Alex Geiger  
Norma Grubb  
Final Project  
8/18/2022

**Sentiment Analysis: A Unique Approach**

Table of Contents

[Data overview and Analysis 1](#_Toc114246466)

[Data Introduction 1](#_Toc114246467)

[Distribution of movie sentiment 2](#_Toc114246468)

[Tokenization a Revolutionary Approach 2](#_Toc114246469)

[POS Tag Removal 2](#_Toc114246470)

[Lowercasing Characters and Stop word Removal 3](#_Toc114246471)

[Stemming 3](#_Toc114246472)

[POS Token Removal 3](#_Toc114246473)

[Modeling 3](#_Toc114246474)

[Cross Validation 3](#_Toc114246475)

[Count Vectorizer 4](#_Toc114246476)

[Input To Model Overview 4](#_Toc114246477)

[Model Overview 5](#_Toc114246478)

[Model Results 5](#_Toc114246479)

[High Level Overview 5](#_Toc114246480)

[Top 25 Folds 6](#_Toc114246481)

[Creating Predictions 6](#_Toc114246482)

# Data overview and Analysis

## Data Introduction

The following dataset is a corpus of Rotten tomatoes movie reviews for the purposes of conducting sentiment analysis. The goal of the dataset was to associate sentiment to each phrase. The sentiment for each phrase is utilized as the model target and was provided by a crowd sourcing platform. The following data was provided by the instructor include both a training and testing datasets. The training dataset contains 156,060 rows and 4 columns. The columns include a phrase ID, Sentence ID, Phrases and the Sentiment associated to the phrase and sentence ID. In the test dataset only the Phrase ID, Sentence ID and Phrase are provided.

## Distribution of movie sentiment

The distribution of the sentiment score appears to be normally distributed and not uniformly distributed. Several techniques which include under sampling or oversampling may be used to balance the data. However, due to the low number of instances of some categories the sentiment of the data was not balanced.

Chart, bar chart

Description automatically generated

# Tokenization a Revolutionary Approach

## POS Tag Removal

The following tokenization process for the training and testing dataset utilized a multi-stage approach to develop two distinct sets of tokens which could be utilized by the model. The first and most unique step is the removal of tokens which have a determiner POS tag. The tokens were derived from a string which referred to the phrase column in the training and testing datasets. The reason why determiners were removed was to reduce the total size of the number of features in the model.

POS stands for Part of speech, which tags tokens and associates a dimensionality of meaning to a word. For example, in the sentence “I had fun doing this assignment” translates to the following tokens and POS tags.

[('I', 'PRP'), ('had', 'VBD'), ('fun', 'NN'),

('doing', 'VBG'), ('this', 'DT'), ('assignment', 'NN')]

Where I is a possessive pronoun, had is a past tense verb, fun is a noun, doing is a present verb, this is a determiner and assignment is a noun. The POS tagger is one of the most thrilling topics covered in the NLP course.

## Lowercasing Characters and Stop word Removal

The next step was to lowercase all the characters in the string and remove stop words. The reason why every letter is lowercased is to not double count certain words. For instance, “This” and “this” would be two separate tokens by lowercasing the characters they become one in the same. Next, stop word removal removes the most frequent irrelevant words in a sentence like “the”, “our”, “my” and many more. By removing these words, you minimize the feature space and potential improve the model’s accuracy.

## Stemming

After, each word in the sentence was Stemmed. Stemming is when the stem of a word or ending of the word is removed. For instance, the word “ending” will become “end” the suffix “ing” will be removed. Examples of stems include but are not limited to “ing”, “ed”, “es” and “er”. The following graphics depicts the processes that are associated to each of the two feature curation processes.

Feature 0

POS Token Removal

Stop Word Removal

Apply Stemmer

Lowercase words

Stop Word Removal

Apply Stemmer

Lowercase words

Feature 1

## POS Token Removal

The algorithm responsible for the POS token removal involved the concepts of tokenization and POS tagging. The algorithm starts by passing a string into the definition known as *removePosToken*. The definition first tokenized the string that had been passed into the definition. After, each token was passed into the NLTK POS tagger. Once each token had been tagged, each token and its corresponding tag were iteratively processed. If the POS tag of a token match the POS tag the algorithm was removing the token would be disregarded else the token would be saved to a modified tokens list which is returned from the definition. In order to apply this algorithm to each phrase of the pandas dataframe was fend into the pandas apply definition which stream the phrases to the definition *removePosToken.* After, the results were store in the column known as feature\_0.

# Modeling

## Cross Validation

The model training utilized K-Fold cross validation. K fold cross validation is when the training dataset is broken into n segments where n number of models are trained. Each time the model is trained one of the n segments is left out from the training dataset and is used to test the model accuracy, precision, recall and F1 Score. In total 10 segments or splits were utilized in the Sklearn object KFold. Once the training and testing datasets are built out two district features are iterated through to train the model these two features are known as feature 0 and feature 1.

## Count Vectorizer

Once the feature is defined the string or cleaned phrase is passed into sklearn count vectorizer. The count vectorizer is an object which converts a collection of documents or phrases into a sparse representation of word counts. The reason why this tool is so effective is because the sparse matrix takes up far less memory than the list of dictionaries required by the NLTK model. The vectorization process works by counting the number of times n-grams appear in the document or corpus which is being analyzed. For instance, consider the document Doc 0:

Doc 0: “the man had gone to the store”

If we do not remove stop words and only vectorize the unigrams we can create the vector [2, 1, 1, 1, 1, 1] by counting the number of times each instance of a unique unigram occurs in the sentence. each rows represents the corpus, and the columns represent a n-gram. Each element represents the number of times that n-gram appeared in the corpus. For instance, since the word “the” occurred twice in Doc 0 the element corresponding to Doc 0 and the n-gram “the” has a value of 2 because the word appeared two times in the document. In a tabular format instead of a matrix we can present the data as follows where each row would be the input to a model.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Doc | The | Man | Had | Gone | to | Store |
| 0 | 2 | 1 | 1 | 1 | 1 | 1 |

The count vectorizer parameters which specifies the n-grams to count in the corpora is known as ngram\_range. The parameter is a two-element set which specifies the lower and upper bound of the range of n-grams which are to be extracted from the corpora and counted. The three distinct count vectorizer objects all utilized different ngram\_range values for the training of models. The different values for the ngram\_range parameter included:

* When ngram\_range = (1,1) only unigrams are extracted from the corpora
* When ngram\_range = (1,2) both unigrams and bigrams are extracted from the corpora
* When ngram\_range = (2,2) only bigrams are extracted from the corpora

## Input To Model Overview

In summary, a total of three count vectorizer objects, and two distinct types of features as described above were utilized to train the models. In total six distinct sets of features were developed to train the model which are outlined below.

* Feature 0, Unigram
* Feature 0, Bigram
* Feature 0, Unigram and Bigram
* Feature 1, Unigram
* Feature 1, Bigram
* Feature 1, Unigram and Bigram

Each of these 6 distinct sets of model inputs were utilized to train 10 different models, one for each of the ten folds. Therefore, in total, sixty models were trained. Each model was evaluated on accuracy, precision, recall, and it’s F1-score. The data that was utilized to evaluate the model was one of the ten folds which was held out for testing.

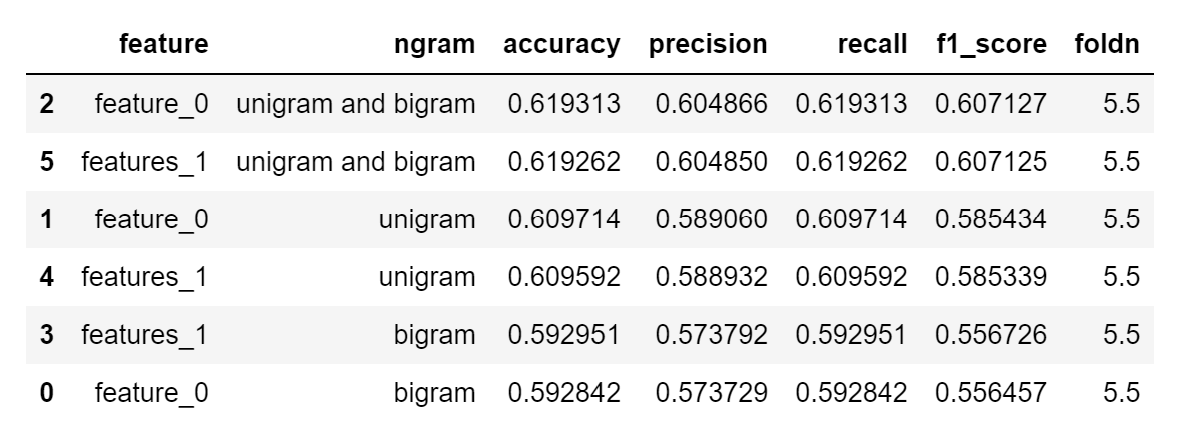
## Model Overview

The model that was trained was a multinomial Naïve Bayes classifier model. The multinomial naïve bayes classifier is perfect for handling discrete values as inputs. Therefore, the count vectorization process pairs perfectly with this particular model since the count vectorization process produces a discrete number of counts of n-grams.

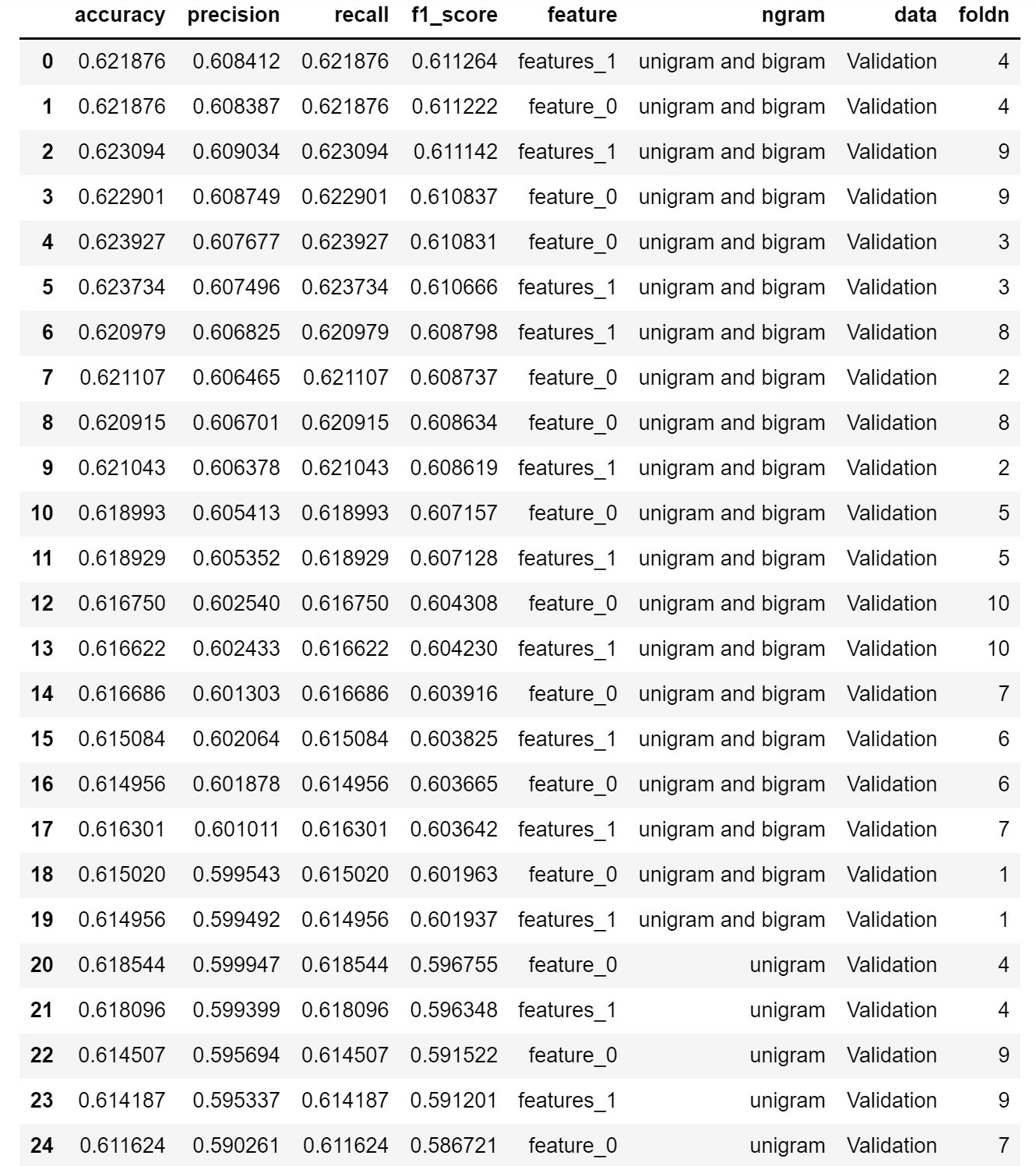
## Model Results

### High Level Overview

The model that had performed the best was the input combination of feature 0 and the unigram and bigrams vectorization. The average KPIs across all the folds included an accuracy of 0.6193, a precision of 0.604866, a recall of 0.619313 and an f1 score of 0.607127.



### Top 25 Folds



## Creating Predictions

The final part of the project was to generate predictions for the phrases in the testing dataset provided in the corpus folder. To generate these predictions, the models associated to the best feature combination were utilized to generate predictions for the class project. Since ten model predictions had to be combined one for every fold the predict\_proba definition was utilized. This predicted the likelihood of each sentiment classification. All 10 predictions were summed and the sentiment category which had the highest probability was chosen. The results were saved to the file predictions.csv.