**SEINFELD**

**GROUP PROJECT**

Group 4: Stephanie Azali

Christopher Bailey

John Leo Bartlett

Micaela Feliciano

Juan Ventosa

Course: IST 736 Text Mining

Professor: Dr. Ami Gates

Assignment: Final Group Project

Date: 3/29/2021

Table of Contents

[Introduction: 2](#_Toc67755541)

[Analysis and Models: 2](#_Toc67755542)

[About the Data: 2](#_Toc67755543)

[Data Cleaning: 3](#_Toc67755544)

[Exploratory Data Analysis (EDA) 4](#_Toc67755545)

[Additional Data Sets: 8](#_Toc67755546)

[Data Transformation: 8](#_Toc67755547)

[Vectorization: 9](#_Toc67755548)

[Defining Parameters 11](#_Toc67755549)

[Models: 12](#_Toc67755550)

[Character Classification: 12](#_Toc67755551)

[Humor Sentiment Analysis: 14](#_Toc67755552)

[Stand-Up Topic Modeling: 15](#_Toc67755553)

[Rating Correlation: 16](#_Toc67755554)

[Results: 17](#_Toc67755555)

[Character Classification: 17](#_Toc67755556)

[Cross Validation 17](#_Toc67755557)

[Confusion Matrix 20](#_Toc67755558)

[Indicative Features 24](#_Toc67755559)

[Humor Sentiment Analysis: 25](#_Toc67755560)

[Stand-Up Topic Modeling: 27](#_Toc67755561)

[Rating Correlation: 27](#_Toc67755562)

[Resulting Character Significance 29](#_Toc67755563)

[Result Summary: 31](#_Toc67755564)

[Conclusion 31](#_Toc67755565)

[Figure 1. Sample Text containing Seinfeld Script 3](#_Toc67362911)

[Figure 2. Screenshot of Cleaned Text Data 4](#_Toc67362912)

[Figure 3. Histogram of Number of Lines: The Top 20 Seinfeld Characters (54,084 Lines Total) 5](#_Toc67362913)

[Figure 4. Histogram of Word Counts by Line of Dialogue (54,084 Lines Total) 5](#_Toc67362914)

[Figure 5. Word Count Stats for Dialogue. Max = 364, Min = 1, Mean = 10.5, STD = 11.3) 6](#_Toc67362915)

[Figure 6. Number of Lines by Character (54,084 lines Total) 6](#_Toc67362916)

[Figure 7. Process of Loading, Vectorizing and Transforming CSV Files to a Data Frame 7](#_Toc67362917)

[Figure 8. Sample of 7 Lines of Dialogue Transformed by Stemming 8](#_Toc67362918)

[Figure 9. Example of a Vectorized Matrix Using Term Frequency of three simple documents 8](#_Toc67362919)

[Figure 10. Illustrative Example of a TFIDF Calculation 9](#_Toc67362920)

[Figure 11. Custom List of 179 English Stop Words Stemmed. (Stemmed Stop Words) 10](#_Toc67362921)

[Figure 12. Percent of Classes by Data Set and Data Size 12](#_Toc67362922)

[Figure 13. Data Sampling/Train-Test Split and 5-Fold Cross Validation Process Performed 12](#_Toc67362923)

[Figure 14. Visualizing SVM Classification Kernels: Linear, Polynomial and Radial Basis 13](#_Toc67362924)

[Figure 15. Illustrative Latent Dirichlet Allocation 14](#_Toc67362925)

[Figure 16. Summary of Train and Test Accuracy Scores from 5 Fold Cross Validation 15](#_Toc67362926)

[Figure 17. Term Frequency Naïve Bayes and Decision Tree Train vs. Test Accuracy Scores 16](#_Toc67362927)

[Figure 18. TFIDF L1 Norm Naïve Bayes and Decision Tree Train vs. Test Accuracy Scores 16](#_Toc67362928)

[Figure 19. TFIDF L2 Norm Naïve Bayes and Decision Tree Train vs. Test Accuracy Scores 16](#_Toc67362929)

[Figure 20. 5 Fold Cross Validation Summary of Precision, Recall & F1 Scores by Class & Model 17](#_Toc67362930)

[Figure 21. Confusion Matrix Results on Multinomial Naïve Bayes Term Frequency Data 18](#_Toc67362931)

[Figure 22. Confusion Matrix Results on Multinomial Naïve Bayes TFIDF L1 Norm Data 18](#_Toc67362932)

[Figure 23. Confusion Matrix Results for Multinomial Naïve Bayes Euclidean (L2) Norm 19](#_Toc67362933)

[Figure 24. Confusion Matrix Results for Decision Tree Term Frequency 19](#_Toc67362934)

[Figure 25. Confusion Matrix Results for Decision Tree L1 Norm Data 20](#_Toc67362935)

[Figure 26. Confusion Matrix Results for Decision Tree Euclidean (L2) Norm 20](#_Toc67362936)

[Figure 27. The 15 Highest Feature Probabilities by Class (Multinomial Naïve Bayes L1 Norm) 21](#_Toc67362937)

Figure 31. Confusion Matrix Results for SVM using Radial and Linear Kernels.......................25

Figure 32. Confusion Matrix Results for SVM using Polynomial Kernel...................................26

Figure 33. Top 20 Features for Funny Sentiment Analysis Linear SVM model..........................27

# Introduction:

P1 – About the Show

P2 - It’s Characters

P3 – Show about nothing

P4 – Defining Funny

# Analysis and Models:

## About the Data:

The source of the data is *Seinfeld text corpus* taken from a Kaggle.com. The text document which contains all the Seinfeld scripts from 179 episodes was contributed by one of its users, Luong Nguyen (<https://www.kaggle.com/luongleanstocode/seinfeld-text-corpus>). A screenshot sample of the document is shown in Figure 1.

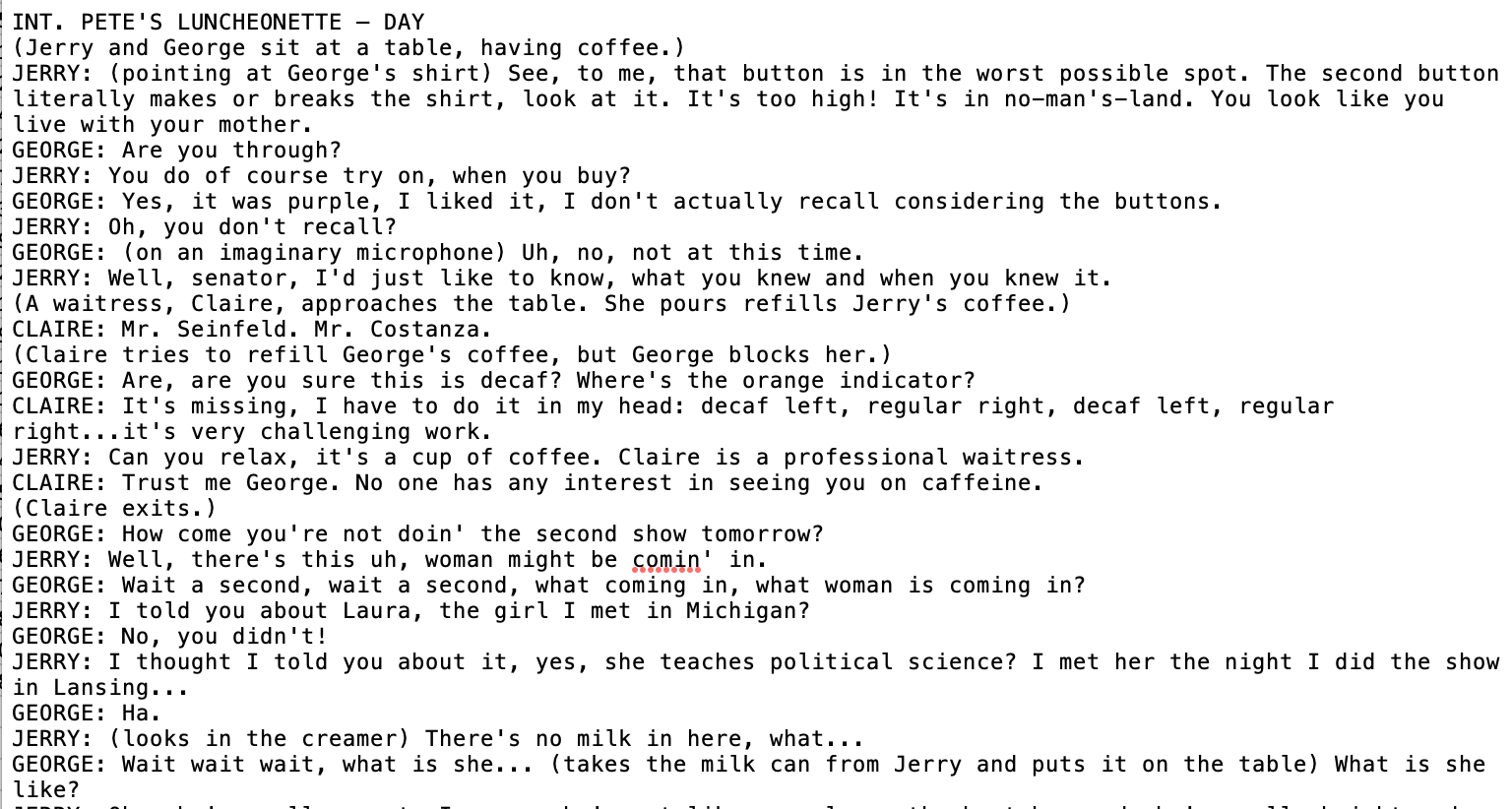


Figure 1. Sample Text containing Seinfeld Script

## Data Cleaning:

Of main interest from the scripts were the character names and their lines of dialogue. These items were extracted from the script for analysis. Parts of the script that were excluded were scene headings, setting descriptions, actions, transitions and camera shots. The text for the character names and the dialogue were fairly clean. Numeric digits, punctuation and unwanted symbols (@, #) were removed the text. In some cases the character names and dialogue contained action descriptions in parenthesis. For example a dialogue could appear as “JERRY: (looks in the creamer) There’s no milk in here, what…” Since the action descriptions did not represent what the characters said, these parenthetical items were also removed from the corpus.

The ‘Character’ and ‘Dialogue’ were placed in a data frame with each representing a column or field. A ‘Label’ column was added to identify the 4 main characters (*Jerry, George, Elaine, Kramer*) plus a fifth label named *Other* to identify all other characters. Lastly a ‘WordCount’ column was added which provided the number of words contained in each line. Figure 2 provides a screenshot of the data frame in csv format.

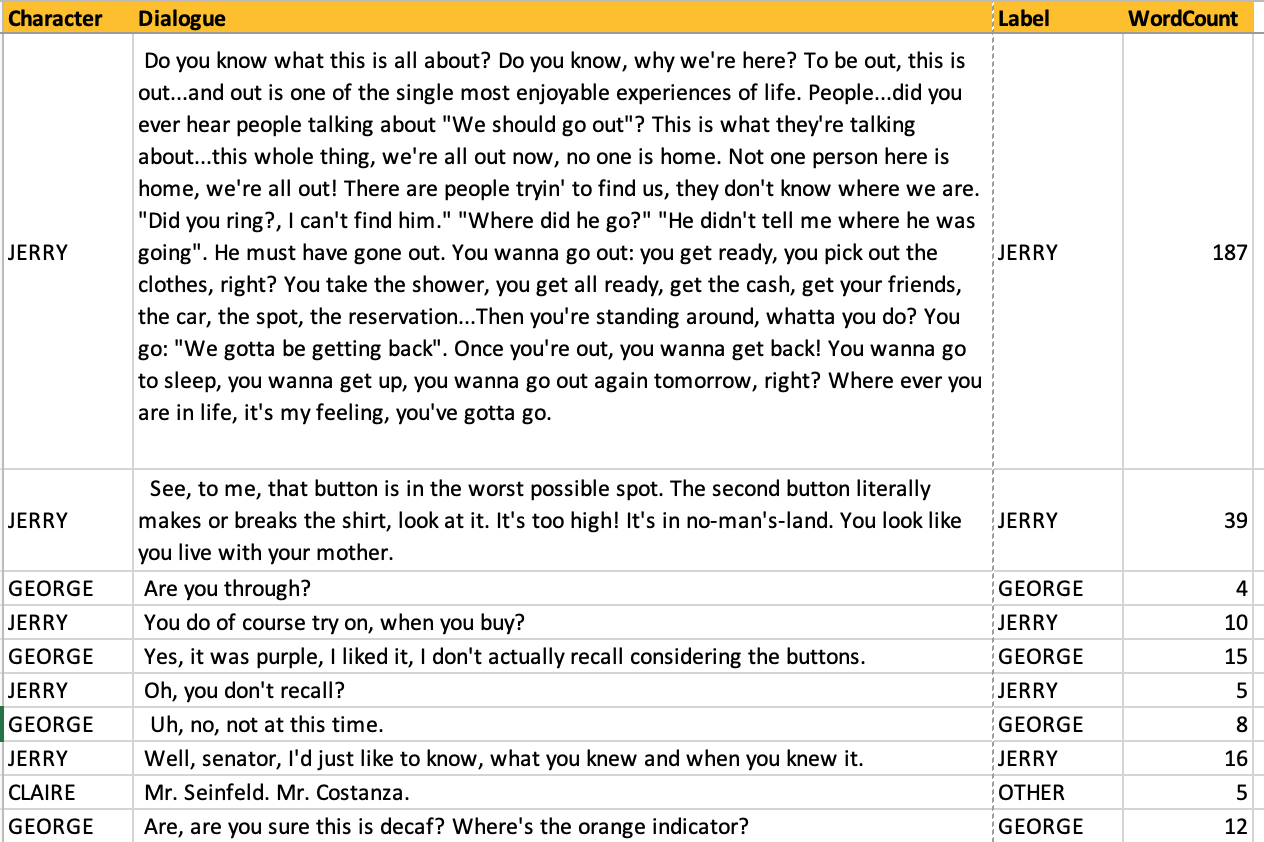


Figure 2. Screenshot of Cleaned Text Data

### Exploratory Data Analysis (EDA)

Before the data transformation process to prepare the dataset for vectorization and model analysis, the data was explored to gain some stats about the data. An inspection of the number of lines spoken by each character revealed that the 4 main characters delivered 72% of the lines in the show with Jerry alone speaking 27% of all lines (Figure 3 Histogram). The vast majority of lines (over 30,000 out of 54,084) had less than 10 words per line (Figure 4 Histogram). The longest lines contained 364 words with a mean of 10.5 words per line (Figure 5). Even with all non-main characters’ (Other) combined Jerry Seinfeld still had the most lines in the show (Figure 6). Looking at the words per episode over different seasons and grouped characters there is a observable trend in the division of lines (Figures 7 & 8).

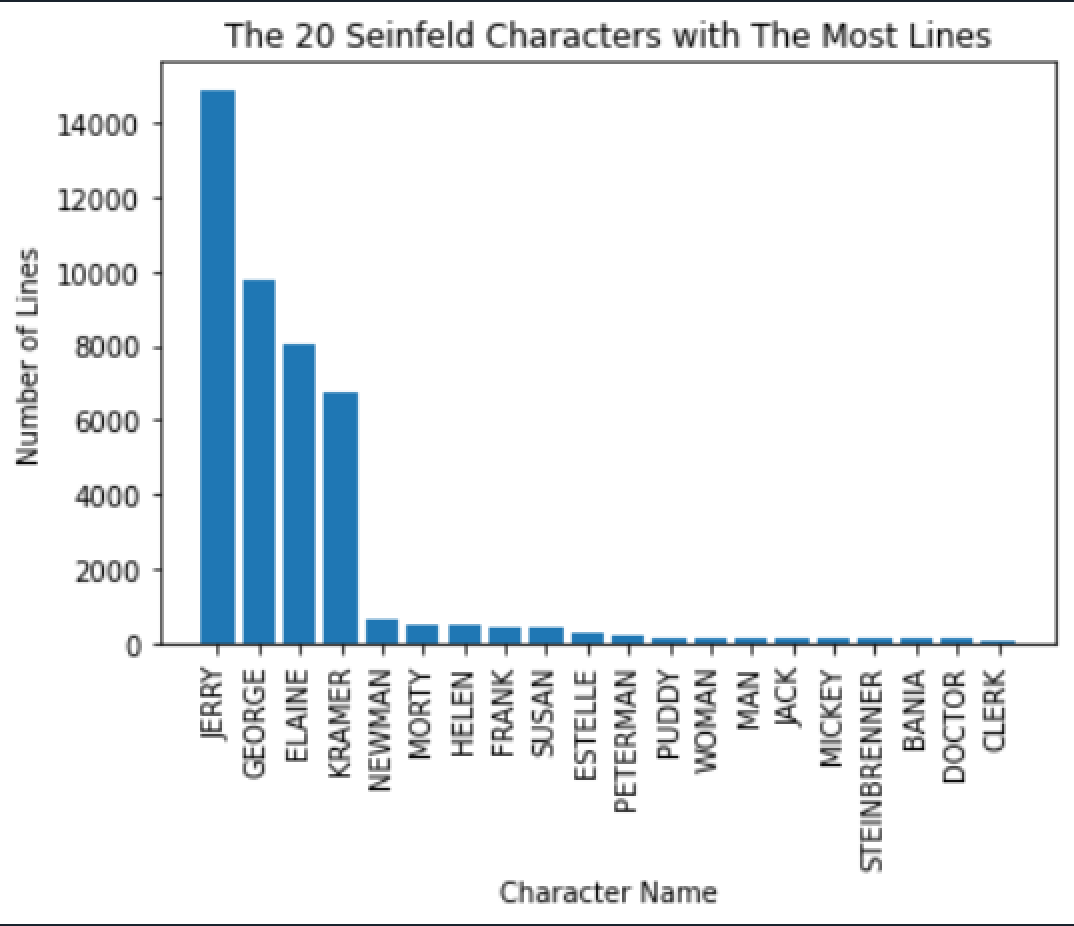


Figure 3. Histogram of Number of Lines: The Top 20 Seinfeld Characters (54,084 Lines Total)

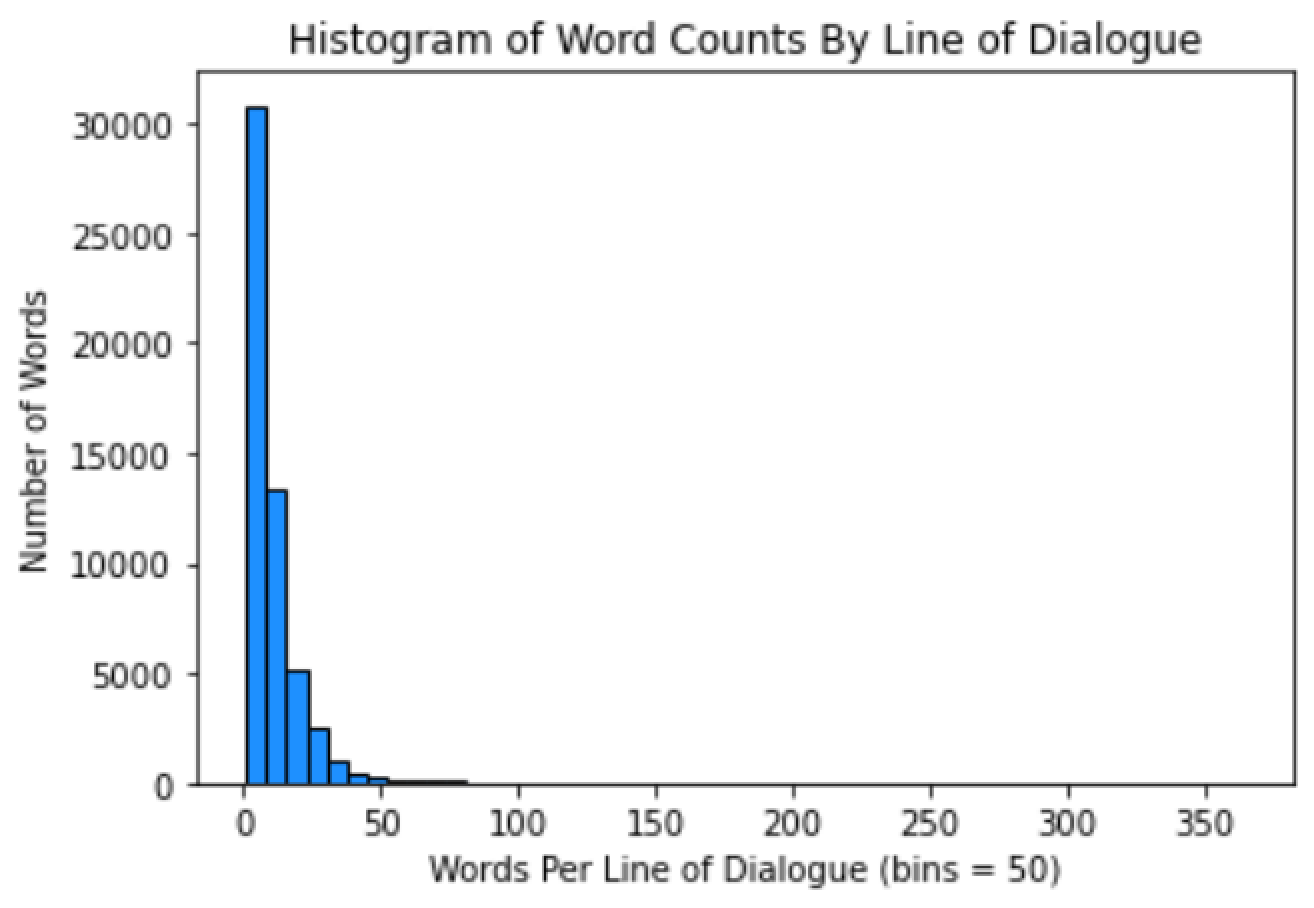


Figure 4. Histogram of Word Counts by Line of Dialogue (54,084 Lines Total)

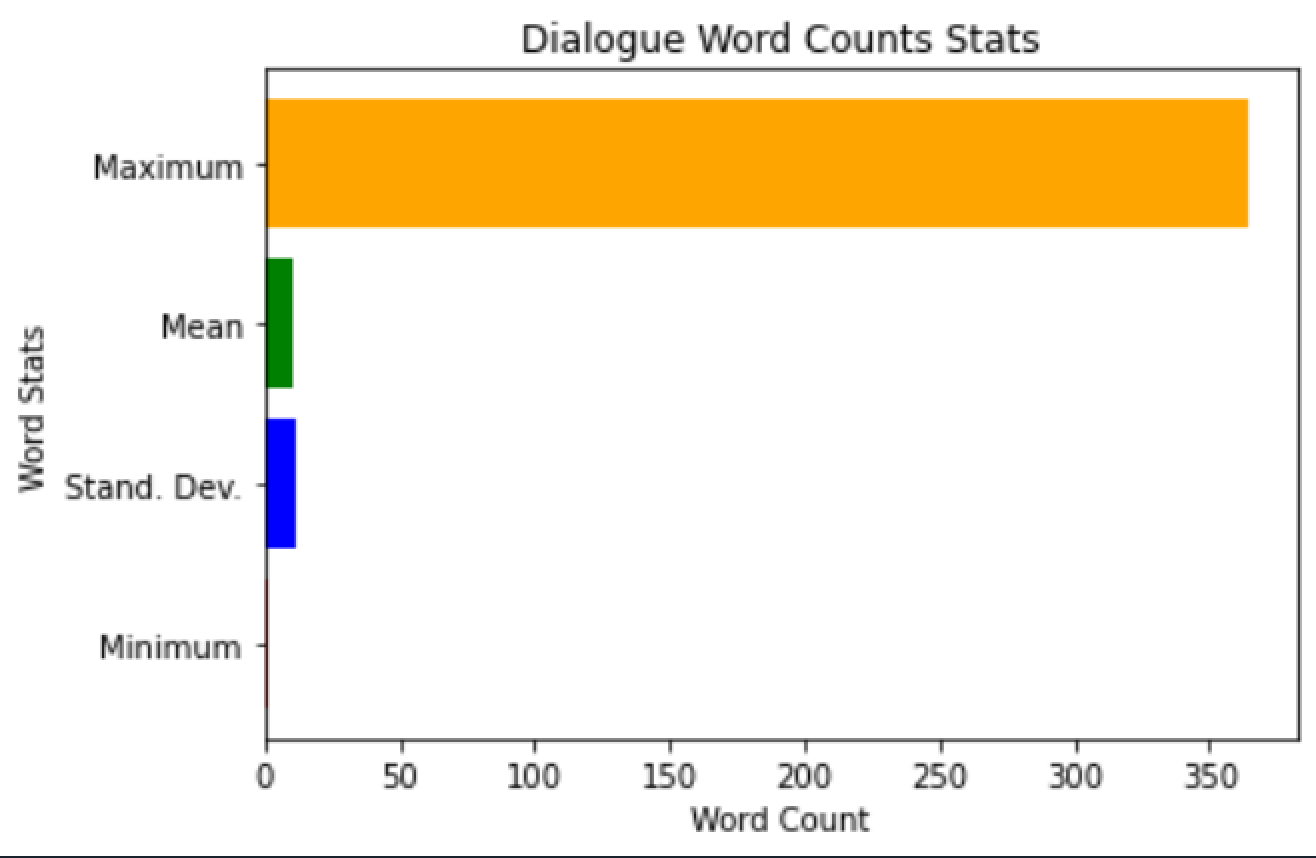


Figure 5. Word Count Stats for Dialogue. Max = 364, Min = 1, Mean = 10.5, STD = 11.3)

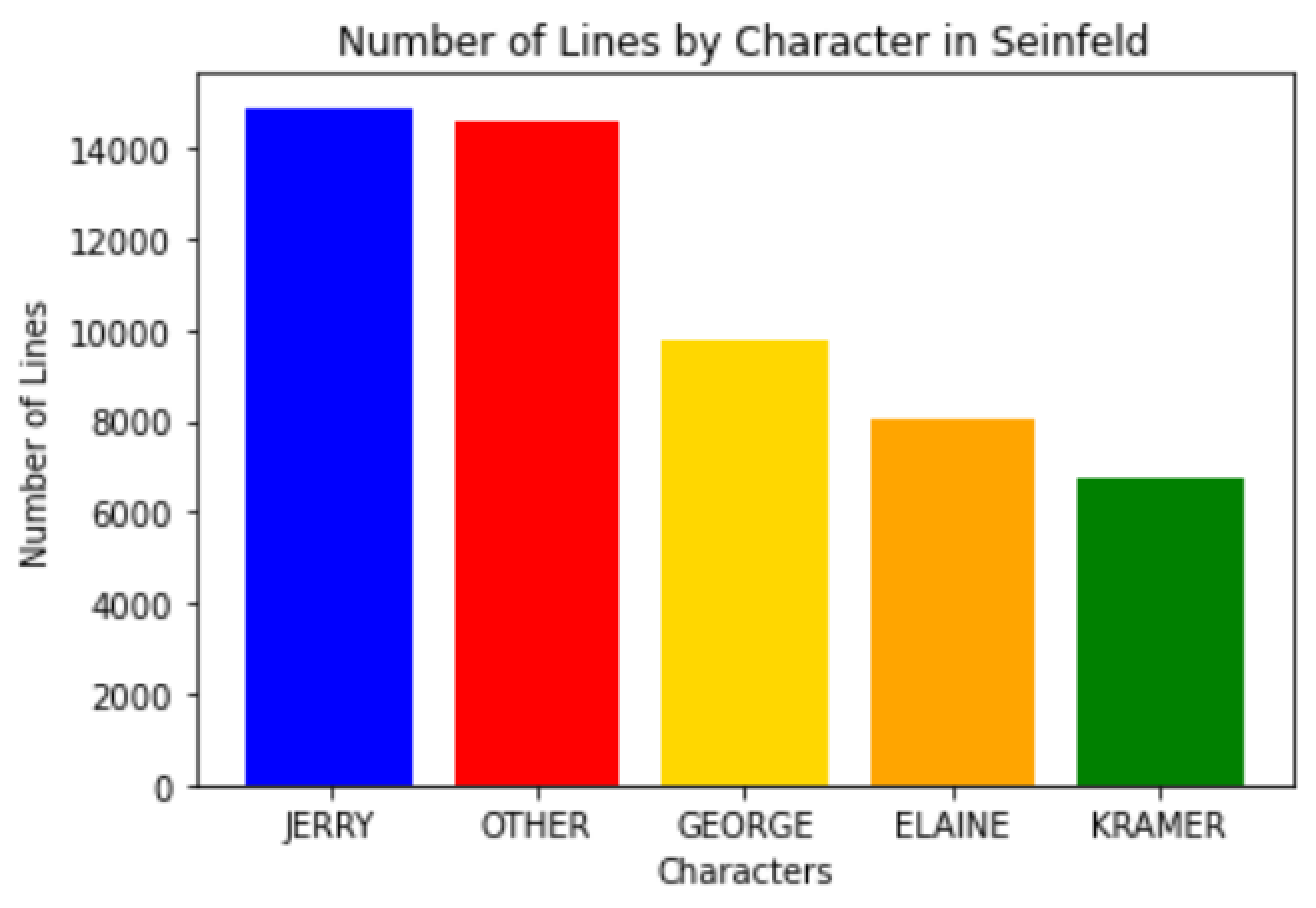
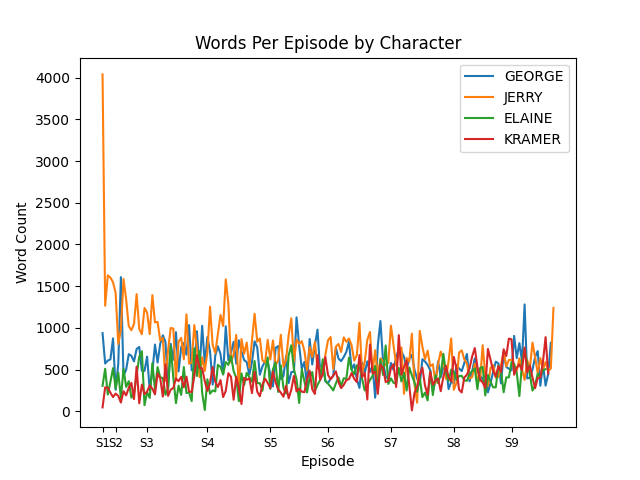


Figure 6. Number of Lines by Character (54,084 lines Total)

 Figure 7. Words per episode by character over time

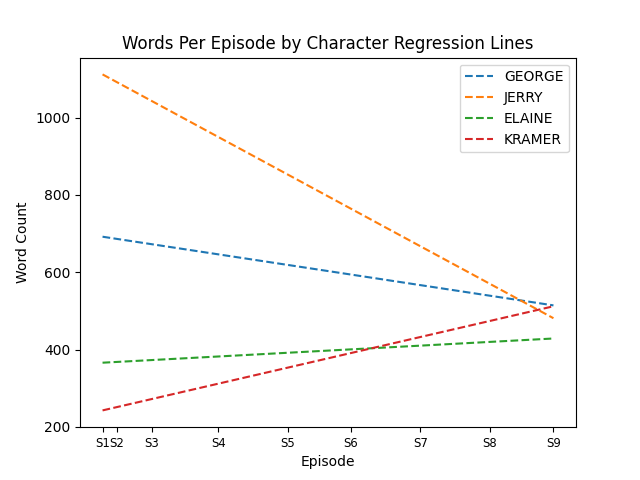


Figure 8. Words per episode by character over time, regression lines

### Additional Data Sets:

To meet the objectives listed below parsing the original data and collecting supplementary data was deemed necessary to meet both the objective requirements and time constraints.

* Classifying Dialogue Lines by Character
  + From the 54,084 rows of data a sample of 5000 rows for train and 3000 rows for test-validation were randomly selected. A more detailed description of this process will be provided in the Models section.
* Funny or Not Funny Sentiment Analysis
  + Every episode of Seinfeld starts with Jerry delivering a stand-up bit in front of an audience. The dialogue from these 191 stand-up lines were extracted from the text data and labeled as “Funny.”
  + News stories containing headlines and brief descriptions were collected using NewsAPI.org. The text was cleaned of unwanted digits, characters and symbols and labeled as “Not Funny.”
  + The Funny and Not Funny data sets were then merged to create a single data set to train and test models on sentiment analysis.
* Topic Modeling Seinfeld Stand Up
  + Jerry’s stand-up bits at the beginning of each show typically contain jokes that are related to the theme of the episode. These jokes cover a gambit of subjects ranging from dating and relationships to laundry, driving and waiting rooms. These 191 lines were extracted from the text data for the purpose of topic modeling.
* Character Dialogue Rating Correlation
  + IMBD ratings of Seinfeld episodes by season was. As well as additional information about in which episode dialogue occurred. The order in which episodes aired and contributing writers and directors.

### Data Transformation:

The contents of the cleaned data in .csv format was converted into a data frame in order to train and test the models. Python was utilized to extract the text data, transform it into a document matrix (sparse matrix), and transfer the data into data frames.

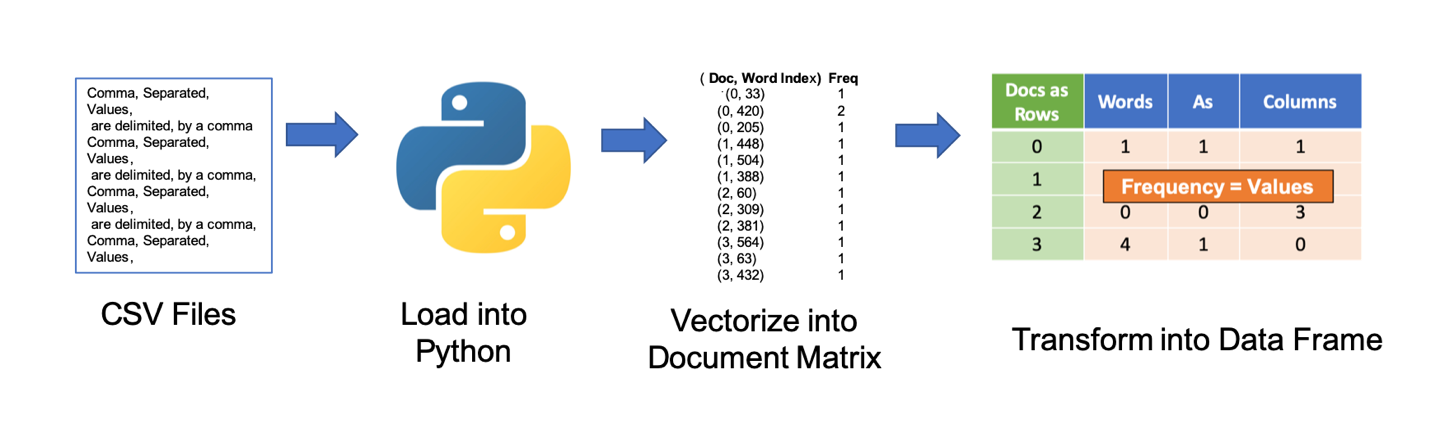


Figure 9. Process of Loading, Vectorizing and Transforming CSV Files to a Data Frame

Given the large number of rows and potential large number of features in the data, the terms were stemmed as defined by the NLTK “Porter Stemmer” module available in Python. Stemming is an attempt to simplify words to their “root” form. For example a verb in present or past tense such as “Jumping” or “jumped” would be transformed to simply lowercase “jump.” The theory behind stemming is that words that have the same essential meaning do not need to be redundantly represented through multiple features thereby reducing the number of features represented in a sparse matrix. A sample of the stemmed data is shown in Figure 8.

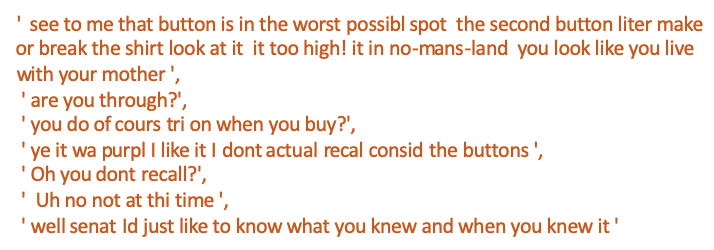


Figure 10. Sample of 7 Lines of Dialogue Transformed by Stemming

## Vectorization:

Three methods of vectorization were employed. The CountVectorizer module in Python was used to vectorize the data using a term frequency method. This metric is especially effective if the length of the documents in the data are relatively the same size. The mean length of words per line of dialogue was 10.5 and over 30,000 lines had less than 10 words.

1. Term Frequency – features are given a value that reflects the number of times a feature appears in a document (Figure 9).



Figure 11. Example of a Vectorized Matrix Using Term Frequency of three simple documents

The next two methods utilizes a Term Frequency Inverse Document Frequency (TFIDF) calculation. Document frequency is defined as the number of documents that contain a term or feature. The inverse would be the document frequency represented as a fraction with the numerator and denominator exchanging positions. The calculation then takes the term frequency multiplied by the log of the inversed document frequency or TF\*log(IDF). See Figure 13 for an illustrative example.

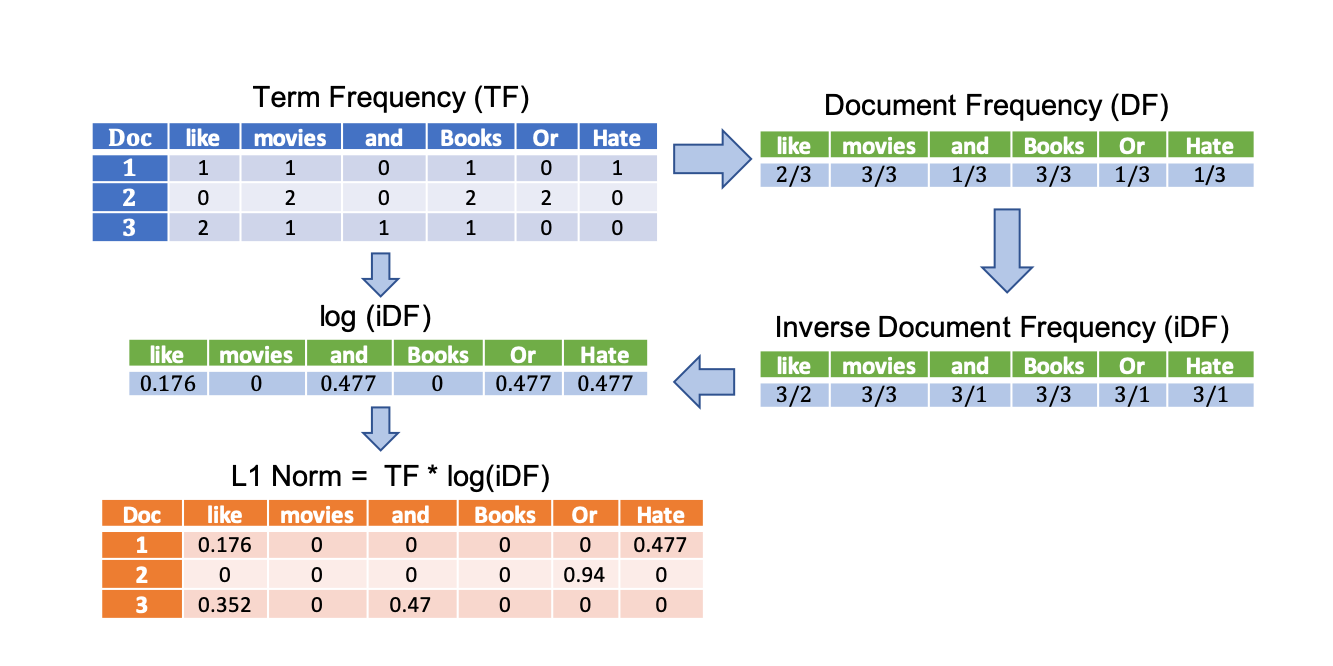


Figure 12. Illustrative Example of a TFIDF Calculation

The methods below define how the TF portion of the formula are normalized.

1. Document Length Normalization – features are given a value that reflects the number of times a feature appears in a document relative to the size or total number of features in a document. (i.e., feature counts in document/total number of features in document)
2. Euclidean Length Normalization – features are given values that reflect the number of times a feature appears in a document relative to the Euclidean length of a documents vector.

An additional consideration will be stop words, which are a list of chosen words that may not add relevant value to an analysis. Some examples of stop words in the English language include articles (the, a, an, etc.), prepositions (of, in, on, etc.) and pronouns (he, she, him, her, etc.). Since the text data was stemmed prior to vectorization, the NLTK English stop words will also be stemmed to accurately match the intended stop words in the data set.

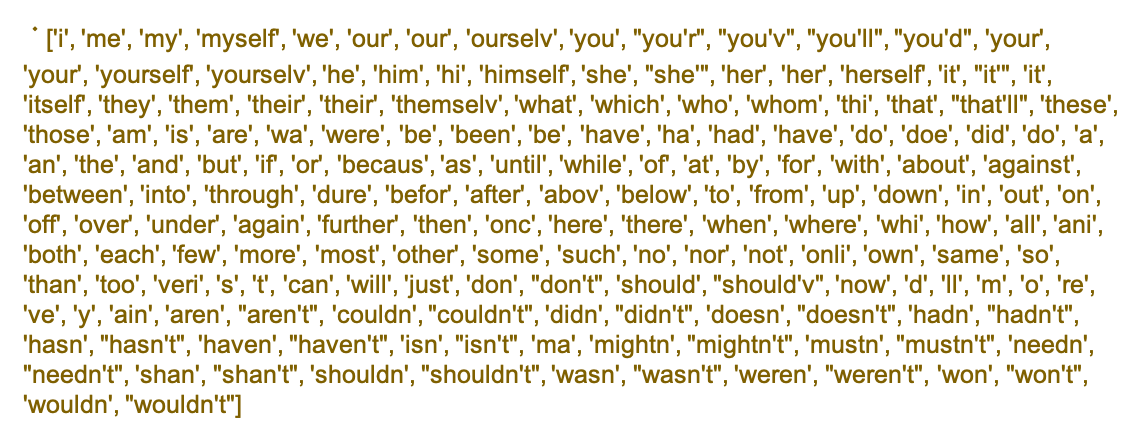


Figure 13. Custom List of 179 English Stop Words Stemmed. (Stemmed Stop Words)

### Defining Parameters

Python’s CountVectorizer module was set to the following parameters:

stop words = custom stem stop words (Figure 7)

analyzer = word

n grams = unigram & bigrams

minimum document frequency = 5

The TfidfVectorizer module in Python was utilized to vectorize the data set using two methods.

Both methods used the following parameters

stop words = custom stem stop words

n grams = unigram & bigrams

analyzer = word

minimum document frequency = 5

Difference between the two vectorizers were based on the normalization parameter.

Method one used norm = L1 normalization (document length normalization)

Method two used norm = L2 normalization (Euclidean length normalization)

## Models:

Given the three objectives of this analysis, each of the models will be separately described according to how they pertain to each objective. The objectives will be discussed in the following order:

1. Character Classification
2. Humor Sentiment Analysis
3. Stand-up Topic Modeling

### Character Classification:

To predict the character classes as either ‘Jerry’, ‘George’, ‘Elaine’, ‘Kramer’ or ‘Other’, Multinomial Naïve Bayes (MNB) and Decision Tree (DT) algorithms were utilized. MNB is a supervised classifier machine learning algorithms known to work well on discrete features. DT is also a supervised machine learning algorithm that can handle both continuous and discrete values in order to perform regression or classification. Each of these algorithms will be trained using each three forms of vectorized data for a total of 6 models.

The model parameters were defined as follows:

Multinomial Naïve Bayes

alpha = 1.0 -- applies Laplace Smoothing

fit prior = True -- to learn class prior probabilities

Decision Tree

criterion = entropy – a measure of uncertainty in which higher value = greater uncertainty

splitter = best – algorithm chooses number of node splits

minimum samples split = 2 – samples required to split a node

minimum samples leaf = 1 – samples required to be a leaf node

#### Training and Testing the Data:

Given 6 model-data combinations and the large size of the data set (54,084 rows), the following procedure of sampling the data was used to reduce the time of processing the training and testing of 6 models. The first step to reduce the size of the data was to only include the 4 main characters (Jerry, George, Elaine and Kramer). All other characters were removed. The 4 main characters represented 73% of all lines of dialogue. This reduced the data to 39,502 rows. The data was then vectorize using all three vectorization methods mentioned in the Vectorization section.

The vectorized data set was randomized and a random sample of 3000 rows were used to represent a Test-Validation set. The 3000 rows were then removed from the original data to eliminate the chance of duplication in the Train set. For the Train data set, a sample of 5000 rows were taken in which classes were chosen randomly but set to equal proportion. Figure 12 shows the proportion of classes in each data set.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **DATA SETS** | **DATA SIZE (ROWS)** | **PERCENT OF DATA SET BY CLASS NAMES** | | | |
| **JERRY** | **GEORGE** | **ELAINE** | **KRAMER** |
| ORIGINAL DATA | 39,502 | 38% | 25% | 20% | 17% |
| TEST-VALIDATION DATA | 3,000 | 39% | 24% | 20% | 17% |
| TRAIN DATA | 5,000 | 25% | 25% | 25% | 25% |

Figure 14. Percent of Classes by Data Set and Data Size

Sampling the data will allow for faster processing times. In an attempt to provide adequate examples for the models, a balanced train data set provided equal amounts of data for each class. The train set was then placed through a fold cross validation process in which 20% of the train data was held out as a test for each fold of the validation without replacement. The models were used to predict both the train and test of each fold in order to collect the accuracy scores of the train error and the test error. The average of the train error and the test error were compared to check for overfitting. In addition the 5-fold cross validation precision, recall and F1-scores for each of the two classes were also recorded for comparison. Figure 16 illustrates the process described above.

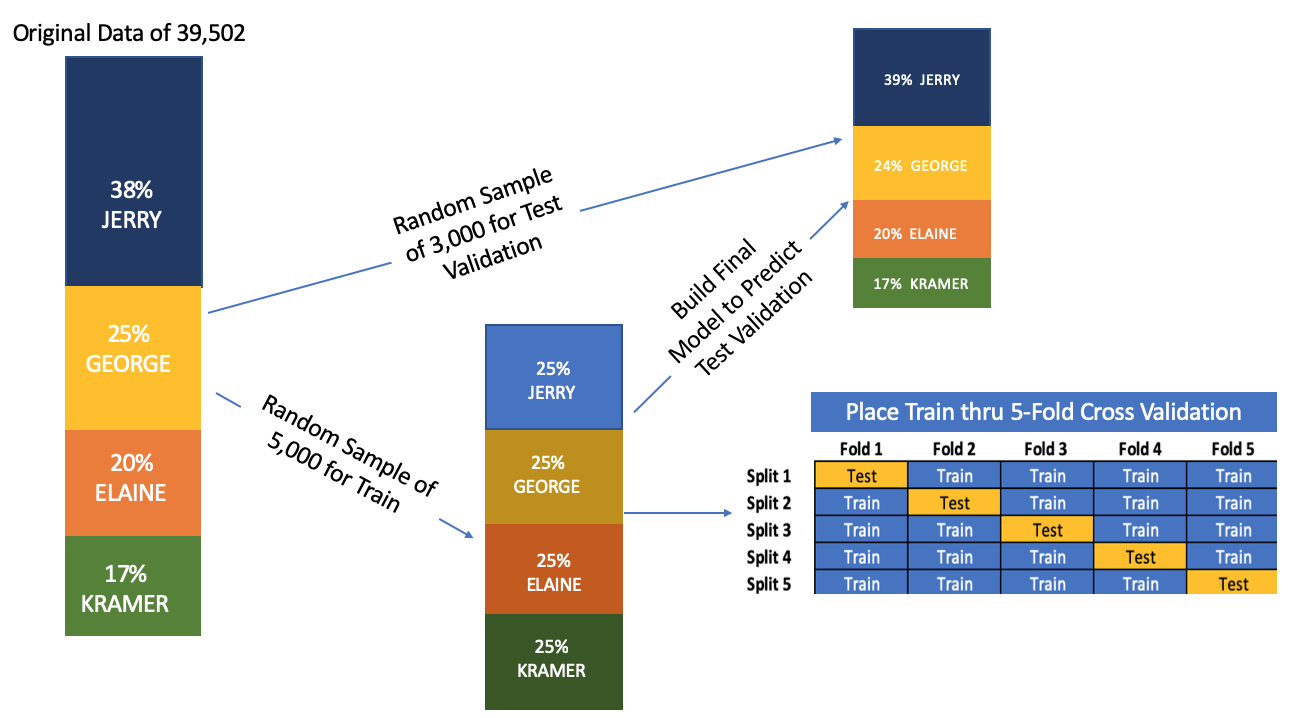


Figure 15. Data Sampling/Train-Test Split and 5-Fold Cross Validation Process Performed

Upon completion of the cross validation process, the models were then trained on the entire train data for a final model. The final models were used to predict the test-validation data to get an overall accuracy measure to be compared with the cross validation results. Confusion matrices of the final models’ predictions on their respective test-validation data were generated to measure precision, recall and accuracy scores. In addition the 20 features with the highest probability for each class were extracted using the Naïve Bayes results.

### Humor Sentiment Analysis:

The idea of humor is quite complex. There is no conclusive “funny formula” in which humans can agree, consequently making the task of detecting humor far tougher for machine learning algorithms. Though this day and age, humans can come to a consensus about one thing: the news is definitively unfunny. Using a combined labeled dataset consisting of opening Seinfeld monologue lines, and a CSV created from multiple News API sources, a humor detection sentiment analysis was performed. For this objective Support Vector Machines (SVM) were chosen to build a model to predict the sentiment analysis of “Funny” or “Not Funny.” As a classifier, the objective of Support Vector Machine’s (SVM) algorithm is to create hyperplanes in N-dimensional space to best categorize features to their defined classes. In text mining N would equal the number of features with their term and document frequency values designating the vector coordinates in multidimensional space. SVM tries to best separate the classes in that space by maximizing the margin of distance between the data points of the classes. The SVM algorithm uses multiple kernels to achieve this task. Three will be used in this analysis: Linear, Polynomial and Radial Basis. Each kernel uses the same concept of separating the data, but how each kernel separates the data differs with linear taking the most straightforward approach while polynomial and radial basis uses mathematical transformations of the vectors that may work best for large complexities in the data. Below is an illustration of how each kernel may handle different conceptual forms of separating data in a two-dimensional space.



Figure 16. Visualizing SVM Classification Kernels: Linear, Polynomial and Radial Basis

Three SVM models were defined with each model using one of the three kernels (linear, polynomial or radial basis). All other parameters for the SVM’s were left at their default settings.

### Stand-Up Topic Modeling:

To determine if coherent topics can be distinguished from the Stand-Up scenes of Seinfeld, the Latent Dirichlet Allocation (LDA) algorithm in Python was used to perform topic modeling. Latent Dirichlet Allocation performs topic modeling through a process of defining documents and words to a given number of topics in a Dirichlet distribution and iteratively associates word topic assignments using Bayesian multinomial distribution that takes into account the term and document frequencies in a text data (Figure 23).

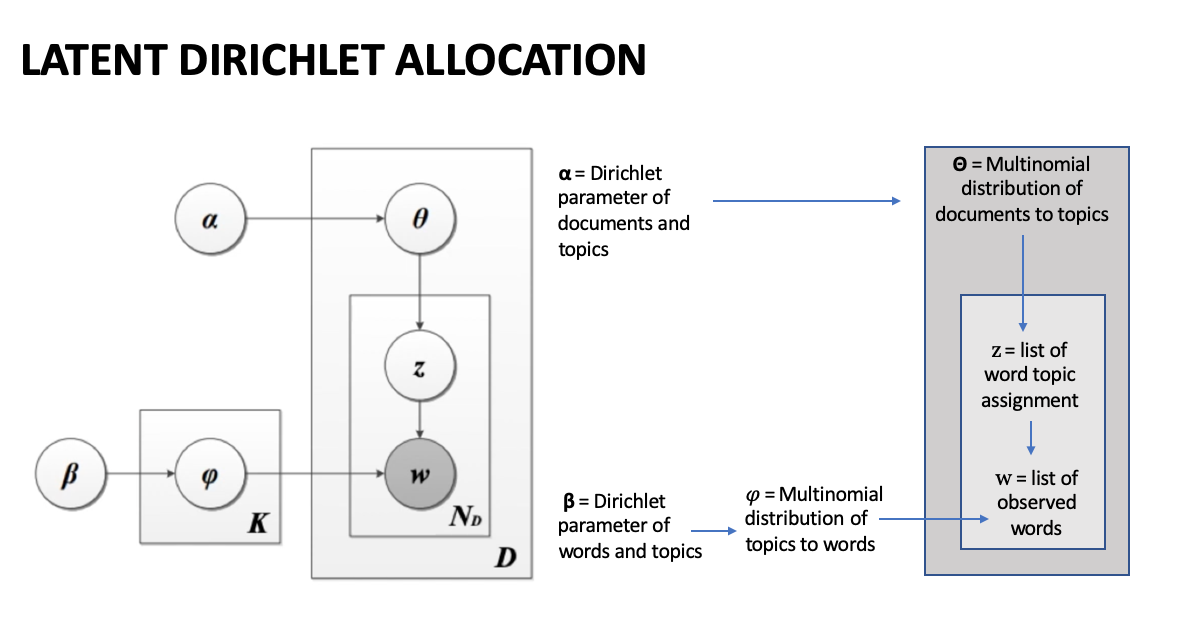


Figure 17. Illustrative Latent Dirichlet Allocation

The LDA models in Python were defined with the following parameters.

n components = 3 - the number of possible topics assumed

max iteration = 1000 – maximum iterations LDA improves word topic association

random state = 23 – to keep process consistent and allow for results to be replicated

### Rating Correlation:

To include ratings into the analysis and additional dataset is required, conveniently IMBD ratings have already been compiled from IMBD API into a csv available online via Kaggle (https://www.kaggle.com/hod101s/seinfeld-imdb-ratings/version/1). Alternatively, IMBD ratings could be compiled manually via IMBD API but obtaining relevant licensing for said API has become more difficult as of late. Obtaining it live should not have any effect on the actual findings so settling for a static csv is appropriate. Additional data about the season and episode number, air date, writer, director will also be needed, and can be easily collected via a separate Kaggle collection (https://www.kaggle.com/thec03u5/seinfeld-chronicles). Aspects of these 2 datasets are then combined into one master info dataset, containing all relevant information about an episode. Finally, the episode info data set is merged onto the dialogue data set according to season and episode.

The scope of characters will also need to be expanded as although 4 main characters delivered 72% of the lines as explained above, the impact of their lines do not necessarily eclipse the impact of the minor characters with the latter share of 28% of total lines. For these reasons, the scope of characters was extended to the characters with the top 10 number of lines. This entails restarting the cleaning process and not creating an Other category of characters in the resulting data frame.

# Results:

Similar to the Models section above, the Results section will be segmented into three parts based on the objectives of the analysis. The order in which the results will be presented will be as follows: 1) Character Classification, 2) Humor Sentiment Analysis, 3) Stand-Up Topic Modeling and 4) Rating Correlation.

## Character Classification:

The classification results based on the Multinomial Naïve Bayes and Decision Tree models were recorded separately for each vectorized data set. Among the results recorded were the average cross validation accuracy scores for the models’ predictions on both the train and test data to evaluate the level of overfitting if any. For each fold of the cross validation, the precision, recall and F1-scores were also recorded for each of the classes. The range of the test accuracies were recorded as baselines for the final model’s accuracy on the test-validation data. A confusion matrix of each of the final models’ prediction on the test validation data was plotted to illustrate the precision, recall and accuracy scores of the final models. Lastly the feature probabilities of the Naïve Bayes models were extracted to compare how and if each model’s performance on each test data set indicated similar or different features associated with each class.

### Cross Validation

All 6 combinations of model-train data sets underwent a 5-fold cross validation. The folds used no replacement meaning the train data was divided into fifths with each quintile alternating as the test data for each fold. A summary of train and test accuracy scores for each model’s performance on the term frequency, TFIDF L1 Norm and TFIDF L2 Norm data sets are provided along with a range of the test accuracy scores recorded during the cross validation (Figure 16). Based on the large difference in the model’s accuracy at predicting the train data vs. the test data during the cross validation process suggest that all six models showed signs of overfitting. This is especially true for the Decision Tree Models. Interestingly the average accuracy scores for predicting the test data all fell within the 0.30 to 0.33 score for all six models. The chance probability for all 6 data sets was 25%. A bar chart comparison of each model’s train vs. test accuracy scores by each fold of the cross validation are shown in Figures 17 through 19.

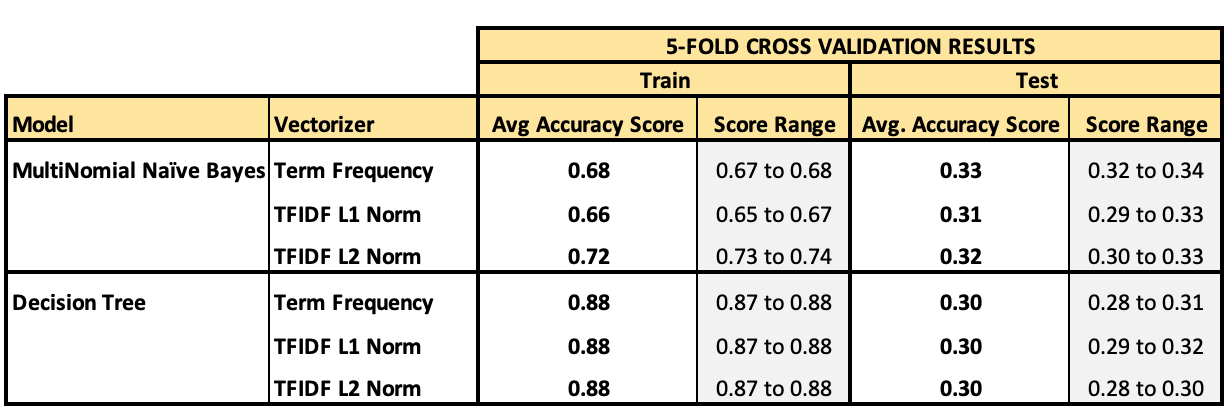


Figure 18. Summary of Train and Test Accuracy Scores from 5 Fold Cross Validation

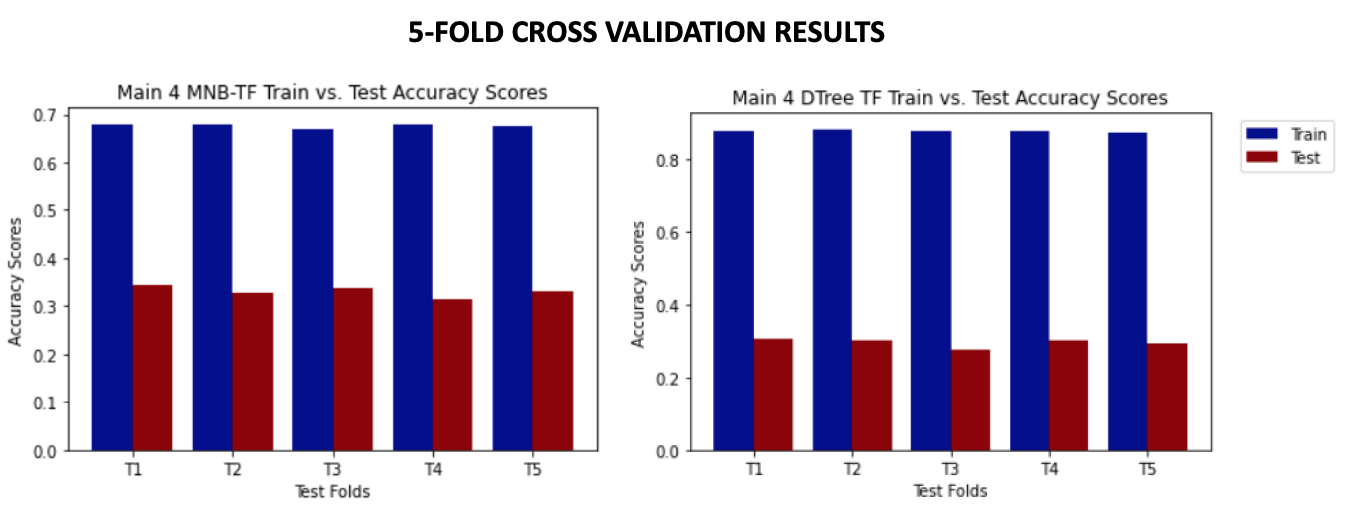


Figure 19. Term Frequency Naïve Bayes and Decision Tree Train vs. Test Accuracy Scores

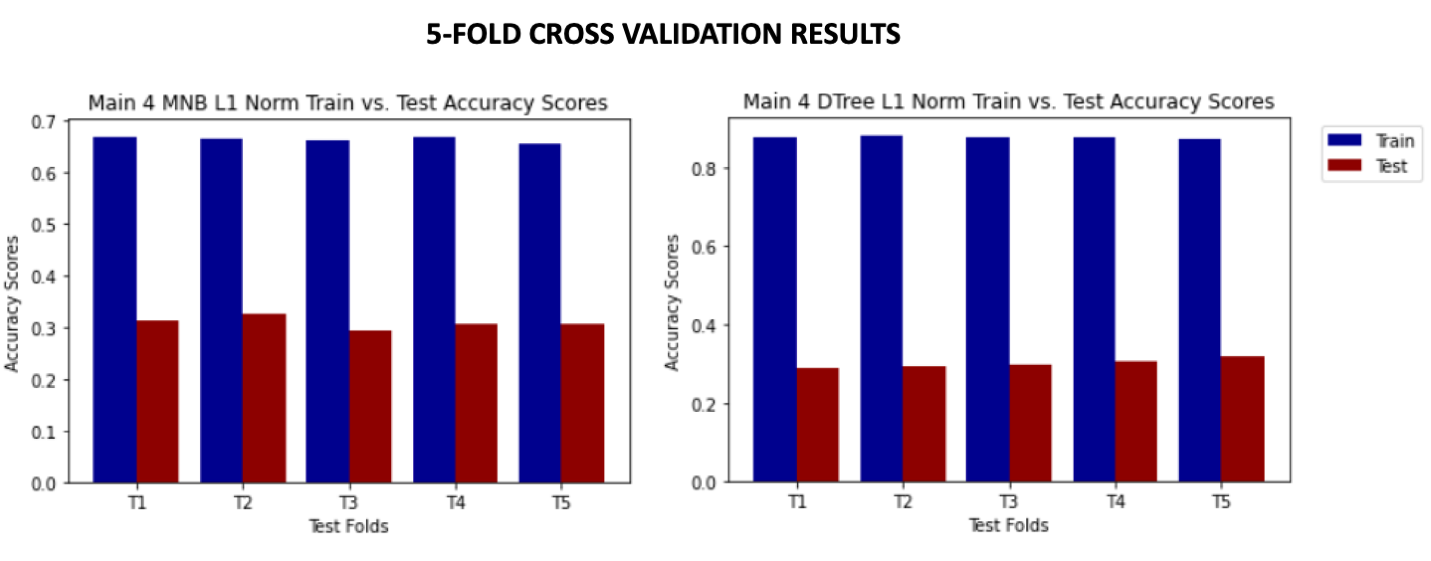


Figure 20. TFIDF L1 Norm Naïve Bayes and Decision Tree Train vs. Test Accuracy Scores

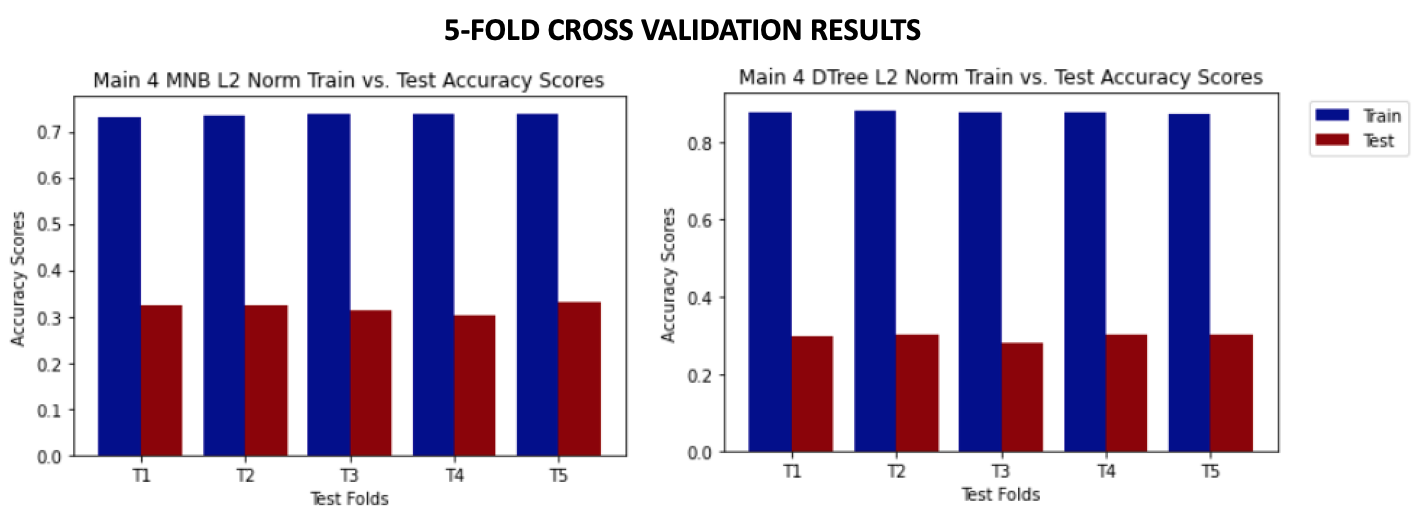


Figure 21. TFIDF L2 Norm Naïve Bayes and Decision Tree Train vs. Test Accuracy Scores

A table summarizing the precision, recall, and F1-scores by model and class for each fold of the cross validation process is available in Figure 20. Based on the results it appears that the Multinomial Naïve Bayes models were most accurate at classifying lines spoken by Kramer. The Decision Tree models were most successful at capturing Elaine’s lines as Elaine’s but were more likely to identify correctly predicted lines as Kramer.

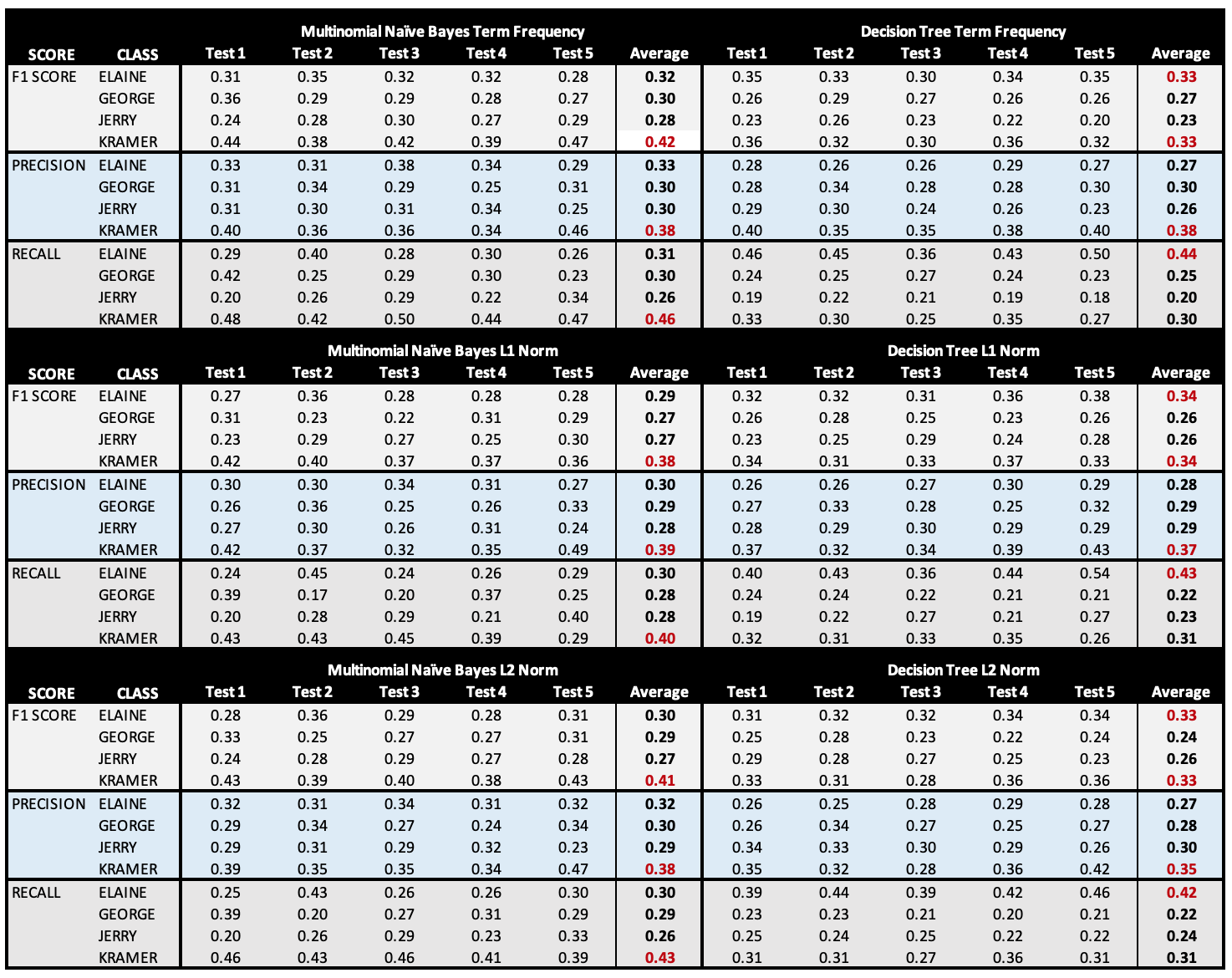


Figure 22. 5 Fold Cross Validation Summary of Precision, Recall & F1 Scores by Class & Model

### Confusion Matrix

With all 6 models achieving similar accuracy scores that were slightly above the chance probabilities of each train data set, the models were then trained on their respective full train data sets to create a final model. The final models were then applied to their respective test-validation data which represented 3,000 lines from the original data with classes closer in proportion to the original data as well. . Each of the models’ prediction results were plotted in a confusion matrix.

Viewing the confusion matrices for all the Multinomial Naïve Bayes models a common theme begins to form on where the models appear to have difficulty in classifying the dialogue lines to the correct character. The first observation is that Jerry’s lines are most likely to be misclassified as another’s character. The second observation is that Kramer’s lines are the least likely to be misclassified as George or Jerry. The third observation that can be seen from the confusion matrices is that a high proportion Jerry, George and Kramer’s lines that were misclassified were assigned to Elaine. See Figures 21 to 23.



Figure 23. Confusion Matrix Results on Multinomial Naïve Bayes Term Frequency Data

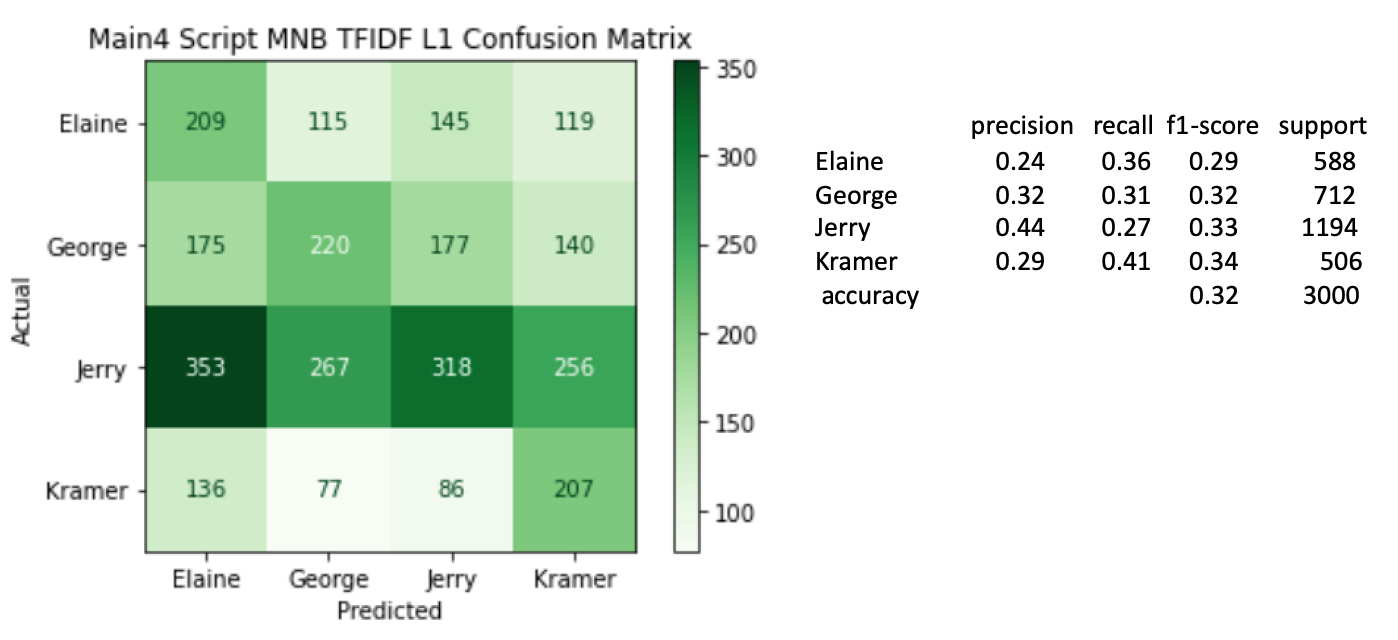


Figure 24. Confusion Matrix Results on Multinomial Naïve Bayes TFIDF L1 Norm Data

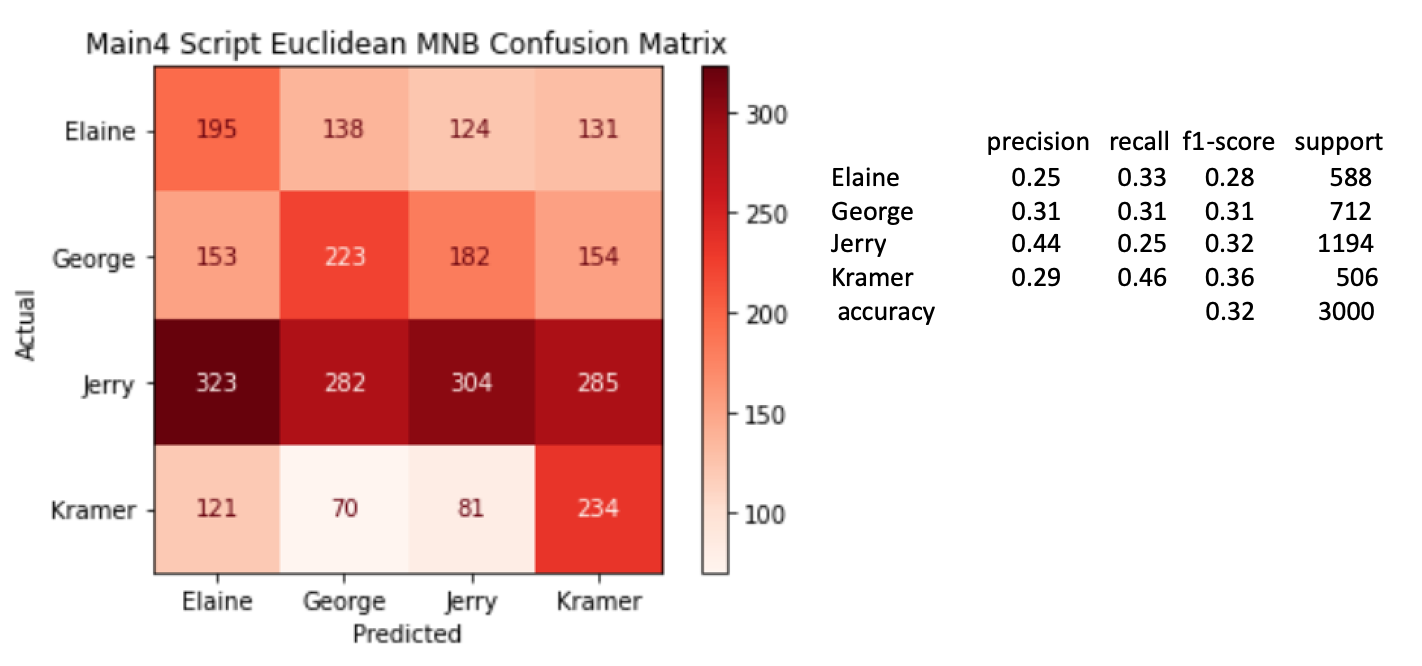


Figure 25. Confusion Matrix Results for Multinomial Naïve Bayes Euclidean (L2) Norm

The confusion matrices resulting from the Decision Tree models showed very similar results as the Multinomial Naïve Bayes models. The only difference worth noting between the two is that the Decision Tree models tended to have a more pronounced difficult distinguishing Jerry’s lines from Elaine. See Figures 24, 25 and 26

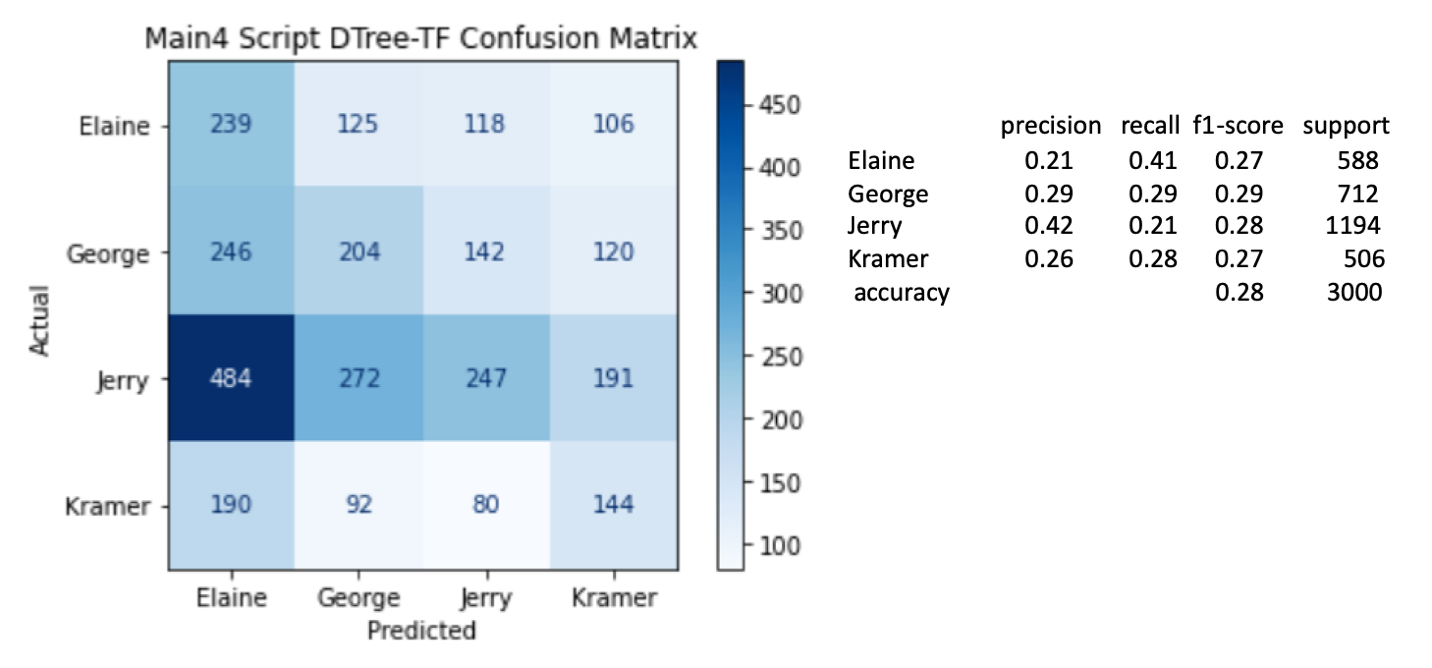


Figure 26. Confusion Matrix Results for Decision Tree Term Frequency



Figure 27. Confusion Matrix Results for Decision Tree L1 Norm Data

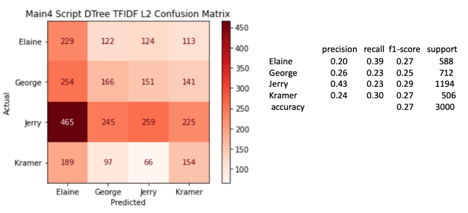


Figure 28. Confusion Matrix Results for Decision Tree Euclidean (L2) Norm

Overall, the accuracy scores for all 6 models were fairly poor at correctly predicting a dialogue line’s class. In fact all the F1 Scores for Jerry and George were below their respective chance probabilities. Despite the poor performance of the models, the confusion matrices did hint at what might be causing some difficulty. One, the majority of Jerry’s lines were being misclassified as belonging to the other characters suggesting Jerry may not have the most distinctive lines or voice in the show. Second a high proportion of the misclassified lines for Jerry, George and Kramer are assigned to Elaine. This seems to suggest that Elaine says many of the same things that the male characters say. This seems especially true with Jerry since Jerry’s lines are most likely to be misclassified as Elaine’s

### Indicative Features

Based on the findings seen from the confusion matrix results the 15 features with the highest probabilities by class were plotted to investigate how the characters’ lines were similar or different (Figure 27). Apparently one of the most spoken words in the show is ‘yeah’ which was identified as the highest probability feature for all four characters. Interestingly the top three features for Jerry and Elaine are identical (‘yeah’, ‘oh’ and ‘know’). Another observation that could make classifying difficult is that at least 50% of the top 15 features for each character are similar to another character. In other words, much of the most frequent words in each of the character’s vocabulary are common to the other characters. This lack of distinction between characters’ vocabulary could make classification difficult.

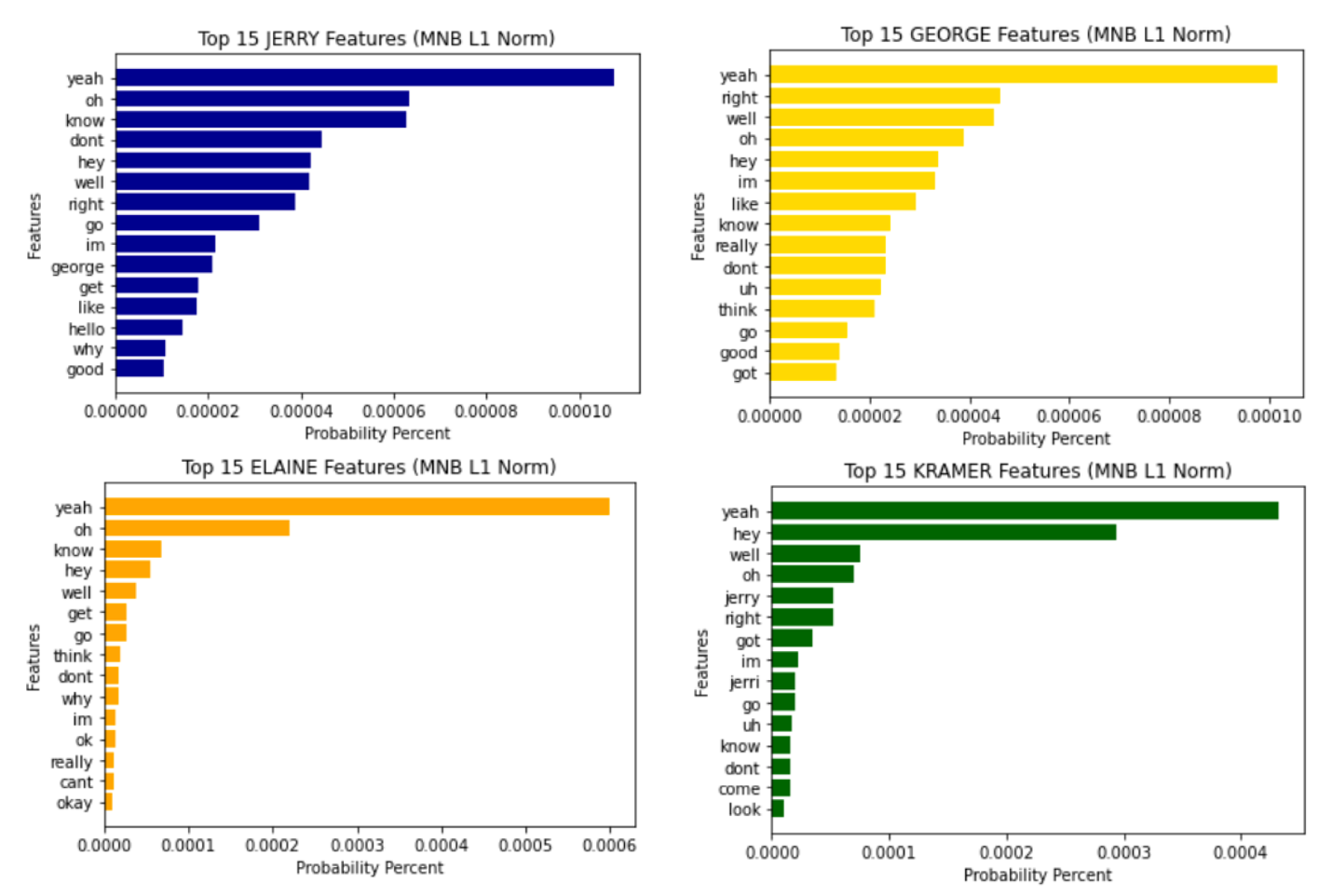


Figure 29. The 15 Highest Feature Probabilities by Class (Multinomial Naïve Bayes L1 Norm)

To further investigate the features into why the models failed to accurately classify the character’s lines a Decision Tree model was defined with the same parameters as described in models except the maximum leaf nodes were set to 20. Acknowledging that this will produce different results from the analysis discussed above, visualizing the top of the decision tree may still provide some guidance on how to improve the model. The model was trained on the TFIDF L1 normalized data. The portion of the resulting tree plot is shown in Figure 28.

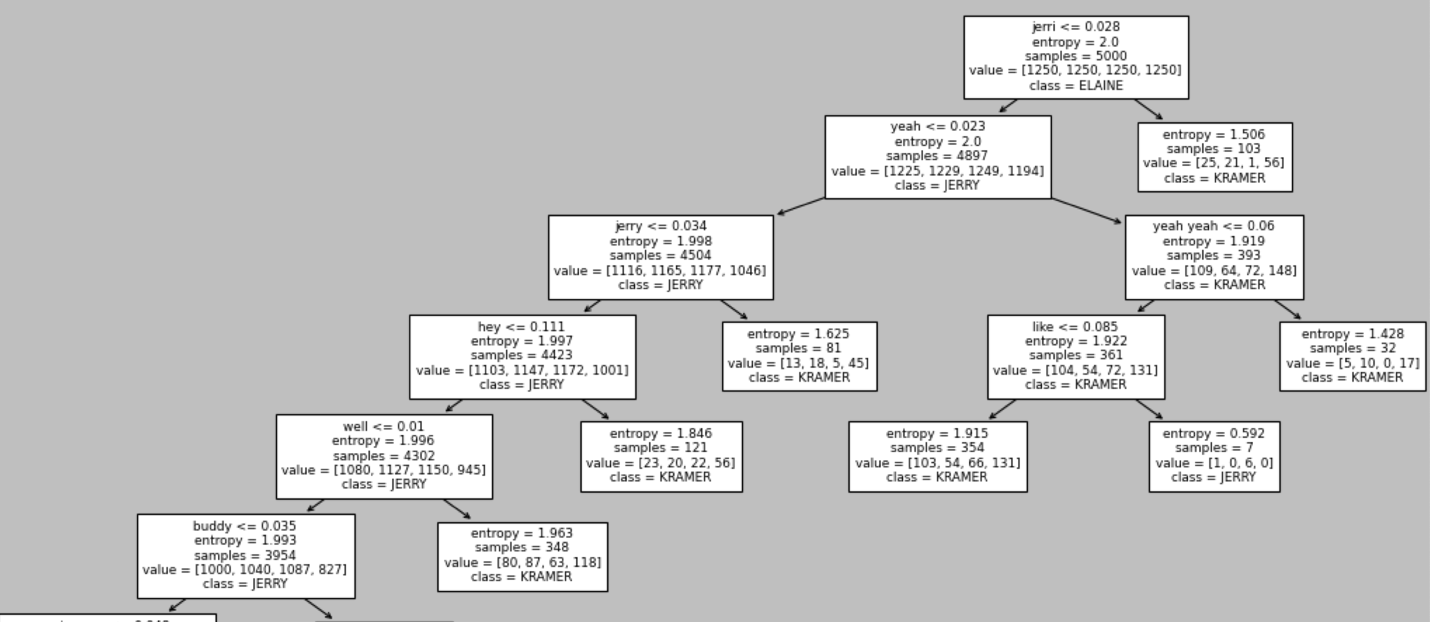
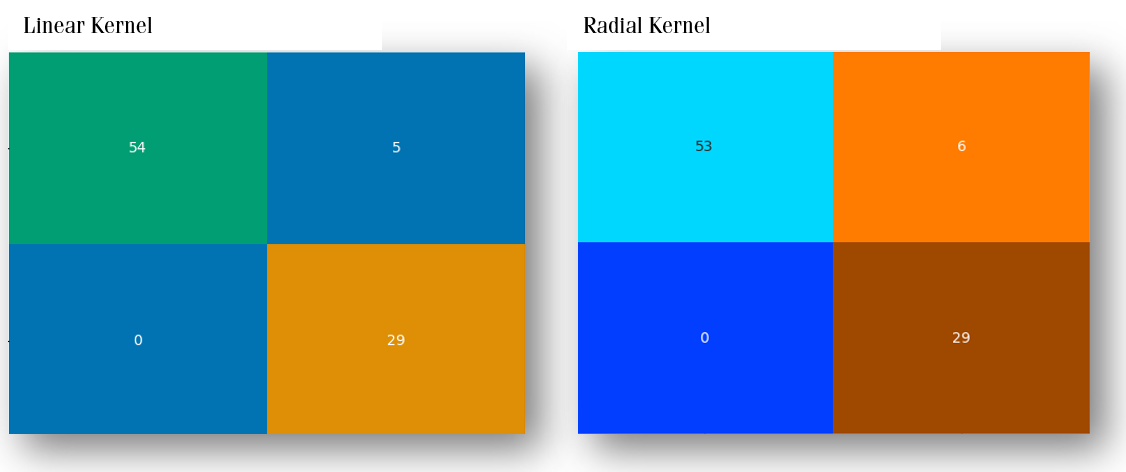


Figure 30. Top Portion of Decision Tree Visualized to Investigate the Leaf Node Splits

From this small portion of the tree plot, the common words seen in the Naïve Bayes probability features (‘yeah’, ‘hey’, ‘well’, ‘like’) can be found in the Decision Tree as well. This further validates how common these words are in the show’s vocabulary. On the other hand, the mention of character’s name in the dialogue can obviously help classify the character speaking it. For instance, the tree plot suggest that Kramer says Jerry’s name more often than Elaine or George. This was also indicated in the feature probabilities in the Naïve Bayes model.

## Humor Sentiment Analysis:

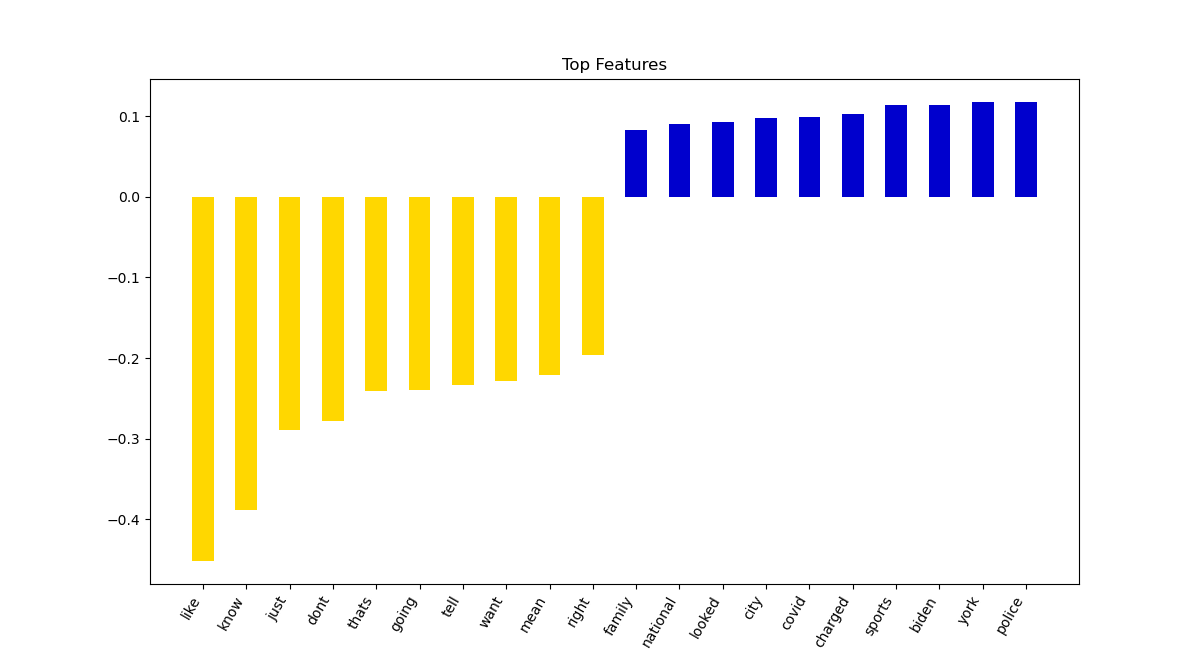
The selection of SVM models for humor sentiment analysis yielded successful results. To ensure the models were properly trained, the News API was executed on various days, with rotating sources, to introduce fresh data. Below are confusion matrices illustrating each of the final models’ prediction. After every iteration of the News API, no model with linear or radial kernels fell below 90% accuracy. The final implementation recorded a 94% accuracy for the linear kernel and a 93% accuracy for the radial kernel.

*Figure 31. Confusion Matrix Results for SVM using Linear and Radial Kernels*

In contrast, the SVM model with the polynomial kernel parameter did not perform as well. Each implementation with the polynomial kernel tested in the 60-70% accuracy range, with the final model registering an accuracy of 67%.

U,{d0a54a3d-63a2-4a94-9c0b-be735914d9c1}{10},3.125,3.125

*Figure 32. Confusion Matrix Results for SVM using Polynomial Kernel*

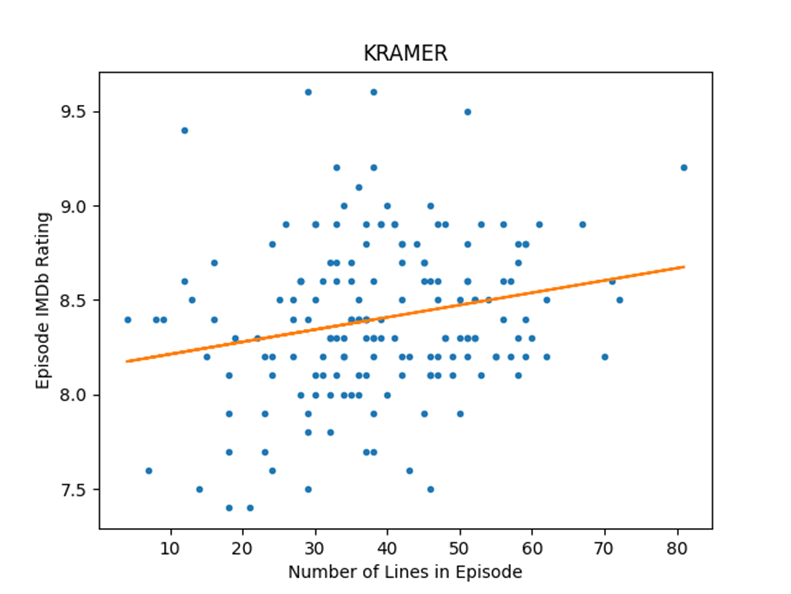


*Figure 33. Top 20 Features for Funny Sentiment Analysis Linear SVM model*

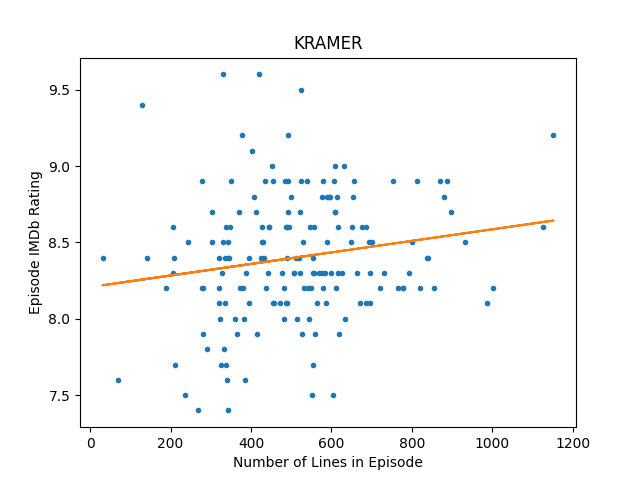
## Stand-Up Topic Modeling:

## Rating Correlation:

The correlations and their respective confidences were calculated and separately recorded 2 separate times, once using the count of lines, and once using the count of words, both per episode. Being that analysis was only done on the characters with the top 10 number of lines, the results can be visualized using a simple histogram and chart containing correlation coefficients and respective p-values. To visualize what the data for each character looks like, see the plots below. They contain wordcount per episode plotted against IMBD rating per episode, and contain an additional line representing the approximated correlation coefficient between the two values. There is a slight observable difference between word count and line count, but the results are parallel.



*Figure 34. Line Count per episode By Rating (Kramer)*



*Figure 35.* *Word Count per episode By Rating (Kramer)*

### Resulting Character Significance

For each character, and the IMBD rating of every episode that character has lines in, a Pearson correlation coefficient and respective p-value that explains the confidence of the correlation is calculated and appended into a dictionary. A resulting dictionary looks something like the table below (Figure 36).

|  | **Character** | **Coefficient** | **p Value** |
| --- | --- | --- | --- |
| **0** | SUSAN | 0.222707 | 0.245554 |
| **1** | KRAMER | 0.217817 | 0.004443 |
| **2** | ESTELLE | 0.196405 | 0.357666 |
| **3** | ELAINE | 0.051687 | 0.504522 |
| **4** | GEORGE | 0.028866 | 0.708647 |
| **5** | MORTY | -0.018068 | 0.938040 |
| **6** | NEWMAN | -0.084117 | 0.582753 |
| **7** | HELEN | -0.169279 | 0.463222 |
| **8** | FRANK | -0.185189 | 0.375489 |
| **9** | JERRY | -0.185490 | 0.015144 |

Figure 36. Correlation Coefficient and Corresponding P-value (Lines)

Below is a visualization of the above dictionary, characters denoted with a \* have a p-value greater than .1, meaning there is a significant chance that the correlation is not valid.

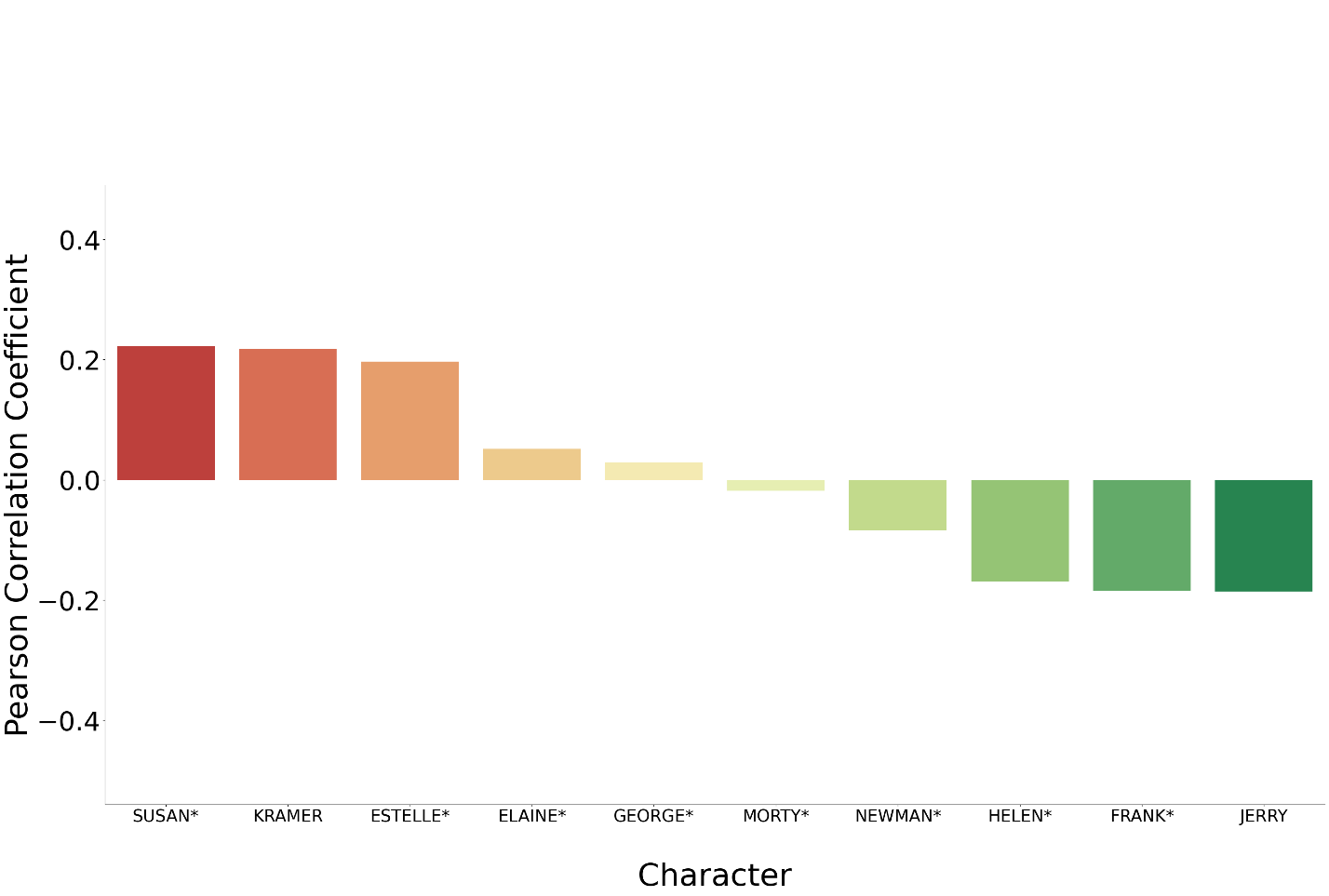
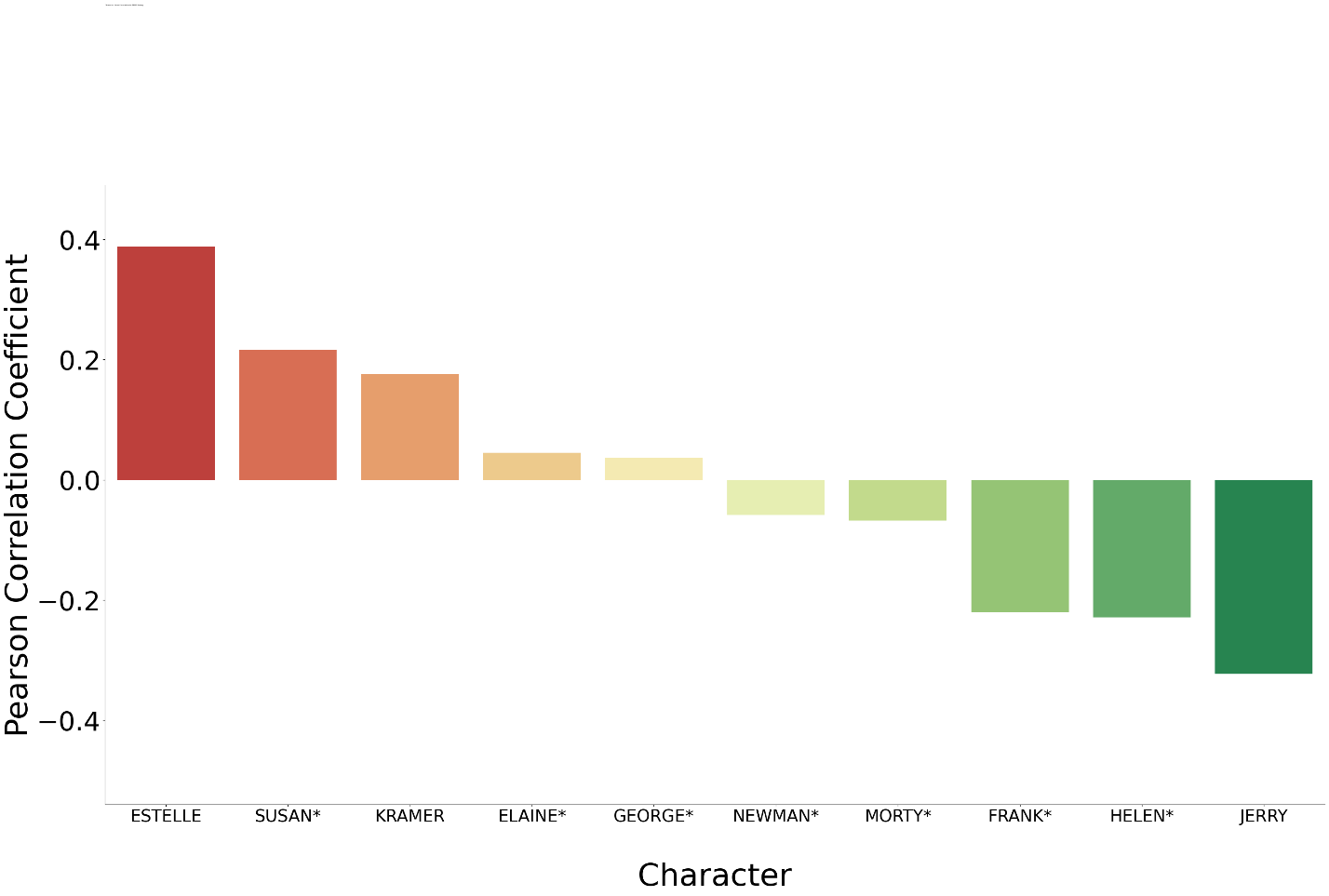


Figure 37. Correlation Coefficient Lines (Episode Ratings: Character Lines per Episode)

A repetition of this process but replacing line count per episode with total word count per episode produces somewhat similar results. Although this second analysis could be said to represent more accurately how much a character talks in an episode. A similar summary of correlation is below.

|  | **Character** | **Coefficient** | **p Value** |
| --- | --- | --- | --- |
| **0** | ESTELLE | 0.388003 | 0.060998 |
| **1** | SUSAN | 0.216408 | 0.259507 |
| **2** | KRAMER | 0.175679 | 0.022332 |
| **3** | ELAINE | 0.044891 | 0.562223 |
| **4** | GEORGE | 0.036734 | 0.634370 |
| **5** | NEWMAN | -0.058758 | 0.701422 |
| **6** | MORTY | -0.067715 | 0.770559 |
| **7** | FRANK | -0.220391 | 0.289769 |
| **8** | HELEN | -0.229010 | 0.318014 |
| **9** | JERRY | -0.323181 | 0.000016 |

Figure 39. Correlation Coefficient and Corresponding P-value (Words)

****Figure 40. Correlation Coefficient Words (Episode Ratings: Character Words per Episode)

There are a few correlations that are apparent in the initial calculation and reenforced by the subsequent calculation. Those correlations being the positive association between Kramer’s amount of dialogue and rating, and the negative association between Jerry’s amount of dialogue rating. The p-value for both these correlations are low in both calculations, suggesting that there is at least a reasonable chance that the correlation is valid. In fact, the p-value for Jerry’s correlation is shockingly low for the word count calculation, almost unbelievably low. It is possible that results for Jerry reflect the shows growth, rather than him having a negative impact on ratings. As looking back at figure 6 & 7, its clear that the spotlight on Jerry’s character fades as the show grows and brings on a larger ensemble of characters. No other character showed an appropriate p-value to be significant except for Estelle in the word count calculation, where she also had the highest positive correlation, perhaps this suggests she is a sort of niche favorite character, or this is a result of her more minor appearances compared to other characters. Ultimately the only assumption that can be confidently made from these calculations is that an episode with a focus on Kramer has positive effects on the ratings, or the opposite for Jerry.

## Result Summary:

The attempt to create a model capable of classifying lines by main character failed to produce accurate predictions that would be considered reliable. Upon investigating the model’s challenges, it was discovered that much of the vocabulary in the show is shared frequently among the four main characters. A possible improvement that may produce better models would be to exclude vocabulary words that every character says often. For example, the word ‘yeah’ is the most frequently used word for all the characters. With such prevalence in the script, the word ‘yeah’ may not add any relevance in determining a class. Its ubiquitous appearance could be seen as simply another stop word. By excluding the word ‘yeah’ and other words frequently used by all the characters, the remaining features in the data may find more distinctive words to associate with each character.

# Conclusion