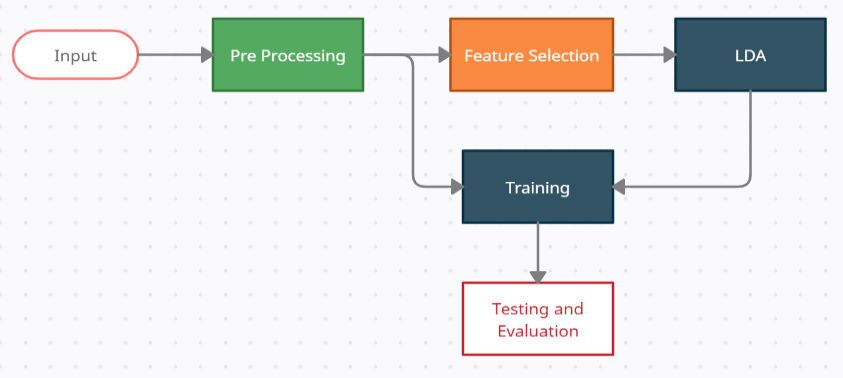
CSL2050 Twitter Sentiment Analysis

Abhishek Jamhoriya(B19CSE003), Aditya Gadhavi(B19EE004), Ajay Meena(B19EE005)

## **Introduction**

With the rise of free speech on social media the need for fast and reliable sentiment analysis becomes important so we can filter hate speech on the platform. In this project we have implemented an end to end machine learning pipeline to achieve the same.

**Machine learning pipeline**



The pipeline starts with the provided data which is preprocessed and ready for training. A separate pipeline applies feature selection and LDA to improve the performance of the model. This pipeline is executed for three classifiers and their results are compared. This pipeline is made by Aditya Gadhavi(B19EE004)

**Preprocessing**

The data has six columns, four of which (date, username, query and id) are dropped as they don’t contain any valuable information. Further processing is done on the tweets. The preprocessing includes:

* Converting all words to lowercase
* Removing HTML, usernames and links
* Removing punctuation, chat words and stop words
* Removing words with uncommon lengths and numbers
* Converting all words to their stem (Stemming)

Finally all the null columns are removed and the data is saved as a csv file. We decided to use Stemming only because sentiment of a word is more closely related to its stem and using lemmatization takes 28 minutes and doesn't reduce any extra features. Along with these we decided to filter words of length less than 2 and greater than 14 because that includes 0.9 percent of total words in english and are mostly spelling mistakes, scientific terms etc.

Stemming is implemented by the natural language toolkit library. The first four steps take around 36 seconds each and Stemming takes 240 seconds on average. After all the preprocessing only 8000 tweets are left null which are removed.The preprocessing is implemented by Abhishek Jamhoriya(B19EE004).

**Feature Selection and LDA**

Here we decided to use the feature filtering method as we have a large number of features to work with, so feature selection methods such as ‘Exhaustive’ selection will take too much calculation space and might not even work for a larger dataset sample. Similarly ‘Forward/Backward’ feature selection methods. Feature Selection is implemented using the Variance Threshold class of sklearn library and it takes 3 seconds to implement.

Linear Discriminant Analysis or LDA is a supervised dimensionality reduction technique which is commonly used in text classification. We chose LDA above PCA because given the number of features it was more important to increase the inter class distance rather than maximising the variance.LDA takes 520 seconds for 5 percent data.

Moreover we used both FS and LDA because we wanted a method to wisely shortlist features and also increase their separability. FS and LDA are implemented by Ajay Meena(B19EE005).

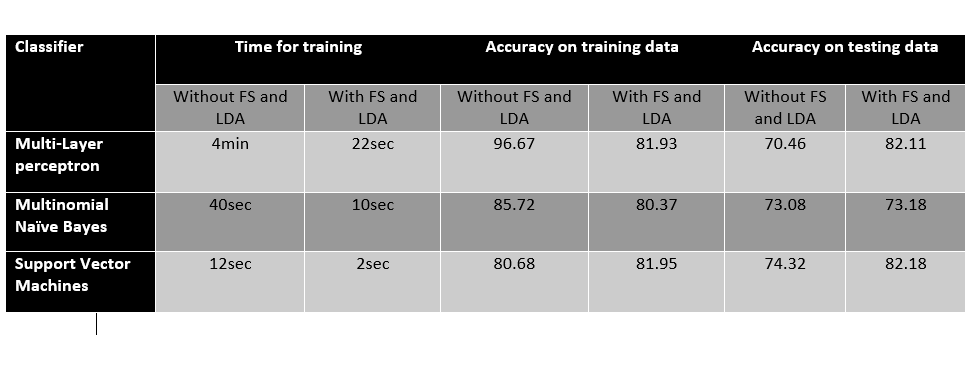
**Classification Models**

We use Tfidf Vectorizer to convert text data to an array using the TfidfVEctorizer in sklearn. This is a universal step and allows finer control over the number of features in the data.

We have used three classification models namely SVM, Multi-Layer perceptron and Multinomial Naive Bayes. Support Vector Machine is the most widely used text classifier since it works well with high dimensional data. MLP is most used in natural language processing but suffers from overfitting. Another widely used classifier is Multinomial NB.

SVM is implemented using the linear SVM classifier of sklearn library. MLP and Multinomial naive bayes are also implemented using sklearn library. SVM takes 12 seconds to train where Multinomial NB takes 40 seconds. The most costly is MLP which takes 4 minutes to train. The performance is greatly improved after FS and LDA. After FS and LDA SVM takes 2 seconds to train where Multinomial NB takes 10 seconds and MLP takes 22 seconds. All models are implemented by Aditya Gadhavi(B19EE004).

**Performance report**



This report is made by Aditya Gadhavi (B19EE004)

**Performance Conclusion**

As we can see FS and LDA not only improve time taken for training but also improve accuracy of the models significantly. Multi-Layer perceptron suffers significantly from training time and overfitting but performs significantly better when used along with FS and LDA.

Multinomial NB is the worst performing of the three in accuracy because we cannot use LDA with it since LDA outputs negative values and z scoring negative values will defeat the purpose of applying it.

Support Vector Machines are the best performing classifier we used, taking the least amount of time with the best accuracy and minimum overfitting on training data.

Note that the training is taken out on only 3 percent of the entire database at best as the training time of Multi Layer Perceptron increases exponentially with the dataset and dimensions. At 10 percent of the database the MLP takes more than 90 minutes to train which is unfeasible to repeatedly train and check out randomness.

**Experimental Finding**

We also used other pipelines and classifiers, tuning the parameters for our classifiers and we found that. Random forest classifier performs well at 68 percent accuracy with 2 minutes training time. RBF svm machines perform astonishing 90 percent accurately without FS or LDA but take even more time than MLP and are even less feasible. Also bagging various classifiers did not report any meaningful increase in accuracy with Random forest with 100 estimators report 70 percent accuracy and Linear SVM with 12 estimators reporting 75 percent accuracy.

These findings are reported by Abhishek Jamhoriya(B19CSE003).s