Introduction :

In Our days, people use social media networks with a unbelievable frequency, writing posts, sharing photos and videos and sending private or public messages. One of th most used social network is Tweeter. Twitter is one of the most popular social media platforms in the world, with 330 million monthly active users and 500 million tweets sent each day. That's why analyzing tweets is very important to understand how people deal with a given subject.Understanding the sentiment of tweets is important for a variety of reasons: business marketing, politics, public behavior analysis, and information gathering are just a few examples. Sentiment analysis of Twitter data can help marketers understand the customer response to product launches and marketing campaigns, and it can also help political parties understand the public response to policy changes or announcements. Since Tweeter generate a huge amount of data (6000 tweets per second).

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.

Objective :

We aim to analyze 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.

SVM.

Decision Tree.

K-nearest neighbors.

Along with using Term Frequency- Inverse Document Frequency (TF-IDF).

The performance of these classifiers is then evaluated using accuracy, ROC-AUC Curve and F1 Scores.

What is Logistic Regression ?

Logistic regression is a statistical analysis method to predict a binary outcome, such as yes or no, based on prior observations of a data set. It is used in statistical software to understand the relationship between the dependent variable and one or more independent variables by estimating probabilities using a logistic regression equation.

Logistic regression is used when your Y variable can take only two values, and if the data is linearly separable, it is more efficient to classify it into two seperate classes.

What is Bernoulli Naive Bayes ?

Bernoulli Naive Bayes implements the naive Bayes training and classification algorithms for data that is distributed according to multivariate Bernoulli distributions.

Bernoulli Naive Bayes is one of the variants of the Naive Bayes algorithm in machine learning. It is very useful to be used when the dataset is in a binary distribution where the output label is either present or absent.

What is SVM ?

Support Vector Machine is a supervised machine learning algorithm used for both classification and regression. Though we say regression problems as well its best suited for classification. The objective of SVM algorithm is to find a hyperplane in an N-dimensional space that distinctly classifies the data points.

Support vector machines are a set of supervised learning methods used for classification, regression and outliers detection. The advantages of support vector machines are: Effective in high dimensional spaces. Still effective in cases where number of dimensions is greater than the number of samples.

What is Term Frequency- Inverse Document Frequency (TF-IDF) ?

Term Frequency gives us information on how often a term appears in a document.

Inverse Document Frequency is a measure of whether a term is common or rare in a given document corpus.

What is Stemming and lemmatization ?

The goal of both stemming and lemmatization is to reduce inflectional forms and sometimes derivationally related forms of a word to a common base form.

For instance:

am, are, is ⇒

be

car, cars, car's, cars' ⇒

car The result of this mapping of text will be something like:

the boy's cars are different colors ⇒

the boy car be differ color

How do we evaluate our model ?

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 : Typically, the accuracy of a predictive model is good (above 90% accuracy)

Execution time : Variable that depends on the machine on which the program was executed, but which can give a small idea of the model that executes the fastest.

F1 score : The F1 score is a weighted harmonic mean of precision and recall such that the best score is 1.0 and the worst is 0.0. F1 scores are lower than accuracy measures as they embed precision and recall into their computation.

ROC-AUC Curve : The Area Under the Curve (AUC) is the measure of the ability of a classifier to distinguish between classes and is used as a summary of the ROC curve. The higher the AUC, the better the performance of the model at distinguishing between the positive and negative classes.

Confusion Matrix with Plot : A Confusion matrix is an N x N matrix used for evaluating the performance of a classification model, where N is the number of target classes. The matrix compares the actual target values with those predicted by the machine learning model. Note that :

Actual values are the columns.

Predicted values are the lines.

	Positive	Negative
Positive	TP	TN
Negative	FP	TN

What is word_tokenize() ?

Tokenization is the act of breaking up a sequence of strings into pieces such as words, keywords, phrases, symbols and other elements called tokens.

word_tokenize() method. It actually returns the syllables from a single word. A single word can contain one or two syllables. Return : Return the list of syllables of words.

Project Pipeline :

The various steps involved in the Machine Learning Pipeline are :

1. Import Necessary Dependencies.
2. Read and Load the Dataset.
3. Exploratory Data Analysis.
4. Data Visualization of Target Variables.
5. Data Preprocessing.
6. Data Visualization after Preprocessing.
7. Splitting our data into Train and Test Subset.
8. Word Embedding and Transforming Dataset using TF-IDF Vectorizer.
9. Function for Model Evaluation.
10. Model Building.
11. Conclusion.

1. IMPORTING THE NECESSARY DEPENDANCIES:

```
# Importing necessary libraries and functions :
import pandas as pd
import numpy as np
import tensorflow as tf
from math import sqrt
import time

# Text processing libraries :
!pip install gensim
import gensim
import re
from spacy.lang.en import English

# Plotting libraries :
import seaborn as sns
import matplotlib.pyplot as plt
from wordcloud import WordCloud
```

```
# Regular Expression library
import nltk
import string
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
from nltk.stem import WordNetLemmatizer
from nltk.tokenize import word_tokenize #tokenization
from transformers import TFAutoModel
from sklearn.feature_extraction.text import TfidfVectorizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from transformers import AutoTokenizer
from collections import Counter
```

```
# sklearn :
import sklearn
from sklearn.svm import LinearSVC
from sklearn.naive_bayes import BernoulliNB
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.tree import DecisionTreeClassifier # Import Decision Tree Classif
from sklearn.model_selection import train_test_split # Import train_test_split
from sklearn import metrics #Import scikit-learn metrics module for accuracy c
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import roc_curve, auc
```

2. READING AND LOADING THE DATASET

n any project related to the manipulation and analysis of data, we always start by collecting the data on which we are going to work. In our case, we will import our data from a .csv file.

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

	target	ids	date	flag	user	text
1594667	4	2192204818	Tue Jun 16 06:40:44 PDT 2009	NO_QUERY	midnightnina	Craving for more ONE TREE HILL! I miss it so m...
1523539	4	2176591297	Mon Jun 15 04:16:21 PDT 2009	NO_QUERY	emirage	is going home.. http://plurk.com/p/112ew1
660990	0	2242938632	Fri Jun 19 12:49:19 PDT 2009	NO_QUERY	honeybc3	@Sexy_Nerd no lol. but its ok. Im gonna take h...
1374526	4	2051448599	Fri Jun 05 22:02:50 PDT 2009	NO_QUERY	AmyLGCampbell	OH MY GOD up at 5:30 this is crazy
382908	0	2053093378	Sat Jun 06 03:14:43 PDT 2009	NO_QUERY	Aodan	Not doin anything today bored to death

3. EXPLORATORY DATA ANALYSIS

In this part, the objective is to know the imported data as much as possible, we analyze a sample, we look for the shape of the dataset, the column names, the data type information, we check if there are null values, in short, we process our data and above all we target the data (columns) that interests us.

```
df.shape
```

```
(1600000, 6)
```

1600000 is the number of records in our dataset.

6 is the number of columns.

```
# Getting info about our dataset :
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1600000 entries, 0 to 1599999
Data columns (total 6 columns):
 #   Column  Non-Null Count    Dtype
---  ------  --------------    -----
 0   target  1600000 non-null  int64
 1   ids     1600000 non-null  int64
 2   date    1600000 non-null  object
 3   flag    1600000 non-null  object
 4   user    1600000 non-null  object
 5   text    1600000 non-null  object
dtypes: int64(2), object(4)
memory usage: 73.2+ MB
```

The range index of the records starts from 0 to 1599999

```
print(df.dtypes)
```

```
target     int64
ids        int64
date      object
flag      object
user      object
text      object
dtype: object
```

The data type of some columns in our dataset is object, which means we still have to process our data before getting into machine learning stuff.

```
# Checking for Null values :
print("number of missing values in the dataframe is {}".format(np.sum(df.isnul
```

Good news so far, there are no missing values in our dataset.

We got four columns with string values (date,flag,user,text) and two other columns with integers (target and ids)

```
# Let's explore our target variable 'target'
print("the number of unique values of the target variable is {}".format(df['
print("unique values of target variable are {0} and {1}".format(df['target']
```

The target column is composed of just 0 and 4

0 stands for negative sentiment.

4 stands for positive sentiment.

The target column is composed of just 0 and 1

0 stands for negative sentiment.

1 stands for positive sentiment.

The number of unique values of the ids feature is 1598315

Since the number of unique values of the Ids is less than the length of our dataset, it means that the Ids have to be repeated. in other words, there might be tweets that have the same ID or repeat each other

Data Visualization of Target Variables :

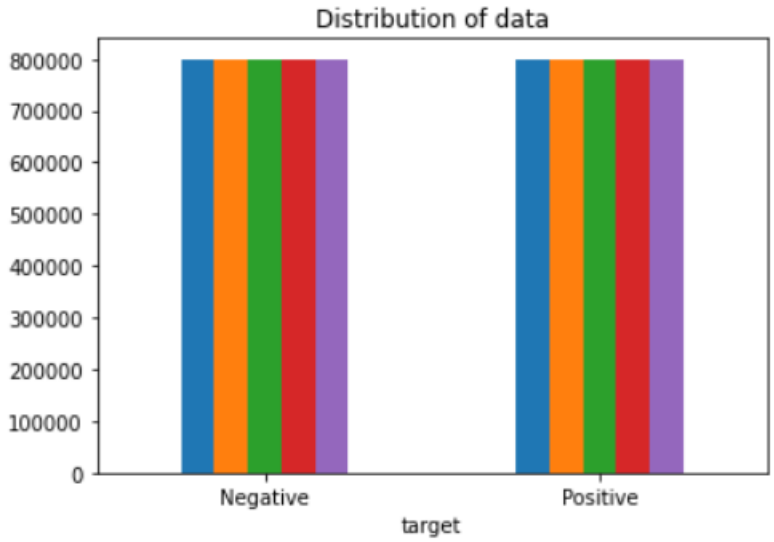
After processing our data and targeting the columns we are interested in, the next step is to have a visual on our data with mathematical plots, the reason for using plots is that a plots makes the data speak more, so it become more understandable.

target	ids	date	flag	user	text
0	800000	800000	800000	800000	800000
1	800000	800000	800000	800000	800000

Since the target column only contains 0 or 4, using the .groupby() function will result in two categories: 0 and 4

```
# Plotting the distribution for dataset :
ax = df.groupby('target').count().plot(kind='bar', title='Distribution of data
# Naming 0 -> Negative , and 4 -> Positive
ax.set_xticklabels(['Negative','Positive'], rotation=0)

# Storing data in lists :
text, sentiment = list(df['text']), list(df['target'])
```

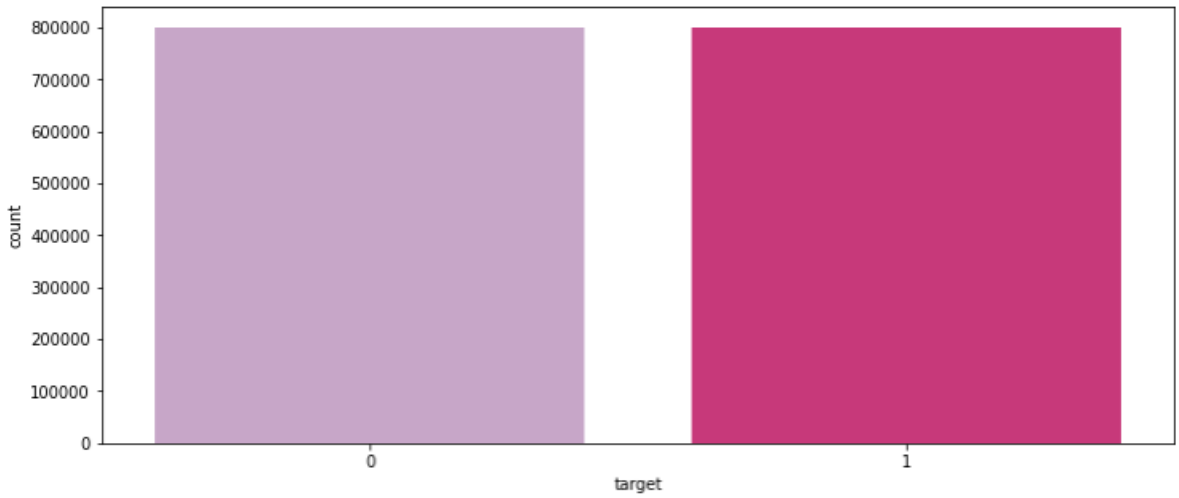


We can see that we have an equal number of tweets with positive sentiments and negative sentiments.

Each color represents one of the columns : ids, date, flag, user and text.

text variable contains the text column.

sentiment variable contains the target column.



Data Preprocessing

Our data generally comes from a variety of different sources and is often in a variety of different formats. For this reason, cleaning our raw data is an essential part of preparing our dataset. However, cleaning is not a simple process, as textual data often contains redundant and/or repetitive words.

Before training the model, we will perform various pre-processing steps on the dataset such as:

Removing stop words.

Removing emojis.

Removing of mentions.

Removal of numbers.

Removal of whitespaces.

Removal of duplicated rows.

Removal of unusual columns.

Converting the text document to lowercase for better generalization.

Cleaning the punctuation (to reduce unnecessary noise from the dataset).

Removing the repeating characters from the words along with removing the URLs/hyperlinks as they do not have any significant importance.

and much more, we will see this in detail later...

We will then perform:

Stemming : reducing the words to their derived stems.

Lemmatization : reducing the derived words to their root form known as lemma for better results.

Lowering Case:

Lowering case is very imprtant since it allows us to make words with same value equal. This will be very useful to reduce the dimensions of our vocabulary.

Removal of Mentions:

In social media, Mentions are used to call/mention another user into our post. Generally, mentions don't have an added value to our model. So we will remove them.

A mention has a special pattern: @UserName, So we will remove all string which starts with @

Removal of Special Characters:

Special characters are every where, since we have punctuation marks in our tweets. In order to treat, for example, hello! and hello in the same way. we have to remove the punctuation mark !

Removal of Stop words:

Stopwords are the most common words in any natural language. For the purpose of analyzing text data and building NLP models, these stopwords might not add much value to the meaning of the document.

Generally, the most common words used in a text are “the”, “is”, “in”, “for”, “where”, “when”, “to”, “at” etc.

Consider this text string – “There is a pen on the table”. Now, the words “is”, “a”, “on”, and “the” add no meaning to the statement while parsing it. Whereas words like “there”, “book”, and “table” are the keywords and tell us what the statement is all about.

Stopword Removal using NLTK:

NLTK, or the Natural Language Toolkit, is a treasure trove of a library for text preprocessing. It’s one of my favorite Python libraries. NLTK has a list of stopwords stored in 16 different languages.

Stopword Removal using spaCy:

spaCy is one of the most versatile and widely used libraries in NLP. We can quickly and efficiently remove stopwords from the given text using SpaCy. It has a list of its own stopwords that can be imported as STOP_WORDS from the spacy.lang.en.stop_words class.

Stopword Removal using Gensim:

Gensim is a pretty handy library to work with on NLP tasks. While pre-processing, gensim provides methods to remove stopwords as well. We can easily import the remove_stopwords method from the class gensim.parsing.preprocessing.

Removal of Links/URLs:

Tweets may contain URLs, which are not significant for our model. That's why we will remove them

okenizing the text feature:

Given a character sequence and a defined document unit, tokenization is the task of chopping it up into pieces, called tokens, perhaps at the same time throwing away certain characters, such as punctuation.

Word Tokenization is the most commonly used tokenization algorithm. It splits a piece of text into individual words based on a certain delimiter. Depending upon delimiters, different word-level tokens are formed.

What is word_tokenize() ?

Tokenization is the act of breaking up a sequence of strings into pieces such as words, keywords, phrases, symbols and other elements called tokens.

word_tokenize() method. It actually returns the syllables from a single word. A single word can contain one or two syllables. Return : Return the list of syllables of words.

What is Stemming and lemmatization ?

The goal of both stemming and lemmatization is to reduce inflectional forms and sometimes derivationally related forms of a word to a common base form.

For instance:

am, are, is ⇒

be

car, cars, car's, cars' ⇒

car The result of this mapping of text will be something like:

the boy's cars are different colors ⇒

the boy car be differ color

Stemming the text feature:

stemming is the process of removing a part of a word, or reducing a word to its stem or root. This might not necessarily mean we're reducing a word to its dictionary root. We use a few algorithms to decide how to chop a word off. This is, for the most part, how stemming differs from lemmatization, which is reducing a word to its dictionary root, which is more complex and needs a very high degree of knowledge of a language. We'll later talk about lemmatization.

Let's assume we have a set of words — send, sent and sending. All three words are different tenses of the same root word send. So after we stem the words, we'll have just the one word — send. Similarly, if we have the words — ask, asking and asked — we can apply stemming algorithms to get the root word — ask. Stemming is as simple as that. But, unfortunately, it's not as simple as that. We will some times have complications. And these complications are called over stemming and under stemming. Let's see more about them in the next sections.

Over stemming : For example, university and universe. Some stemming algorithms may reduce both the words to the stem univers, which would imply both the words mean the same thing, and that is clearly wrong.

Under stemming: For example, consider the words "data" and "datum." Some algorithms may reduce these words to dat and datu respectively, which is obviously wrong.

Porter stemmer is a widely used stemming technique. nltk.stem provides the utility function to stem 'PorterStemmer'

Lemmatizing the text feature:

Lemmatization, unlike Stemming, reduces the inflected words properly ensuring that the root word belongs to the language. In Lemmatization root word is called Lemma. A lemma (plural lemmas or lemmata) is the canonical form, dictionary form, or citation form of a set of words.

Data Visualization after Preprocessing :

Before performing the machine learning, let's have a general idea of the accuracy of our data, to do this we will use a word cloud which is a collection, or group, of words represented in different

sizes. The bigger and bolder the word appears, the more often it is mentioned in a given text and the more important it is.

Which means that in our case we expect a word cloud to contain a sample of words representing the category we are plotting

we'll generate a word cloud for positive tweets and another for negative tweets to see which are the most commonly used words for each tweet category.

Splitting our data into Train and Test Subset

random_state is basically used for reproducing your problem the same every time it is run. If we do not use a random_state in train_test_split, every time you make the split we might get a different set of train and test data points and will not help in debugging in case we get an issue.

X contains data.text

y contains = data.target

X_train contains 95% of data.text

X_test contains 5% of data.text

y_train contains 95% of data.target

y_test contains 5% of data.target

Word Embedding and Transforming Dataset using TF-IDF Vectorizer

NLP experts developed a technique called word embeddings that convert words into their numerical representations. Once converted, NLP algorithms can easily digest these learned representations to process textual information. Word embeddings map the words as real-valued numerical vectors. It does so by tokenizing each word in a sequence (or sentence) and converting them into a vector space. Word embeddings aim to capture the semantic meaning of words in a sequence of text. It assigns similar numerical representations to words that have similar meanings.

Simply, these words need to be made meaningful for machine learning or deep learning algorithms. Therefore, they must be expressed numerically. Algorithms such as One Hot Encoding, TF-IDF, Word2Vec, FastText enable words to be expressed mathematically as word embedding techniques used to solve such problems.

One-hot encoding is an important step for preparing our dataset for use in machine learning. One-hot encoding turns your categorical data into a binary vector representation. Pandas get dummies makes this very easy!

This means that for each unique value in a column, a new column is created. The values in this column are represented as 1s and 0s, depending on whether the value matches the column header.

For example, with the help of the get_dummies function, we turn this table below :

Gender

Male

Female

Male

Male

To this :

Gender	Male	Female
Male	1	0
Female	0	1
Male	1	0
Male	1	0

Bag Of Words:

The bag-of-words model is a simplifying representation used in natural language processing and information retrieval (IR). In this model, a text (such as a sentence or a document) is represented as the bag (multiset) of its words, disregarding grammar and even word order but keeping multiplicity.

Bag of Words (BOW) is a method to extract features from text documents. These features can be used for training machine learning algorithms. It creates a vocabulary of all the unique words occurring in all the documents in the training set.

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 : Typically, the accuracy of a predictive model is good (above 90% accuracy)

ROC-AUC Curve : The Area Under the Curve (AUC) is the measure of the ability of a classifier to distinguish between classes and is used as a summary of the ROC curve. The higher the AUC, the better the performance of the model at distinguishing between the positive and negative classes.

Confusion Matrix with Plot : A Confusion matrix is an N x N matrix used for evaluating the performance of a classification model, where N is the number of target classes. The matrix compares the actual target values with those predicted by the machine learning model.

Actual values are the columns.

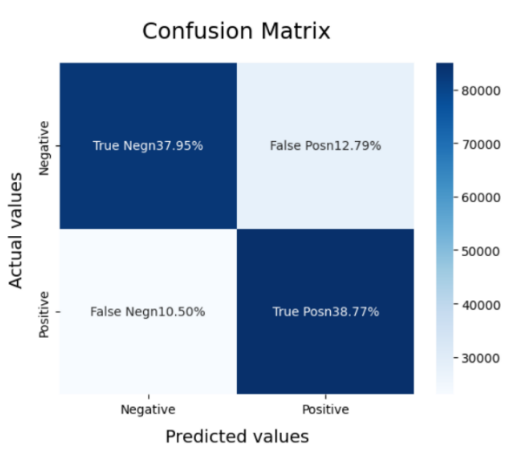
Predicted values are the lines.

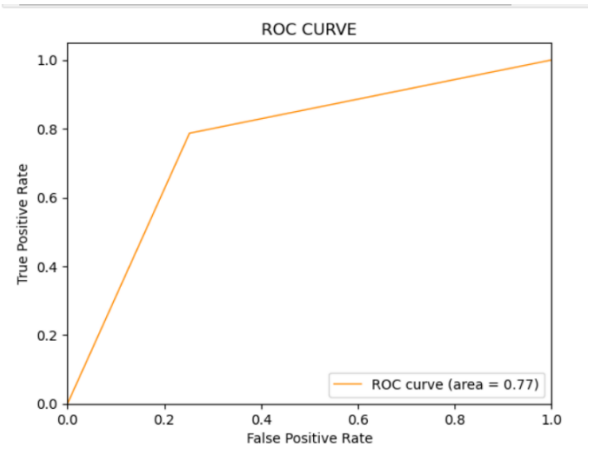
Positive	Negative	
Positive	TP	TN
Negative	FP	TN

MODEL BUILDING

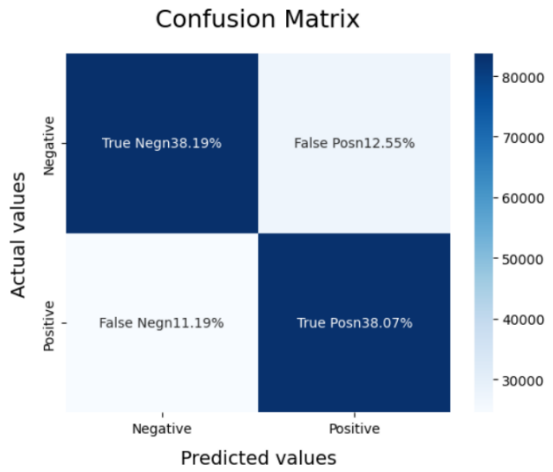
In the problem statement we have used different models:

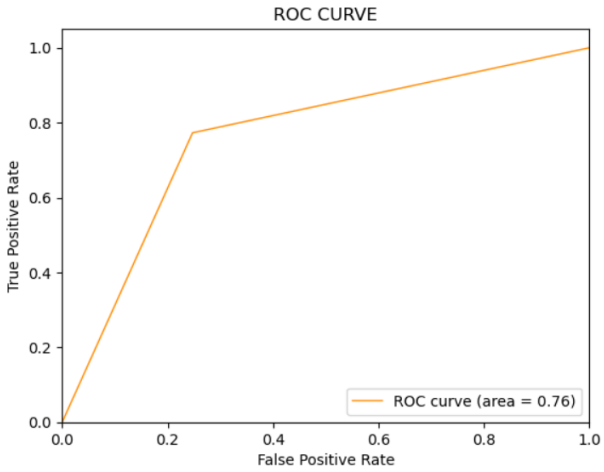
MODEL 1: BERNOULLI NAÏVE BAYES



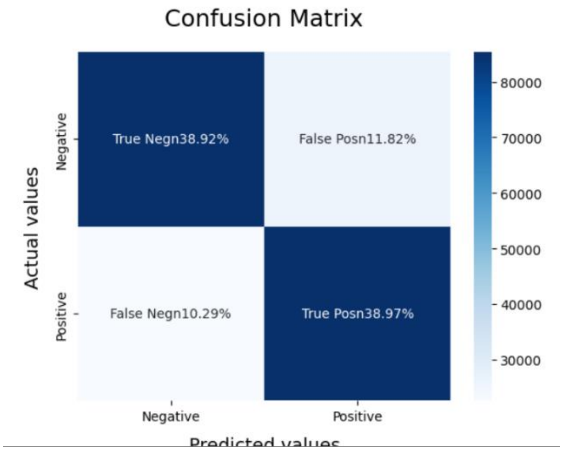


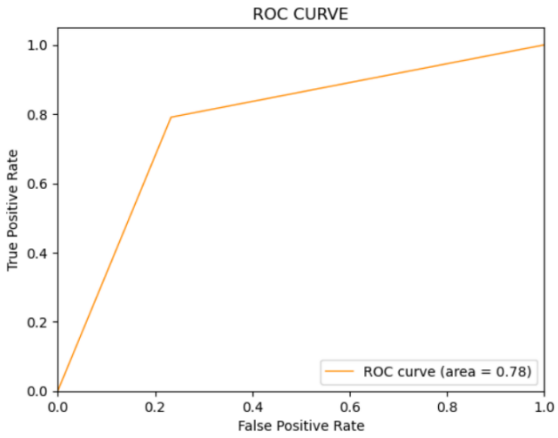
MODEL 2: SUPPORT VECTOR MACHINE



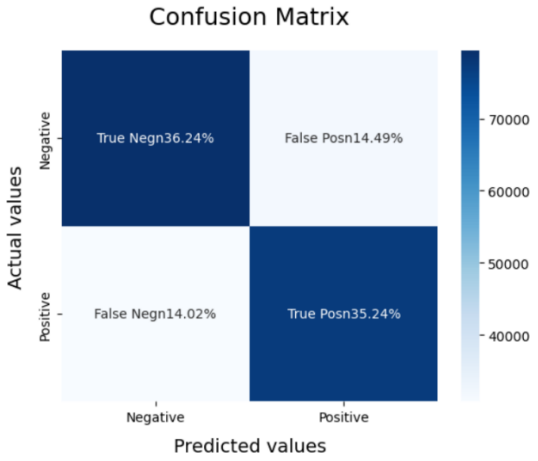


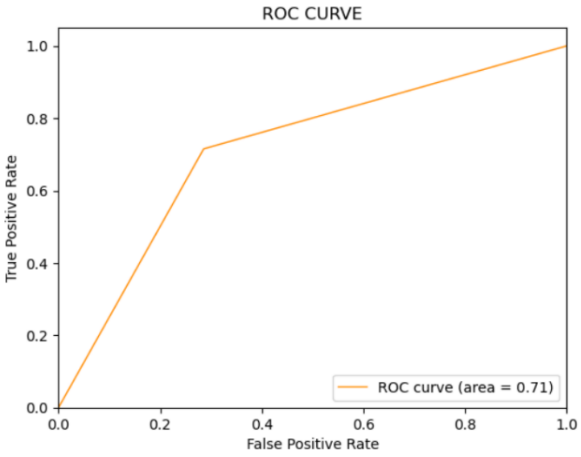
MODEL 3: LOGISTIC REGRESSION





MODEL 4: DECISION TREE





Conclusion :

After evaluating all models, we can conclude the following details :

Model	Accuracy	F1-score (class 0)	F1-score (class 1)	AUC Score	Execution time
Bernoulli Naive Bayes (BNB)	80%	80%	80%	80%	0.69 seconds

Model	Accuracy	F1-score (class 0)	F1-score (class 1)	AUC Score	Execution time
Decision Tree (DT)	69%	69%	69%	69%	0.76 seconds
Support Vector Machine (SVM)	82%	81%	82%	82%	28.32 seconds
Logistic Regression (LR)	83%	83%	83%	83%	163.56 seconds

- **Execution time** : When it comes to comparing the running time of models, Bernoulli Naive Bayes performs faster than SVM, which in turn runs faster than Logistic Regression.
- **Accuracy** : When it comes to model accuracy, 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 :
    - For **class 0** (negative tweets) :

      accuracy : BNB (= 0.80) < SVM (=0.81) < LR (= 0.83)

    - For **class 1** (positive tweets) :

      accuracy : BNB (= 0.80) < SVM (=0.82) < LR (= 0.83)

- **AUC Score** : All three models have the same ROC-AUC score.

  AUC score : BNB (= 0.80) < SVM (=0.82) < LR (= 0.83)

- We therefore conclude that **logistic regression** is the **best model** for the above dataset.(although it took much longer to run than other models).

- In our problem statement, **logistic regression** follows **Occam's razor principle** which defines that for a particular problem statement, if the data has no assumptions, then the simplest model works best. Since our **dataset has no assumptions** and **logistic regression is a simple model**, so the concept holds true for the dataset mentioned above.