Machine Learning Project Report

YouTube Comments Sentiment Analysis

Name: Vedant Vartak
PRN: 24070243057

Problem Statement

Content creators, marketers and platform moderators needs to understand what exactly the viewers feel. YouTube comments are unstructured and varied, reading and analysing them manually will take too much time. This project aims to create machine learning model that helps to classify positive, negative or neutral comments. This will help to improve the user engagement and brand reputation.

Objective

1. To analyse the sentiments of the YouTube comments and classify them as the positive, negative or neutral.
2. To preprocess the data by applying the language filter, stopwords removal, contraction expansion, lemmatization, tokenization and emoji handling.
3. To compare the performance of different machine learning models and choosing which gives the best one based on the evaluation metrics.

Methodology

1. Dataset Description

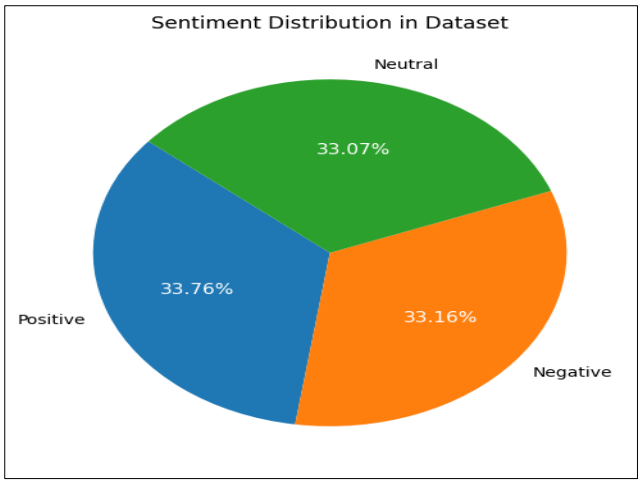
Two different datasets are downloaded from Kaggle and are merged to form the variations in the dataset.

Dataset Sources:

I. Dataset containing around 18,000 datapoints.
Link: https://www.kaggle.com/datasets/atifaliak/youtube-comments-dataset

II. Dataset containing around 1 Million datapoints.
Link:https://www.kaggle.com/datasets/amaanpoonawala/youtube-comments-sentiment-dataset

Total Comments: Around 1 Million datapoints with balanced sentiments but after filtering it became around 8,80,000 datapoints.



2. Preprocessing Techniques

I. Language Detection and filtering

Detects and retain only comments which are in English language. This is achieved by the langdetect library, it helps to avoid the errors which are caused by non-English text during model training. After filtering, datapoints got reduced around 8,80,000.

II. Stop Words Removal

Removing the irrelevant stop words from the sentences and keeping only relevant stop words which are necessary to identify the negative sentiments. Words such as "can", "did", "has", "is", "not", "but", "no", "very", "shall", "will", "would" etc. are kept to identify sentiments more effectively.

III. Contractions

This is used to convert the informal or shortened words into their full forms.

can't → cannot

it's → it is

IV. Removing Name Entity

Removing named entities (like names, places, and organizations) from the text which helps reduced noise and prevent bias. Specific name can influence the sentiment in unintended way.

V. Tokenization & lemmatization

Tokenization is used to split the text into words which helps for easy analysis and transformation. Lemmatization is used to convert the word into base or root form to reduce the dimensionality.

VI. Removing Punctuations

Punctuations are removed to get only words in vocabulary instead of special characters.

“Hello, how are you?" → "Hello how are you"

VII. Handling Emojis

Converting the emoji's to words so that the context of sentence remains preserved.

👍 → :thumbs_up:

3. Feature Extraction (TF-IDF Vectorization)

Converts the text data into format which can be processed by the machine learning models. TF-IDF helps to give the importance to the words and reduces the impact of commonly occurring words. Term Frequency counts how frequently the word appears and Inverse Document Frequency reduces the weight of commonly used words.

Hyperparameter set for the TFIDF:

- **ngram_range:** The ngram range is set to (1,3) which will try all the combinations of unigrams, bigrams and trigrams. The models were also tested using unigrams and bigrams as well.

4. Machine Learning Models

All the models were trained on 80% of the data and tested on 20% of the data.

Below are the models mentioned which are used for YouTube comment Sentiment Analysis:

1. **Logistic Regression**
   It is simple and effective for text classification also providing balance performance across all the sentiment classes.
2. **Naïve Bayes**
   Naïve Bayes works well with the text data. It is very fast and efficient for larger dataset and gives optimal performance.
3. **Decision Tree**
   It helps to understand which words matters most but it may not work with the large data.
4. **Gradient Boost Classifier**
   Gradient Boost classifier was chosen because it improves result step by step. It learns from the previous mistakes and works well with different types of data.
5. **XGBoost Classifier**
   It can handle large data well and can give better result if tuned properly.

Results & Discussion

Below are the classification reports of each model and short description about how it was achieved.

1. Logistic Regression

	precision	recall	f1-score	support
negative	0.70	0.72	0.71	62109
neutral	0.62	0.64	0.63	54810
positive	0.77	0.72	0.75	59486
accuracy			0.70	176405
macro avg	0.70	0.70	0.70	176405
weighted avg	0.70	0.70	0.70	176405

- In logistic regression hyperparameters set were multi_class='multinomial' for the multiclass classification and solver='lbfgs' for optimization.
- Logistic Regression gave very stable result, it gives accuracy around 70% and also maintains the balance of precision, recall and f1 score for all the sentiments.
- Neutral sentiment is slightly descent but it performs better when compared to other models, which makes it more reliable for sentiment analysis.
- TF-IDF vector with unigrams gave good accuracy of 70%, but when bigrams and trigrams were included, the accuracy got dropped by 1% as well as other metrics.

2. Naïve Bayes

	precision	recall	f1-score	support
negative	0.66	0.84	0.74	62109
neutral	0.71	0.51	0.59	54810
positive	0.76	0.75	0.75	59486
accuracy			0.70	176405
macro avg	0.71	0.70	0.69	176405
weighted avg	0.71	0.70	0.70	176405

- In Naïve Bayes the hyperparameter alpha=1 was set which gave the accuracy around 65% and it was trained on unigrams.
- In second test, ngram range was set to (1,3) which tries all combination from unigram to trigrams, it gave around 68% accuracy. But it was giving bad results for neutral class, resulting low recall and f1-score.
- After several test, alpha was set to 0.1 which gave the very good result for negative and positive class but decent result for recall and f1 score of neutral class. The model gave around 70.37% accuracy.

3. Decision Tree

	precision	recall	f1-score	support
negative	0.48	0.44	0.46	62109
neutral	0.34	0.69	0.46	54810
positive	0.59	0.08	0.14	59486
accuracy			0.40	176405
macro avg	0.47	0.41	0.35	176405
weighted avg	0.47	0.4	0.35	176405

Decision tree has performed poorly on the large datasets. The above result is achieved when minimum depth is set to 20, when the depth was increased the model gave more poor results.

4. Gradient Boost Classifier

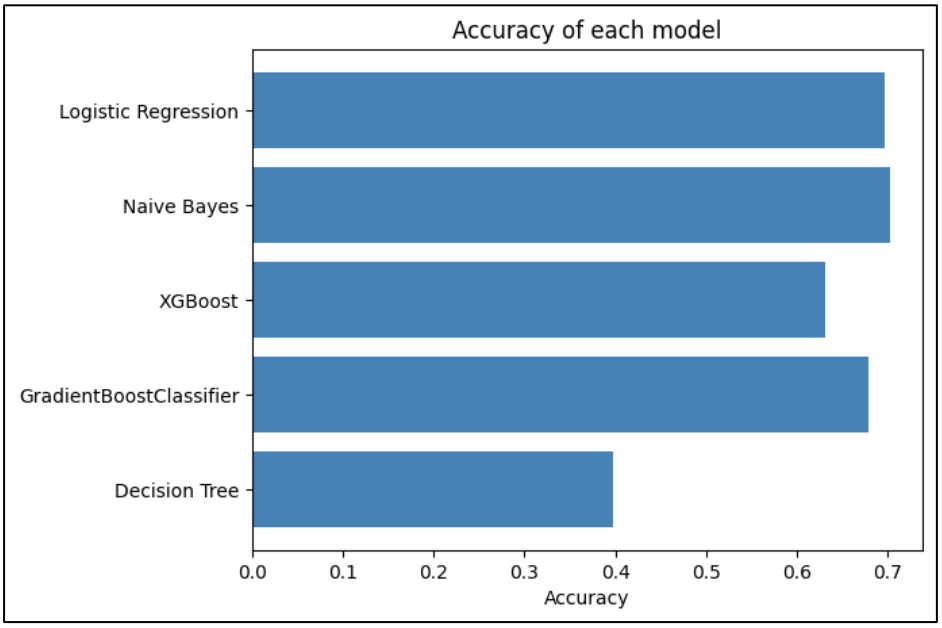
	precision	recall	f1-score	support
negative	0.67	0.70	0.69	62109
neutral	0.60	0.63	0.61	54810
positive	0.77	0.70	0.73	59486
accuracy			0.68	176405
macro avg	0.68	0.68	0.68	176405
weighted avg	0.68	0.68	0.68	176405

- The Gradient boost depth was set to 10 it gave accuracy of 62%, later depth was set to 30 which gave accuracy of 65%.
- When the depth was set to 100, the model achieves the accuracy of 68% which suggests that increasing the depth improves the model performance.
- Due to lack of computational power further tuning was restricted. Further hyperparameter tuning may improve the model performance and might surpass the Logistic Regression.
- The model performs well by achieving accuracy of 68%, but it falls short when compared to Logistic Regression and Naïve Bayes.
- Precision and Recall for positive and negative class are strong making it reliable for sentiment analysis.
- Achieves better overall performance than XGBoost, suggesting that it generalizes little better.

5. XGBoost Classifier

	precision	recall	f1-score	support
negative	0.62	0.66	0.64	62109
neutral	0.53	0.61	0.57	54810
positive	0.78	0.62	0.69	59486
accuracy			0.63	176405
macro avg	0.64	0.63	0.63	176405
weighted avg	0.65	0.63	0.63	176405

- The model gave 63% accuracy, which is less compared to naïve bayes and logistic regression that gave around 70% accuracy.
- The precision, recall and f1-score of neutral class is quite low which shows that model struggles in classifying neutral classes.
- In first test, model estimator was set to 200 with max depth 8, gamma=0.3 and learning rate 0.1, but gave around 61% accuracy.
- In second test, estimators were set to 300 with max depth 6, gamma=0.2 and learning rate lower to 0.05, aiming for better generalization. The model improved the accuracy 63%.
- XGBoost Classifier full potential cannot be unleashed due to lack of the computational power, which restricted the further hyperparameter tuning.



Findings

- **Balanced Performance:** Both the models Logistic Regression and Naïve Bayes achieved the accuracy around 70%, but Logistic Regression provides more balance performance across all sentiment classes, whereas Naïve Bayes struggles with the neutral classes.
- **Best for Positive and Negative Sentences:** If the goal is to prioritize the positive and negative sentences while giving less preference to neutral sentence then Naïve Bayes is the best choice as it gave good results for negative and positive class, it achieves the highest recall (0.84).
- **Best for Multi-Class Sentiment Detection**: If the primary goal is to detect all three classes, then Logistic Regression is better choice as it is stable for detecting all three classes.

- **Limitations of XGBoost and Gradient Boost:** The full potential of both XGBoost and Gradient Boost classifier was not unleashed due to lack of the computational power. With further optimizations such as adjusting learning rates, increasing boosting rounds and leveraging GPU based training, these models may outperformed the Logistic Regression and Naïve Bayes.
- **Decision Tree Model:** The decision tree model performs poorly on large datasets. Increasing depth didn't improve the performance instead it led to overfitting.
- **Training Time:** Naïve Bayes was fastest model to train and test. Logistic Regression also took average time to train while GradientBoost and XGBoost were computationally expensive but showed improvement with training.
- **Real World Application Challenges:** Traditional ML model struggles in sentiment analysis of YouTube comments due to sarcasm, context, slangs, etc. Future work can involve using pre-trained embeddings (Word2Vec, BERT).