PRML PROJECT ANALYSIS

Twitter sentiment analysis

OJASWIT GAUTAM(B19EE059)
NIMIT KHANNA(B19EE057)

INDIAN INSTITUTE OF INFORMATION TECHNOLOGY, JODHPUR

Abstract— Twitter Sentiment Analysis means, using advanced text mining techniques to analyze the sentiment of the text (here, tweet) in the form of positive, negative and neutral. Sentiment analysis can be defined as a process that automates mining of attitudes, opinions, views and emotions from text, speech, tweets and database.

Keywords— Twitter,Machine learning,Random forest,Logistic regression,Naive Bayes,NN,XGB

INTRODUCTION

Labels : labels are the targets which are sentiments of the tweets in the project.

Inputs : Inputs are the data that we feed into machine learning for eg “tweets texts” are the inputs in our project.

Training Data: We use training data when we have to train the models with the help of split_test_train.

Validation Data: This is used while training the model. We use this data to evaluate the performance of how the model performs on training time.

	0	1467810369	Mon Apr 06 22:19:45 PDT 2009	NO_QUERY	_TheSpecialOne_	@switchfoot http://twitpic.com/2y1zl - Awww, that's a bummer. You shoulda got David Carr of Third Day to do it. ;D
0	0	1467810672	Mon Apr 06 22:19:49 PDT 2009	NO_QUERY	scotthamilton	is upset that he can't update his Facebook by ...
1	0	1467810917	Mon Apr 06 22:19:53 PDT 2009	NO_QUERY	mattycus	@Kenichan I dived many times for the ball. Man...
2	0	1467811184	Mon Apr 06 22:19:57 PDT 2009	NO_QUERY	ElleCTF	my whole body feels itchy and like its on fire
3	0	1467811193	Mon Apr 06 22:19:57 PDT 2009	NO_QUERY	Karoli	@nationwideclass no, it's not behaving at all....
4	0	1467811372	Mon Apr 06 22:20:00 PDT 2009	NO_QUERY	joy_wolf	@Kwesidei not the whole crew

Testing Data: After training we test our model. We use this data to evaluate the performance of how the model performs after training.

Preparing the input features for training-

We removed-
URLs: `def remove_urls(text):`
Spaces: `def remove_spaces(text):`
Emojis: `def remove_emoji(string):`
Punctuation: `def remove_punctuation(text):`
Multiple spaces: `data["text"].apply(lambda text: " ".join(text.split()))`
Removing non English words
Removing numbers

After preparing all the tweets, now we have to separate/split the tweets into training data and testing data.

- 80% tweets will be used in the training
- 20% tweets will be used to test the performance of the model

```
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.20,random_state=33)
```

```
F1 score = 73.15864859794002

************************************

Precision score :   73.18247850180384

****************************************
precision_cnb :
0.7318247850180384
****************************************

Recall score : 73.16351638654398
```

Implementing Tensorflow based model for training

Step 1:- The input to the model is 470 words because these are the number features/words that we extracted above from the text of tweets.

Step 2:- With the help of Embeddings the presentation of words and their relative meanings are enhanced. We are feeding the limit of maximum words, length of input words and the inputs of the previous layer.

Step 3:- LSTM (long short term memory) is a sequence prediction of next coming words. Used LSTM that saves the words and predicts the next words based on the previous words.

Step 4:- The **dense layer** is a neural network layer that is connected deeply, which means each neuron in the dense layer receives input from all neurons of its previous layer. Dense layer uses all the inputs of previous layer neurons and performs calculations and sends 256 outputs.

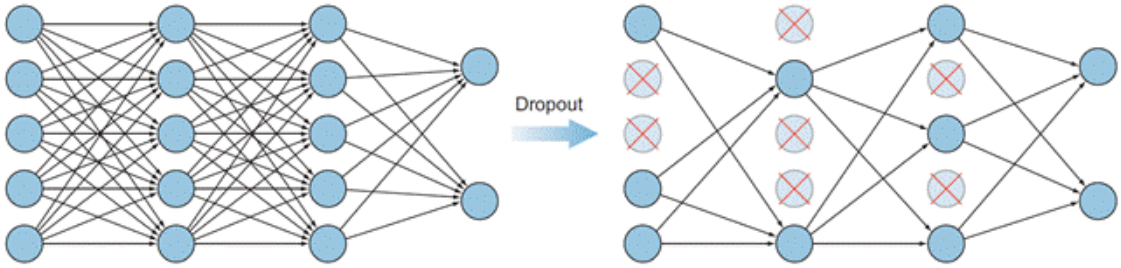
Step 5:- Activation function is a node that is put at the end of all layers of neural network model or in between neural network layers. Activation functions help to decide which neuron should be passed and which neuron should fire. So the activation function of a node defines the output of that node given an input or set of inputs.

Step 6:- The Dropout layer randomly sets input units to 0 with a frequency of rate at each step during training time, which helps prevent overfitting. In overfitting, models give good accuracy on training time but not good on testing time. Dropout layer only applies when training is set to True such that no values are dropped during inference.

----------the accuracy of the model on test data is given below----------

Test set

Accuracy: 0.7737414836883545



MODEL COMPILATION

We are using 2 classes so we set "binary_crossentropy".

```
model = tensorflow_based_model() # here we are calling the function of created model
```

Optimizers are algorithms or methods used to change the attributes of the neural network such as weights and learning rate to reduce the lossesThese are used to solve optimization problems by minimizing the function.So the learning rate of the neural network to reduce the losses is defined by the optimizer.

Then we have calculated the percentage of correct predictions over all predictions on the validation set, by setting the metrics = accuracy.

Training and validating with parameter tuning-

```
history=model.fit(X_train,Y_train,batch_size=84,epochs=3, validation_split=0.1)
```

Here we are starting the training of models by feeding the training data getting 10% data for validation from training data.

We set the following parameters:

- Batch size =84 so the model took 84 tweets in each iteration and trained them. Batch size is a term that refers to the number of training examples utilized in one iteration.

- Epochs =3 so the model will train on the data 3 times.An epoch means training the neural network with all the training data for one cycle. In an epoch, we use all of the data exactly once.
- Batch_size, and epochs can be changed by the user as we want so we set some values and train the model. If the model does not give good results we can change it and keep trying unless we get satisfactory results for the training of the model. This process is called parameter tuning.

We need to do all the above configurations to train the model.

Testing the Trained model on test data-

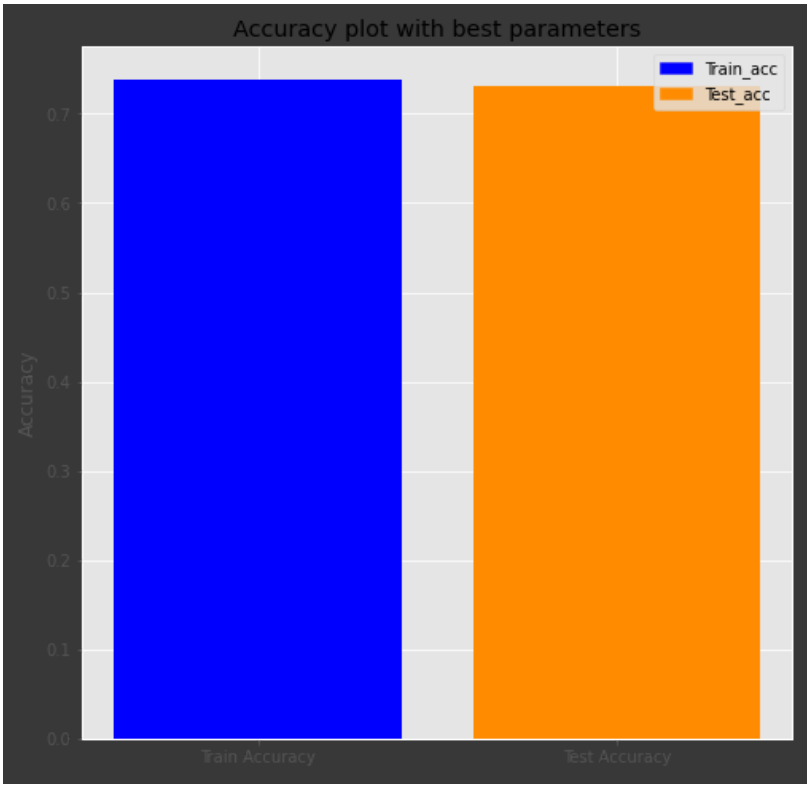
1) Getting predictions/classifying the sentiments (positive/negative) on the test data using a trained model.

```
accr1 = model.evaluate(X_test,Y_test) #we are starting to test the model here
```

Accuracy-

- Accuracy= Total number of correct predictions/Total number of predictions.
- Accuracy is the number of correctly classified tweets from all the tweets of positive and negative.

Cross Validation score = [0.73217813 0.73100625 0.73190312 0.73182813 0.73120541]
Train accuracy =73.89% Test accuracy =73.16%



----------the accuracy of the model on test data is given below----------
Test set

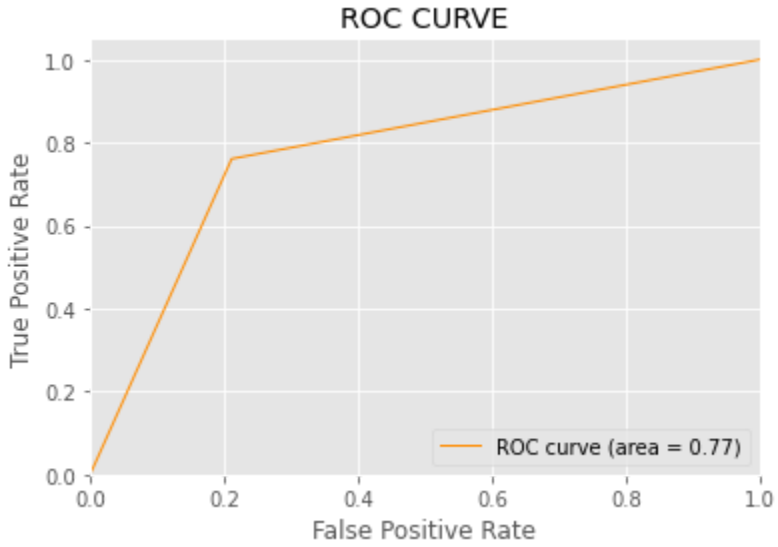
Accuracy: 0.7737414836883545

confusion matrix-
[[5693 1726]
 [1600 5681]]

- A confusion matrix is a summary of prediction results on a classification problem. The number of correct and incorrect predictions are summarized with count values and broken down by each class. It shows the ways in which your classification model.
- 5693 and 5681 are the correct predictions with the trained model and 1726 and 1600 show the wrong predictions.
- 5693 tweets correctly predicted negative sentiments. 1726 tweets predicted positive sentiments but that were actually negative sentiments.
- 5681 tweets correctly predicted positive sentiments. 1600 tweets predicted negative sentiments but that were actually positive sentiments.

ROC CURVE

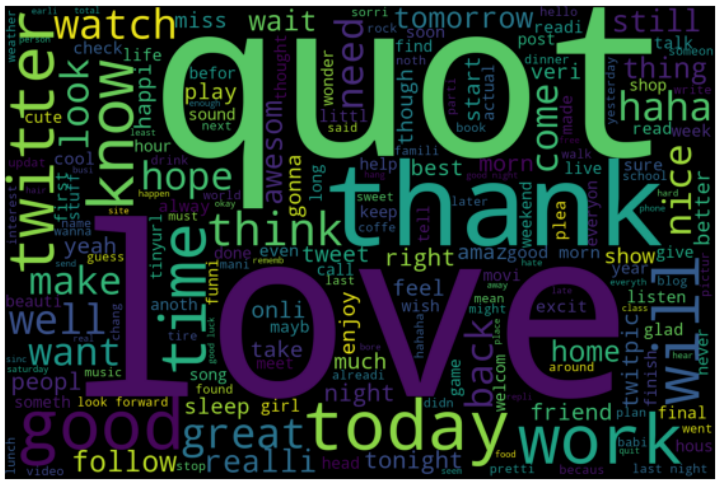
An ROC curve is a graph showing the performance of a classification model at all classification thresholds. This curve plots two parameters: True Positive Rate VS False Positive Rate.
We notice that the model started from the 0% predictions and then moved to true positive predictions.
Area Under ROC = 0.774



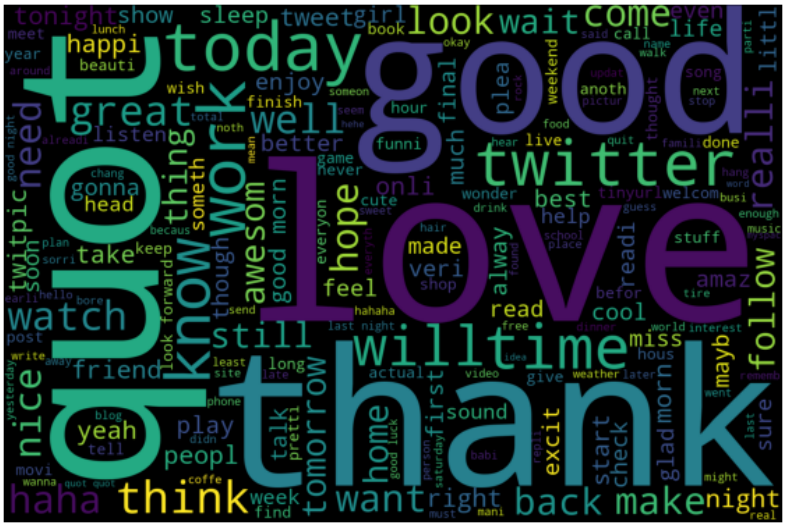
Naive Bayes classifiers are a collection of classification algorithms based on Bayes' Theorem. In simple terms, a Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature.

	precision	recall	f1-score	support
0	0.73	0.75	0.74	160097
4	0.74	0.72	0.73	159903
accuracy			0.73	320000
macro avg	0.73	0.73	0.73	320000
weighted avg	0.73	0.73	0.73	320000

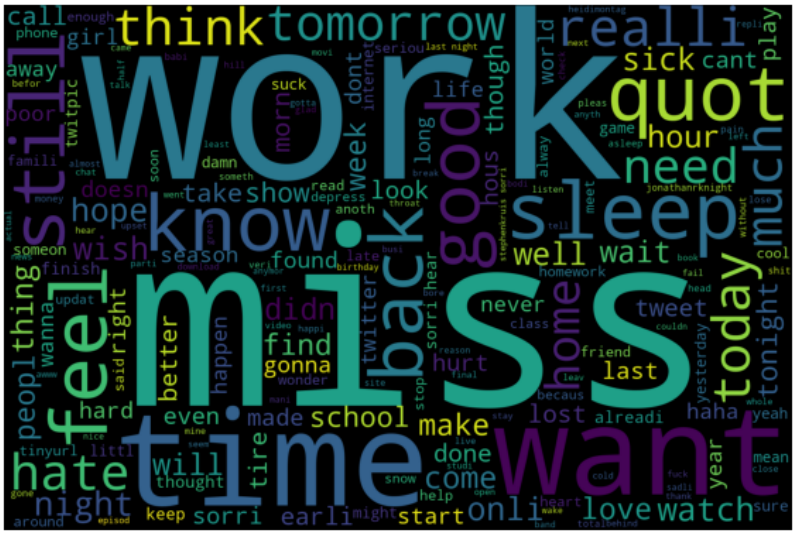
Area Under the Curve = 0.7316351638654397



**********tweets for positive sentiment**********



**********tweet for depressive sentiments**********



****

MultinomialNB

Accuracy Score :

96.44

****

	precision	recall	f1-score	support
0	0.00	0.00	0.00	0
4	1.00	0.96	0.98	7620
accuracy			0.96	7620
macro avg	0.50	0.48	0.49	7620
weighted avg	1.00	0.96	0.98	7620

****

RandomForestClassifier

Accuracy Score :

96.5

****

	precision	recall	f1-score	support
0	0.05	0.59	0.09	22
4	1.00	0.97	0.98	7598
accuracy			0.96	7620
macro avg	0.52	0.78	0.54	7620
weighted avg	1.00	0.96	0.98	7620

****

SVC

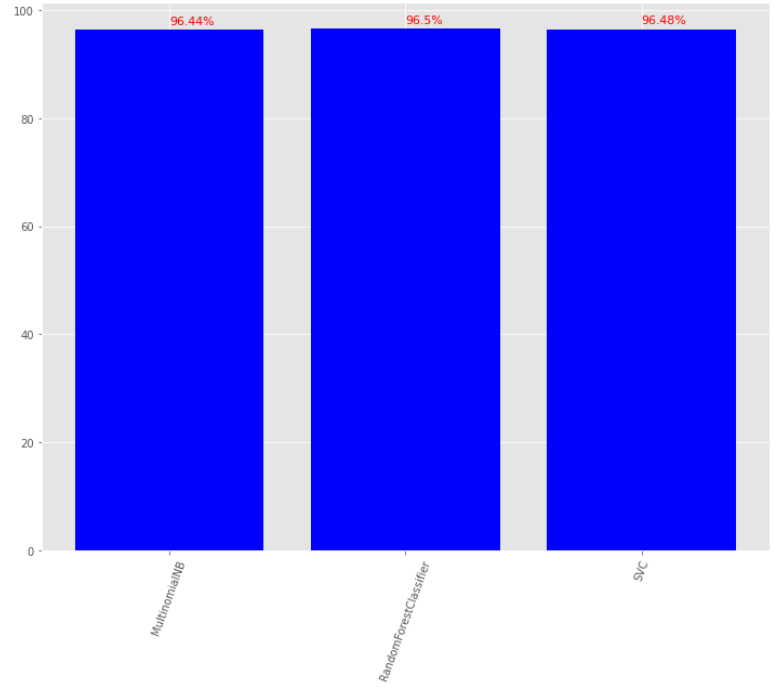
Accuracy Score :

96.48

****

	precision	recall	f1-score	support

0	0.01	1.00	0.02	3
4	1.00	0.96	0.98	7617
accuracy			0.96	7620
macro avg	0.51	0.98	0.50	7620
weighted avg	1.00	0.96	0.98	7620



****
LogisticRegression
Accuracy Score :
96.43
****

	precision	recall	f1-score	support
0	0.00	0.00	0.00	1
4	1.00	0.96	0.98	7619
accuracy			0.96	7620
macro avg	0.50	0.48	0.49	7620
weighted avg	1.00	0.96	0.98	7620

****
DecisionTreeClassifier
Accuracy Score :
94.4
****

	precision	recall	f1-score	support
0	0.13	0.16	0.14	228
4	0.97	0.97	0.97	7392
accuracy			0.94	7620
macro avg	0.55	0.56	0.56	7620

weighted avg 0.95 0.94 0.95 7620

****

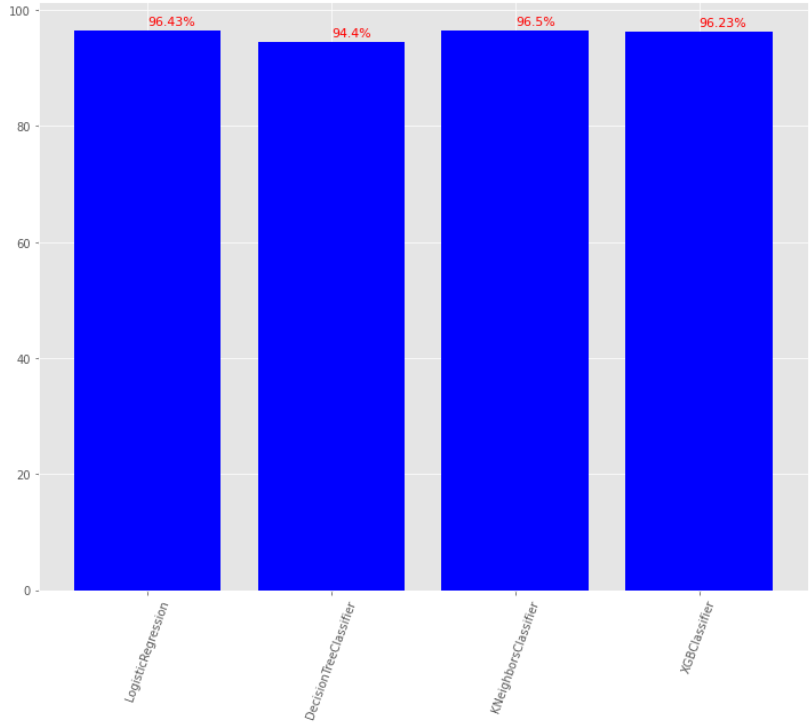
KNeighborsClassifier
Accuracy Score :
96.5
****

	precision	recall	f1-score	support
0	0.01	1.00	0.03	4
4	1.00	0.96	0.98	7616
accuracy			0.96	7620
macro avg	0.51	0.98	0.51	7620
weighted avg	1.00	0.96	0.98	7620

****

XGBClassifier
Accuracy Score :
96.23
****

	precision	recall	f1-score	support
0	0.04	0.29	0.07	38
4	1.00	0.97	0.98	7582
accuracy			0.96	7620
macro avg	0.52	0.63	0.53	7620
weighted avg	0.99	0.96	0.98	7620



CONCLUSIONS

We used the twitter sentiment analysis dataset and explored the data in different ways.
We prepared the text data of tweets by removing the unnecessary things.
We trained a model based on tensorflow with all settings.
We evaluated the model with different evaluation measures.
We worked on the classification problem and specifically we call it binary classification which is two class classification.

Accuracy for Naive Bayes, Random Forest Classifier and SVM
[96.44, 96.5, 96.48]
Accuracy for Logistic Regression, Decision Tree, K-Neighbors ,XGBoost
[96.43, 94.4, 96.5, 96.23]
RF and KNN were found to have maximum accuracy.

All the classifiers used gave pretty good results, amongst them we found the Decision tree classifier a little back in accuracy. This is found by running code with different hyper parameters.
RandomForest Classifier was slow with these hyperParameters (n_estimators=999, random_state=25).

ACKNOWLEDGMENT

We would like to express our sincere gratitude to several individuals and TAs for supporting us. First, we wish to express my sincere gratitude to ourProfessor Dr Richa Singh and Dr. Romi Banerjee for her enthusiasm, patience, insightful comments, helpful information, practical advice and unceasing ideas that have helped me tremendously at all times in our project.

CONTRIBUTION

- **Nimit Khanna(B19EE057)**
  - **Naive Bayes classifier**
  - **Neural network**
  - **Random Forest Classification**
  - **Logistic Regression Classification**
  - **Decision Tree Classifier**
  - **And other code.**
- **Ojaswit Gautam(B19EE059)**
  - **Analysis of project with conclusion**
  - **Applied KNN**
  - **XGBoast**
  - **SVM**
  - **Collecting resources**

REFERENCES

- https://towardsdatascience.com/detecting-disaster-from-tweets-classical-ml-and-lstm-approach-4566871af5f7
- https://en.wikipedia.org/wiki/Naive_Bayes_classifier
- https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm
- https://en.wikipedia.org/wiki/Random_forest
- https://github.com
- https://en.wikipedia.org/wiki/Decision_tree_learning