Final Project: Kaggle Movie Review Text Classification

IST 664: Natural Language Processing

Professor: Michael Larche

Submitted By:

Leena Balagalu

Due Date:

1/14/2025

1. INTRODUCTION

The project focuses on developing a text classification model using the Kaggle movie review dataset in which phrases are labeled with sentiments ranging from "positive" to "negative." The main objective of the project is to analyze the text data, preprocess it, and extract helpful features so that a good sentiment analysis model can be generated. Tokenization along with unigrams and bigrams as features will be used for preparing data for classification.

Within the scope of the project, the implementation and comparison of different machine learning models , such as Naive Bayes classifiers to find the optimal operation. The models will be evaluated using precision, recall, and f1-score metrics. Apart from the standard feature, a unique feature function will be designed and tested to explore effects on accuracy and performance enhancement of the model.

Ultimately, the project aims to showcase the significance of preprocessing, feature selection, and algorithmic comparison in Natural Language Processing (NLP). The data can provide valuable insight into the applicability of various techniques as well as the importance of purposeful feature engineering in achieving accuracy and reliability in sentiment classification. The project aspires to achieve is that it will develop good sentiment classification models and draw insight into structures of the dataset by combining advanced machine learning methods with feature engineering. The data is basically prepared through the processes of feature extraction, model training, evaluation, and parameter optimization-it's a full process-oriented approach.

2. OVERVIEW OF DATASET

The Kaggle Movie Review Dataset consists of 156060 training phrases extracted from movie reviews along with their associated sentiment labels. The dataset has a structured way to analyze and classify textual data into sentiment categories, making it an great resource for building and evaluating sentiment classification models.

2.1 Key Features of the Dataset:

2.1.1 Data Composition:

The dataset contains training data of 156,060 phrases derived from movie reviews.Each phrase represents a specific portion of a review, ranging from individual words to complete sentences.

Sentiment Labels:

The dataset categorizes phrases into one of five sentiment labels:

- 0: Very Negative
- 1: Negative
- 2: Neutral
- 3: Positive
- 4: Very Positive

Structure:

The dataset is provided in TSV format.

Key columns in data:

PhraseId: A unique identifier for each phrase.

SentenceId: Links the phrase to its originating sentence.

Phrase: The text of the phrase.

Sentiment: The sentiment label for the phrase.

2.2 Key Statistics

- Total phrases: 156,060
- Total sentences: ~11,855
- Sentiment distribution (based on train.tsv):
    - 0 (Negative): ~10.2%
    - 1 (Somewhat Negative): ~20.1%

- 2 (Neutral): ~51.2%
- 3 (Somewhat Positive): ~15.4%
- 4 (Positive): ~3.1%

2.3 Applications of data in project:

The dataset serves as a basic foundation for training and testing machine learning models in sentiment analysis in the project. It provides various opportunities for feature engineering, such as exploring unigram and bigram features, leveraging lexicons, and experimenting with advanced NLP techniques.

3. METHODOLOGY

3.1 Data preprocessing

The data set is preprocessed before sentiment analysis, where the cleaning and organizing the data takes place and one sentence will be divided into individual words, also known as tokens through tokenization. Processing was done removing only those elements that do not add much value, which are stop words. Common stop words include "and," "the," and the rest. The data has negation words as well like not, never, no, none, nothing, nowhere, neither, nor, noone, hardly, scarcely, and rarely. These negation words in a sentence may significantly change the sentiment of the sentence like "Not good" is different from "good."

After tokenizing and lowercasing the data, the preprocessed step involved handling negation and removing some extra stop words, not included in standard libraries which were noticed most frequently, such as "'s," "'re," "'ve," "'d," "maybe," "would," "could," "should," "might," "must," "also," "thing," "lot," "etc," "ok," "okay," "oh," "uh," and "um." These words, show relatively no significance to the sentiment derived out of a sentence, and thus were excluded from the texts.Therefore, cleaning steps transformed the processed data into a structured and meaningful form, holding only significant words within a sentence. Ultimately, the model of sentiment analysis would primarily focus on more relevant features rendering it more accurate in sentiment classification.

3.2 FEATURE ENGINEERING

To prepare the data for machine learning models , feature engineering is required to shape the given dataset. It also prepares raw text to be transformed into a meaningful features that helps the model understand the sentiment expressed in each phrase. I have used below features for the Kaggle dataset

1. Unigram Bag of Words

Unigram means single words from each phrase with frequency in the vector format. The result feature vector can detect the existence as well as nonexistence of any specific words in this regard with reference to sentiment.

Example:

Phrase: "A series of escapades demonstrating the adage"

Unigrams: ["A", "series", "of", "escapades", "demonstrating", "the", "adage"]

Feature Representation: {"A": 1, "series": 1, "of": 1, "escapades": 1, "demonstrating": 1, "the": 1, "adage": 1}

Words like "escapades" or "adage" may carry clues about the sentiment being expressed.

2. Negations

Negations detect the impact of negation words in altering the sentiment of a phrase. Words like "not," "never," "no," or "none" are identified in the text. When these words are present, they frequently change the sentiment of phrases. For example, "not good" has an opposite meaning compared to "good." Negations are critical for accurate sentiment analysis since they can flip the sentiment of an otherwise positive or negative phrase.

3. Bigrams

Bigrams capture the relationship between 2 consecutive words to understand context better. Sentiment can often depend on the relationship between words. For example, the bigram "not happy" shows negative sentiment, whereas "very happy" shows positive sentiment.

4. Part-of-Speech (POS) Tagging

Part-of-Speech (POS) Tagging helps to understand the structure of phrases to determine each word's role in sentiment. Words like adjectives and adverbs often carry strong sentiment.

Example:

Phrase: "demonstrating the adage"

POS Tags: [("demonstrating", "VBG"), ("the", "DT"), ("adage", "NN")]

5. Subjectivity Analysis

Subjectivity Analysis determine whether a phrase expresses personal opinion (subjective) or actual information(objective). Each phrase is analyzed for its subjectivity score. Subjective phrases are more likely to contribute to sentiment, whereas objective ones may be neutral.

Example:

Phrase: "A heartwarming experience"

Subjectivity Score: High

6. LIWC (Linguistic Inquiry and Word Count) Features

LIWC all kinds of attributes from phrases using LIWC lexicons. LIWC analyzes phrases for categories like positive emotions, negative emotions, and neutral emotions.

Example:

Phrase: "A series of escapades"

LIWC Features: High in "Cognitive Processes," moderate in "Positive Emotion."

7. VADER Sentiment Lexicons

VADER quantify the sentiment intensity of a phrase, particularly for short text. VADER uses a pre-built lexicon to assign scores for positive, negative, and neutral sentiment, considering nuances like punctuation, capitalization, and negations. VADER is effective for understanding sentiment in short phrases, including the impact of stylistic elements.

Example:

Phrase: "A series of escapades demonstrating the adage"

VADER Scores: Positive: 0.2, Negative: 0.1, Neutral: 0.7

3.3 CLASSIFIERS

Random Forest - This algorithm uses multiple decision trees to predict data and is very excellent for holding complex data patterns and relationships among features and avoids the problem of overfitting, which is, highly suitable for datasets that involve mixed features like text and numerical data.

Naive Bayes - This one works on the kernel function of Bayes' Theorem. It assumes that words are independent and it works very well on text classification tasks such as sentiment analysis with features comprises of word frequency, like Bag of Words or TD-IDF.

Logistic Regression - This creates the chance of correct points to belong within its class. A linearly separable and smaller dataset is very interpretable and efficient, perfect for such.

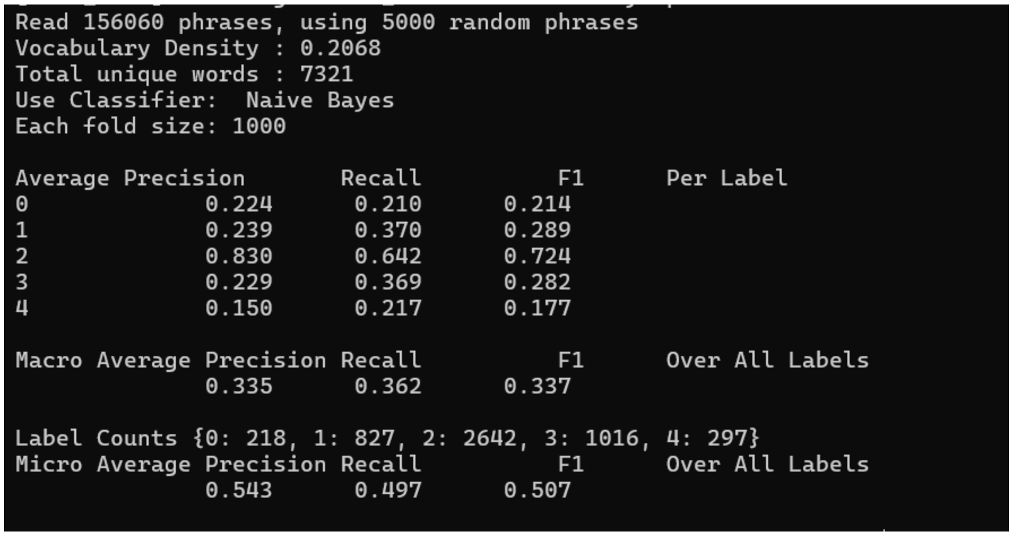
XGBoost - This algorithm particularly excels in constructing complicated but nonlinear relations and has a very nice performance over structured data. It usually achieves the highest level of classification and regression tasks.

RNNs (Recurrent Neural Networks) - An architecture in deep learning designed for data-like text where order becomes crucial, processed sentence by sentence, word by word. By doing so, they also recall the context from earlier in the sentence to gain a better understanding of the full title of phrases-words.

4 RESULTS AND ANALSYSIS

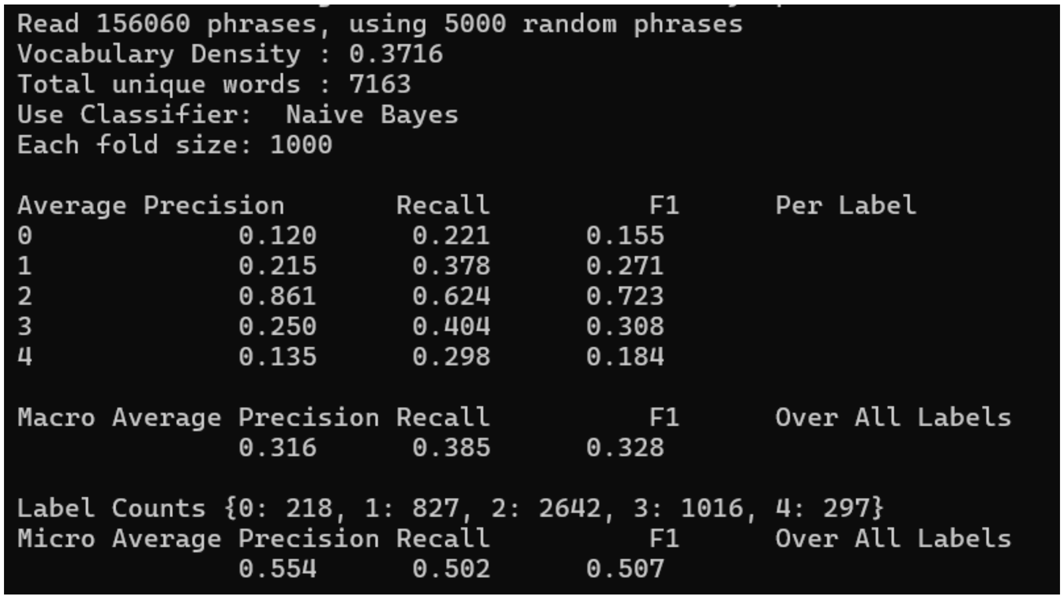
4.1 EXPERIMENTS

4.1.1 Experiment on Unigrams with and without removal of stop words and punctuations with Naïve Bayes



```
Read 156060 phrases, using 5000 random phrases
Vocabulary Density : 0.2068
Total unique words : 7321
Use Classifier:  Naive Bayes
Each fold size: 1000

Average Precision       Recall          F1      Per Label
0             0.224      0.210      0.214
1             0.239      0.370      0.289
2             0.830      0.642      0.724
3             0.229      0.369      0.282
4             0.150      0.217      0.177

Macro Average Precision Recall          F1      Over All Labels
              0.335      0.362      0.337

Label Counts {0: 218, 1: 827, 2: 2642, 3: 1016, 4: 297}
Micro Average Precision Recall          F1      Over All Labels
              0.543      0.497      0.507
```

With stopwords and punctuation

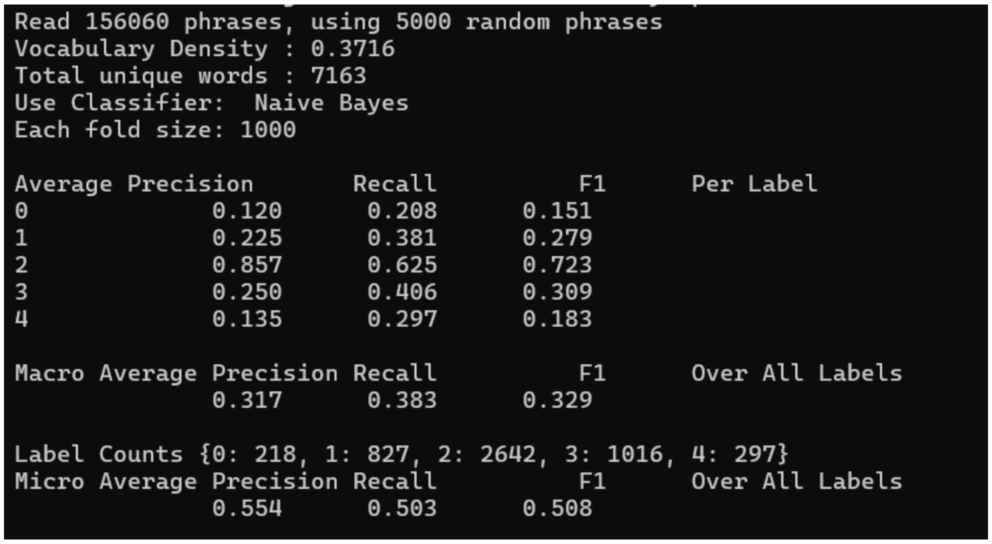


```
Read 156060 phrases, using 5000 random phrases
Vocabulary Density : 0.3716
Total unique words : 7163
Use Classifier:  Naive Bayes
Each fold size: 1000

Average Precision       Recall          F1      Per Label
0             0.120      0.221      0.155
1             0.215      0.378      0.271
2             0.861      0.624      0.723
3             0.250      0.404      0.308
4             0.135      0.298      0.184

Macro Average Precision Recall          F1      Over All Labels
              0.316      0.385      0.328

Label Counts {0: 218, 1: 827, 2: 2642, 3: 1016, 4: 297}
Micro Average Precision Recall          F1      Over All Labels
              0.554      0.502      0.507
```

Without stopwords and punctuation

This experiment deals with the impact of stopwords and punctuation using naïve bayes classifier. The results showed that the dataset that by removing stopwords and punctuation had a noticeably higher vocabulary density (0.3716 instead of 0.2068) and a higher recall, which is important for identifying true positives. Although there was a decrease in precision, there was no a big overall difference in terms of accuracy (micro-average F1 score), standing at 0.507. Removing stopwords and punctuation removes noise, even though it impacts vocabulary to be more meaningful and denser. For that reason, the dataset without stopwords and punctuations is selected as baseline for the remaining experiments which ensures better balance and efficiency.

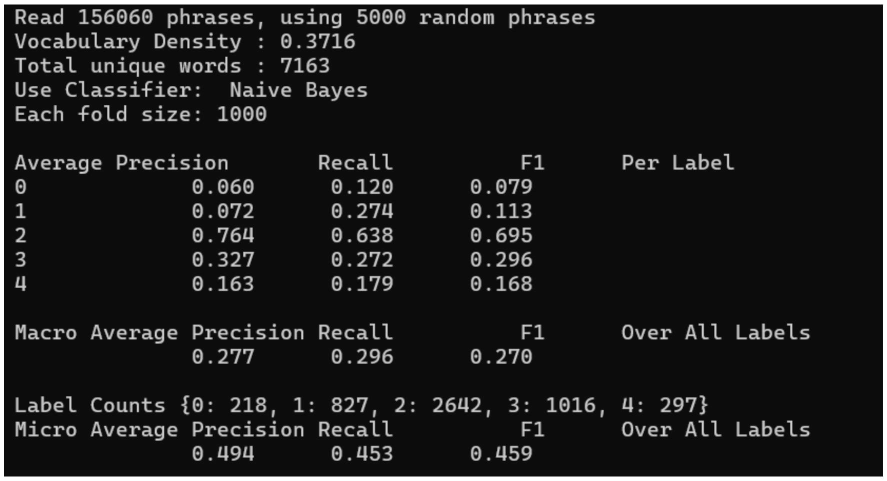
4.1.2 Experiment using unigrams and negation with naïve bayers



```
Read 156060 phrases, using 5000 random phrases
Vocabulary Density : 0.3716
Total unique words : 7163
Use Classifier:  Naive Bayes
Each fold size: 1000

Average Precision       Recall          F1      Per Label
0             0.120      0.208      0.151
1             0.225      0.381      0.279
2             0.857      0.625      0.723
3             0.250      0.406      0.309
4             0.135      0.297      0.183

Macro Average Precision Recall          F1      Over All Labels
              0.317      0.383      0.329

Label Counts {0: 218, 1: 827, 2: 2642, 3: 1016, 4: 297}
Micro Average Precision Recall          F1      Over All Labels
              0.554      0.503      0.508
```

The negation word increased the F1 score in performance from 0.507 to 0.508. Negations made the model to better grasp sentiment changes, particularly in expressions as "not good" or "never bad,". Thus, negation increased the contextual understanding of the classifier while improving overall performance.

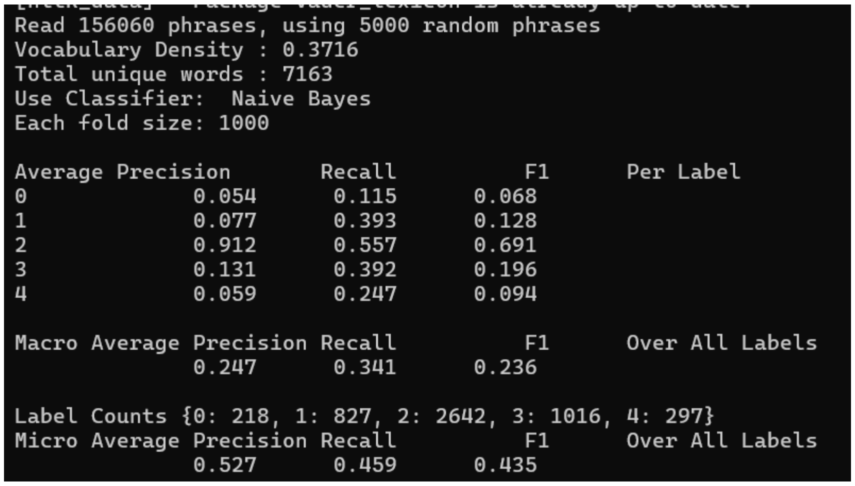
4.1.3 Experiment Using POS Tagging with naïve bayers



```
Read 156060 phrases, using 5000 random phrases
Vocabulary Density : 0.3716
Total unique words : 7163
Use Classifier:  Naive Bayes
Each fold size: 1000

Average Precision       Recall          F1      Per Label
0              0.060      0.120      0.079
1              0.072      0.274      0.113
2              0.764      0.638      0.695
3              0.327      0.272      0.296
4              0.163      0.179      0.168

Macro Average Precision Recall          F1      Over All Labels
               0.277      0.296      0.270

Label Counts {0: 218, 1: 827, 2: 2642, 3: 1016, 4: 297}
Micro Average Precision Recall          F1      Over All Labels
               0.494      0.453      0.459
```

This experiment used POS tagging with dataset without stopwords and punctuation as baseline to enhance feature extraction. The micro-average F1 score was lower without POS tagging at 0.507 and with negation at 0.508. POS tagging does reflect grammatical structure and context, but the Naive Bayes classifier may not exploit such extra richness so that F1 score has reduced to 0.459. Adding an additional linguistic dimension through POS tagging does not return respectable results with simple probabilistic models such as the Naive Bayes Classifier.

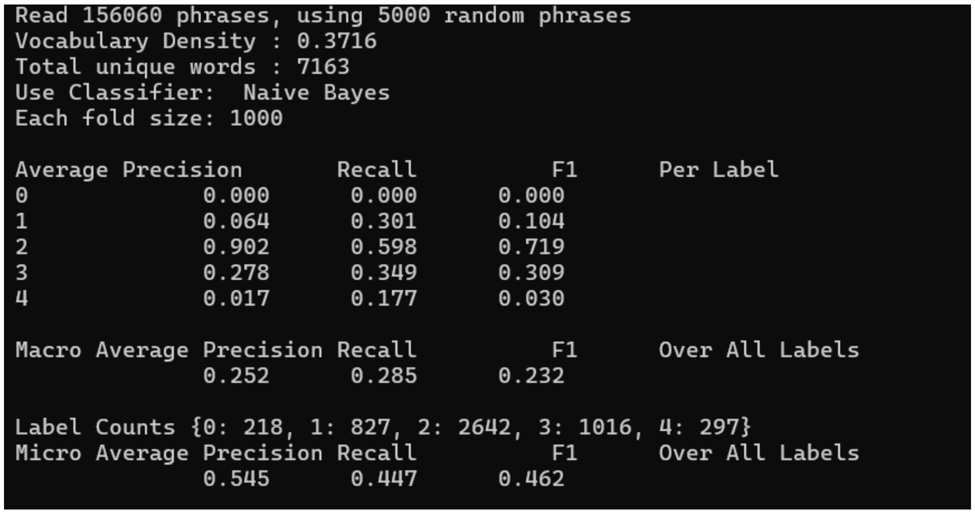
4.1.4 Experiment using bigram feature with naïve bayers



```
Read 156060 phrases, using 5000 random phrases
Vocabulary Density : 0.3716
Total unique words : 7163
Use Classifier:  Naive Bayes
Each fold size: 1000

Average Precision       Recall          F1      Per Label
0              0.054      0.115      0.068
1              0.077      0.393      0.128
2              0.912      0.557      0.691
3              0.131      0.392      0.196
4              0.059      0.247      0.094

Macro Average Precision Recall          F1      Over All Labels
               0.247      0.341      0.236

Label Counts {0: 218, 1: 827, 2: 2642, 3: 1016, 4: 297}
Micro Average Precision Recall          F1      Over All Labels
               0.527      0.459      0.435
```

In this experiment, bigram features were added to the dataset without stopwords and punctuation, using Naive Bayes as the classifier. The micro-average F1 score has decrease of 0.462 compared to the baseline dataset (F1 = 0.507) without bigrams. This shows that bigrams are capable of capturing relationships between word pairs, they may add complications that Naive Bayes may have been unable to deal with effectively. For the future, more sophisticated models like SVM and deep learning models must be put into effect for the sake of using bigrams most beneficially.

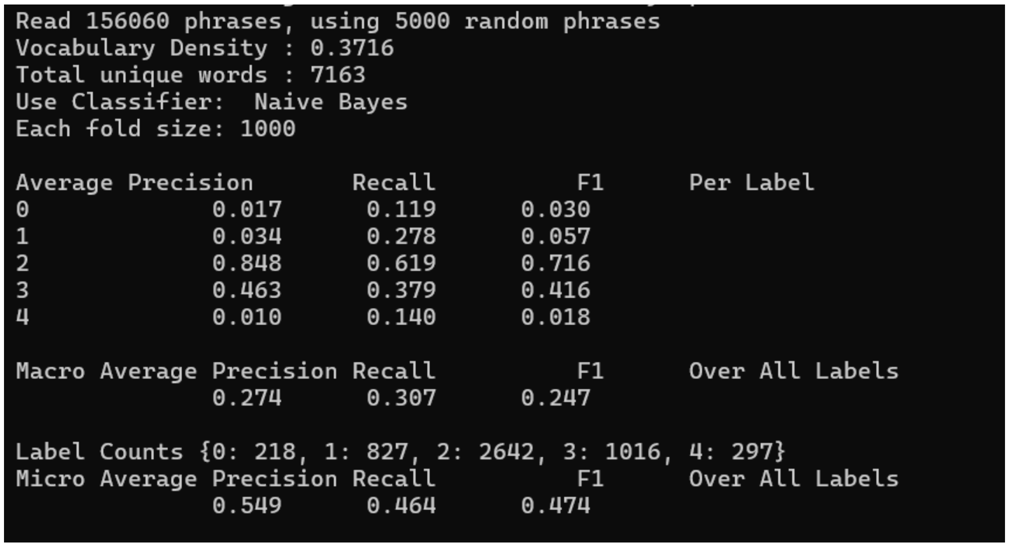
4.1.5 Experiment using Subjectivity analysis with naïve bayers



```
Read 156060 phrases, using 5000 random phrases
Vocabulary Density : 0.3716
Total unique words : 7163
Use Classifier:  Naive Bayes
Each fold size: 1000

Average Precision       Recall          F1      Per Label
0              0.000      0.000      0.000
1              0.064      0.301      0.104
2              0.902      0.598      0.719
3              0.278      0.349      0.309
4              0.017      0.177      0.030

Macro Average Precision Recall          F1      Over All Labels
               0.252      0.285      0.232

Label Counts {0: 218, 1: 827, 2: 2642, 3: 1016, 4: 297}
Micro Average Precision Recall          F1      Over All Labels
               0.545      0.447      0.462
```

In this experiment, subjectivity analysis was added as a feature to the dataset without stopwords and punctuation, using Naive Bayes as the classifier. The micro-average F1 score results to 0.462 as compared to the dataset baseline F1 score of 0.507. In the analyses , subjectivity captured the differentiation of subjective and objective phrases but adding the specific feature had not made significant improvement in the error performance of the Naïve Bayes model. Both are the precision and recall minority classes (Classes 0 and 4) are low, meaning that subjectivity cannot contribute to the improvement of performance in this set-up alone.

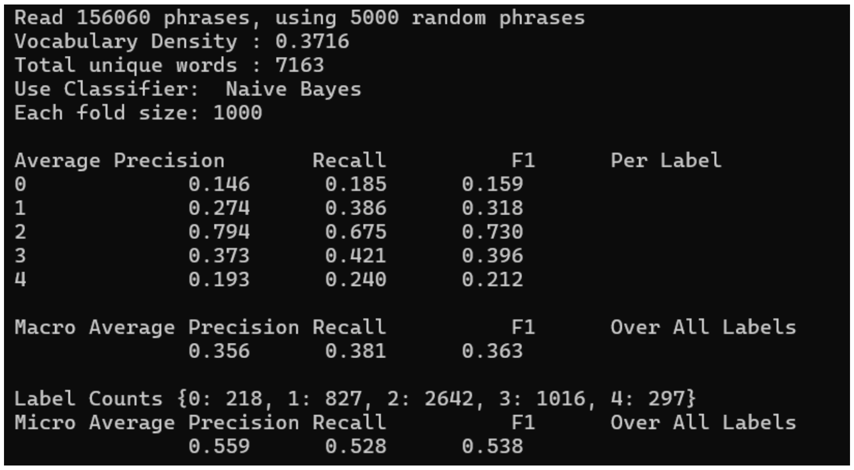
4.1.6 Experiment using LIWC using naïve bayes



```
Read 156060 phrases, using 5000 random phrases
Vocabulary Density : 0.3716
Total unique words : 7163
Use Classifier:  Naive Bayes
Each fold size: 1000

Average Precision       Recall          F1      Per Label
0             0.017      0.119      0.030
1             0.034      0.278      0.057
2             0.848      0.619      0.716
3             0.463      0.379      0.416
4             0.010      0.140      0.018

Macro Average Precision Recall          F1      Over All Labels
              0.274      0.307      0.247

Label Counts {0: 218, 1: 827, 2: 2642, 3: 1016, 4: 297}
Micro Average Precision Recall          F1      Over All Labels
              0.549      0.464      0.474
```

In this experiment, LIWC (Linguistic Inquiry and Word Count) features were added to the dataset without stopwords and punctuation, using Naive Bayes as the classifier. The micro-average F1 has declined up to 0.474 when compared against a baseline of F1=0.507, and other experiments. LIWC features adddresses psychological, emotional, and social issues from within, adds great deepening of its feature set. But the scores were still very low on the minority classes: Classes 0 and 4, whose recall and precision values were most similar to zero. It seems that the LIWC features had given some linguistic insights but did not have any extraordinary results because the Naive Bayes classifier was very simple and not so efficient a tool for extracting the sophistication of such complex features. A more advanced model could probably help in better utilizing LIWC features

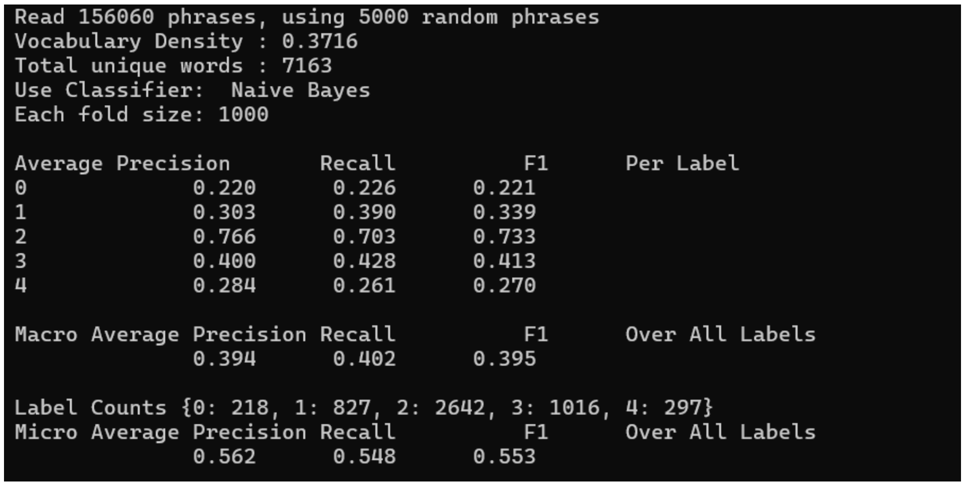
4.1.7 Experiment using combined feature(Unigram,negation,pos tagging ,subjectivity ,liwc , bigrams) using naïve bayes



```
Read 156060 phrases, using 5000 random phrases
Vocabulary Density : 0.3716
Total unique words : 7163
Use Classifier:  Naive Bayes
Each fold size: 1000

Average Precision       Recall          F1      Per Label
0            0.146      0.185      0.159
1            0.274      0.386      0.318
2            0.794      0.675      0.730
3            0.373      0.421      0.396
4            0.193      0.240      0.212

Macro Average Precision Recall          F1      Over All Labels
             0.356      0.381      0.363

Label Counts {0: 218, 1: 827, 2: 2642, 3: 1016, 4: 297}
Micro Average Precision Recall          F1      Over All Labels
             0.559      0.528      0.538
```

In this experiment, a combination of features, including unigrams, negations, POS tagging, subjectivity analysis, LIWC, and bigrams, was used with the Naive Bayes classifier on the dataset without stopwords and punctuation. The combined approach increasethe micro-average F1 score to **0.538**, compared to the baseline (F1 = 0.507). Having different elements combined, a good representation of text was created for better capturing subtle nuances of sentiment by the model. Out of neutrality, 0.730 was observed to be the highest F1 score. As revealed in the experiment, it shows forces of linguistic, contextual, as well as even psychological features enhancing overall model performance.

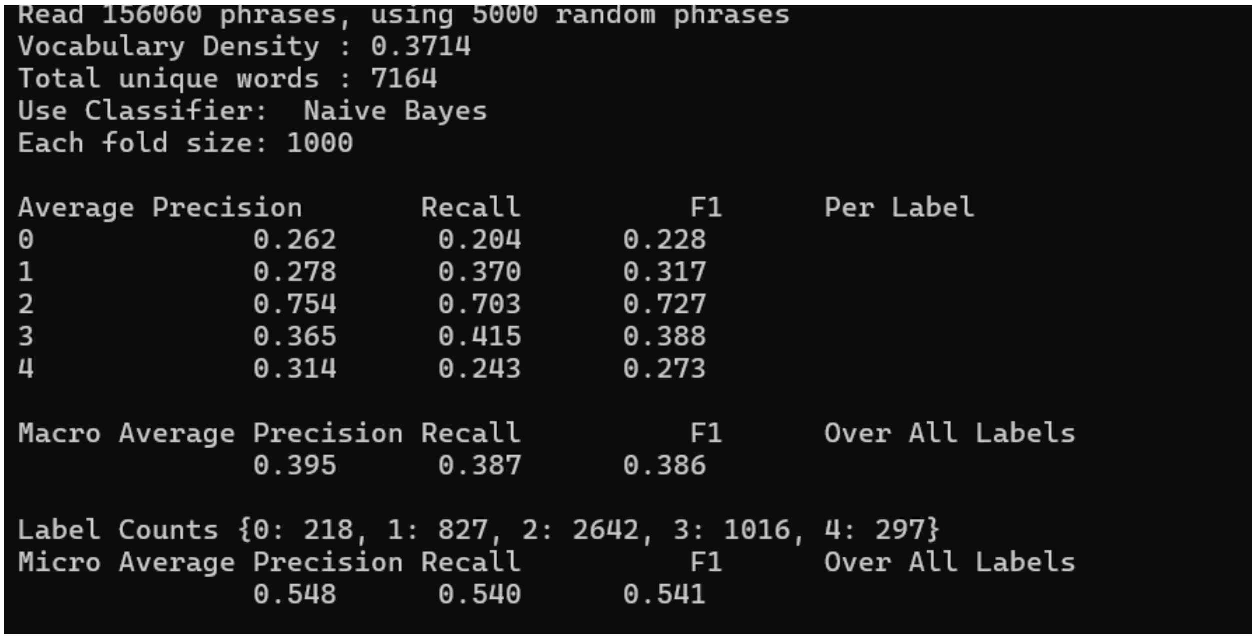
4.1.8 Experiment using combined feature(Unigram ,pos tagging,subjectivity,liwc) using naïve bayes



```
Read 156060 phrases, using 5000 random phrases
Vocabulary Density : 0.3716
Total unique words : 7163
Use Classifier:  Naive Bayes
Each fold size: 1000

Average Precision       Recall          F1      Per Label
0            0.220      0.226      0.221
1            0.303      0.390      0.339
2            0.766      0.703      0.733
3            0.400      0.428      0.413
4            0.284      0.261      0.270

Macro Average Precision Recall          F1      Over All Labels
             0.394      0.402      0.395

Label Counts {0: 218, 1: 827, 2: 2642, 3: 1016, 4: 297}
Micro Average Precision Recall          F1      Over All Labels
             0.562      0.548      0.553
```

In this experiment, a combination of features including unigrams, POS tagging, subjectivity analysis, and LIWC was used with the Naive Bayes classifier on the dataset without stopwords and punctuation. The micro-average F1 score has increased to **0.553**, compared to the baseline (F1 = 0.507) when bigrams are removed from combination. All other measures have also significantly improved when compared to combination where bigrams were used. So bigrams were removed to focus on better performed combination.

4.1.9 Experiment using combined feature(Unigram ,pos tagging,subjectivity,liwc,VADER) using naïve bayes



```
Read 156060 phrases, using 5000 random phrases
Vocabulary Density : 0.3714
Total unique words : 7164
Use Classifier:  Naive Bayes
Each fold size: 1000

Average Precision       Recall          F1      Per Label
0             0.262      0.204      0.228
1             0.278      0.370      0.317
2             0.754      0.703      0.727
3             0.365      0.415      0.388
4             0.314      0.243      0.273

Macro Average Precision Recall          F1      Over All Labels
              0.395      0.387      0.386

Label Counts {0: 218, 1: 827, 2: 2642, 3: 1016, 4: 297}
Micro Average Precision Recall          F1      Over All Labels
              0.548      0.540      0.541
```

In this experiment, a combination of features including unigrams, POS tagging, subjectivity analysis, LIWC, and VADER sentiment scores was used with the Naive Bayes classifier on the dataset without stopwords and punctuation. The micro-average F1 value achieved was 0.541, which significantly falls when compared to the experiment under similar feature sets, without including the VADER (F1 = 0.553).The addition of VADER sentimental scores included an additional layer of sentiment intensity analysis, but it was still unable to significantly improve the performance. The considerably more superior performance continued to be displayed by the Neutral sentiment (Class 2) followed by scoring of 0.727. So, this was just an indication of VADER bringing with it valuable insights into a certain sentiment. I haved used this combination to test with different advanced classifiers

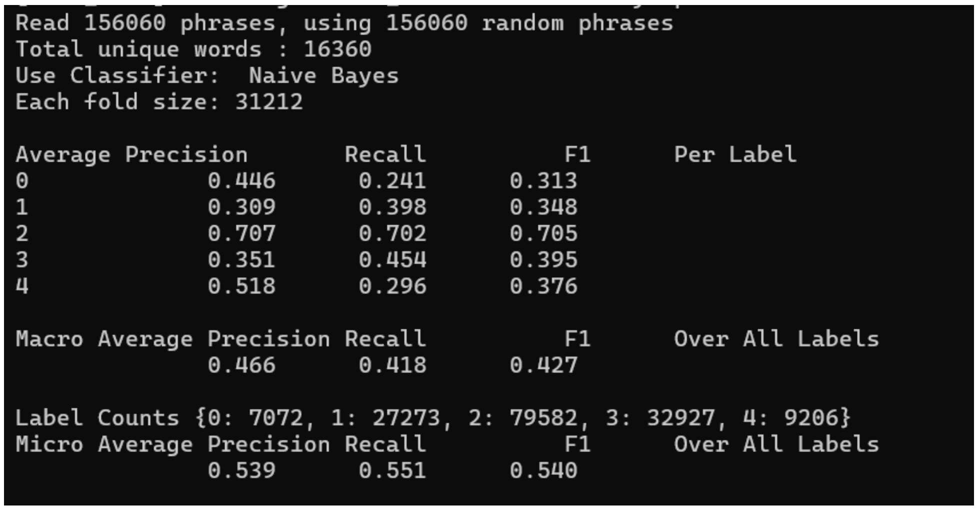
4.2 RESULT SUMMARY

Many different features have been combined, including Unigrams, POS tagging, Subjectivity Analysis, and LIWC as part of experiments. However, the highest micro-average F1 score was 0.553 which is without VADER. This configuration can maintain the capture of linguistic, context, and psychological nuances. Adding negations achieved the terms of understanding and sentiment of data. Addition of vader which is a sentiment lexicon added limited value that might be due to simplicity in the Naive Bayes model. As far as all experiments are concerned, neutral sentiment (Class 2) performed well. The findings also show that adding complementary features significantly enhances overall performance. However, even better potential is particularly possible in predicting underrepresented classes with more advanced models as well as correcting imbalances in class distribution. VADER has added additional layers to sentiment analysis so combination of Unigrams, POS tagging, Subjectivity Analysis, LIWC, and VADER were used with different classifiers to train on whole training dataset of 156060 phrases.

Measures	Baseline	Negation	POS Tagging	SL analysis	LIWC	Combined features with bigram and negation	combined feature without bigram and negation
Precision	0.554	0.554	0.494	0.545	0.549	0.559	0.562
Recall	0.502	0.503	0.453	0.447	0.464	0.528	0.548
F-1 Score	0.507	0.508	0.459	0.462	0.474	0.538	0.553

4.3 ADVANCED CLASSIFIERS

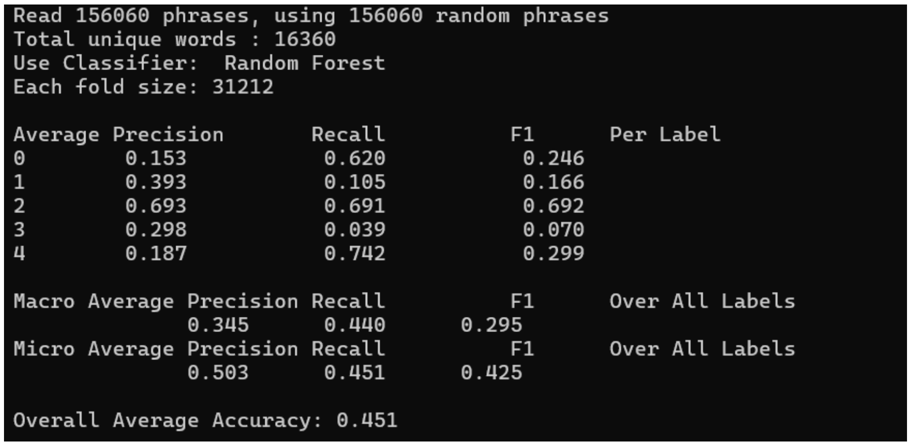
4.3.1. Naïve Bayes



```
Read 156060 phrases, using 156060 random phrases
Total unique words : 16360
Use Classifier:  Naive Bayes
Each fold size: 31212

Average Precision       Recall          F1      Per Label
0              0.446      0.241      0.313
1              0.309      0.398      0.348
2              0.707      0.702      0.705
3              0.351      0.454      0.395
4              0.518      0.296      0.376

Macro Average Precision Recall          F1      Over All Labels
               0.466      0.418      0.427

Label Counts {0: 7072, 1: 27273, 2: 79582, 3: 32927, 4: 9206}
Micro Average Precision Recall          F1      Over All Labels
               0.539      0.551      0.540
```

In this experiment, a combination of **Unigrams, POS tagging, Subjectivity Analysis, LIWC, and VADER** features was applied to the full training dataset using the Naive Bayes classifier. The model had a **micro-average F1 score of 0.540** and a recall of **0.551**. The best performances to be achieved were due to the neutral sentiment class (Class 2), which accounted for an F1 of 0.705. However, under-represented classes like negative (class 0) and positive (class 4) performed far worse, gaining F1 scores of 0.313 and 0.376, respectively, because of class imbalance. This is due to the diverse forms of features incorporated in the model ,but made it less apparent, making it necessary to think that the Naive Bayes might not fully handle the complex features.

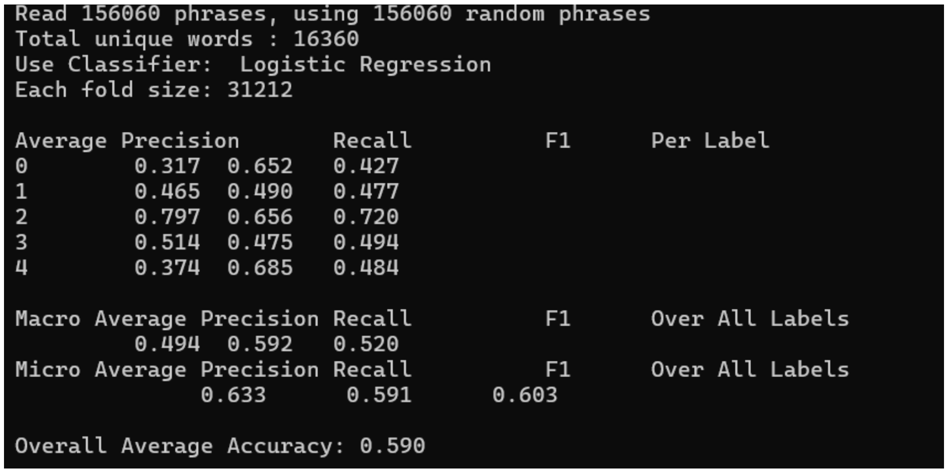
4.3.2 Random Forest



```
Read 156060 phrases, using 156060 random phrases
Total unique words : 16360
Use Classifier:  Random Forest
Each fold size: 31212

Average Precision       Recall          F1      Per Label
0       0.153           0.620           0.246
1       0.393           0.105           0.166
2       0.693           0.691           0.692
3       0.298           0.039           0.070
4       0.187           0.742           0.299

Macro Average Precision Recall          F1      Over All Labels
             0.345      0.440      0.295
Micro Average Precision Recall          F1      Over All Labels
             0.503      0.451      0.425

Overall Average Accuracy: 0.451
```

In this experiment, the combination of Unigrams, POS tagging, Subjectivity Analysis, LIWC, and VADER was used with the Random Forest classifier on the full training dataset. At a micro average level, the highest scores included an F1 point of 0.425 as well as average recall of 0.451 with an accuracy of 0.451 throughout the model. Positive sentiment (Class 4) came out ahead, with an F1 score of 0.299, while the other classes display relatively low performance-particularly Class 3 (Somewhat Positive). The macro average has been reduced ,which is a reflection of the uneven performances, mainly due to class imbalance.

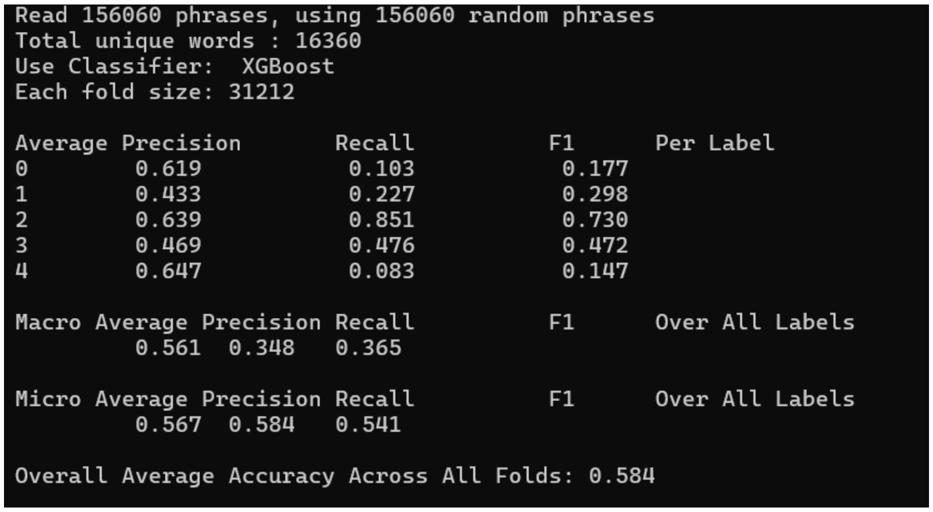
4.3.3. Logistic Regression



```
Read 156060 phrases, using 156060 random phrases
Total unique words : 16360
Use Classifier:  Logistic Regression
Each fold size: 31212

Average Precision       Recall          F1      Per Label
0        0.317  0.652   0.427
1        0.465  0.490   0.477
2        0.797  0.656   0.720
3        0.514  0.475   0.494
4        0.374  0.685   0.484

Macro Average Precision Recall          F1      Over All Labels
         0.494  0.592   0.520
Micro Average Precision Recall          F1      Over All Labels
              0.633      0.591      0.603

Overall Average Accuracy: 0.590
```

In this experiment, the combination of Unigrams, POS tagging, Subjectivity Analysis, LIWC, and VADER was applied using the Logistic Regression classifier on the full training set. The model showed an average accuracy of 0.590, with the best micro-average F1 score at 0.603 for neutral sentiment (Class 2) in the sentiment classes-second only to an F1 score of 0.720. The model had very good recall for negative (Class 0) but suffered its worst precision at a low F1 score of 0.427. Both Class 1 (Somewhat Negative) and Class 4 (Positive) performed worse, generating F1 scores of 0.477 and 0.484 respectively. Logistic Regression was the highest performer against Naive Bayes and Random Forests on the metrics of classification accuracy and overall F1

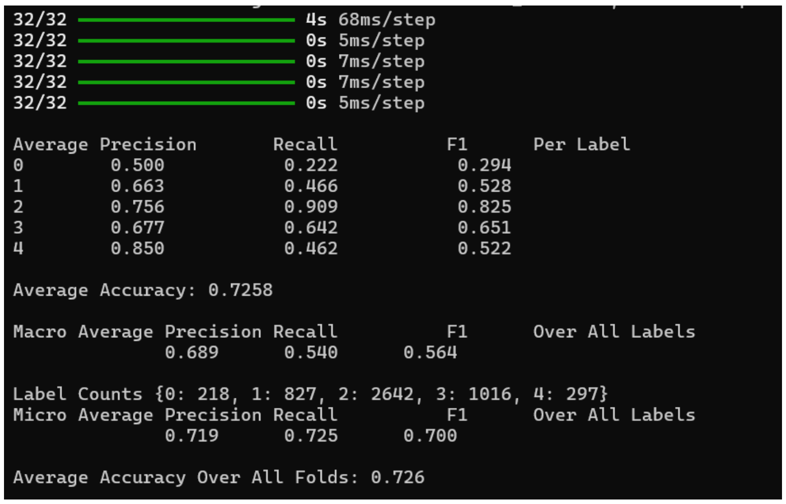
4. XGBOOST



```
Read 156060 phrases, using 156060 random phrases
Total unique words : 16360
Use Classifier:  XGBoost
Each fold size: 31212

Average Precision       Recall          F1      Per Label
0        0.619           0.103           0.177
1        0.433           0.227           0.298
2        0.639           0.851           0.730
3        0.469           0.476           0.472
4        0.647           0.083           0.147

Macro Average Precision Recall          F1      Over All Labels
         0.561  0.348   0.365

Micro Average Precision Recall          F1      Over All Labels
         0.567  0.584   0.541

Overall Average Accuracy Across All Folds: 0.584
```

In this experiment, the combination of Unigrams, POS tagging, Subjectivity Analysis, LIWC, and VADER was applied using XGBoost on the full training dataset, with a micro-average F1 score of 0.541 and overall accuracy of 0.584. The model performed best for neutral sentiment (Class 2) with an F1 score of 0.730. Although on the negative and positive grounds, performance-wise, the model came up unfriendly since their recall and F1 scores were lower. The macro F1 score hit the bottom of 0.365, thereby indicating extreme class imbalance. XGBoost gave the best performance in neutrality.

4.3.5. Recurrent Neural Network



```
32/32 ━━━━━━━━━━━━━━━━━━━━ 4s 68ms/step
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step

Average Precision       Recall          F1      Per Label
0        0.500           0.222           0.294
1        0.663           0.466           0.528
2        0.756           0.909           0.825
3        0.677           0.642           0.651
4        0.850           0.462           0.522

Average Accuracy: 0.7258

Macro Average Precision Recall          F1      Over All Labels
              0.689      0.540      0.564

Label Counts {0: 218, 1: 827, 2: 2642, 3: 1016, 4: 297}
Micro Average Precision Recall          F1      Over All Labels
              0.719      0.725      0.700

Average Accuracy Over All Folds: 0.726
```

In this experiment, the combination of Unigrams, POS tagging, Subjectivity Analysis, LIWC, and VADER was used with a Recurrent Neural Network (RNN), with an accuracy of 0.7256. F1 score of 0.825 was obtained for the neutral (Class 2) sentiment class, and 0.528 for the rather negative (Class 1) class. However, the F1 scores of these classes were poorer for their positive (Class 4) and negative (Class 0) counterparts, achieving 0.522 and 0.294, respectively, in the evaluations. Furthermore, the model had high performance compared to other models, which confirms better addressing the class imbalance in minority-class prediction.

4.4 SUMMARY FOR CLASSIFIERS

The highest accuracy was obtained by the RNN on 0.7256, a micro-average F1 score of 0.700 in point of sentiment, and an F1 score of 0.825 in neutral sentiment (Class 2), which were trained with full training data on Unigrams, POS tagging, Subjectivity Analysis, LIWC, and VADER. Logistic Regression illustrated highly desirable results with an F1 score of 0.603, while XGBoost had a micro-average F1 score of 0.541. Naive Bayes and Random Forest gave modest performance, especially for minority classes, because of class imbalance. However, the best performance had been given by RNN, which also had problems with class imbalance and constraints and which requires further optimization.

Measures	Naïve Bayes	Random Forest	Logistic Regressiom	XGBOOST	RNN
Precision	0.539	0.503	0.633	0.567	0.719
Recall	0.551	0.451	0.591	0.584	0.725
F-1 Score	0.54	0.425	0.603	0.541	0.7
Accuracy	54%	45.10%	59%	58.40%	72.60%

5. DISCUSSION

5.1 OBSERVATIONS

1. **Best Performance for Neutral Sentiment:** Across all models, the highest F1 score was achieved with an F1 score of 0.817 by the RNN, which classifies neutral sentiment (Class 2) as the easiest to accomplish likely due to more balanced and clear features in the dataset.

2. **Minority Classes issues:** The performances were low especially for Negative (Class 0) and Positive (Class 4) sentiments for their recall. For example, Class 0 has an F1 score as low as 0.177 in Naive Bayes and an F1 score of 0.387 in RNN. It is challenging, and the challenging part is the class imbalance, where minority classes are often misclassified.

3. **Variation in model Performance:** Recurrent Neural Networks (RNNs) achieved the overall best performance with accuracy 0.7256 and F1 score 0.541, and also superior to other models. But then, the performances of both Naive Bayes and Random Forest were not encouraging at all, particularly for the minority classes. Logistic Regression and XGBoost showed moderate performance, but did not outshine the complete effectiveness of RNNs.

4. **Impact of feature Set:** Using unigrams, POS tagging, subjectivity analysis, LIWC, and VADER combined improves the performance across the models. However, these features make the simpler classifiers like Naive Bayes less effective, with the deep learning models (RNNs) taking better advantage of the characteristics.

5.2. LESSONS LEARNED

1. **Effect of class imbalance:** Even though advanced features were used, there was an influence of the class imbalance on performance at higher levels specifically for classes positive and negative. This implies that there should be oversampling or conditioning with class weights in models resampled to address this imbalance.

2. **Deep Learning Models:** The deep learning models performed better than the traditional methods. They showed better performance in the overall measures, showing that complex models like RNN are much better handling the kinds of feature combinations employed in this project.

3. **Importance of Feature Selection:** The increased number of features, such as LIWC or VADER, POS tagging, and subjectivity analysis greatly improved the accuracy of the model. However, the real efficacy of these features was highly dependent on their employment within different models. Naive Bayes is not able to quite handle the complexity of these features when it comes to such rich input cases as with RNNs.

4. **Model Complexity versus Model Performance:** The simplicity of naive Bayes or Random Forest didn't work best with the features used in this experiment. Advanced models like RNNs were ideally suited to extracting value from complex features, which simpler models may not necessarily capture due to some capitation.

5. **Efficient Performance in the Minority Class Group:** Among these classes, neutral sentiment was well predicted. Still, it becomes equally important to improve the effectiveness of minority classes (negative and positive in sentiments) in balanced sentiment analysis. Implements could be extended to the advanced methods of class balancing, tuning fine hyperparameters, modeling that indeed better deals

5.3. LIMITATIONS

1. **Class Imbalance:** significant problem in all experiments has turned out to be the class imbalance visible in the dataset. Underrepresentation of the minority groups- negative sentiment class (Class 0) and positive sentiment group (Class 4)- resulted in performance degradation, as indicated by lower recall and F1 scores. Consequently, the resulting outcomes, where it is clear the model does much more well than neutral sentiment in reality is more frequent comparing to other classes performances, showed just how the model became less significant due to class imbalance.
2. **Model Limitations:** Models like Naive Bayes and Random Forest are unable to fully realize the effectiveness of combining the advanced technologies such as VADER, LIWC, and POS tagging. These models did not have the capability to handle the contextual as well as linguistic relationships buried within such data, making poor performances due to the above large number of features.
3. **Resource Intensive:** Deep learning models (RNNs) are more resource-consuming than traditional machine learning models, like Naive Bayes or Random Forest. Training RNNs on a large dataset is hardware intensive, and these resources may not be accessible and might limit the applicability of RNNs to resource-poor environments

4. **Feature Redundancy:** Incorporation of the features with Unigrams, Part of Speech (POS) tagging, Subjectivity, LIWC model, and VADER increased the depth of the analysis and brought in the redundancy which made the models complex. Therefore, the emphasis should be placed on prior feature selection to refine feature set for some models like Naive Bayes that may not benefit from such a complicated feature set.

5. **Lack of Fine-Tuning:** The models worked quite successfully based on default settings, and it permitted only very minor hyperparameter tuning. Optimizing hyperparameters like learning rate, batch size for an RNN, or tree depth for Random Forest could very well lead to better results, although this was beyond the scope of experiments.

6. CONCLUSION

Feature selection was one of the most important steps in the project. I have trained several machine learning models and tested their performance in sentiment analysis using a combination of features including Unigrams and POS tagging, and some other features like Subjectivity Analysis, LIWC, and VADER features. Among the models traditionally developed for sentiment analysis, Recurrent Neural Networks emerged as the prominent one, resulting in a better accuracy and micro average F1 score than Naive Bayes and Random Forest on neutral sentiment. Still, some classes were hard to predict as the class imbalance problem persisted. Even with such imbalanced classes, minority classes such as negative (Class 0) and positive (Class 4) did not exhibit good performance. Although feature engineering made a significant contribution to model performance, simpler models were not potent on very complex feature sets. In conclusion, it can be observed that the RNN with other features is the most favorable and can marshal complex information efficiently, but to have rich performance, these output results must be improved in some areas of imbalance in class and optimization.