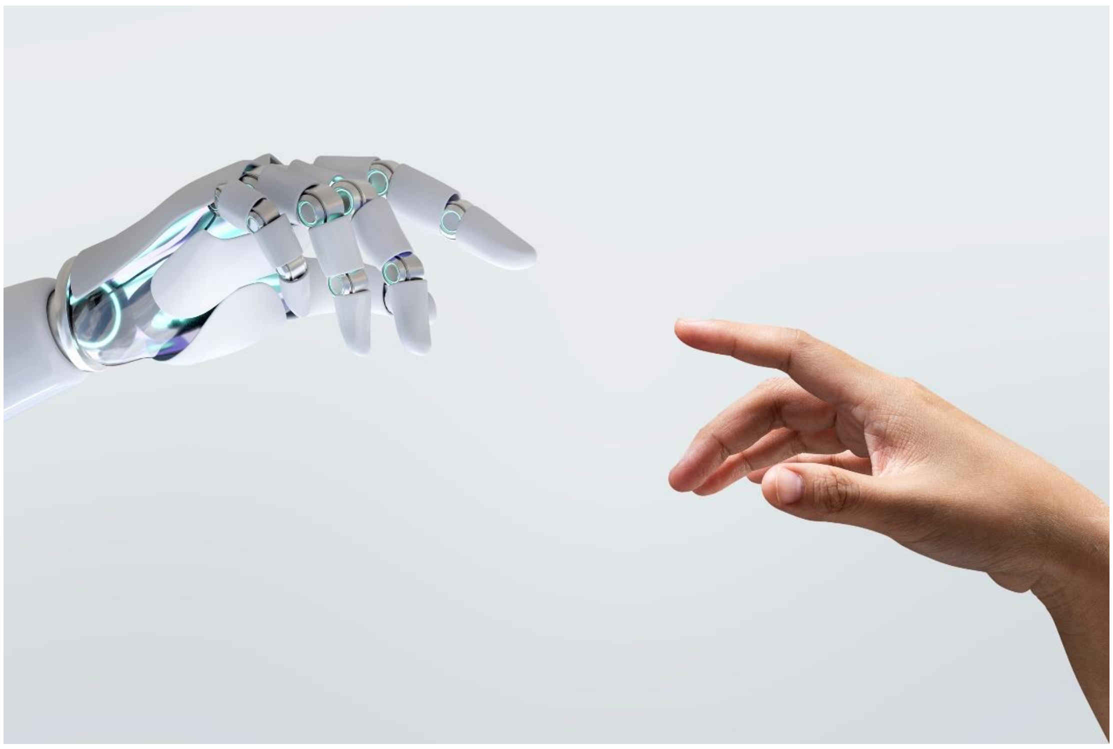
Analysis of 20 Newsgroups Dataset Using Various Classification Models



CETM 47 Machine Learning and Data Analysis

Assignment 1

Supervised by: Ashley Williamson

University of Sunderland

Student Name: Mahmoud Al-Mbaydeen

Student ID: 219217301

1. Introduction:

Text Classification is a popular natural language processing job in a variety of commercial situations. The goal of this project is to categorise all of the text documents. Data from "The 20 Newsgroups data" is what we'll be working with. About 20,000 newsgroup documents are sorted into 20 newsgroups in the 20 Newsgroup data collection. The scikit-learn API is used to obtain the data needed for this task.

2. EXPLORATORY DATA ANALYSIS (EDA)

I made the decision to separate the EDA process into two distinct groups: fundamental pre-processing procedures that were utilised by all vectorizers and models, and specific pre-processing procedures that were incorporated into the process as alternate options for evaluating the effectiveness of the model either with or without them. The level of a model's accuracy is directly correlated to how well it performs on test data, which is why it was decided to use accuracy as a criterion for comparing models.

The following typical pre-processing procedures were used:

1. Changing the uppercase to lowercase

2. Elimination of commas and full stops

3. Elimination of all alphabetic symbols

4. Remove all commas and dashes

5. Vectorization: The TfIdfVectorizer was employed. It was found that the models using Count Vectorizer were more accurate. TfIDFVectorizer was chosen as the default Vectorizer since it provided better results in all circumstances.

The following steps were added as optional to the pre-processing phases to examine how model performance altered with and without them:

1. Stemming

2. Lemmatization

3. Using Unigrams/Bigrams

Analysis of output labels in the training data was done to evaluate if any of the classes had an unfair advantage. Figure 1 depicts this pattern per grade level. There appears to be a nearly equal distribution of output variables, as can be seen in the graphic. Since no resampling was required, the classification models will not be biassed towards any one class. Additionally, the distribution of scores by class is depicted in Figure 2.

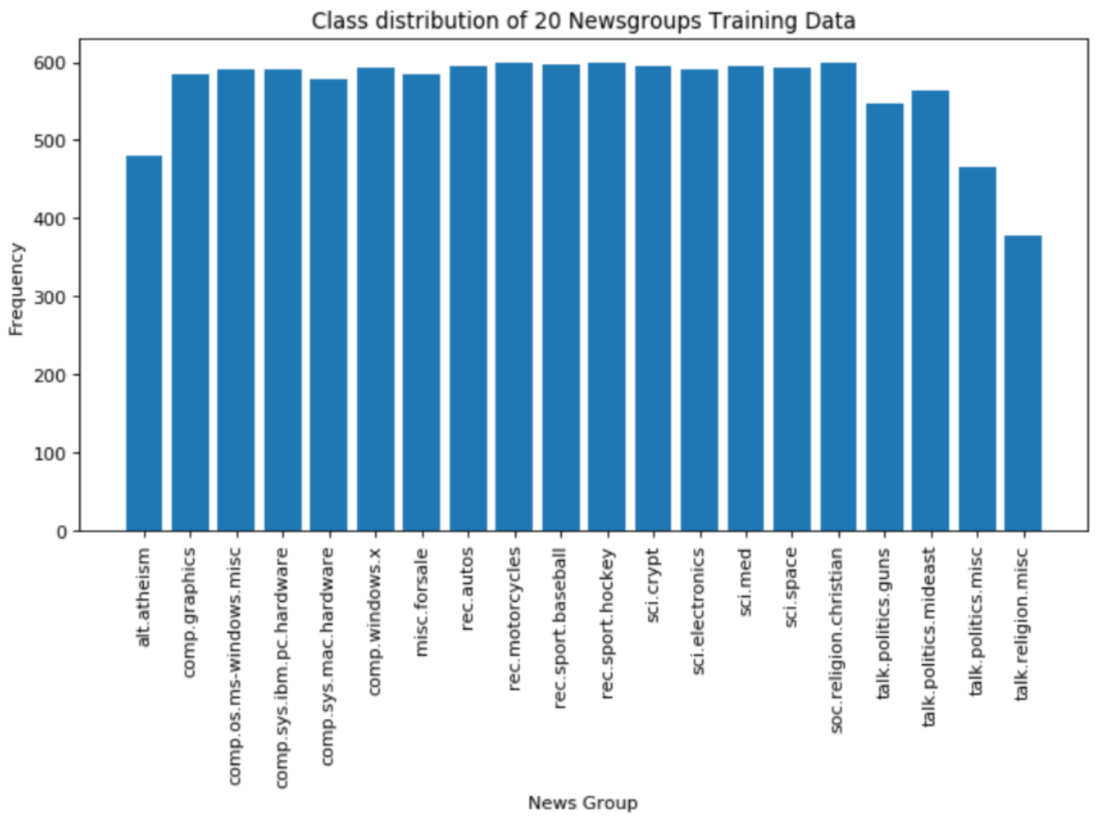


Figure (1): Classification of 20 Newsgroups in the Testing Dataset.

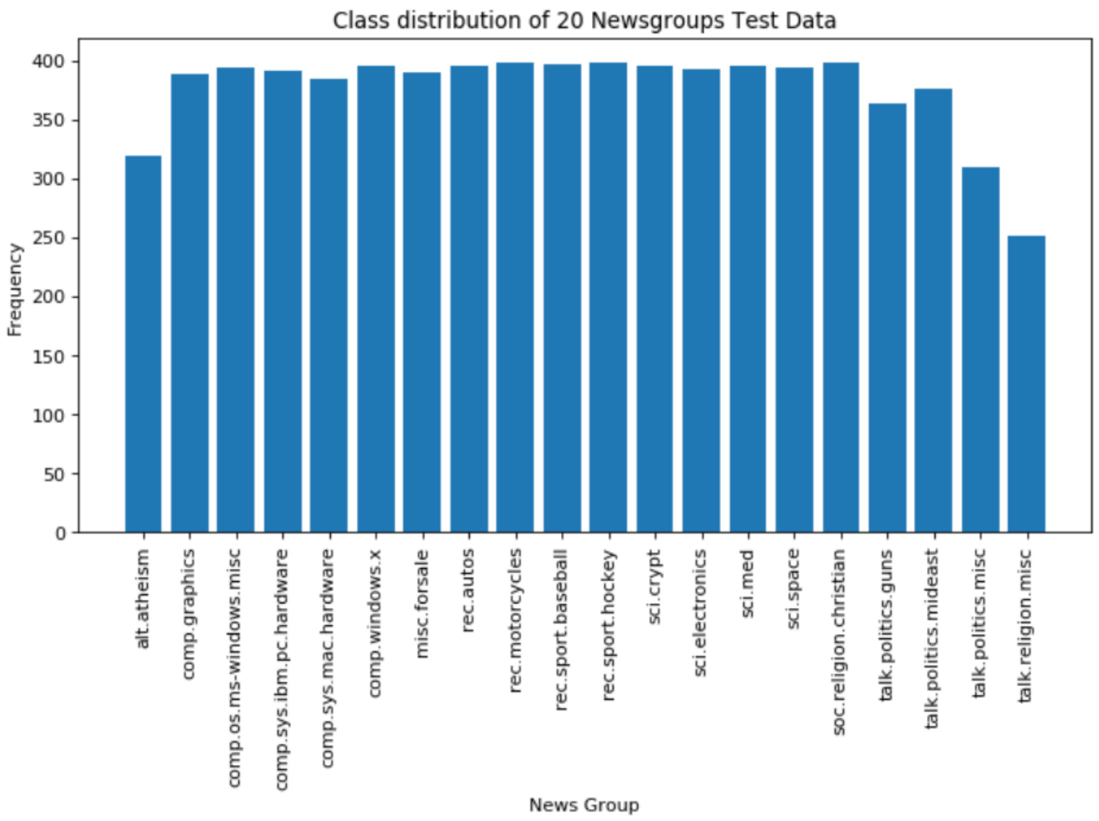
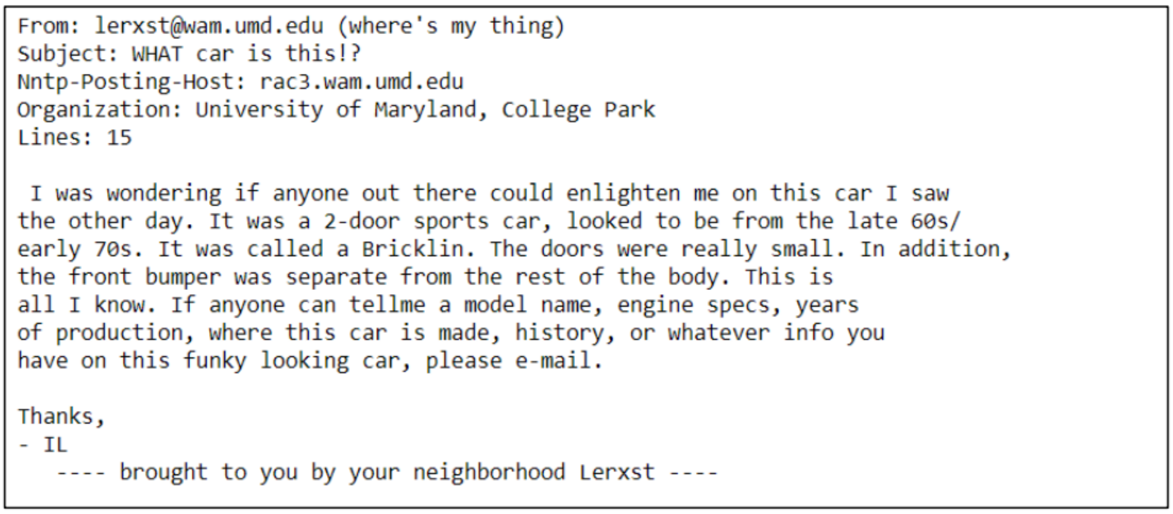


Figure (2): Classification of 20 Newsgroups in the Testing Dataset.

Furthermore, accessible data (text) comprised a header, footer, and quotations, and looked somewhat like:

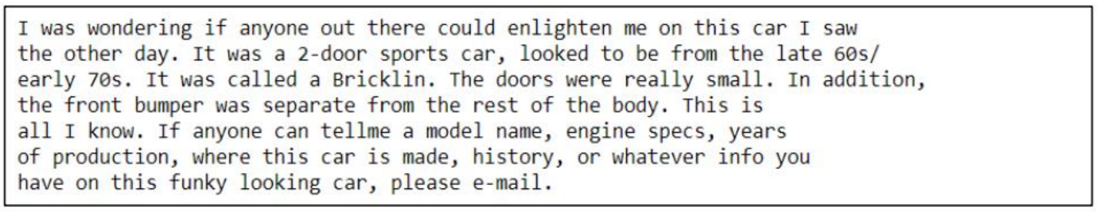


```
From: lerxst@wam.umd.edu (where's my thing)
Subject: WHAT car is this!?
Nntp-Posting-Host: rac3.wam.umd.edu
Organization: University of Maryland, College Park
Lines: 15

 I was wondering if anyone out there could enlighten me on this car I saw
the other day. It was a 2-door sports car, looked to be from the late 60s/
early 70s. It was called a Bricklin. The doors were really small. In addition,
the front bumper was separate from the rest of the body. This is
all I know. If anyone can tellme a model name, engine specs, years
of production, where this car is made, history, or whatever info you
have on this funky looking car, please e-mail.

Thanks,
- IL
   ---- brought to you by your neighborhood Lerxst ----
```

I extracted the main body of text by deleting the header, footer, and quotations, and the result was as follows:



```
I was wondering if anyone out there could enlighten me on this car I saw
the other day. It was a 2-door sports car, looked to be from the late 60s/
early 70s. It was called a Bricklin. The doors were really small. In addition,
the front bumper was separate from the rest of the body. This is
all I know. If anyone can tellme a model name, engine specs, years
of production, where this car is made, history, or whatever info you
have on this funky looking car, please e-mail.
```

I was receiving better accuracies [10% more] with the first version, but I assume this was due to the bias placed into the data. Most writers write on a certain set of themes, and our algorithms used that knowledge to categorize as well. This meant that the classifier assigned less weight to what the text really said. When we observe test data that lacks such information [author, server, organization], this may be a serious problem. So, in order to train a decent general model that simply considers the words spoken, we extracted them.

3. METHODOLOGY:

A number of TF-IDF Vectorizer properties are scrutinised in this research in order to identify the optimal setup for distinct classification models. I begin the modelling process in the vectorizer with no stemming or lemmatization in mind. After that, stemming and lemmatization will be used in the vectorizer. Using only unigrams as the TF-IDF Vectorizer's producing features in each of the three settings, I compare the results to those obtained when both unigrams and bigrams are included. This is followed by an examination of the performance of four different classification models on the test data. A sampling of these methods includes Naive Bayes, Logistic Regression, Stochastic Gradient Descent Classifier, and KNN. The sklearn package is used to perform the analysis in Python.

3.1. Not performing Stemming and Lemmatization in the TF-IDF Vectorizer:

It's possible to use TF-IDF Vectorizer to extract the "bag of words" from the text and apply TF-IDF weights. Sublinear tf scaling (1 + log(tf)) enhanced overall results, as I learned through try and error. Based on 11000 documents and 68,000 words of training data (without stemming or lemmatization), the TF-IDF Vectorizer developed a Document Term Matrix.

To begin, I extract just unigrams from the text input and use those to build the vectorizer. Then, I test four classifiers: Naive Bayes, Logistic Regression, Stochastic Gradient Descent Classifier, and k Nearest Neighbors. For Logistic Regression and k Nearest Neighbors, grid search was used.

The L1 and L2 norm values for Logistic Regression were applied to the parameter 'penalty.' In the penalization, Logistic Regression performs best when the "L2" norm is used. When using K Nearest Neighbors, the parameters 'n neighbours' and 'weights' were set to 5, 10, 100, and 200, respectively. Setting 'n neighbours' to 5 and 'weights' to 'distance' produced the best results for k Nearest Neighbors.

After that, we develop the TF-IDF Vectorizer by extracting unigrams and bigrams from the text input. Our evaluation of Logistic Regression and k Neighbors uses the grid search approach once more. The results of the tests are shown in Table 1.

ngrams	**Classifier Accuracy (%)**			
	MultinomialNB	**LogisticRegression**	**SGDClassifier**	**KNeighborsClassifier**
Unigram	66.93	68.96	69.67	8.83
Unigram+Bigram	65.57	68.34	70.39	7.73

Table (1): Classifiers Accuracy on Test Data

3.2. Applying only Stemming in the TF-IDF Vectorizer:

Since then, I've switched to the stemming method, in which I break down long words into their simplest forms. The Snowball English Stemmer algorithm from the NLTK package is what I'm using in Python, and I've constructed a class to represent it. First, I test the four classifiers with only unigrams and then with both unigrams and bigrams in the vectorizer.

Grid searches on Logistic Regression produced the best results when the 'L2' norm was used in the penalization. The best results were obtained with n neighbours set to 5 and weights set to distance while using k Nearest Neighbors. The results of the tests are shown in Table 2.

ngrams	**Classifier Accuracy (%)**			
	MultinomialNB	**LogisticRegression**	**SGDClassifier**	**KNeighborsClassifier**
Unigram	66.49	69	70.18	8.42
Unigram+Bigram	65.59	68.34	70.66	8.5

Table (2): Classifiers' Performance on Test Data

3.3. Applying only Lemmatization in the TF-IDF Vectorizer:

Our final step is to use lemmatization to get the term in a semantically acceptable normal form. Word Net Lemmatizer and part-of-speech word tagging from the NLTK package are used, as is the creation of a class to represent the Lemma Tokenizer. The four classifiers are tested again in the vectorizer, this time with both unigrams and bigrams.

Grid searches on Logistic Regression produced the best results when the 'L2' norm was used in the penalization. The best results were obtained with n neighbours set to 5 and weights set to distance while using k Nearest Neighbors. The results of the tests are shown in Table 3.

ngrams	**Classifier Accuracy (%)**			
	MultinomialNB	**LogisticRegression**	**SGDClassifier**	**KNeighborsClassifier**
Unigram	66.87	69.01	69.92	8.71
Unigram+Bigram	65.72	68.36	70.58	8.03

Table (3): Performance of Classifiers on Test Data

3.4. Results and Discussions:

Now I compare data from the preceding sub-sections. Figures 3–6 demonstrate the performance of each classifier. Figures 3 and 6 show that the classifiers Multinominal Nave Bayes and k Nearest Neighbors perform best in the TF-IDF Vectorizer when no stemming or lemmatization is performed and only unigrams are recovered. Figure 4 shows that Logistic Regression performs best when Lemmatization is employed with just unigrams collected using the TF-IDF Vectorizer. Finally, Figure 5 shows that when Stemming is employed with both unigrams and bigrams retrieved by the TF-IDF Vectorizer, the Stochastic Gradient Descent Classifier performs the best.

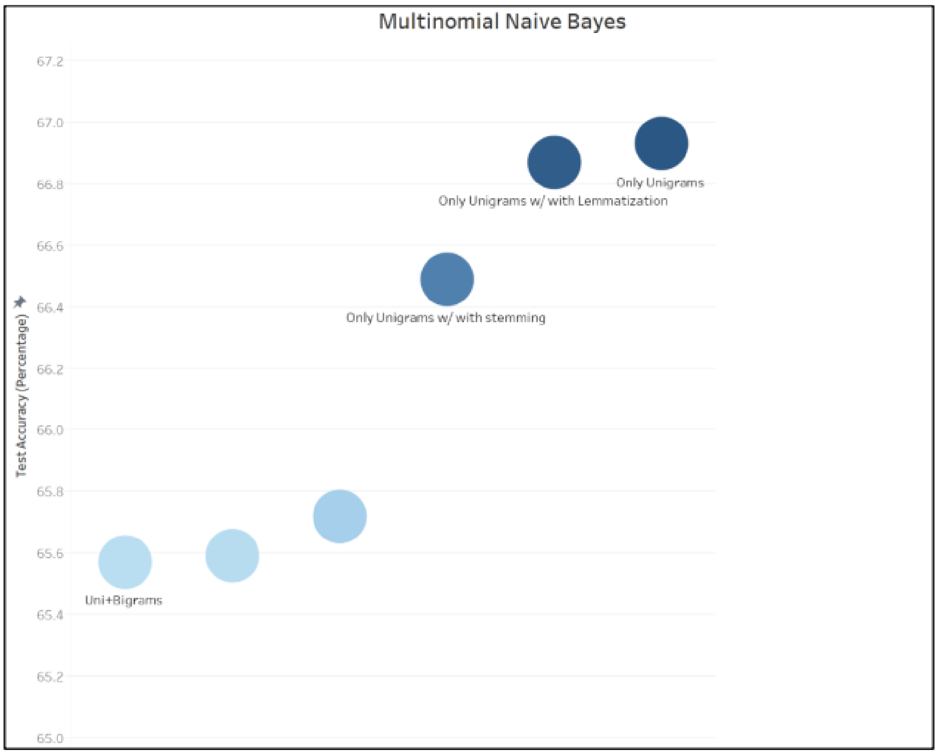


Figure (3): Multinominal Naive Bayes Performance

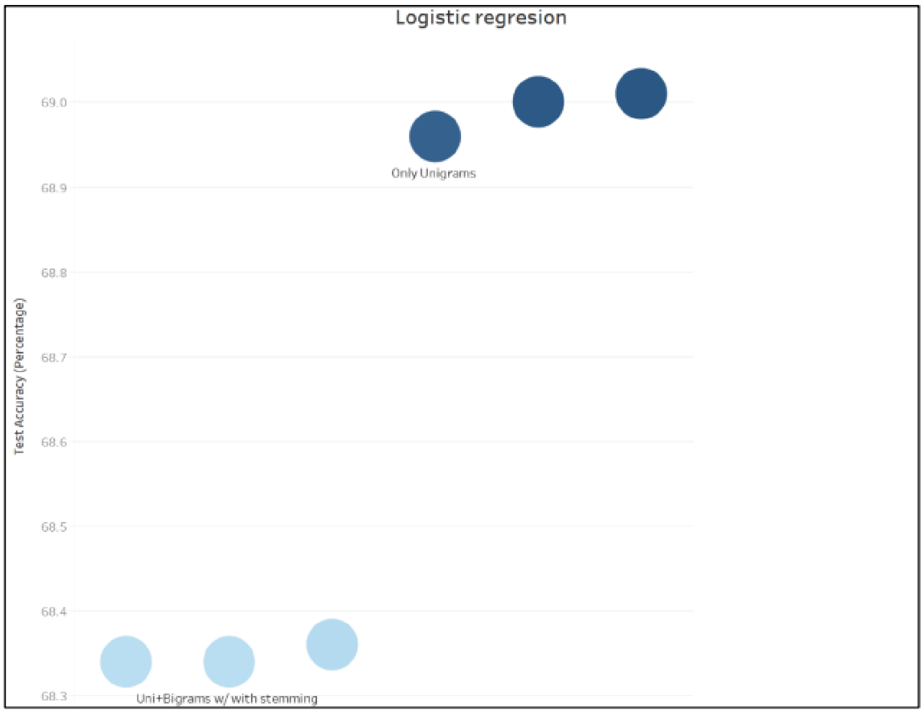


Figure (4): Logistic Regression Performance

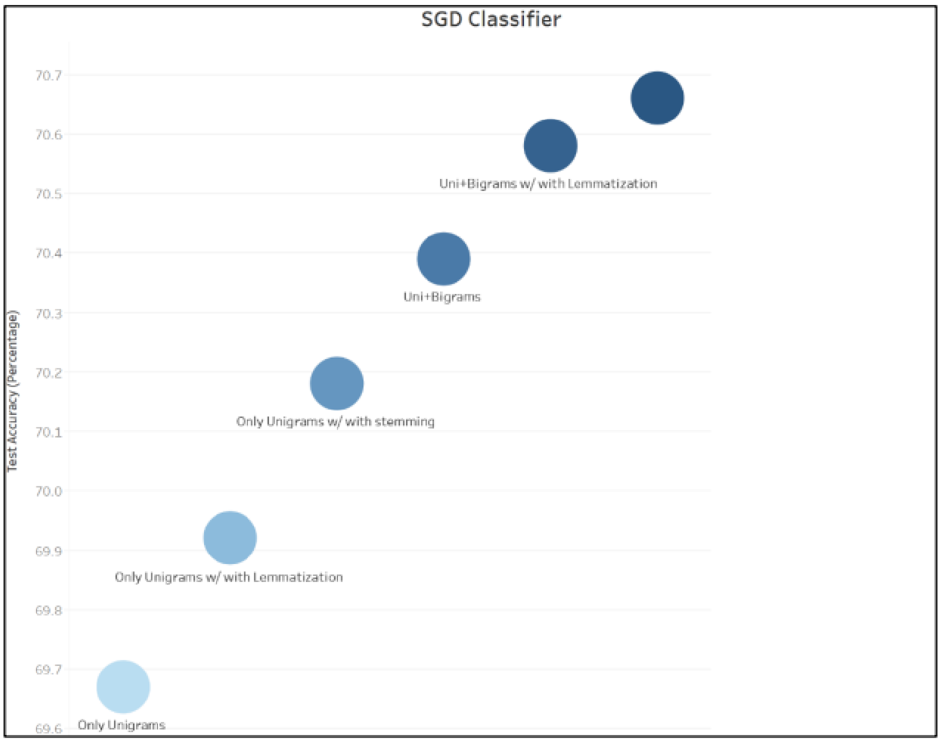


Figure (5): SGD Classifier Performance

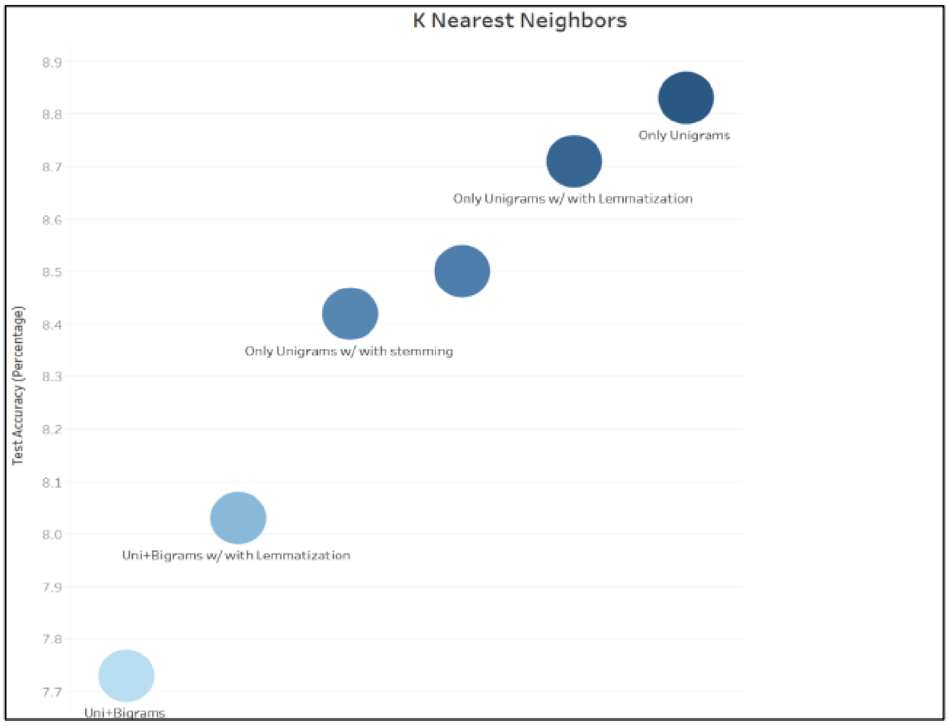
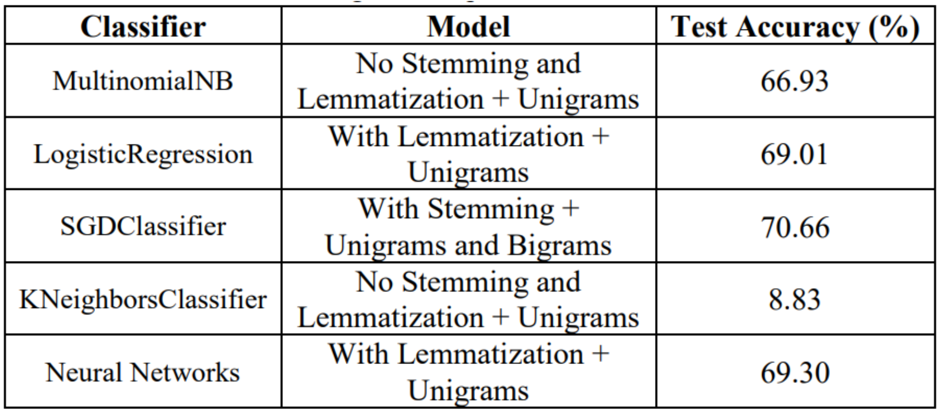


Figure (6): k Nearest Neighbors Performance

Table 4 displays and visualizes the best performance attained for each classifier. The Stochastic Gradient Descent Classifier achieves the greatest accuracy of 70.66 percent, followed by Logistic Regression and Multinominal Nave Bayes. We can observe that k Nearest Neighbors performs horribly. This is because k Nearest

Neighbors is known to suffer from the curse of dimensionality when a high number of characteristics are present. When the dimension of the data expands sufficiently, its distance measure becomes meaningless. The chart also illustrates the performance of neural networks (second best), which is discussed in further depth in Section 3.5. Finally, in Figure 8, we provide the performance parameters such as accuracy, recall, and f1-score for the best performing Stochastic Gradient Descent Classifier for each of the 20 classes.



Classifier	Model	Test Accuracy (%)
MultinomialNB	No Stemming and Lemmatization + Unigrams	66.93
LogisticRegression	With Lemmatization + Unigrams	69.01
SGDClassifier	With Stemming + Unigrams and Bigrams	70.66
KNeighborsClassifier	No Stemming and Lemmatization + Unigrams	8.83
Neural Networks	With Lemmatization + Unigrams	69.30

Table (4): Summarizing the Best Performance from Each Classifier

4. Conclusion:

This report provides a method for text cleaning 20 Newsgroup datasets in order to reorganize the material in an organized manner. The first step is to delete any useless data from the dataset. The next step is to remove the stop word list. The stemmer method is used in the third step to transform the words to the root. Finally, it is necessary to remove the empty documents from the dataset. We conducted studies, and the findings suggest that text cleaning approaches greatly decreased the size of the dataset. Furthermore, the 20 Newsgroup dataset versions were examined using a TF-IDF Vectorizer model, and I used Naive Bayes, Logistic Regression, SGD Classifier, and k Nearest Neighbors to choose which performance to utilize for analyzing the dataset.