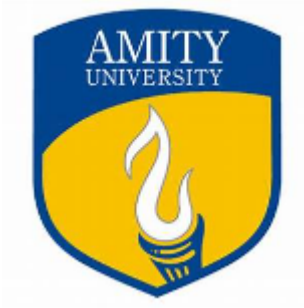
Lab Project File

Of

Fundamentals of Data Analytics (AIML301)



Submitted By:

Umang Mittal (A2305219025)

Dev Walia (A2305219017)

Class: 6CSE-1X

Submitted To:

Mr. Roshan Lal

Designation: Assistant Professor III

Department: CSE

Topic: SENTIMENT ANALYSIS MODEL ON SOCIAL MEDIA (TWITTER DATA)

Department of Computer Science and Engineering

Amity School of Engineering and Technology

Amity University, Noida, Uttar Pradesh

Session 2021-2022

ABSTRACT

In this project, we try to implement a **Twitter sentiment analysis model** that helps to overcome the challenges of identifying the sentiments of the tweets. The necessary details regarding the dataset are:

- The dataset provided is the **Sentiment140 Dataset** which consists of **1,600,000 tweets** that have been extracted using the Twitter API. The various columns present in the dataset are:
    - **target:** the polarity of the tweet (positive or negative)
    - **ids:** Unique id of the tweet
    - **date:** the date of the tweet
    - **flag:** It refers to the query. If no such query exists then it is “NO QUERY”
    - **user:** It refers to the name of the user that tweeted
    - **text:** It refers to the text of the tweet

INTRODUCTION

Sentiment analysis refers to identifying as well as classifying the sentiments that are expressed in the text source.

Tweets are often useful in generating a vast amount of sentiment data upon analysis. These data are useful in understanding the opinion of the people about a variety of topics.

Therefore we need to develop an Automated Machine Learning Sentiment Analysis Model in order to compute the customer perception. Due to the presence of non-useful characters (collectively termed as the noise) along with useful data, it becomes difficult to implement models on them.

With the help of this project, we aim to analyse the sentiment of the tweets provided from the Sentiment140 dataset by developing a machine learning pipeline involving the use of three classifiers (Logistic Regression, Bernoulli Naive Bayes, and SVM) along with using Term Frequency- Inverse Document Frequency (TF-IDF). The performance of these classifiers is then evaluated using accuracy and F1 Scores.

LITERATURE REVIEW

Sentiment Analysis is the process of 'computationally' determining whether a piece of writing is positive, negative or neutral. It's also known as opinion mining, deriving the opinion or attitude of a speaker.

Types of Sentiment Analysis

Sentiment analysis focuses on the polarity of a text (positive, negative, neutral) but it also goes beyond polarity to detect specific feelings and emotions (angry, happy, sad, etc), urgency (urgent, not urgent) and even intentions (interested v. not interested).

Depending on how you want to interpret customer feedback and queries, you can define and tailor your categories to meet your sentiment analysis needs. In the meantime, here are some of the most popular types of sentiment analysis:

- **Graded Sentiment Analysis** - If polarity precision is important to your business, you might consider expanding your polarity categories to include different levels of positive and negative:
    - Very positive
    - Positive
    - Neutral
    - Negative
    - Very negative

    This is usually referred to as graded or fine-grained sentiment analysis, and could be used to interpret 5-star ratings in a review, for example:

    Very Positive = 5 stars

    Very Negative = 1 star

- **Emotion detection** - Emotion detection sentiment analysis allows you to go beyond polarity to detect emotions, like happiness, frustration, anger, and sadness. Many emotion detection systems use lexicons (i.e. lists of words and the emotions they convey) or complex machine learning algorithms.

One of the downsides of using lexicons is that people express emotions in different ways. Some words that typically express anger, like bad or kill (e.g. your product is so bad or your customer support is killing me) might also express happiness (e.g. this is badass or you are killing it)

- **Aspect-based Sentiment Analysis** - Usually, when analysing sentiments of texts you'll want to know which particular aspects or features people are mentioning in a positive, neutral, or negative way. That's where aspect-based sentiment analysis can help, for example in this product review: "The battery life of this camera is too short", an aspect-based classifier would be able to determine that the sentence expresses a negative opinion about the battery life of the product in question
- **Multilingual sentiment analysis** - Multilingual sentiment analysis can be difficult. It involves a lot of pre-processing and resources. Most of these resources are available online (e.g. sentiment lexicons), while others need to be created (e.g. translated corpora or noise detection algorithms), but you'll need to know how to code to use them.

  Alternatively, you could detect the language in texts automatically with a language classifier, and then train a custom sentiment analysis model to classify texts in the language of your choice.

Why Is Sentiment Analysis Important?

Since humans express their thoughts and feelings more openly than ever before, sentiment analysis is fast becoming an essential tool to monitor and understand the sentiment in all types of data. Automatically analysing, such as opinions in survey responses and social media conversations, allows brands to learn what makes customers happy or frustrated so that they can tailor products and services to meet their customers' needs.

For example, using sentiment analysis to automatically analyse 4,000+ open-ended responses in your customer satisfaction surveys could help you discover why customers are happy or unhappy at each stage of the customer journey. Maybe you want to track brand sentiment so you can detect disgruntled customers immediately and respond as soon as possible. Maybe you

want to compare sentiment from one quarter to the next to see if you need to take action. Then you could dig deeper into your qualitative data to see why sentiment is falling or rising.

The overall benefits of sentiment analysis include:

- **Sorting Data at Scale** - Can you imagine manually sorting through thousands of tweets, customer support conversations, or surveys? There's just too much business data to process manually. Sentiment analysis helps businesses process huge amounts of unstructured data in an efficient and cost-effective way.
- **Real-Time Analysis** - Sentiment analysis can identify critical issues in real-time, for example, is a PR crisis on social media escalating? Is an angry customer about to churn? Sentiment analysis models can help you immediately identify these kinds of situations, so you can take action right away.
- **Consistent criteria** - It's estimated that people only agree around 60-65% of the time when determining the sentiment of a particular text. Tagging text by sentiment is highly subjective and influenced by personal experiences, thoughts, and beliefs.

By using a centralized sentiment analysis system, companies can apply the same criteria to all of their data, helping them improve accuracy and gain better insights.

Applications of Sentiment Analysis

The applications of sentiment analysis are endless. So, to help you understand how sentiment analysis could benefit your business, let's take a look at some examples of texts that you could analyse using sentiment analysis. Then, we'll jump into a real-world example of how Chewy, a pet supplies company, was able to gain a much more nuanced (and useful!) understanding of their reviews through the application of sentiment analysis.

Sentiment Analysis Examples

- Disliking horror movies is not uncommon. (negation, inverted word order)
- Sometimes I really hate the show. (adverbial modifies the sentiment)
- I love having to wait two months for the next series to come out! (sarcasm)
- The final episode was surprising with a terrible twist at the end (negative term used in a positive way)
- The film was easy to watch but I would not recommend it to my friends. (difficult to categorize)

We have used the following three classifiers in our project:

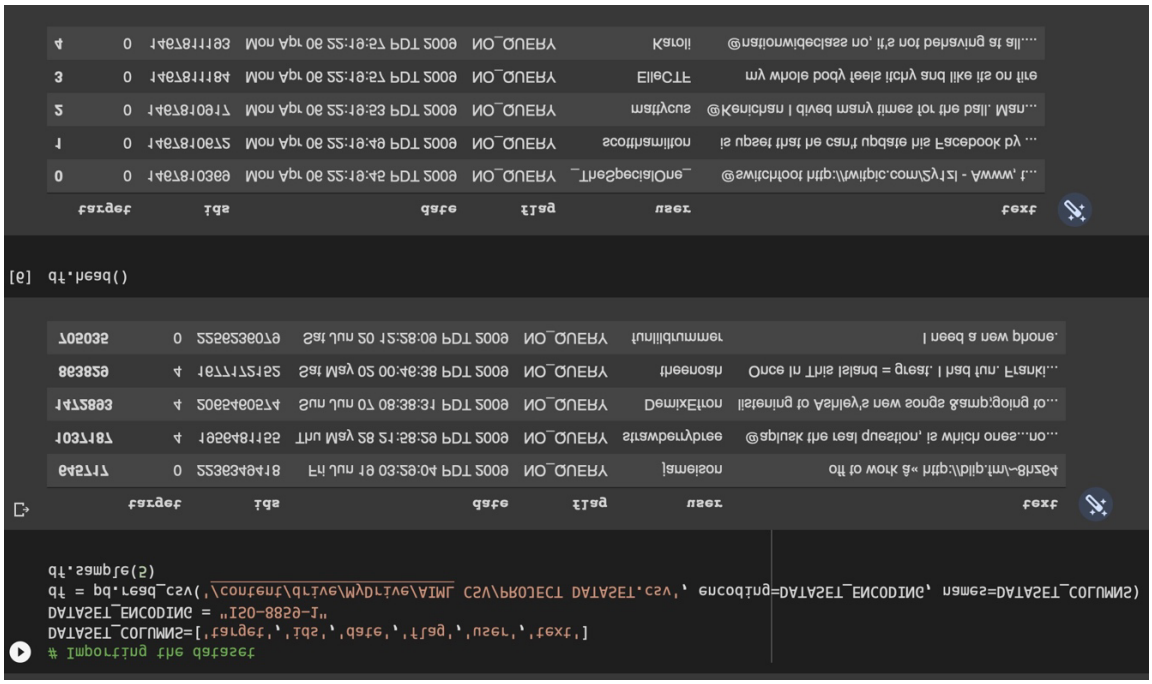
- **Logistic Regression:** It is a powerful supervised ML algorithm used for binary classification problems (when the target is categorical). The best way to think about logistic regression is that it is a linear regression but for classification problems. For example, to predict whether an email is spam (1) or (0) or whether the tumour is malignant (1) or not (0)
- **Bernoulli Naive Bayes:** It implements the naive Bayes training and classification algorithms for data that is distributed according to multivariate Bernoulli distributions; i.e., there may be multiple features but each one is assumed to be a binary-valued (Bernoulli, Boolean) variable.
- **Support Vector Machine (SVM):** It is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.

Relation between Logistic Regression and Bernoulli Naive Bayes

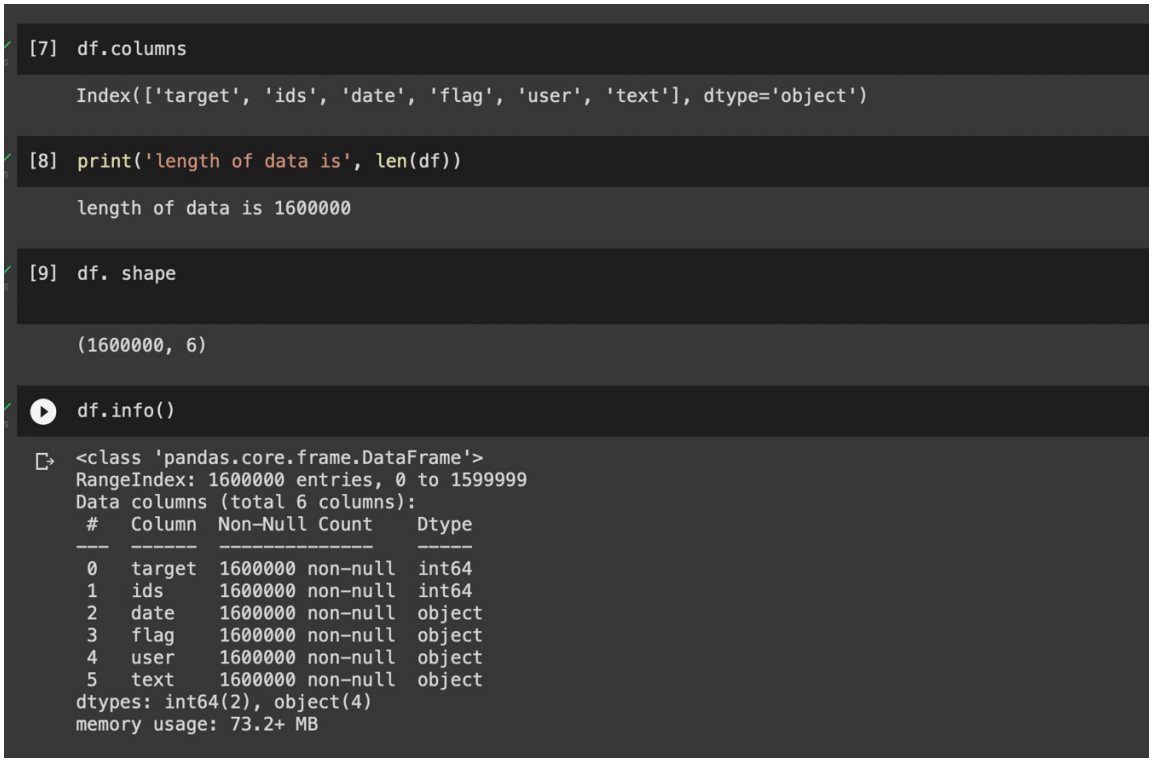
Logistic regression assumes the response is conditionally Bernoulli distributed, given the values of the features. The Bernoulli distribution has one parameter, the probability of the positive class. Logistic regression also specifies a specific functional form for this probability in terms of the features.

METHODOLOGY

- Import necessary dependencies
- Read and load the dataset



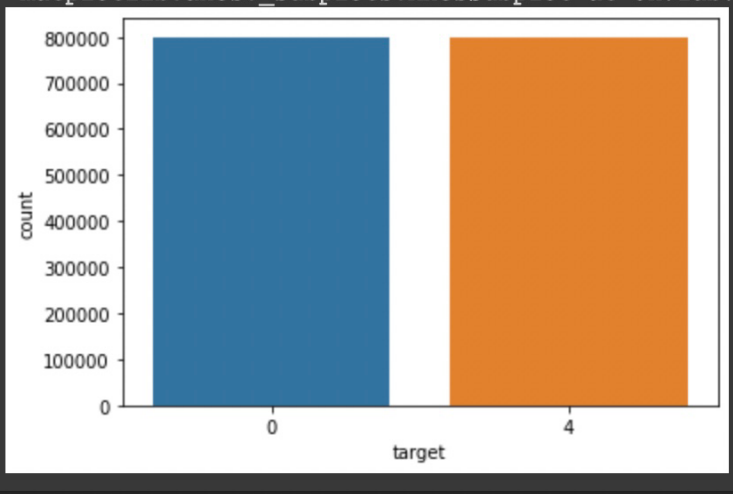
- Exploratory data analysis – 5 top records, features in data (columns) and their data types, length and shape of the dataset, data information, checking for null values, no. of rows and columns in the dataset, checking unique and total target values



- Data visualisation of target variables

```
import seaborn as sns
sns.countplot(x='target', data=df)
```

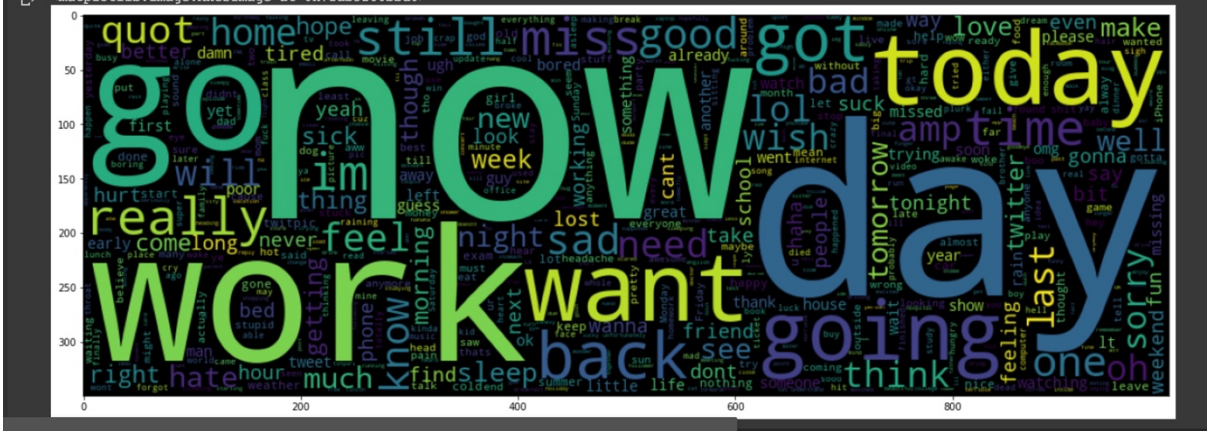
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fab73a49910>
```



- Data pre-processing – selecting text and target column from the dataset for further analysis, separating positive and negative tweets, taking 1/4th of the data so we can run on our machine easily and then combining positive and negative tweets, defining a set that contains all stop words in English, cleaning and removing above stop words from tweet text, cleaning and removing punctuations, repeating characters, URL's and numeric numbers, tokenisation of tweet text, applying stemming and lemmatizer, plotting word-cloud for negative and positive tweets

```
X=data.text
y=data.target
data_neg = data['text'][:800000]
plt.figure(figsize = (20,20))
wc = WordCloud(max_words = 1000 , width = 1000 , height = 350,
               collocations=False).generate(" ".join(data_neg))
plt.imshow(wc)
#Negative Tokens
```

```
<matplotlib.image.AxesImage at 0x7fab5260bbd0>
```



Word-Cloud for Negative Tweets

Word-Cloud for Positive Tweets

- Splitting data into train and test subset
- Function for model evaluation

  After training the model we then apply the evaluation measures to check how the model is performing. Accordingly, we use the following evaluation parameters to check the performance of the models respectively :

  - Accuracy Score
  - Confusion Matrix with Plot
  - ROC-AUC Curve
- Model Building

  In the problem statement we have used three different models respectively :

  - Bernoulli Naive Bayes
  - SVM (Support Vector Machine)
  - Logistic Regression

  The idea behind choosing these models is that we want to try all the classifiers on the dataset ranging from simple ones to complex models and then try to find out the one which gives the best performance among them.

RESULTS

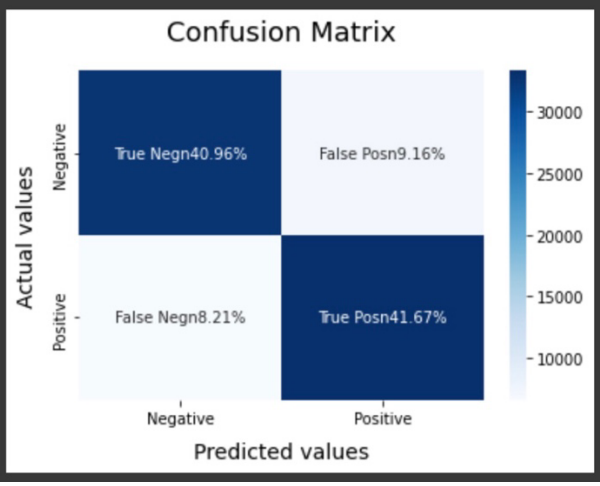
Model 1: Logistic Regression Model

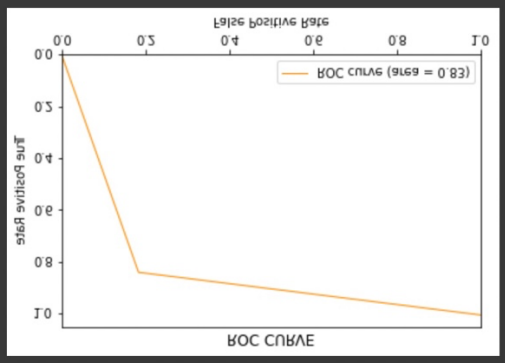
```
LRmodel = LogisticRegression(C = 2, max_iter = 1000, n_jobs=-1)
LRmodel.fit(X_train, y_train)
model_Evaluate(LRmodel)
y_pred3 = LRmodel.predict(X_test)
```

```
              precision    recall  f1-score   support

           0       0.83      0.82      0.83     40100
           1       0.82      0.84      0.83     39900

    accuracy                           0.83     80000
   macro avg       0.83      0.83      0.83     80000
weighted avg       0.83      0.83      0.83     80000
```






```
plt.show()
plt.legend(loc="lower right")
plt.title('ROC CURVE')
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.ylim([0.0, 1.02])
plt.xlim([0.0, 1.0])
plt.plot(fpr, fpr, color='darkorange', lw=1, label='ROC curve (area = %0.2f)' % roc_auc)
plt.figure()
roc_auc = auc(fpr, fpr)
fpr, fpr, fprgbmodel = roc_curve(y_test, y_pred3)
from sklearn.metrics import roc_curve, auc
```

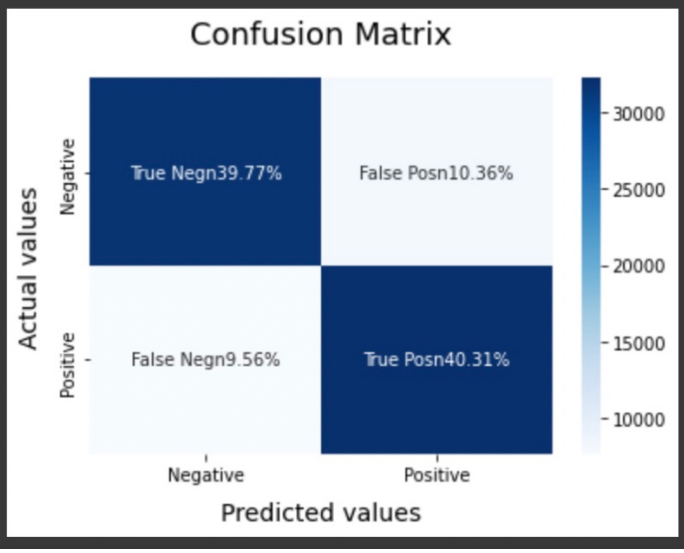
Model 2: Bernoulli Naive Bayes Model

```
BNBmodel = BernoulliNB()
BNBmodel.fit(X_train, y_train)
model_Evaluate(BNBmodel)
y_pred1 = BNBmodel.predict(X_test)
```

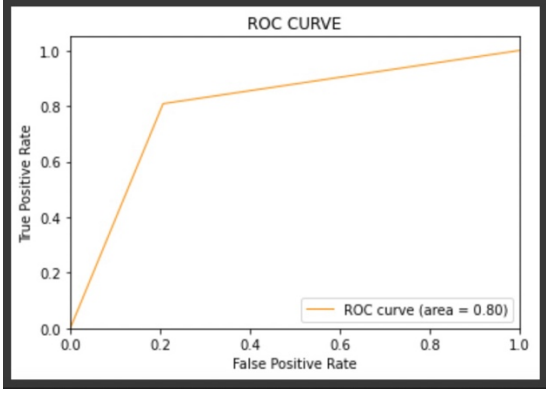
```
              precision    recall  f1-score   support

           0       0.81      0.79      0.80     40100
           1       0.80      0.81      0.80     39900

    accuracy                           0.80     80000
   macro avg       0.80      0.80      0.80     80000
weighted avg       0.80      0.80      0.80     80000
```



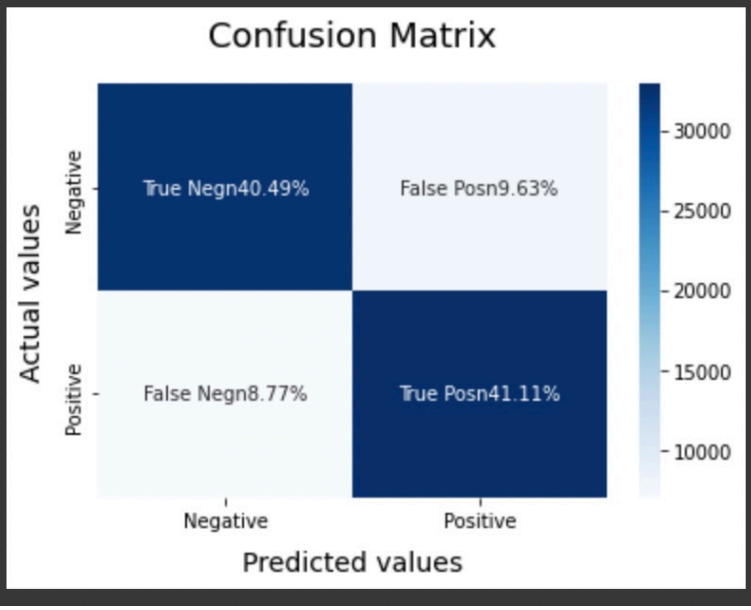

```
from sklearn.metrics import roc_curve, auc
fpr, tpr, thresholds = roc_curve(y_test, y_pred1)
roc_auc = auc(fpr, tpr)
plt.figure()
plt.plot(fpr, tpr, color='darkorange', lw=1, label='ROC curve (area = %0.2f)' % roc_auc)
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC CURVE')
plt.legend(loc="lower right")
plt.show()
```

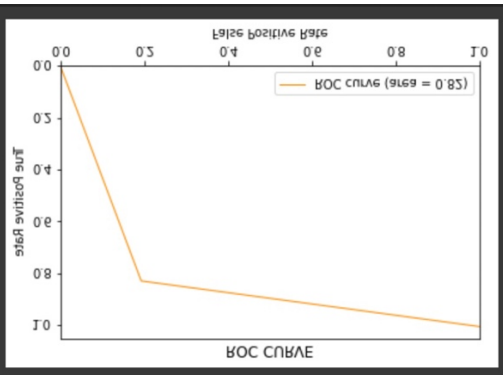


Model 3: Support Vector Machine (SVM)

```
SVCmodel = LinearSVC()
SVCmodel.fit(X_train, y_train)
model_Evaluate(SVCmodel)
y_pred2 = SVCmodel.predict(X_test)
```

	precision	recall	f1-score	support
0	0.82	0.81	0.81	40100
1	0.81	0.82	0.82	39900
accuracy			0.82	80000
macro avg	0.82	0.82	0.82	80000
weighted avg	0.82	0.82	0.82	80000






```
from sklearn.metrics import roc_curve, auc
fpr, tpr, thresholds = roc_curve(y_test, y_pred2)
roc_auc = auc(fpr, tpr)
plt.figure()
plt.plot(fpr, tpr, color='darkorange', lw=1, label='ROC curve (area = %0.2f)' % roc_auc)
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.02])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC CURVE')
plt.legend(loc="lower right")
plt.show()
```

CONCLUSION

Upon evaluating all the models we can conclude the following details i.e.

- **Accuracy:** As far as the accuracy of the model is concerned Logistic Regression performs better than SVM which in turn performs better than Bernoulli Naive Bayes.
- **F1-score:** The F1 Scores for class 0 and class 1 are :
  (a) For class 0: Bernoulli Naive Bayes(accuracy = 0.90) < SVM (accuracy =0.91) < Logistic Regression (accuracy = 0.92)
  (b) For class 1: Bernoulli Naive Bayes (accuracy = 0.66) < SVM (accuracy = 0.68) < Logistic Regression (accuracy = 0.69)
- **AUC Score:** All three models have the same ROC-AUC score.

We, therefore, conclude that the Logistic Regression is the best model for the above-given dataset. In our problem statement, Logistic Regression is following the principle of Occam's Razor which defines that for a particular problem statement if the data has no assumption, then the simplest model works the best. Since our dataset does not have any assumptions and Logistic Regression is a simple model, therefore the concept holds true for the above-mentioned dataset.

REFERENCES

- Dataset: https://www.kaggle.com/datasets/kazanova/sentiment140
- https://monkeylearn.com/sentiment-analysis/
- https://techvidvan.com/tutorials/python-sentiment-analysis/
- https://www.geeksforgeeks.org/twitter-sentiment-analysis-using-python/