ABSTRACT

Machine Learning is the field of study that offers computers the potential to grasp knowledge without being explicitly programmed. ML is among the most exciting technologies that one would have ever encounter. because it is clear from the name, it gives the computer that which makes it more almost like humans: the power to find out . Machine learning is actively getting used today, perhaps in more places than one would expect .In my project I used the machine learning techniques during which the info set provided is split into to parts first is test set and other is train set. The algorithm uses 80% data-set to train and uses 20% data-set to test.

In this project I used a binary classified dataset to predict the sentiment of the user, positive or negative, after analyzing the users tweet. The Project is implemented in python and features a Naive Bayes Classifier. After the implementation of the project I can conclude that my model can predict sentiments of the users with 80% accuracy.

CONTENTS

1. INTRODUCTION

Sentiment analysis is the process of using natural language processing, text analysis, and statistics to analyze customer sentiment. The best businesses understand the sentiment of their customers—what people are saying, how they're saying it, and what they mean. Customer sentiment can be found in tweets, comments, reviews, or other places where people mention your brand. Sentiment Analysis is the domain of understanding these emotions with software, and it's a must-understand for developers and business leaders in a modern workplace.

As with many other fields, advances in deep learning have brought sentiment analysis into the foreground of cutting-edge algorithms. Today we use natural language processing, statistics, and text analysis to extract, and identify the sentiment of words into positive, negative, or neutral categories.

1.1 PURPOSE

NLP(Natural Language Processor) works by converting words into numbers.

These numbers are then used to train on AI/ML model to make predictions.

Predictions could be sentiment inferred from social media posts and product reviews.

AI/ML based sentiment analysis is crucial for companies to predict if the consumer is happy or not.

Process could be done automatically instead of manually through thousands of tweets and customer reviews.

1.2 SCOPE OF THE PROJECT

Sentiment analysis for brand monitoring

One of the most well documented uses of sentiment analysis is to get a full 360 view of how your brand, product, or company is viewed by your customers and stakeholders. Widely available media, like product reviews and social, can reveal key insights about what your business is doing right or wrong. Companies can also use sentiment analysis to measure the impact of a new product, ad campaign, or consumer's response to recent company news on social media. Private companies like Unamo offer this as a service.

Sentiment analysis for customer service

Customer service agents often use sentiment or intent analysis to automatically sort incoming user email into "urgent" or "not urgent" buckets based on the sentiment of the email, proactively identifying frustrated users. The agent then directs their time toward resolving the users with the most urgent needs first. As customer service becomes more and more automated through machine learning, understanding the sentiment and intent of a given case becomes increasingly important.

Sentiment analysis for market research and analysis

Sentiment analysis is used in business intelligence to understand the subjective reasons why consumers are or are not responding to something (e.x. why are consumers buying a product? What do they think of the user experience? Did customer service support meet their expectations?). Sentiment analysis can also be used in the areas of political science, sociology, and psychology to analyze trends, ideological bias, opinions, gauge reactions, etc.

1.3 ADVANTAGES OF THE SYSTEM

Upselling opportunities

Happy customers are more likely to be receptive to upselling. With sentiment analysis, you can easily identify your happiest customers.

This helps you recognise chatters who might be receptive to spending more, as well as avoiding upsetting disgruntled customers with any unwelcome sales pitches.

Agent Monitoring

Sentiment analysis gives you a clear overview of customer satisfaction, agent by agent. This means you can keep an eye on the quality of service each team member is offering customers, as well as their more subtle ability to create happy customers.

Adaptive customer service

Human agents are great at providing flexible service, but it can be difficult to identify the best approach for each customer. With sentiment analysis, it's easier for your team to adapt their service to the mood of the customer early on.

2. Literature Survey

2.1 Existing System

Sentiment analysis is done using algorithms that use text analysis and natural language processing to classify words as either positive, negative, or neutral. This allows companies to gain an overview of how their customers feel about the brand.

Algorithms used in Sentiment Analysis

Implementing sentiment analysis in your apps is as simple as calling our REST API. There are no servers to setup, or settings to configure. Sentiment analysis can be used to quickly analyze the text of research papers, news articles, social media posts like tweets and more.

Social Sentiment Analysis is an algorithm that is tuned to analyze the sentiment of social media content, like tweets and status updates. The algorithm takes a string, and returns the sentiment rating for the "positive," "negative," and "neutral." In addition, this algorithm provides a compound result, which is the general overall sentiment of the string**.**

Input Exaple:

```
{"sentenceList": [
"I like double cheese pizza",
"I love black coffee and donuts",
"I don't want to have diabetes"]}
```

Output Example:

```
[{
"positive": 0.455,
"negative": 0,
"sentence": "I like double cheese pizza",
"neutral": 0.545,
"compound": 0.3612
},
{
"positive": 0.512,
"negative": 0,
"sentence": "I love black coffee and donuts",
"neutral": 0.488,
"compound": 0.6369
},
{
"positive": 0,
"negative": 0.234,
"sentence": "I don't want to have diabetes",
"neutral": 0.766,
"compound": -0.0572
}]
```

Existing approaches to sentiment analysis can be grouped into three main categories: knowledge-based techniques, statistical methods, and hybrid approaches.

Knowledge-based techniques classify text by affect categories based on the presence of unambiguous affect words such as happy, sad, afraid, and bored. Some knowledge bases not only list obvious affect words, but also assign arbitrary words a probable "affinity" to particular emotions.

Statistical methods leverage elements from machine learning such as latent semantic analysis, support vector machines,"bag of words", "Pointwise Mutual Information" for Semantic Orientation, and deep learning**.**

Challenges faced by existing systems

Sentiment analysis runs into a similar set of problems as emotion recognition does – before deciding what the sentiment of a given sentence is, we need to figure out what “sentiment” is in the first place. Is it categorical, and sentiment can be split into clear buckets like happy, sad, angry Or is it dimensional, and sentiment needs to be evaluated on some sort of bi-directional spectrum.

In addition, There are multiple layers of meaning in any human-generated sentence. People express opinions in complex ways; rhetorical devices like sarcasm, irony, and implied meaning can mislead sentiment analysis.

Current thinking in sentiment analysis happens in a categorical framework: sentiment is analyzed as belonging to a certain bucket, to a certain degree. For example, a given sentence may be 45% happy, 23% sad, 89% excited, and 55% hopeful. These numbers don’t add up to 100.

Another challenge in sentiment analysis is deciding how to train the model you’d like to use. There are a number of pre-trained models available for use in popular Data Science languages. These modules are quick, but for the best long term results own model should be implemented.

Getting access to labeled training data for sentiment analysis can be difficult, but it’s key to building models.

2.2 Proposed System

In this project, we will train a Naive Bayes classifier to predict sentiment from thousands of Twitter tweets. This project could be practically used by any company with social media presence to automatically predict customer's sentiment (i.e.: whether their customers are happy or not). The process could be done automatically without having humans manually review thousands of tweets and customer reviews.

We will apply Python Libraries to import and Visualize Dataset. Perform exploratory data analysis and Evaluate the performance of trained Naive Bayes Classifier Model using confusion matrix.

3. Software requirement specification

3.1 Functional Requirements

- A functional requirement defines a function of a system or its component.

- Functional requirements may be calculations, technical details, data manipulation and processing and other specific functionality that define what a system is supposed to accomplish.

- The function requirements of this system are:

  1. Training the system according to csv files data.
  2. Testing on 25 percent portion of data.
  3. Implementation of Data Visualization in the from of graphs.

3.2 Non-Functional Requirements

- A non-functional requirement is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviors.

- The non-functional requirements in this system are:

  1. Performance

  2. Capacity

  3. Availability

  4. Reliability

  5. Serviceability

6. Efficiency

4 Requirement Analysis

4.1.1 Hardware Requirement

1) Processor: 2 or more Logical cores

2) Disk Space: 1 GB available space (Windows Installation is < 200 MB)

4.1.2 Software Requirement

1) Operating System : Windows 7 (64 bit) or later

2) Software: Jupyter Notebook/Anaconda

3) Programming Languages/Commands:

✓ **Python**

5 System Design

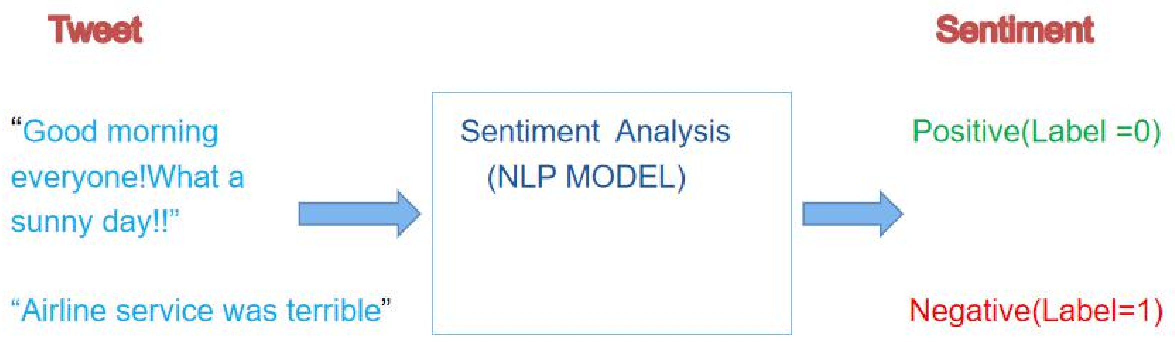
NLP(Natural Language Processor) works by converting words into numbers.

These numbers are then used to train on AI/ML model to make predictions.

Predictions could be sentiment inferred from social media posts and product reviews.

AI/ML based sentiment analysis is crucial for companies to predict if the consumer is happy or not.

Process could be done automatically instead of manually through thousands of tweets and customer reviews.


Tweet
"Good morning everyone!What a sunny day!!"
"Airline service was terrible"
Sentiment Analysis (NLP MODEL)
Sentiment
Positive(Label =0)
Negative(Label=1)

This is a binary classification problem, with Positive sentiments labeled as 0 and Negative sentiments labeled as 1.

Data Cleaning : Perform Data Cleaning using pandas on the Data-set to remove Stop-words. Stop-words : are words which do not provide any sentimental meaning. They're neutral words.

Example: The tweet " Beautiful start to the day with such a pleasant morning." becomes "Beautiful start such pleasant " after removing the stop-words.

Data Cleaning also includes performing inbuilt lowercase operation to remove redundancy. 'World' and 'world' are considered as same.

Tokenization : ML model does not accept strings, so first the cleaned data-set containing tweets is tokenized. The tweets are converted into numerical tokens, which are then ready to be fed into the model.

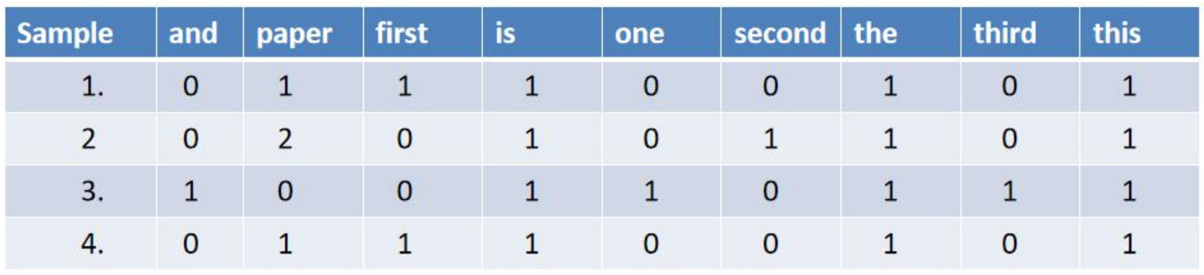
Sample :

1. This is the first paper.

2. This paper is the second paper.

3. And this is the third one.

4. Is this the first paper.



Sample	and	paper	first	is	one	second	the	third	this
1.	0	1	1	1	0	0	1	0	1
2	0	2	0	1	0	1	1	0	1
3.	1	0	0	1	1	0	1	1	1
4.	0	1	1	1	0	0	1	0	1

It takes all the unique words from all the samples . Then charts individual samples and if they have the word and how many times the sample has.

Sample 2 has 'paper' twice in it so it is charted as 2.

Creates tokenised/vectorised vector of a string.

Eg: 'This is the first paper' = 011100101

' Is this the first paper ' = 011100101

Creating the Model : We are creating a Naive Bayes Classifier Model. Theory and Intuition behind Naive Bayes

It is a classification technique based on Bayes' theorem.

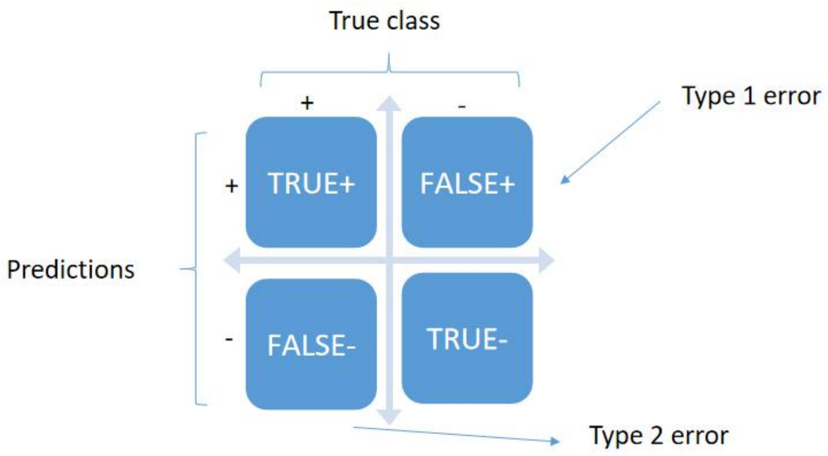
Bayes theorem provides a way of calculating the posterior probability, P(c|x), from P(c), P(x), and P(x|c). Naive Bayes classifier assume that the effect of the value of a predictor (x) on a given class (c) is independent of the values of other predictors. ... P(c) is the prior probability of class.

$$\underset{\text{Posterior Probability}}{\underbrace{P(c \mid x)}} = \frac{\overset{\text{Likelihood}}{\overbrace{P(x \mid c)}}\,\overset{\text{Class Prior Probability}}{\overbrace{P(c)}}}{\underset{\text{Predictor Prior Probability}}{\underbrace{P(x)}}}$$

$$P(c \mid X) = P(x_1 \mid c) \times P(x_2 \mid c) \times \cdots \times P(x_n \mid c) \times P(c)$$

Testing the Model : The efficiency and performance of the model is judged using a Confusion Matrix. confusion matrix, also known as an error matrix, is a specific table layout that allows visualization of the performance of an algorithm, typically a supervised learning one.

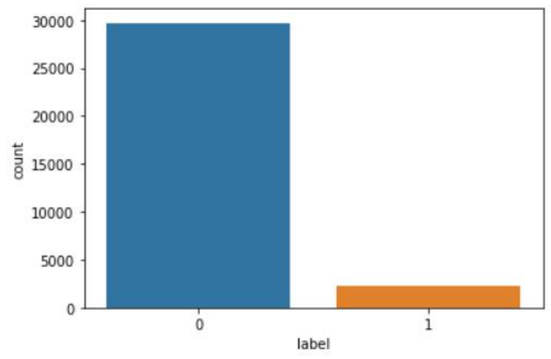
CONFUSION MATRIX



TRUE +
When model predicts the tweet is positive and true class of the tweet is also positive
TRUE –
When the model predicts the tweet is negative and the true class of the tweet is also negative
FALSE+ TYPE 1 error
When model predicts the tweet is positive but the true class of the tweet is negative
FALSE – TYPE 2 error
When model predicts tweet is negative but the True class of tweet is positive.

6 System Testing:

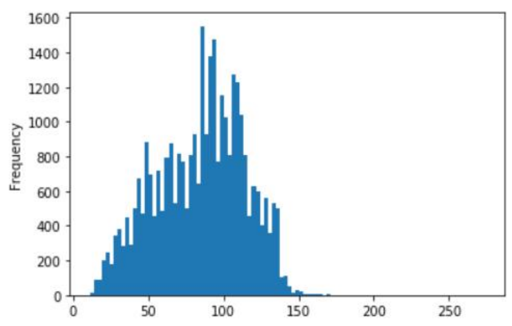
```
In [3]:  #plotting our dataset which has max label=0 ie positive tweets and a few label=1 ie negative tweets

         sns.countplot(tweets_df['label'], label= 'count')

Out[3]: <matplotlib.axes._subplots.AxesSubplot at 0x1ac533f9d88>
```



Plotting Negative and Positive tweets label1 and 0, its evident that our data-set contains more positive tweets than negative tweets.

```
In [6]:  #plotting a histogram based on length
         tweets_df['length'].plot(bins=100, kind = 'hist')

         tweets_df.describe() #shows min and max lenght of tweets, here min is 11 average is 84, also shows std & count

Out[6]:
```

	label	length
count	31962.000000	31962.000000
mean	0.070146	84.739628
std	0.255397	29.455749
min	0.000000	11.000000
25%	0.000000	63.000000
50%	0.000000	88.000000
75%	0.000000	108.000000
max	1.000000	274.000000



plotting a Histogram based on the length of tweets in our data set.

7 Project Planning:

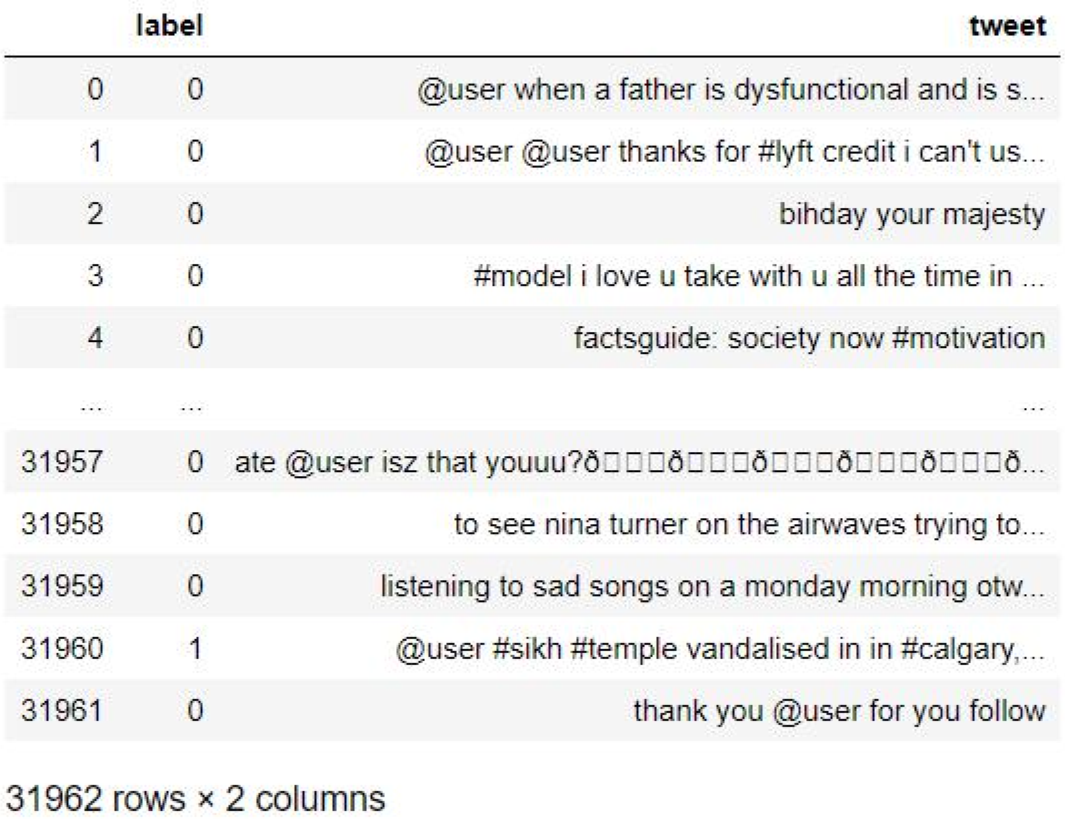
- Information is simply a lot for human processing and examination. There is simply a lot of information being delivered and without a robotized framework dependent on AI to gain from such frameworks, we are no place.

- It’s getting better with new deep learning systems, data engineering and pre-processing costs are being dropped.

- With the help of this project I can predict the sentiments of the consumer based on their tweets and the companies can know how customers are reacting to their product/services.

8 Implementation:

Data set:

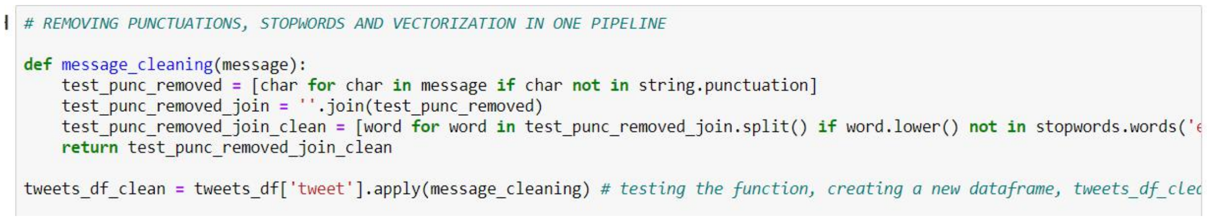


	label	tweet
0	0	@user when a father is dysfunctional and is s...
1	0	@user @user thanks for #lyft credit i can't us...
2	0	bihday your majesty
3	0	#model i love u take with u all the time in ...
4	0	factsguide: society now #motivation
...	...	...
31957	0	ate @user isz that youuu?ð□□□□ð□□□□ð□□□□ð□□□□ð□□□□ð...
31958	0	to see nina turner on the airwaves trying to...
31959	0	listening to sad songs on a monday morning otw...
31960	1	@user #sikh #temple vandalised in in #calgary,...
31961	0	thank you @user for you follow

31962 rows × 2 columns

Screenshot displaying Data-set consisting of tweets. There are two coloumns namely, label and the tweet. This is a real data-set and the users are anonymous to protect privacy.

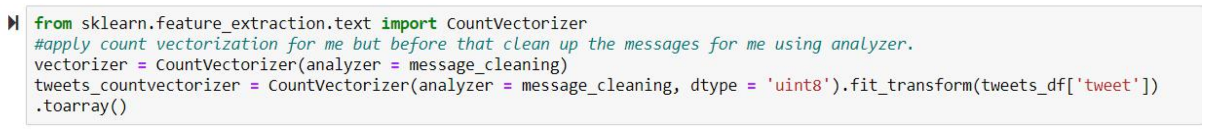
Data Cleaning :



```
# REMOVING PUNCTUATIONS, STOPWORDS AND VECTORIZATION IN ONE PIPELINE

def message_cleaning(message):
    test_punc_removed = [char for char in message if char not in string.punctuation]
    test_punc_removed_join = ''.join(test_punc_removed)
    test_punc_removed_join_clean = [word for word in test_punc_removed_join.split() if word.lower() not in stopwords.words('e
    return test_punc_removed_join_clean

tweets_df_clean = tweets_df['tweet'].apply(message_cleaning) # testing the function, creating a new dataframe, tweets_df_clea
```

Screenshot of code performing data cleaning.

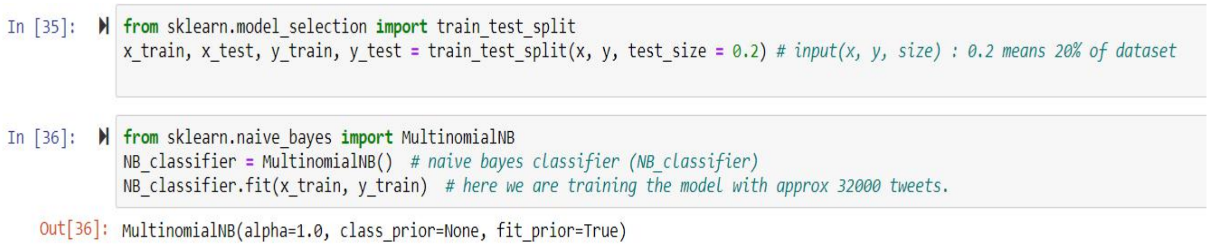
Tokenization:



```
from sklearn.feature_extraction.text import CountVectorizer
#apply count vectorization for me but before that clean up the messages for me using analyzer.
vectorizer = CountVectorizer(analyzer = message_cleaning)
tweets_countvectorizer = CountVectorizer(analyzer = message_cleaning, dtype = 'uint8').fit_transform(tweets_df['tweet'])
.toarray()
```

Screenshot of code calling Count Vectorizer to implement tokenization to the cleaned data-set.

Creating the Model :

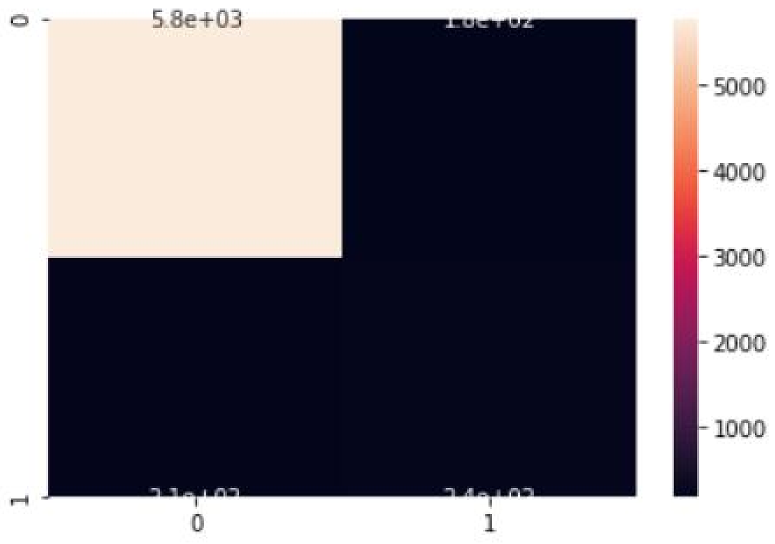


```
In [35]:  from sklearn.model_selection import train_test_split
          x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2) # input(x, y, size) : 0.2 means 20% of dataset

In [36]:  from sklearn.naive_bayes import MultinomialNB
          NB_classifier = MultinomialNB()  # naive bayes classifier (NB_classifier)
          NB_classifier.fit(x_train, y_train)  # here we are training the model with approx 32000 tweets.

Out[36]:  MultinomialNB(alpha=1.0, class_prior=None, fit_prior=True)
```

Screenshot of code creating a model using Naive Bayes Classifier. Tokenized tweets are fed to the model and the model splits the data-set to train using 80% and test usding 20% of the data set

Assessing the performance of the model:

```
Out[38]: <matplotlib.axes._subplots.AxesSubplot at 0x1acfb853888>
```



```
In [39]:  print(classification_report(y_test, y_predict_test))

              precision    recall  f1-score   support

           0       0.96      0.97      0.97      5939
           1       0.57      0.54      0.55       454

    accuracy                           0.94      6393
   macro avg       0.77      0.75      0.76      6393
weighted avg       0.94      0.94      0.94      6393
```

Performance of the model is assessed using a Confusion Matrix and a classification report displaying precision, recall, f1-score, support of the model

9 CONCLUSION

Sentiment analysis or opinion mining is an important topic in machine learning however, we are yet to analyze sentiment from texts very accurately because of the complexities in Language. In this project we tried to show the basic way of classifying tweets into positive or negative category using Naive Bayes as baseline and how language models are related to the Naive Bayes and can produce better results. We can further improve our classifier by trying to extract more features from the tweets, trying different kinds of features, tuning the parameters of the naïve Bayes classifier, or trying another classifier all together. The programming language used in the project is python.

By doing this project I enhanced my knowledge in the field of Machine Learning which if used properly for data analysis can benefit a lot .I further plan to improve my knowledge in this field and utilize it for different projects

REFERENCES

[1] Course Lesson(Coursera)

[2] Data Collection(Datahub.com)

[3] GeeksForGeeks.com

[4] Wikipedia.com

[5] Fuchun Peng. 2003, Augmenting Naive Bayes Classifiers with Statistical Language Models