by the raw message.

Dataset does not require any kind of cleaning, wrangling and there is no null value in any column .

I split to classes to know common things between them and explanation will covered in next section .

In [77]:
```
data=data.replace(['ham','spam'],[0,1])
```

In [78]:
```
# Collecting ham messages
ham=data[data.v1==0]
ham_count  = pd.DataFrame(pd.value_counts(ham['Count'],sort=True).sort_index())
print ("Number of ham messages in data set :", ham['v1'].count(),'\n')
print ("Ham Count value :", ham_count['Count'].count(),'\n')
print(ham.head())
```

```
Number of ham messages in data set : 4825

Ham Count value : 272

   v1                                                 v2  Count
0   0  Go until jurong point, crazy.. Available only ...    111
1   0                      Ok lar... Joking wif u oni...     29
3   0  U dun say so early hor... U c already then say...     49
4   0  Nah I don't think he goes to usf, he lives aro...     61
6   0  Even my brother is not like to speak with me. ...     77
```

In [79]:
```
# Collecting spam messages
spam=data[data.v1 == 1]
spam_count=pd.DataFrame(pd.value_counts(spam['Count'],sort=True).sort_index())
print("Number of spam messages in data set :" ,spam['v1'].count(),'\n')
print("Spam count value :",spam_count['Count'].count(),'\n')
print(spam.head())
```

```
Number of spam messages in data set : 747

Spam count value : 122

    v1                                                 v2  Count
2    1  Free entry in 2 a wkly comp to win FA Cup fina...    155
5    1  FreeMsg Hey there darling it's been 3 week's n...    148
8    1  WINNER!! As a valued network customer you have...    158
9    1  Had your mobile 11 months or more? U R entitle...    154
11   1  SIX chances to win CASH! From 100 to 20,000 po...    136
```

4.2- Data Visualization :

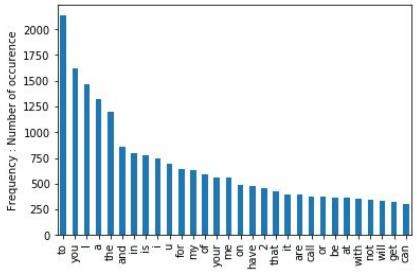
There are a lot of common words in the same class of messages , Since each category of messages has some similar type of keywords i.e. in spam message words like (Free, Winner, win, won, award, congrats, congratulation, selected, urgent, and Cash) it is easier to classify them and in ham message some words like (go , ok ,…)so I will display graph that demonstrate most frequency words that appear in data set to make it easy to understand which words common in both two types and avoid them in prediction .so I will use Techniques that are based on word count and term-frequency inverse document-frequency (tf-idf) transform .

In [74]:
```
data.groupby('v1').describe()
```

Out[74]:

	v2			
	count	unique	top	freq
v1				
ham	4825	4516	Sorry, I'll call later	30
spam	747	653	Please call our customer service representativ...	4

In [75]:
```
# Visualization of the most freq words in the dataset
cunt1=Counter(" ".join(data['v2']).split()).most_common(30)
df1 =pd.DataFrame.from_dict(cunt1)
df1=df1.rename(columns={0:'word',1:'count'})
fig=plt.figure()
ax=fig.add_subplot()
df1.plot.bar(ax=ax,legend=False)
xticks = np.arange(len(df1['word']))
ax.set_xticks(xticks)
ax.set_xticklabels(df1['word'])
ax.set_ylabel('Frequency : Number of occurence')
plt.show()
```



4.3 Algorithms and Techniques :

1- SVM :

SVM is an algorithm of supervised learning which is used for both classification and regression. In most of cases it is used as a classifier. It is used for many NLP tasks in text classification. Here each data item is plot as n-dimensional space where n represents no. of features extracted. SVM comes with a unique feature that it includes both types: positive & negative training sets. It represents each text-document as a vector where its dimensions are the no. of different keywords. But if the size of text document is large then there will be a number of dimensions in hyper-space which may increase computational cost of the process .

2- Naive Bayes :

Naive bayes is a probabilistic classifier which do not work on any single algorithm rather it work on a family of algorithm that all work on a single principle of classification. All of the extracted features using this classifier are independent of each other. The advantage of using this classifier is that it work good on both numeric as well as textual data and moreover it is easier to implement. The disadvantage of this classifier is that its performance gets poorer when the extracted features are correlated to each other .

3- Logistic Regression :

Logistic Regression is another way to determine a class label, depending on the features. Logistic regression takes features that can be continuous (for example, the count of words in an email) and translate them to discrete values (spam or not spam).

4- Decision Tree :

Decision tree is another classifier algorithm which is widely used for the purpose of classification. It works on a series of some test questions & conditions applied on it. It is represented in a tree form of structure where the branches of tree represents “weight” and each leaf is a different “class”. Decision tree is good in learning disjunction expressions and can handle noisy data. But training a decision-tree may act as an expensive process. If there is a mistake in very higher level then there will be flaws in whole sub tree and whole structure may act as invalid .

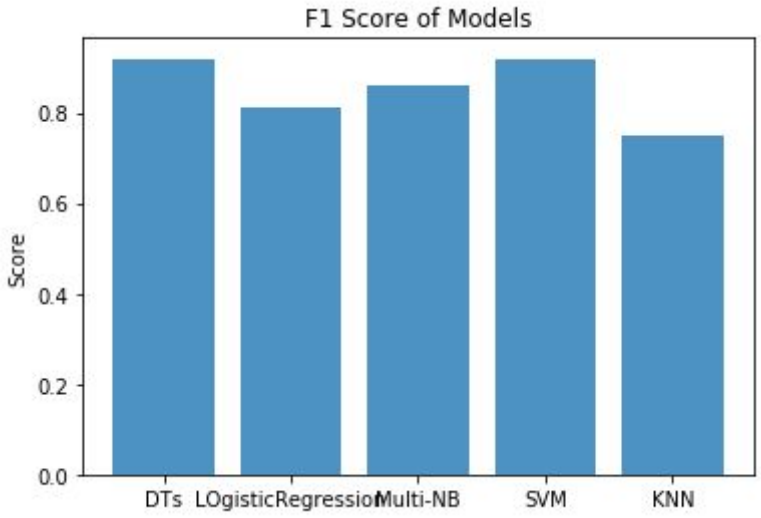
5- K-Nearest Neighbors (KNN):

KNN is another classification algorithm, it predicts label of any data point by looking at the ‘K’ closest labeled data points by taking a majority vote if K=3 will classify according to the nearest 3 points .

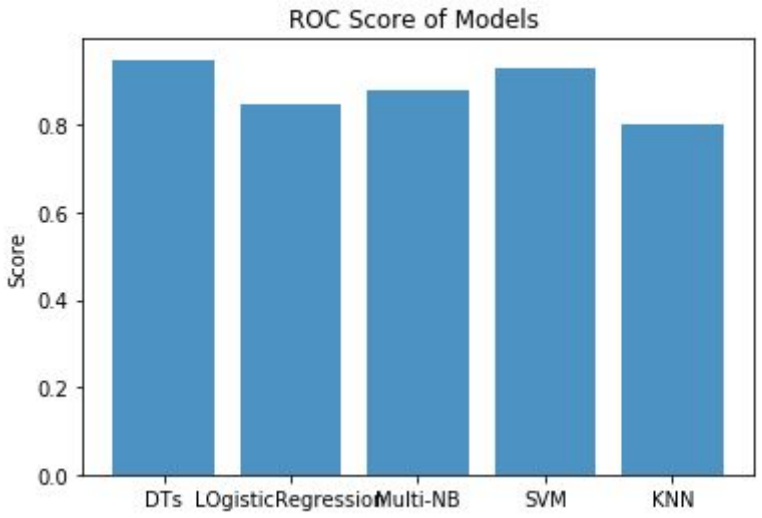
```
DecisionTree Accuracy: 0.9157894736842104
LogisticRegression Accuracy: 0.8134556574923548
NaiveBayse Accuracy: 0.8605341246290802

/home/bahaa/.local/lib/python3.6/site-packages/sklearn/linear_model/logistic.py:432: FutureWarning: Default solver
will be changed to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)

SVC Accuracy: 0.9192200557103063
KNN Accuracy: 0.7483870967741935
```



```
DecisionTree Accuracy: 0.9492577815333955
LogisticRegression Accuracy: 0.8469196191339042
NaiveBayse Accuracy: 0.879165178454757
/home/bahaa/.local/lib/python3.6/site-packages/sklearn/linear_model/logistic.py:432: FutureWarning: Default solver
will be changed to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
SVC Accuracy: 0.9306892526417576
KNN Accuracy: 0.8024170013328571
```



4.4- BenchMark model:

The result shows that Naive Bayes work better on the dataset with an accuracy_score of 86% for F1 Score and 87% for ROC score .

III- Methodology

As specified before dataset does not require any sort of cleaning. Column “v1” contains categorical value(class labels) ham and spam since classifier require numeric value ham is replaced with 0 and spam is replaced with 1 ,the only things I do here is to ensure that message body does not contain any type of Regular Expression the type of it found in Wikipedia

(https://en.wikipedia.org/wiki/Regular_expression) Other thing I do is Data processing To train our classifier we need to use term frequency– inverse document frequency (tf–idf), it is a numerical statistic that is intended to reflect how important a word is to a document in a corpus. It is used as a weighting factor in information retrieval, text mining, and user modelling. $tf.idf(t,d)=tf(t,d)\times idf(t)$ The tf-idf value increases proportionally to the number of times a word appears in the document but is offset by the frequency of the word in the corpus, which helps to adjust for the fact that some words appear more frequently in general. Remove stop words , all details about it found here (https://pythonspot.com/nltk-stop-words/) This's sample from our data after process :

```
# change capital words  to lower case

processed=processed.str.lower()
processed.head()
```

```
0    go until jurong point crazy available only in ...
1                            ok lar joking wif u oni
2    free entry in numbr a wkly comp to win fa cup ...
3          u dun say so early hor u c already then say
4    nah i don t think he goes to usf he lives arou...
Name: v2, dtype: object
```

I used 5 classifiers "DecisionTree,LogisticRegression,Naive Bayse, SVC , KNN" And initialize all of them

- For the 'svc' i choose 'linear' kernel because this is a classification problem and we have linear data
- For the 'KNN' first of all i set "n_neighbors=10" and it's F1 score almost = 22% it was so bad so i have increased the n_neighbors value to 100 .

In [33]:
```
# use models  form sklearn to predict

from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import MultinomialNB
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import f1_score
from sklearn.model_selection import learning_curve,validation_curve
from sklearn.model_selection import cross_val_score
from sklearn.metrics import roc_auc_score

objects=( 'DTs', 'LOgisticRegression','Multi-NB', 'SVM','KNN')
```

In [34]:
```
# Training and predicting

# function to train model
def train_classifier(clf, X_train, y_train):
    clf.fit(X_train, y_train)

# function to predict features
def predict_labels(clf, features):
    return (clf.predict(features))
```

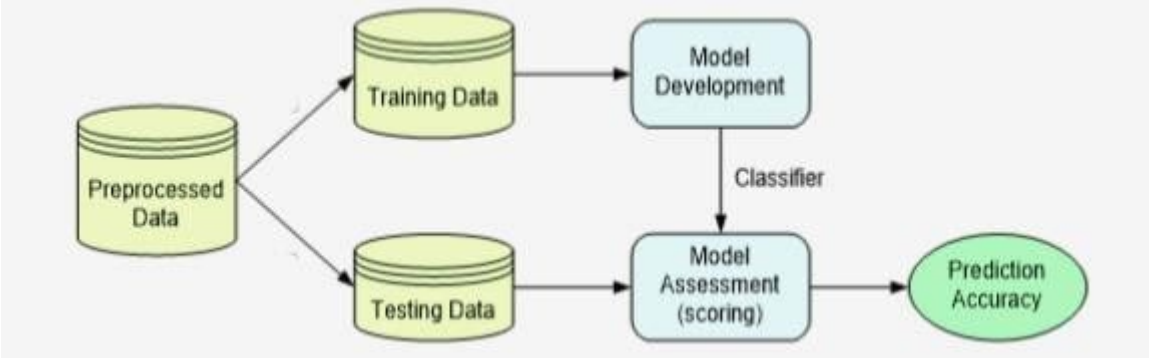
In [35]:
```
# initialize Classifiers
DecisionTree = DecisionTreeClassifier(random_state=42)
LogisticRegression= LogisticRegression()
NaiveBayse = MultinomialNB(alpha=1.0,fit_prior=True)
SVC = SVC(kernel = 'linear')
KNeighborsClassifier=KNeighborsClassifier(n_neighbors=100)


clf = [DecisionTree,LogisticRegression,NaiveBayse,SVC,KNeighborsClassifier]
names =['DecisionTree','LogisticRegression','NaiveBayse','SVC','KNN']
```

IV- Implementation And Results

All classification algorithms needs all the data to be numeric, this was handled during the preprocessing step and it was mentioned in the above section. Pre-processing is the actual first and most important step that needs to be handled carefully and with patience as it takes 80% of the work time on the project, preprocessing the data well leads to a better results in the prediction of any model. After analyzing the data to explore trends or it's characteristics, I have preprocessed the data to be ready and suitable for modeling and prediction. The final output of the preprocessing phase is a clean dataset that's labeled numerically and encoded without any unnecessary feature that doesn't contribute to the prediction of the target variable. The dataset is then divided into 75% for training and 25% testing data. The classifiers are initialized for Naive Bayes, Decision Tree, Logistic Regression, and SVM, and the training for the model starts, then the testing outputs the final predictions and

compares them with the test labels to measure the accuracy using the accuracy_score . I think this problem is a straight forward in this case data set is very good and simple and message it self also good I make some pre-processing on it but isn't cause me a problem .

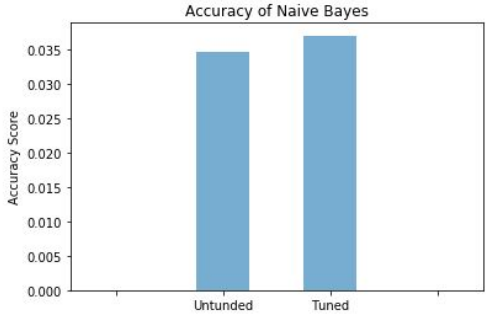


I mentioned the accuracy score for F1 & ROC before with screenshots from the code and it's result .

Accuracy after and before Tuning :

The process is Remove Stop words from model make it faster and more accurate than un-tuned one .

```
# ploating data for Accuracy Score
# ploating data for Accuracy of Models between 1.00 - 0.90 for better visualization
objects = ('','Untunded', 'Tuned','')
y_pos = np.arange(4)
y_val = [0,0.03470790378,0.037062937063,0 ]
plt.bar(y_pos,y_val, align='center',width = 0.5, alpha=0.6)
plt.xticks(y_pos, objects)
plt.ylabel('Accuracy Score')
plt.title('Accuracy of Naive Bayes')
plt.show()
```



In Result , it is possible to say that the result is trustworthy and

the project is finished under satisfaction the precision of 88%. The only thing negative is the time it takes for training, maybe 10 to 15 seconds but the output has correct labels and that is the most important .

I used cross validation score to validate the chosen classifier “the solution” actually i did it for all classifiers that i used in the project

```
In [35]: # initialize Classifiers
DecisionTree = DecisionTreeClassifier(random_state=42)
LogisticRegression= LogisticRegression()
NaiveBayse = MultinomialNB(alpha=1.0,fit_prior=True)
SVC = SVC(kernel = 'linear')
KNeighborsClassifier=KNeighborsClassifier(n_neighbors=100)


clf = [DecisionTree,LogisticRegression,NaiveBayse,SVC,KNeighborsClassifier]
names =['DecisionTree','LogisticRegression','NaiveBayse','SVC','KNN']
scores=[]
for j in range(0,5):
    scores.append(cross_val_score(clf[j], X_train, y_train, cv=5))
    print ("{} : {}".format(names[j],scores[j]))

DecisionTree : [0.97013142 0.97966507 0.98086124 0.96646707 0.96886228]
LogisticRegression : [0.96415771 0.96889952 0.95933014 0.95329341 0.95808383]
NaiveBayse : [0.96774194 0.96650718 0.95813397 0.96287425 0.96047904]

/home/bahaa/.local/lib/python3.6/site-packages/sklearn/linear_model/logistic.py:432: FutureWarning: Default solver
will be changed to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)

SVC : [0.98446834 0.98684211 0.98086124 0.98443114 0.97724551]
KNN : [0.95937873 0.94736842 0.93899522 0.95449102 0.95329341]
```

Justification :

The result of the models was very satisfactory, comparing to the benchmark. All the final models worked better than benchmark even with half the dataset, it is possible to say that the result is trustworthy and the project is finished under satisfaction the precision of 88% . I think that the graph of accuracy model provides a good comparison with results for this situation all model is better than benchmark model .

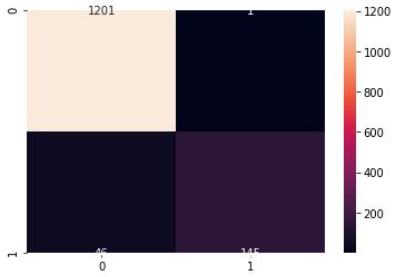
V - Conclusion

In this project, we tried to analyse different methods to identify spam messages. We used different approach, based on word count and term-frequency inverse document-frequency transform to classify the messages. Since test data have very high meaning for human and very difficult for the machine to understand, the biggest challenge was to analyse the test data and convert it into some meaningful numeric data without disturbing the relation between categories. Best method to convert the test data into meaningful numeric data is tf-idf vectorizer. This was the most interesting part of the whole project to convert huge amount of text data into numeric data. Best result was generated using tf-idf vectorizer with SVM and DT classifier, both achieved accuracy approx. 91%, which is a satisfactory result .

- **Visualization**
  I used confusion matrix. A confusion matrix is a table that is often used to describe the performance of a classification model (or "classifier") on a set of test data for which the true values are known.

```
In [43]: from sklearn.metrics import confusion_matrix
         import seaborn as sns
         %matplotlib inline
         best_clf=NaiveBayse
         pred = best_clf.predict(X_test)
         sns.heatmap(confusion_matrix(y_test, pred), annot = True, fmt = '')
```

```
Out[43]: <matplotlib.axes._subplots.AxesSubplot at 0x7f8be3014400>
```



The process used for this project can be summarized using the following steps:
1 - Getting the dataset
2- Exploratory data analysis.
3- Data preprocessing to prepare it for modeling.
4-Build benchmark model and evaluate ie .
5 - building proposed model and evaluate it.
6- model validation.

I think preprocessing is interesting and very important also it improve the model accuracy and we show that when make tuned of remove stop words the model become better .

Improvement :

Most Models work good so I think if we use ANN will achieve results Better than one that I have now , because Artificial neural-networks work on the concept of human brain consisting neurons. It consists of a layered arrangement of neurons where the input vectors are converted into the same form of output. ANN is considered to be a good classifier because it can better handle multiple categories and work well on it. It supports fast testing phase.ANN is when combined with naive bayes algorithm it comes up with a new idea which is called knowledge based neural networks and this is more efficient in handling noisy data .

References :

1- https://en.wikipedia.org/wiki/Natural_language_processing
2- https://www.nltk.org/book/ch06.html
3- https://towardsdatascience.com/handling-imbalanced-datasets-in-machine-learning-7a0e84220f28
4- https://en.wikipedia.org/wiki/Mean_squared_error
5- https://archive.ics.uci.edu/ml/datasets/sms+spam+collection
6- https://en.wikipedia.org/wiki/Regular_expression
7- https://www.oreilly.com/ideas/evaluating-machine-learning-models/page/3/evaluation-metrics