Twitter Sentiment Analysis

Krishna Patil Saumitra Agrawal Harshit Goyal Mukund Gupta
Aarohi Dharmadhikari Gouri Patidar

April 21, 2024

CSL2050:Pattern Recognition and Machine Learning - Course Project
IIT Jodhpur

Abstract

A text can provide extensive insight into the sentiment conveyed by the author. Consequently, sentiment analysis derived from textual input is a well-established problem statement in the domains of machine learning and natural language processing (NLP). In this project, we aimed to address this challenge by employing a conventional machine learning methodology to analyze a dataset of over 30,000 tweets from Twitter. The objective was to implement and analyze different classifiers on the Sentiment Analysis Dataset, utilizing various preprocessing techniques (LDA & PCA) and tokenizers (BERTTokenizer and TFIDtokenizer). The project explored and thoroughly analyzed the performance of classifiers- Decision Trees, Random Forests, SVMs, Naive-Bayes, Perceptron and Logistic Regression, optimizing each to yield maximum accuracy by modifying any parameters or hyperparameter tuning. Finally, ensemble learning was employed to achieve optimum performance of the model. It was observed that TFID gave the maximum accuracy, which reached as high as 71.5

Contents

1 Introduction

Language serves as a medium for the exchange of both knowledge and emotions. A text can provide extensive insight into the sentiment conveyed by the author. Therefore, sentiment analysis derived from textual input is a well-established problem statement in the domains of machine learning and natural language processing (NLP). In this project, we aimed to address this challenge by employing a conventional machine learning methodology to analyze a dataset of over 30,000 tweets from Twitter. We provide a brief overview of our approach, highlighting key steps such as data preprocessing and the utilization of traditional ML algorithms for classification tasks. The primary focus was on evaluating performance across various methodologies, including the comparison of different vectorization techniques and the application of ensemble learning techniques to enhance accuracy.

The subsequent sections of this report will delve into the specific methodologies employed and experimental results obtained.

1.1 Citations

1. Kaggle Sentiment Analysis Dataset: https://www.kaggle.com/datasets/abhi8923shriv/sentiment-analysis-dataset?select=train.csv
2. Scikit-learn documentation: https://scikit-learn.org/stable/
3. Numpy documentation: https://numpy.org/doc/
4. Pandas documentation: https://pandas.pydata.org/docs/
5. Joblib documentation: https://joblib.readthedocs.io/en/stable/
6. Complement Naive Bayes (CNB) algorithm: https://www.geeksforgeeks.org/complement-naive-bayes-cnb-algorithm/
7. Reference video for data-preprocessing: https://www.youtube.com/watch?v=4YGkfAd2iXM
8. Reference video for data-preprocessing: https://www.youtube.com/watch?v=RLfUyn3HoaE
9. Google: http://www.google.com
10. LLM (ChatGPT 3.5) as a search engine and debugging tool: https://chat.openai.com/
11. LaTeX editor: https://www.overleaf.com/

1.2 Figures

1.3 Tables

2 Approaches Tried

The project involved critical comparative analysis of different approaches to achieve maximum accuracy on the test data. We experimented with various techniques at multiple levels such as data preprocessing(vectorization), dimensionality reduction, classification and prediction.

2.1 Text Vectorization:

After some basic EDA, data cleaning and preprocessing, the text data had to be converted to numeric features. We tried different approaches for text vectorization. These include::

- BertTokenizer :It returns a 768 dimensional vector containing the BERT encoding of the text. The BertTokenizer is a specialized tool designed to tokenize text for models built on BERT (Bidirectional Encoder Representations from Transformers), an advanced model for natural language understanding. It segments input text into individual tokens, including special tokens like [CLS] for classification tasks and [SEP] to denote sentence boundaries. Each token is then mapped to its corresponding index in the BERT vocabulary, resulting in a sequence of numerical indices representing the input text.
- Principal Component Analysis(PCA) : Since the BertTokenizer results in a very large (768) dimensional vector, PCA was applied for dimensionality reduction. After performing grid search, n_components = 100 was considered to be a good tradeoff between performance and compute power.
- Linear Discriminant Analysis(LDA) : LDA was applied as another approach of dimensionality reduction and class separation on the result of the BertTokenizer.
- Tfid Vectorizer:The TfidfVectorizer is a classic technique used for vectorizing text data based on the Term Frequency Inverse Document Frequency (TF IDF) formula. It first tokenizes the input text into individual words or terms. Then, it calculates the TF IDF weight for each term, which measures the importance of a term in a document relative to its frequency in the entire corpus of documents. This process results in a numerical representation of each document, where each dimension corresponds to a unique term in the vocabulary, and the value in each dimension represents the TF IDF weight of that term in the document.

2.2 Machine Learning Classifier Models:

The objective of the project being use of classical ML approaches, the following classifier models were used to make predictions on the test data:

- Decision Tree: Decision Tree is a versatile algorithm that recursively splits the dataset into subsets based on the most significant feature, aiming to maximize information gain or minimize impurity at each node. The depth of the tree was varied to get best results.
- Random Forest: Forest is an ensemble learning technique that constructs multiple decision trees and aggregates their predictions through voting or averaging. By introducing randomness in feature selection and bootstrapping, Random Forest mitigates overfitting and often yields robust performance across various datasets. The number of trees was varied to get a good trade-off between accuracy and computational cost.
- Support Vector Machines (SVM): SVM is a powerful algorithm for classification tasks that aims to find the hyperplane that best separates different classes in the feature space. It works by maximizing the margin between the closest data points of different classes and is effective in high-dimensional spaces, making it suitable for complex classification problems. Different kernels were tried out to achieve maximum accuracy.
- Naive Bayes: Naive Bayes is a simple yet effective probabilistic classifier based on Bayes' theorem with an assumption of independence between features. Despite its simplistic nature, Naive Bayes can perform well on text classification and other tasks with high dimensional feature spaces. Laplace smoothing parameter was varied using grid_search and accuracy was compared.

- Perceptron Learning Algorithm: The Perceptron Learning Algorithm is one of the earliest forms of neural networks, consisting of a single-layer linear classifier. It iteratively updates weights based on misclassified samples to minimize classification errors, making it suitable for binary classification tasks. For multiclass classification problems like ours, it uses the 'one-vs-rest' strategy.

- Logistic Regression: Logistic Regression is a linear classification algorithm that models the probability of belonging to a certain class using a logistic function. It's widely used for binary classification tasks and can be extended to handle multiclass classification with techniques like one-vs-rest or softmax regression.

A comparative study of results obtained from above classifiers was performed, and the models giving the best results were clubbed together to create an ensemble model. Additionally, boosting was applied to this ensemble to get improved results.

3 Experiments and Results

3.1 Dataset Description

Kaggle's "Sentiment Analysis Dataset" (link: https://www.kaggle.com/datasets/abhi8923shriv/sentiment-analysis-dataset , file used:train.csv, test.csv) was used for training and testing the dataset. The dataset consists of the fields: 'UserID', 'text', 'sentiment', 'Time of Tweet', 'Age of User', 'Country', 'Population', 'Land' and 'Density'. It conprises a total of 31015 data-points. A short analysis of the datasets explaining its structure and features is provided below:

Data Sample:

	textID	text	sentiment	Time of Tweet	Age of User	Country	Population -2020	Land Area (Kmï¿½)	Density (P/Kmï¿½)
0	cb774db0d1	I'd have responded, if I were going	neutral	morning	0-20	Afghanistan	38928346	652860.0	60
1	549e992a42	Sooo SAD I will miss you here in San Diego!!!	negative	noon	21-30	Albania	2877797	27400.0	105

Dataset info:

```
RangeIndex: 31015 entries, 0 to 31014
Data columns (total 9 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   textID              31015 non-null  object
 1   text                31014 non-null  object
 2   sentiment           31015 non-null  object
 3   Time of Tweet       31015 non-null  object
 4   Age of User         31015 non-null  object
 5   Country             31015 non-null  object
 6   Population -2020    31015 non-null  int64
 7   Land Area (Kmï¿½)    31015 non-null  float64
 8   Density (P/Kmï¿½)    31015 non-null  int64
```

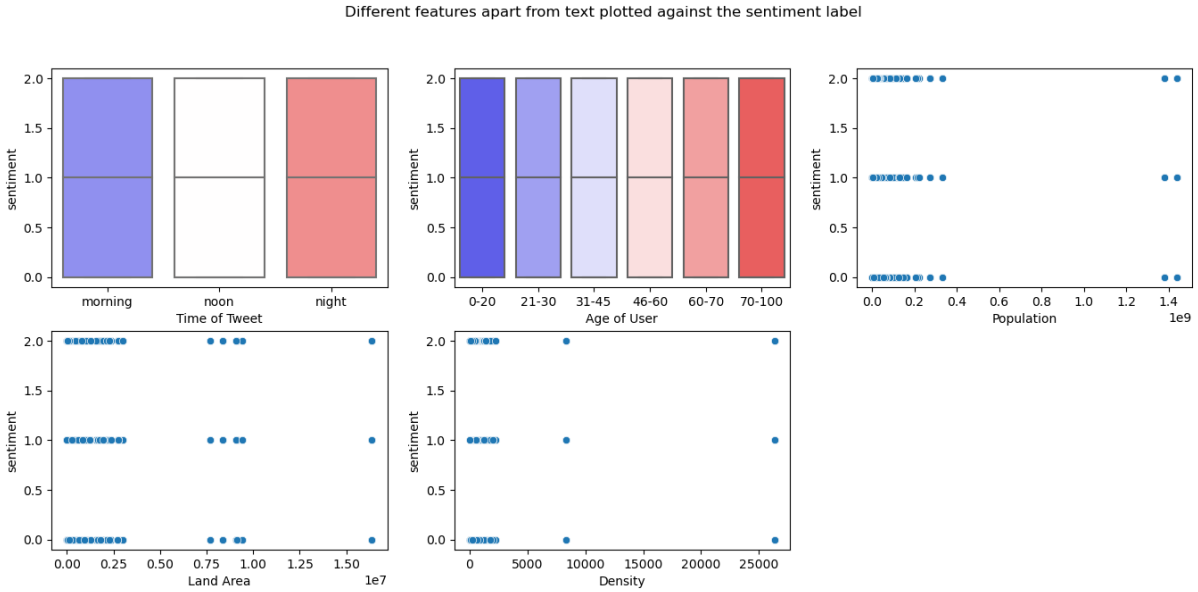
Sentiment Analysis Dataset: Features
Continunous features: 'text ID, text, Population, Land Area, Density'
Categorical features with value counts:

```
neutral     12548
positive     9685
negative     8782
Name: sentiment, dtype: int64
```

```
morning    10339
noon       10338
night      10338
Name: Time of Tweet, dtype: int64
```

```
0-20     5171
21-30    5170
31-45    5170
46-60    5168
60-70    5168
70-100   5168
Name: Age of User, dtype: int64
```

```
Afghanistan          169
Ecuador              169
Chile                169
China                169
Colombia             169
                    ...
Singapore            144
...
Slovenia             144
Solomon Islands      144
Zimbabwe             144
Name: Country, Length: 195, dtype: int64
```



Brief dataset analysis

'Text ID' and 'Text' are considered as non-categorical features since there are too many unique values to be of categorical importance. The target variable is the 'sentiment' feature with three possible classes: 'positive', 'negative', and 'neutral'. Text ID will be neglected while training the ML models since it

does not affect the sentiment of the tweet. Additionally, since the dataset is so evenly distributed, the fields 'Country' 'Population', 'Land Area' and 'Density' are also ignored since it won't really affect the prediction as all countries samples have same distribution of each sentiment. These fields also contained heavy outliers which would impair results.

Hence, only 'text', 'age of user' and 'time of tweet' are considered for training the model.

3.2 Data Pre Processing:

Before applying the classifiers, the data had to be heavily cleaned and text had to be vectorized to be converted to numeric values. Following data cleaning and preprocessing approaches were taken:

- Dropping Unwanted Features: Only the 'text', 'Time of Tweet' and 'Age of User' and 'sentiment' columns were left and all others were dropped since they were not to be used for training.
- Processing categorical features: 'Time of Tweet' is a categorical feature with possible values 'morning', 'noon' and 'night'. These values were encoded to 0,1 and 2 respectively for further training.
- 'Age of User' was also a categorical variable with possible values as range in years (eg- 0-20, 21-30, 60-70, etc). Every range entry was replaced by its corresponding mean value for simplicity.
- Text Processing:The text-column consists of raw tweet text, with the emotes removed. The text was first cleaned by removing all non alphabetical symbols (numbers, punctuations, @, etc). All the characters in the text were converted to lower case for uniformity. The processed text was then stemmed to remove english stopwords using the PorterStemmer module.

The frequently occurring words for each sentiment were visualised using a wordcloud.

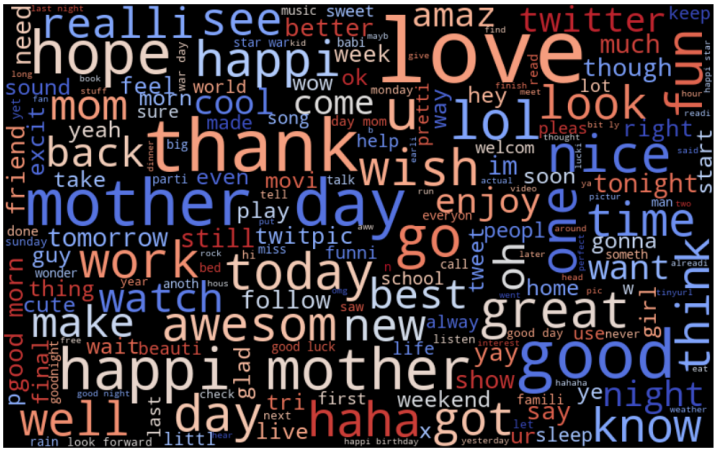


Fig-2 : Wordcloud for sentiment = positive

Fig-3 : Wordcloud for sentiment = negative

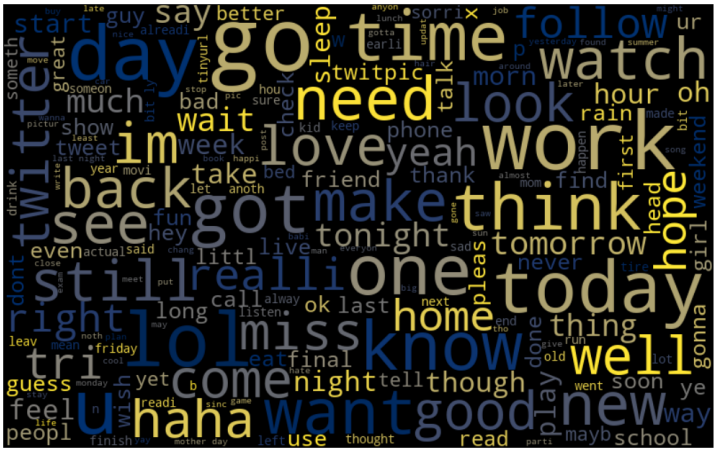


Fig-4 : Wordcloud for sentiment = neutral

3.3 Text Vectorization

After the above preprocessing, the text data had to be converted to numeric features. We tried different approaches for text vectorization and the later sections provide a comparative analysis of performance of the ML models on the differently processed datasets.

As mentioned in the above section, the following techniques were implemented for text vectorization:

- BertTokenizer: Converts the dataset to 31015 x 771 dataframe with 'time of tweet', 'age of user', 768 dimensional bert encoding and 'sentiment'.
- Principal Component Analysis (PCA): Since the BertTokenizer results in a very large (768) dimensional vector, PCA was applied for dimensionality reduction. After performing grid search, n_components = 100 was considered to be a good trade-off between performance and compute power.
- Linear Discriminant Analysis (LDA) : LDA was applied as another approach of dimensionality reduction and class separation on the result of the BertTokenizer.
- TfidVectorizer: It converts every datapoint to a large dimensional sparse matrix.

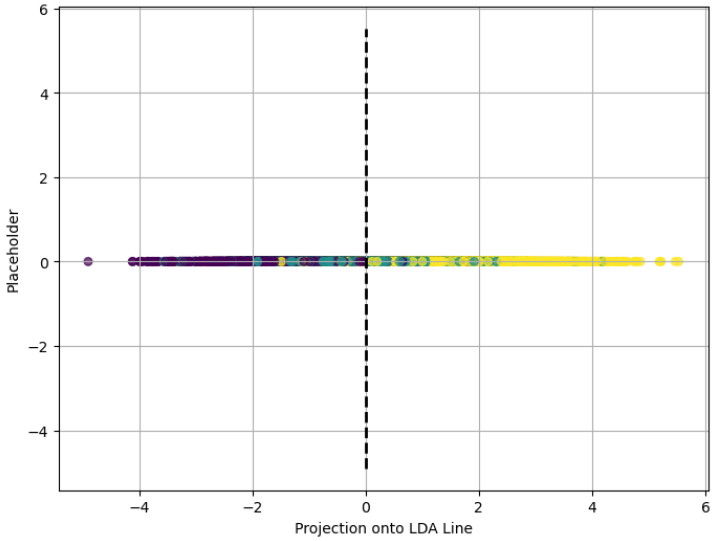


Fig-5 : Data Projection after applying LDA

3.4 Classification

Four datasets were prepared by using different vectorization approaches and were tested separately to determine which one yielded maximum accuracy. The datasets were referred to as 'BERT', 'PCA', 'LDA', and 'Tfid'.

All classification models used below are imported from the scikit-learn python library. The different classification models implemented are as follows:

3.4.1 Decision tree:

Observations:

The decision tree classifier was applied to the four datasets. The parameter 'max_depth' of the tree was varied between (1,20). The following graphs represent the variation in accuracy obtained as a function of model complexity (depth of trees) for each of the datasets.

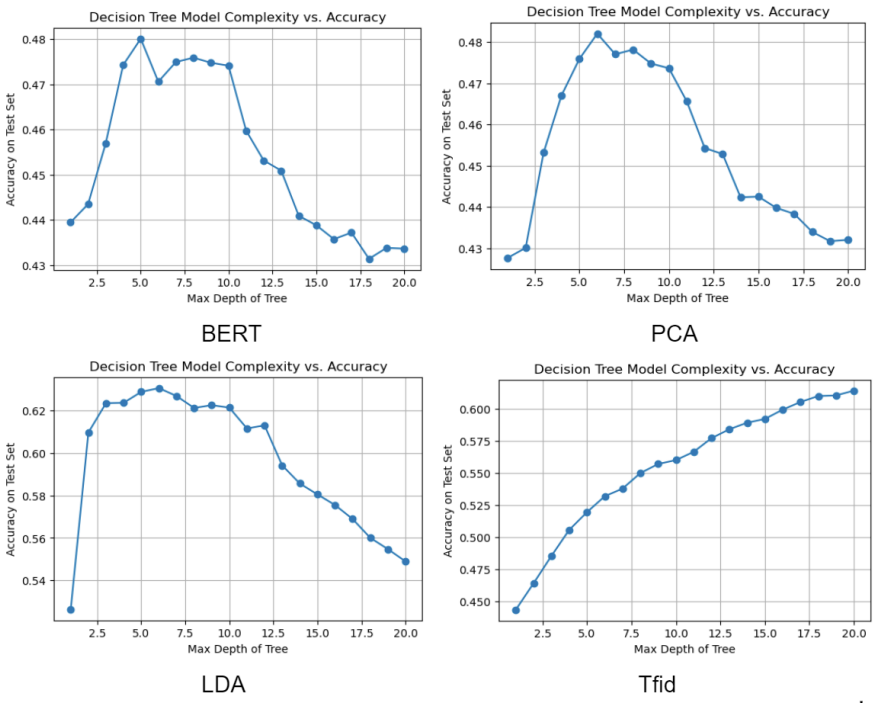


Fig-6 : max depth vs accuracy score for decision tree classifier for different datasets.

The following bar chart summarizes the maximum accuracy obtained for each dataset using the Decision Tree classifier.

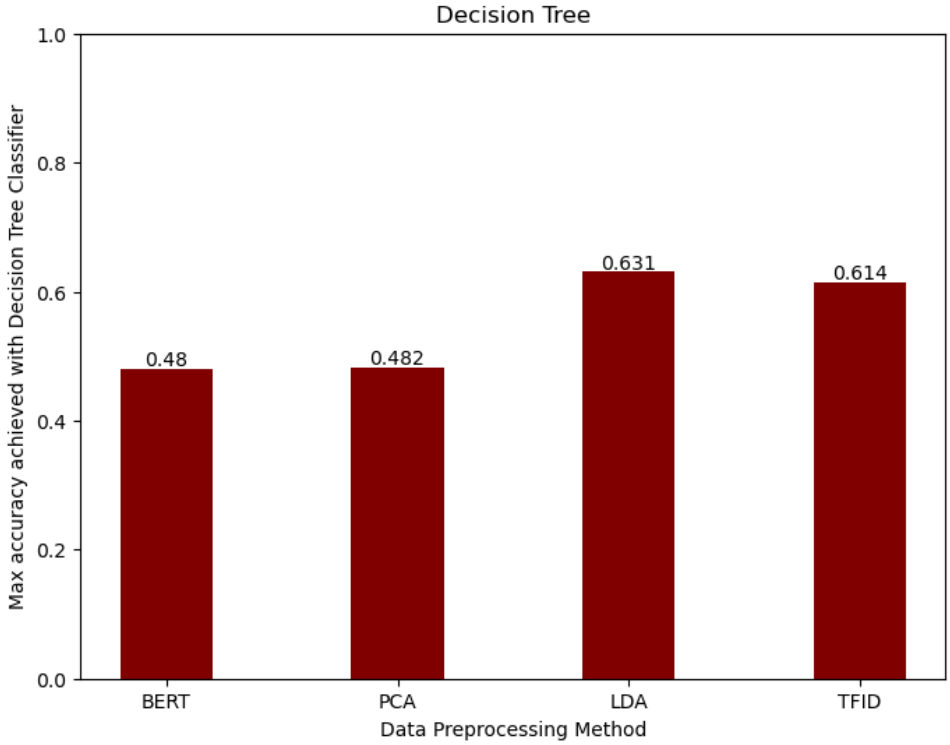


Fig-7 : Bar graph showing best performance of decision tree for the datasets.

Dataset	Depth	Class	Precision	Recall	F1-score	Accuracy
BERT	5	0 (Neg)	0.48	0.21	0.30	0.48
		1 (Neu)	0.47	0.71	0.56	
		2 (Pos)	0.52	0.42	0.46	
PCA	6	0 (Neg)	0.44	0.36	0.39	0.482
		1 (Neu)	0.48	0.58	0.53	
		2 (Pos)	0.49	0.44	0.46	
LDA	6	0 (Neg)	0.64	0.55	0.59	0.631
		1 (Neu)	0.60	0.68	0.64	
		2 (Pos)	0.68	0.64	0.66	
TFID	20	0 (Neg)	0.79	0.29	0.43	0.614
		1 (Neu)	0.53	0.87	0.66	
		2 (Pos)	0.78	0.56	0.65	

Table-1 : Classification report for decision tree on different datasets.

Inferences Drawn:

- The model performed poorly on the datasets with large number of dense features, for example BERT and PCA, with an accuracy less than 0.5. This might be because of overfitting or reduced feature importance, since these datasets have 100+ features.
- It performed best on LDA, given its low dimensional size and the fact that the projections are calculated to maximize distance between the datapoints, to make classification easier.
- The model performed decently on the TFID dataset, but we observe that the max_depth for the TFID dataset was much greater than the other datasets. This can be due to the nature of TFID data, a sparse matrix. With a large max_depth, the decision tree can potentially create highly complex splits to partition the feature space effectively. In the case of sparse data, finding optimal splits that effectively separate different classes or categories may require exploring a large number of possible split points, leading to better performance on higher max_depth.

3.4.2 Random Forest Classifier

Random Forest classifier was applied to the datasets and accuracies were reported for several values of the number of trees used for each dataset.

Observations: The values used for grid search on number of forests were 1, 5, 10, 20, 50, 100, 200, 500. The following charts describe the variation of accuracy as a function of the number of trees used for each of the 4 datasets.

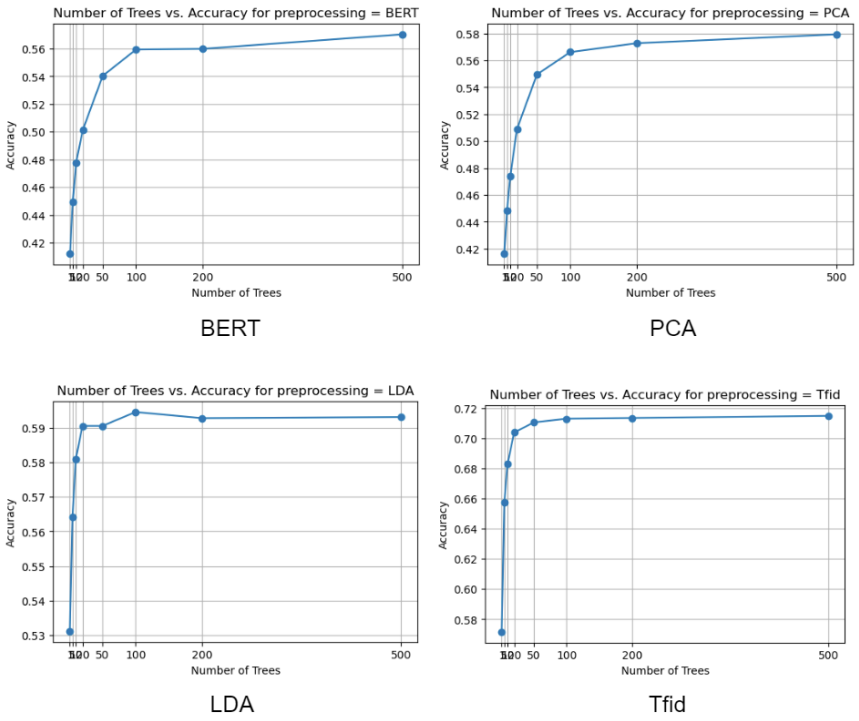


Fig-8 : Number of trees vs accuracy_score of random forest classifier for different datasets.

Dataset	No. of trees for maximum accuracy	Class	Precision	Recall	F1-score	Accuracy
BERT	500	0 (Neg)	0.67	0.30	0.42	0.57
		1 (Neu)	0.52	0.79	0.63	
		2 (Pos)	0.64	0.53	0.58	
PCA	500	0 (Neg)	0.66	0.31	0.42	0.579
		1 (Neu)	0.51	0.80	0.63	
		2 (Pos)	0.67	0.49	0.56	
LDA	100	0 (Neg)	0.59	0.54	0.57	0.595
		1 (Neu)	0.58	0.61	0.60	
		2 (Pos)	0.62	0.62	0.62	
TFID	500	0 (Neg)	0.75	0.62	0.68	0.715
		1 (Neu)	0.67	0.75	0.71	
		2 (Pos)	0.74	0.76	0.75	

Table-2 : Classification report for random forest on different datasets.

The following bar graph summarizes the maximum accuracy obtained using Random Forest for each of the datasets.

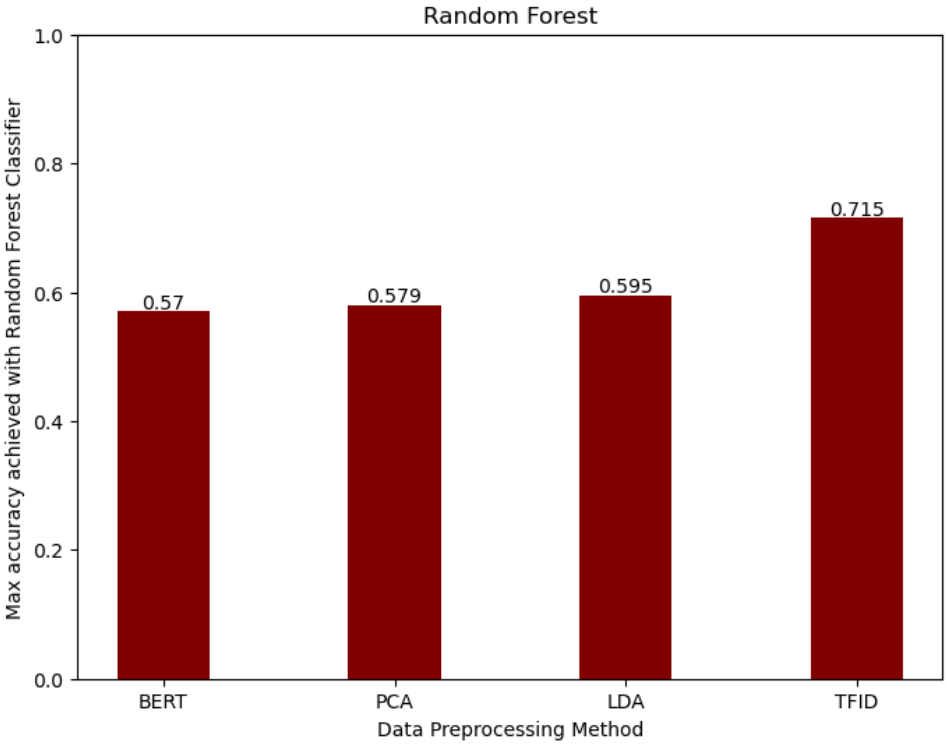


Fig-9 : Bar graph showing best performance of random forest for the datasets.

Inference Drawn:

- For this model, the variation accuracy with number of trees is more or less of similar nature since more number of trees would naturally provide better predictions.
- For LDA it is observed that the variation peaks at n=100. This may be due to the lower dimensionality compared to other datasets. The accuracy on LDA processed dataset seems to have dropped compared to normal decision tree. This may be because we did not vary the max_depths of trees in the random forest, which is by default set to 10. From the fig() it is evident that the accuracy drops after max_depth=6 for this dataset.
- While the model performed on all the bert encoded and further preprocessed datasets to give an accuracy around 0.57-0.58, the performed much better on the Tfid dataset with an accuracy of 0.715. This can be due to high dimensional data with lots of zero values.

3.4.3 Support Vector Machines(SVM)

All four aforementioned datasets were trained over by 4 different kernel selections :

1.LinearSVC 2.LinearKernelSVC (linear kernel used) 3.PolyKernelSVC (polynomial kernel used) 4.RBFKernelSVC (RBF kernel used)

Hence, 16 different combinations of training data and kernel were tried and following results were obtained.

Dataset	Kernel used	Accuracy obtained
BERT	None	0.5725
	Linear	0.6320
	Polynomial	0.4653
	RBF	0.6149
PCA	None	0.5731
	Linear	0.6086
	Polynomial	0.4499
	RBF	0.6045
LDA	None	0.6369
	Linear	0.6382
	Polynomial	0.5060
	RBF	0.6399
Tfid	None	0.6766
	Linear	0.7134
	Polynomial	0.4601
	RBF	0.4111

Table-3 : Accuracy scores for different SVM kernel on different datasets.

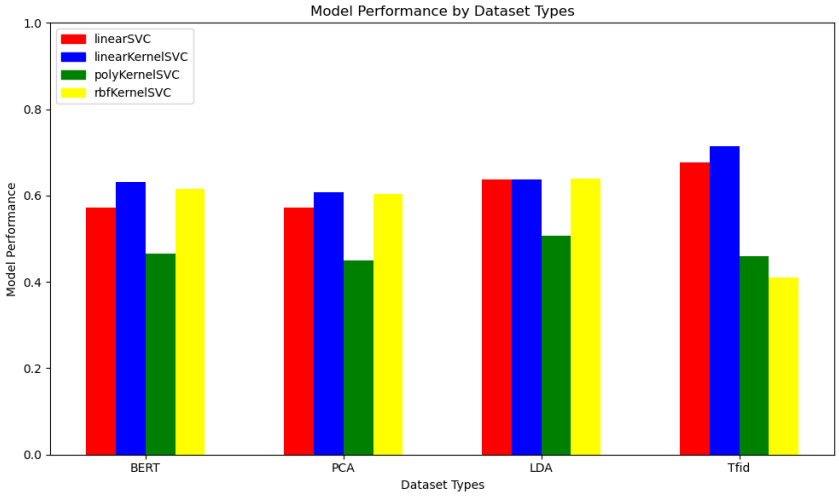


Fig-10 : Bar graph showing best performance of SVC kernels for the datasets.

In the above table and diagram, kernel = None means that the LinearSVC() model from sklearn was used. Inference Drawn

- The models LinearSVC() and SVC(kernel='linear') nearly give similar results for all datasets because internally they have almost same implementation, only the hyperparameter c is more precisely fit by the SVC model.
- Highest accuracy is achieved by the linear kernel, proving that the data is somewhat linearly separable.
- For the polynomial kernel, degree=7 and gamma value equal to 5 was used, which clearly did not perform well given the nature of the dataset.

3.4.4 Naive-Bayes Classifier

Although negative values hold importance for BERT classifier, but they can't be processed by the Naive-Bayes classifier since it works with probabilities. We handled it by taking modulus for negative values but adding one to positive values to distinguish the sign. We experimented with different types of naive

bayes models provided by sklearn library. The model complimentNB(), which stands for compliment naive bayes was found to provide the best results. It is basically a new type of classifier derived from the standard multinomial naive bayes classifier, which is better suited for handling imbalanced datasets. Like other Naive Bayes classifiers, ComplementNB assumes that features are conditionally independent given the class label. However, unlike the standard Naive Bayes algorithm, it models the distribution of features for each class by taking into account the complementary class. Then, hyperparameter tuning

was performed to identify the best values for alpha (laplace smoothing parameter) using gridsearch. Observations:

The following plots show the variation in accuracy as we change values of alpha for all the different processings of the dataset.

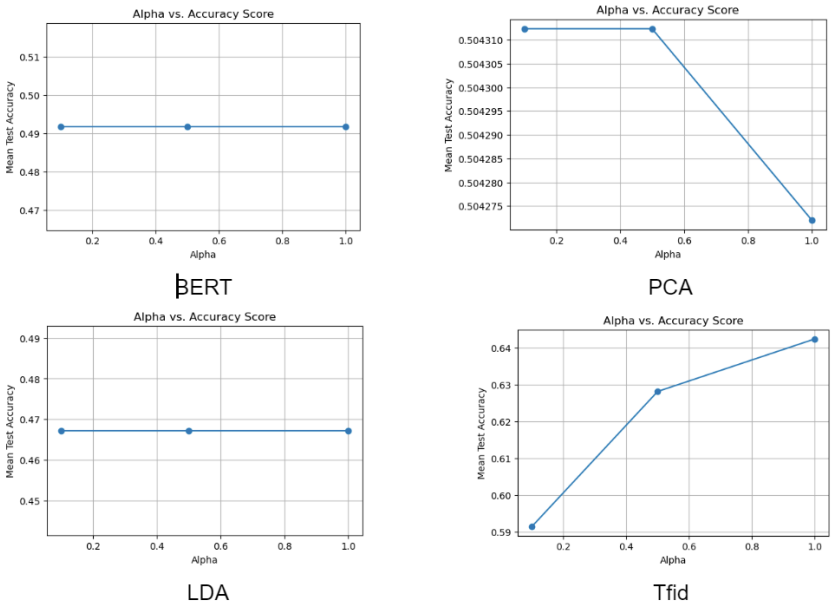


Fig-11 : Laplace smoothing parameter(alpha) vs accuracy score of naive bayes classifier for different datasets.

Dataset	Class	Precision	Recall	F1-score	Accuracy
BERT	0 (Neg)	0.47	0.30	0.36	0.487
	1 (Neu)	0.71	0.13	0.22	
	2 (Pos)	0.36	0.89	0.51	
PCA	0 (Neg)	0.49	0.43	0.45	0.507
	1 (Neu)	0.56	0.46	0.51	
	2 (Pos)	0.47	0.64	0.54	
LDA	0 (Neg)	0.51	0.77	0.61	0.452
	1 (Neu)	0.55	0.03	0.06	
	2 (Pos)	0.43	0.77	0.55	
TFID	0 (Neg)	0.66	0.62	0.64	0.648
	1 (Neu)	0.63	0.63	0.63	
	2 (Pos)	0.66	0.70	0.68	

Table-4 : Classification report for Compliment Naive Bayes on different datasets.

The following plot summarizes the maximum accuracy achieved on all the datasets.

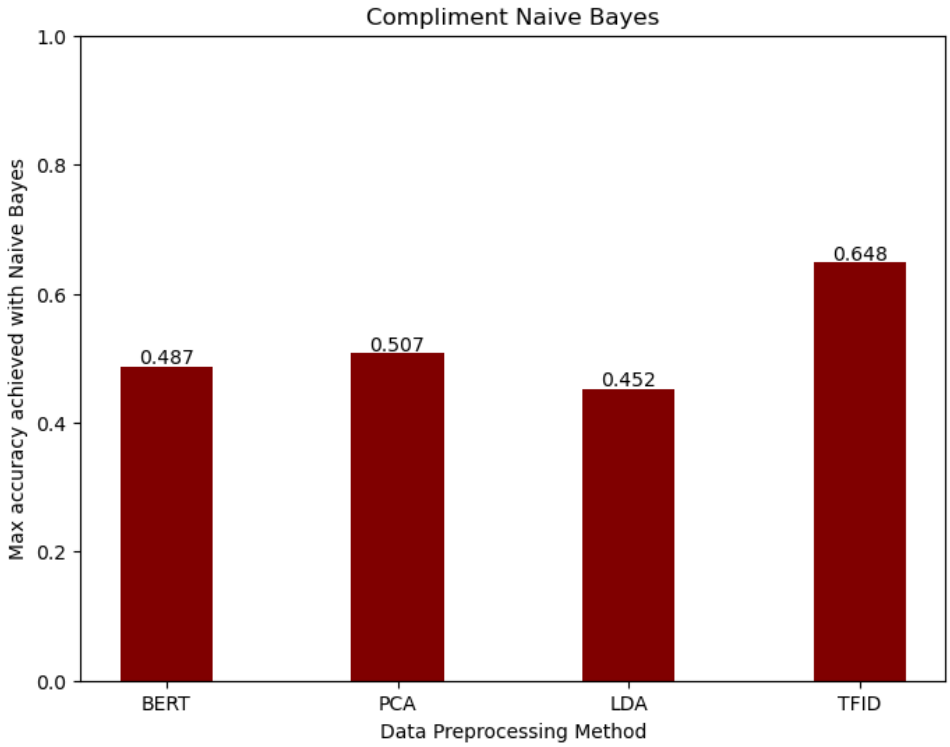


Fig-12 : Bar graph showing best performance of complimentNB classifier for the datasets.

Inference Drawn:

- We observe that the accuracy of model on PCA dataset drastically drops for alpha=1, this may be due to the fact that PCA dataset holds more heavily important features compared to other datasets, because of dimensionality reduction. In such a case, a larger alpha value, meaning aggressive smoothing could lead to dilution of importance of the features.

- We observe that the accuracy score is much higher for the Tfid data compared to the rest. This can be due to the independence assumptions naive bayes algorithms make. Although this assumption makes the model less complex, it may not be accurate for features derived from BERT tokenization, which captures intricate contextual connections between words and phrases. On the other hand, TF IDF vectorization assigns a separate value to each term based on how often it appears in the document corpus, which is more in line with the assumption of independence.

3.4.5 Perceptron

Observations: Firstly, using GridSearch, the optimum values of learning rate and tolerance were determined for application of the perceptron learning algorithm on all the datasets which were obtained as follows:

Dataset	Learning rate	Tolerance
BERT	0.01	0.01
PCA	0.1	0.01
LDA	0.1	0.0001
TFID	0.01	0.01

Table-5 : Best values of learning rate and tolerance for perceptron on different datasets.

Dataset	Class	Precision	Recall	F1-score	Accuracy
BERT	0 (Neg)	0.72	0.42	0.53	0.625
	1 (Neu)	0.57	0.76	0.65	
	2 (Pos)	0.66	0.64	0.65	
PCA	0 (Neg)	0.48	0.48	0.48	0.511
	1 (Neu)	0.55	0.56	0.56	
	2 (Pos)	0.59	0.56	0.58	
LDA	0 (Neg)	0.41	0.89	0.56	0.431
	1 (Neu)	0.45	0.42	0.44	
	2 (Pos)	0.92	0.01	0.02	
TFID	0 (Neg)	0.30	1.00	0.46	0.33
	1 (Neu)	0.50	0.00	0.00	
	2 (Pos)	0.90	0.15	0.26	

Table-6 : Classification report for perceptron on different datasets.

The following plot summarizes the maximum accuracy obtained on all the datasets.

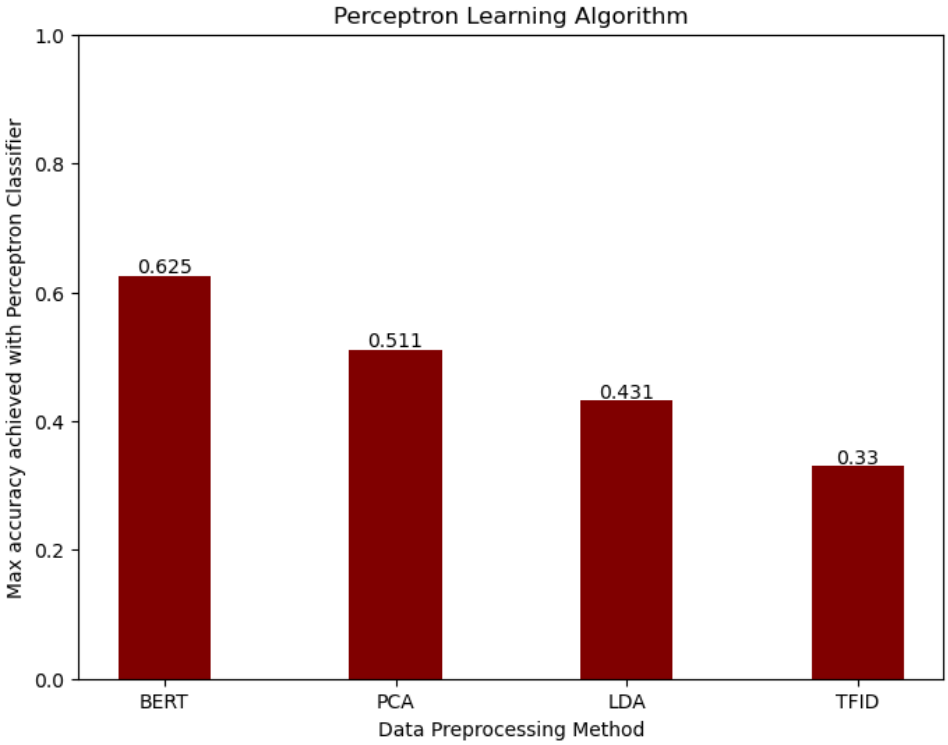


Fig-13 : Bar graph showing best performance of perceptron for the datasets.

Inference Drawn: Overall, the perceptron model performed poorly on all the datasets. This may be because it is more suited for linearly seperable binary datasets. Its strategy for multiclass classification is one-over-rest, which does not work well with the sparsely spread Tfid dataset. While sparse representations are memory-efficient, they might pose challenges for linear classifiers like the Perceptron, as they may not effectively capture complex patterns and relationships in the data. Sparse representations may also lead to sparse gradients during training, which can slow down convergence.

3.4.6 Logistic Regression

Observations:

The best value of 'C' (regularization parameter) was determined using grid search and then the logistic regression classifier was implemented on all the four datasets and following accuracies were reported. Regularization is a technique used to penalize large coefficients in the logistic regression model, effectively reducing model complexity and improving generalization performance on unseen data.The regularization parameter (C) in logistic regression controls the trade-off between fitting the training data well and preventing overfitting by penalizing large coefficients.

Dataset	Optimum C value
BERT	1
PCA	0.01
LDA	0.01
TFID	1000

Table-7 : Optimum regularization parameter for logistic regression on different datasets.

The above data was inferred from the following plots which show variation of mean cross-validation accuracies with C,

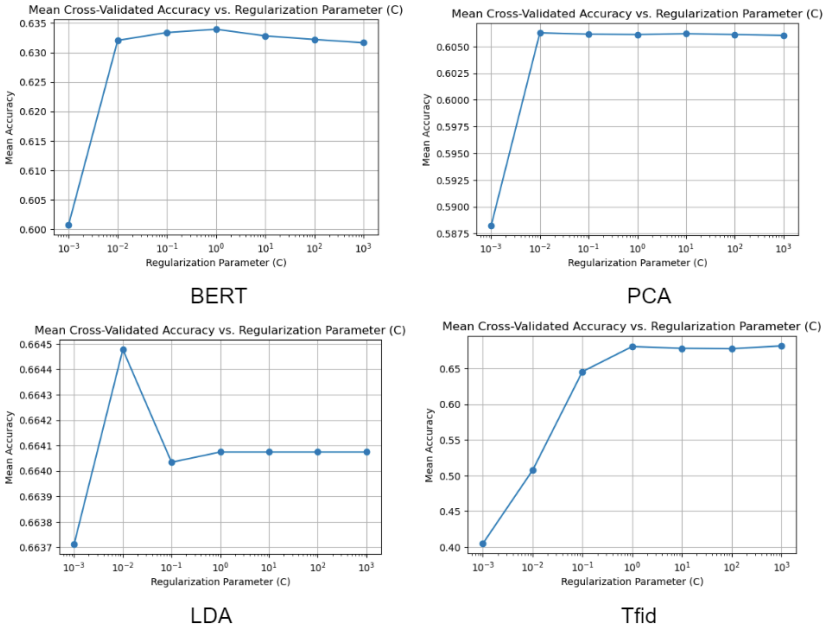


Fig-14 : Regularization parameter (C) vs accuracy score of logistic regression for different datasets.

The following table summarizes the quality parameters obtained by logistic regression on all the datasets.

Dataset	Class	Precision	Recall	F1-score	Accuracy
BERT	0 (Neg)	0.64	0.54	0.59	0.6336
	1 (Neu)	0.61	0.68	0.64	
	2 (Pos)	0.66	0.65	0.66	
PCA	0 (Neg)	0.58	0.46	0.51	0.6115
	1 (Neu)	0.56	0.64	0.60	
	2 (Pos)	0.60	0.61	0.61	
LDA	0 (Neg)	0.66	0.56	0.61	0.6403
	1 (Neu)	0.61	0.69	0.65	
	2 (Pos)	0.67	0.65	0.66	
TFID	0 (Neg)	0.75	0.56	0.64	0.6845
	1 (Neu)	0.62	0.78	0.69	
	2 (Pos)	0.77	0.66	0.71	

Table-8 : Classification report for logistic regression on different datasets.

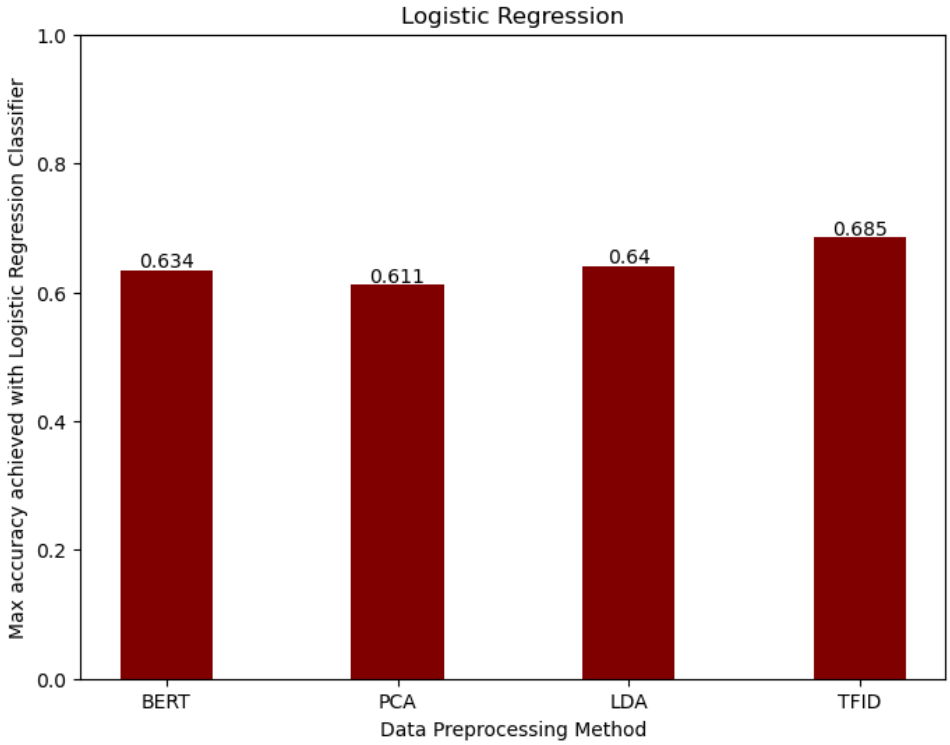


Fig-15 : Bar graph showing best performance of logistic regression for the datasets.

Inference Drawn

- Again due to its low dimensionality, the LDA dataset requires a low C value since it is a simpler dataset and doesnt require complex decision boundary, compared to the other datasets.
- Tfid data requires very high C value due to most of the entries in the matrix being zeros, which means a more precise fitting is required to capture
- the features of the model. The model performs with very similar accuracy scores for all the datasets, meaning it can handle large variety of data by fine-tuning the hyper-parameter 'C'.

3.5 Ensemble Learning and Boosting

In order to combine the best results from the different models, we created an ensemble including each classifier with their best hyper-parameters. Since the data input had to be kept consisted with one of the dataset types, the one with best performance was chosen.

Classifier	Dataset that yielded maximum accuracy	Maximum Accuracy
Decision Tree	LDA	0.631
Random Forest	TFID	0.715
SVM	TFID	0.7134
Naive-Bayes	TFID	0.648
Perceptron	BERT	0.625
Logistic Regression	TFID	0.6845

Table-9 : Best results for different classifiers.

As it is evident from the above table, Tfid was the best choice.
Conflicts in predictions were resolved by taking the mean value of the prediction given by each model.

With ensemble learning, an consistent accuracy of around 70-72% was achieved. The increase is not very significant. The reason could be the underperformance on the Tfid dataset by models like perceptron and decision tree. It is possible that majority of the models struggle in classifying some specific datapoints.

We experimented with applying the boosting algorithm using Sklearn's AdaBoostClassifier class, with decision tree as the base classifier. The accuracy score however seemed to drop by a margin of 0.03% (around 0.71 without boosting, to 0.69 with boosting). The drop may be due to attempts in resolving some misclassifications leading to newer misclassifications.

3.6 Conclusion

- In-depth comparative analysis of different ML classifiers gave insights into the impact of data preprocessing on model accuracy. Classifiers like random forest and linear SVM seemed to work well with the dataset considered.
- Vectorization of text data using Sklearn's TfidVectorizer() proved to yield better results with majority of the classifiers compared to the BertTokenizer().
- In many cases dataset with LDA applied to the BERTt tokenized data proved to yield drastically better results. This shows how it is an effective technique for dimensionality reduction in classification tasks.
- Overall after comparing the classifiers, max standalone accuracy is produced by the random forest classifier on the Tfid processed data (accuracy_score=0.715).
- Ensemble learning slightly improved the accuracy, since it incorporated multiple classifiers based on different algorithms for prediction.
- Applying boosting algorithm to the ensemble of models did not bear better results, possibly due to the overfitiitng and recurring misclassifications.

4 Summary

Again, sentiment analysis from text input is a well-known problem statement in the field of machine learning and NLP. We tried to work on the same problem, taking a traditional machine learning approach. In this project, we performed sentiment analysis on a twitter dataset with 30k+ tweets, giving a detailed analysis of performance obtained with different techniques. To summarize the project and results:

The objective of the project was to implement (and subsequently analyse) different machine learning classifiers on the Sentiment Analysis Dataset.

Firstly, pre-processing was done on the original dataset using different techniques and tokenizers:

1.Bert tokenizer
2.Bert tokenizer followed by PCA
3.Bert tokenizer followed by LDA
4.TFID tokenizer Then, several machine learning classifiers including Decision Trees, Random Forests,

SVM, Naive-Bayes, Perceptron, Logistic Regression were implemented. Each of these classifiers were optimized to generate the maximum possible accuracy on each dataset by varying different parameters and hyper-parameters (if any).

The observations can be summarized as follows:

Following this, it was concluded from the table() that TFID tokenizer was the best suited for making predictions. Then, ensemble learning and boosting was done to finally return the modal value of the predictions returned by all the classifiers after processing user input by this TFID tokenizer.

A Contribution of each member

1. Krishna Patil (B22CS078) : Performed data pre-processing, organized github repository, video presentation, report writing.
2. Saumitra Agarwal (B22AI054) : Implemented Random Forest, Ensemble Learning and Boosting, Inference function and latex editing of report.
3. Harshit Goyal (B22CS024) : Implemented SVM Classifier, project page design, presentation and contributed to the report content.
4. Mukund Gupta (B22CS086) : Implemented Decision Tree Classifier, project page desing and presentation making.
5. Aarohi Dharmadhikari (B22AI001) : Implemented Naive Bayes Classifier, contributed to report making.
6. Gouri Patidar (B22AI020) : Implemented Perceptron and Logistic Regression Classifier, contributed to report making.