PRML Course Project

Topic : Twitter Sentiment Analysis

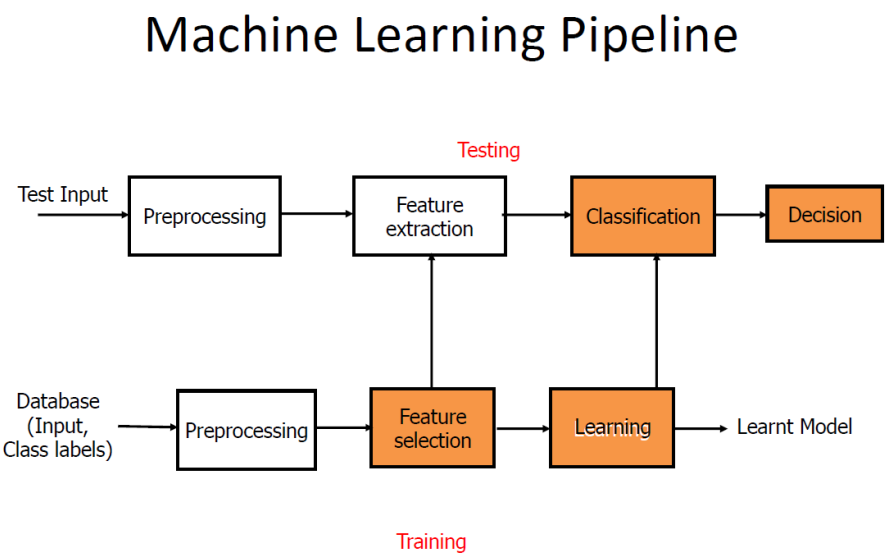
Group Members

1. Abhishek Dua (B19EE002)
2. Akshat Agarwal (B19EE007)
3. Barun Shakya (B19CSE020)

Abstract

Twitter is an online social network. Users on twitter create short messages called tweets to be shared with other twitter users who interact by retweeting and responding. Twitter employs a message size restriction of 280 characters or less which forces the users to stay focused on the message they wish to disseminate. This very characteristic makes messages on twitter very good candidates for the Machine Learning (ML) task of sentiment analysis. Sentiment Analysis falls under Natural Language Processing (NLP) which is a branch of ML that deals with how computers process and analyze human language.Twitter sentiment analysis iis the automated process of identifying and classifying subjective information in text data. This might be an opinion, a judgment, or a feeling about a particular topic or product feature. The most common type of sentiment analysis is classifying tweets as positive or negative based on the words in the tweet. Sentiment Analysis refers to the use of Natural Language Processing to determine the attitude, opinions and emotions of a speaker, writer, or other subject within an online mention.

Machine learning Pipeline Used in the Project:



Steps Involved According to Pipeline:

Preprocessing

- **Data Loading:** We loaded our tweet sentiment analysis dataset.
- **Separating Dependent and Independent Variables**: In this step, the dependent and independent variables are separated. The dependent variable is sentiment and the independent variable is text of tweets.
- **Data analysis:** The data analysis is performed by describing the data, counting the instances of each class, etc.

- **Data Cleaning:** In this step we remove all the special characters, links, punctuations, hashtags, etc.
  In this step, the whole text is converted into lowercase letters.
- **Remove Stopwords:** Stopwords are frequently occurring common words and they have no use in the classification process. So we remove stopwords from the tweet dataset. We also plotted wordcloud to get a view of frequently occurring words.

Feature extraction:

- **Vectorization:** In this step, the data of each tweet is converted into a vector. A machine learning model can easily understand these vectors.
- **Bag of Words:** In Bag of words model:
  - Counted the frequency of each word in a tweet
  - Weight the count, so frequent words get lower weight, i.e., inverse document matrix.
  - Normalize the vectors to unit length, to abstract from the original text length.
- **Term Document and Inverse Document Frequency:** TF-IDF stands for term frequency-inverse document frequency, and the tf-idf weight is a weight often used in information retrieval and text mining. The importance 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.
  - **Term Frequency:** Term Frequency, which measures how frequently a term occurs in a document. Since every document is different in length, it is possible that a term would appear much more times in long documents than shorter ones. Term Frequency(word,w) = No. of times w appears in a document / Total no. of words in a document
  - **Inverse Term Document:** ITD is used to determine how important a particular word is.
    IDF(t) = loge(Total number of documents / No. of documents with term t in it)

Model Training

Bayes Classification: Bayesian classification is based on Bayes' Theorem. Bayes classifiers are the statistical classifiers. Bayesian classifiers can predict class membership probabilities such as the probability that a given tuple belongs to a particular class.

Important terms in Bayes Classifier:

- Prior Probability: P(A)
- Likelihood: P(B|A)
- Evidence: P(B)
- Posterior Probability: P(A|B)

By Bayes Theorem we have:

Posterior Probability = (Likelihood * Prior) / (Evidence)

Naïve Bayes Classification:

The Naïve bayes classification is Bayes Classification with assumptions. It assumes that each feature makes an Independent and Equal contribution to the outcome.

Pipeline

In this step we created a machine learning pipeline to manage all the preprocessing steps. Pipeline workflow is created which lets all the preprocessing steps, model and grid search, cross validation to execute in one step.
Pipelines make code more readable, allow cross validation on the model workflow, help avoid leaking statistics from your test data into the trained model in cross-validation, by ensuring that the same samples are used to train the transformers and predictors.

Cross validation: Cross validation is done using k-fold cross validation. We used grid search CV to cross validate the training process and to select the best hyperparameters.GridSearchCV allows you to define a ParameterGrid with hyperparameter configuration values to iterate over. All combinations are tested and scored and the best model returned.

Grid Search CV: We passed a combined 8 parameters to the GridsearchCV object and 10 folds for the cross validation which means that for every parameter combination, the grid will run 10 different iterations with a different test set every time.
After performing model training on all the combinations of hyperparameters the Grid Search CV returns the best performing model which we use for classifying our tweet data on the basis of sentiment.

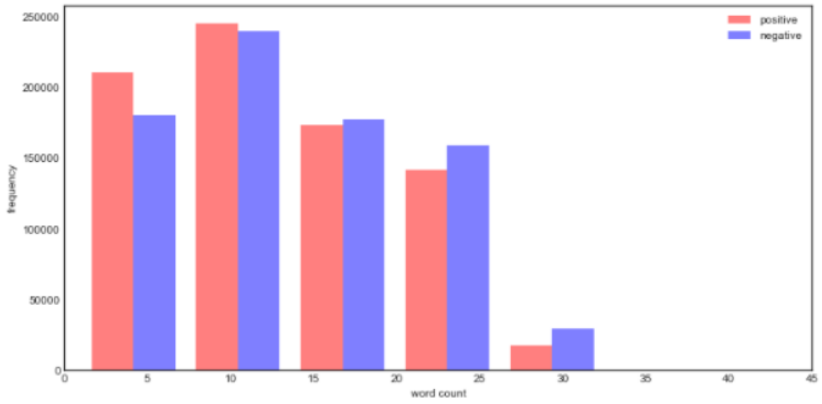
Model Evaluation: In this step we evaluated our model.

- **Accuracy Score:** We calculated accuracy score on the test dataset.
- **Confusion matrix:** We obtained a confusion matrix using actual and predicted labels. Confusion matrix lets us know the true positive, true negative, false positive and false negatives of our model prediction.We can calculate precision, accuracy, sensitivity, specificity using confusion matrix.
- **Classification report:** Classification report has Precision, Recall and f1-score of the model.

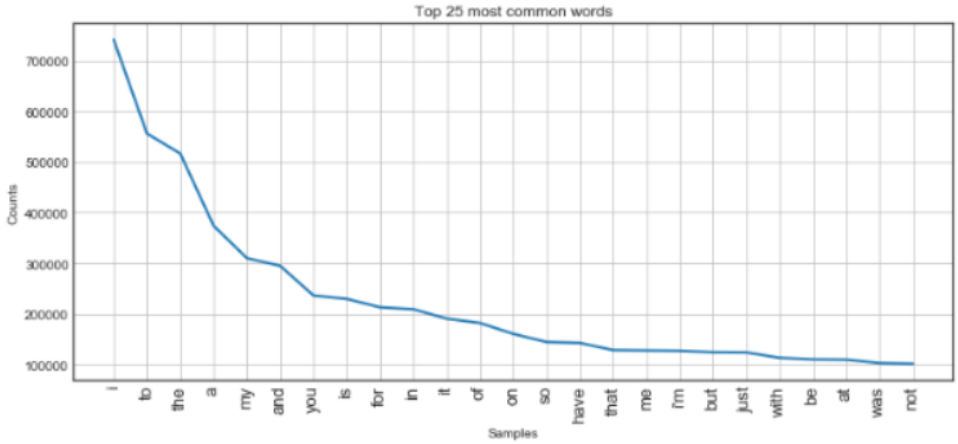
Model Testing in Real World: In this step we tested our sentiment analysis model on data acquired through twitter API.

Results and Conclusion:

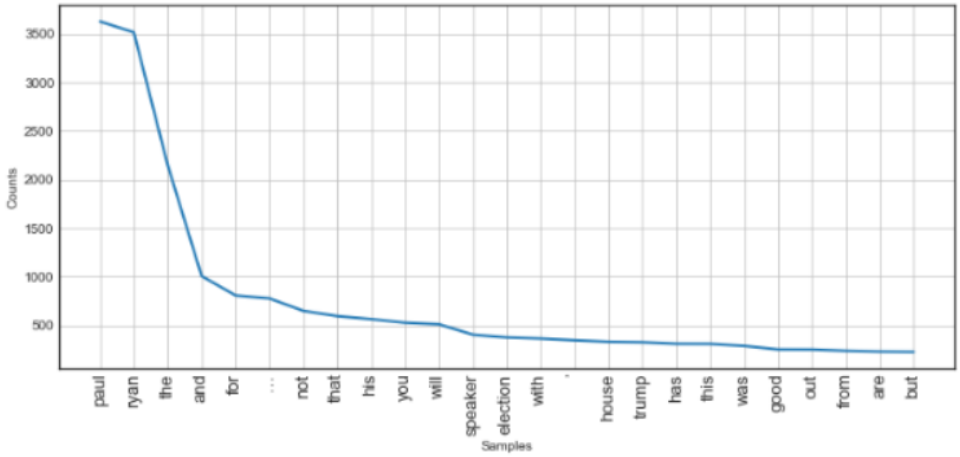
Word Count for positive and negative classification:



Top 25 most frequent words before Preprocessing:



Top words after Preprocessing:



Word Cloud:

Model Classification Report:

	precision	recall	f1-score	support
0	0.80	0.90	0.84	713
1	0.63	0.44	0.51	287
avg / total	0.75	0.76	0.75	1000

- **Precision:** 99% for label 0 and 93% for label 1. This number tells us what proportion of the labels were predicted correctly out of the total predictions for that class.
- **Recall:** 90% for label 0 and 100% for label 1. This is the number of correct predictions out of true labels for that class.
- **f1 -score:** This is the weighted average of precision and recall for that class. It generally gives the bigger picture of how the model is performing for that label and obviously the higher this number is the better. 94 % for negative sentiment and 96% for positive sentiment.

Accuracy: 76.4 % : The model correctly classified 76.4 percent of data correctly into positive and negative sentiment.

Confusion Matrix: [[639 74
162 125]]

Interpretation of Confusion matrix:

- The model predicted 553 labels correctly as Negative and 799 labels correctly as Positive.

- We also got 4 labels predicted as Positives even though they are Negatives (False Negatives).
- Another thing we can tell from the CM is that the model predicted 62 labels as Negatives but they turned out to be Positives (False Positives).

References:

- https://monkeylearn.com/blog/sentiment-analysis-of-twitter
- https://www.analyticsvidhya.com/blog/2018/07/hands-on-sentiment-analysis-dataset-python/
- https://developer.twitter.com/en/docs/tutorials/how-to-analyze-the-sentiment-of-your-own-tweets
- https://scikit-learn.org/stable/modules/generated/sklearn.pipeline.Pipeline.html
- https://www.digitalvidya.com/blog/twitter-sentiment-analysis-introduction-and-techniques/