# Classification of 20 Newsgroups Dataset using machine learning classifiers

**Your Name:**

**Your ID:**

Table of Contents

[Classification of 20 Newsgroups Dataset using machine learning classifiers 1](#_Toc104941731)

[1 Introduction: 3](#_Toc104941732)

[2. The Exploratory Data Analysis: 3](#_Toc104941733)

[3. Methodology: 5](#_Toc104941734)

[3.1 TF-IDF vector without Lemmatization/Stemming: 6](#_Toc104941735)

[3.2 TF-IDF vector with Stemming: 7](#_Toc104941736)

[3.3 TF-IDF vector with Lemmatization: 7](#_Toc104941737)

[4. Results: 8](#_Toc104941738)

[5. Evaluation & Discussion: 11](#_Toc104941739)

[6. Future Work: 11](#_Toc104941740)

[7. Conclusion: 12](#_Toc104941741)

[Figure 1 Outline of the project 4](#_Toc104941742)

[Figure 2 Distribution of class in training set 5](#_Toc104941743)

[Figure 3 Distribution of class in testing set 6](#_Toc104941744)

[Figure 4 Naive Bayes Performance 9](#_Toc104941745)

[Figure 5 Logistic Regression Performance 9](#_Toc104941746)

[Figure 6 SGD Performance 10](#_Toc104941747)

[Figure 7 KNN Performance 10](#_Toc104941748)

[Figure 8 Model comparison 11](#_Toc104941749)

[Figure 9 SGD Performance Metrics 11](#_Toc104941750)

[Table 1 Performance using the testing data 7](#_Toc104941753)

[Table 2 Accuracies on testing data with stemming 8](#_Toc104941754)

[Table 3 Accuracies on the models after Lemmatization 8](#_Toc104941755)

[Table 4 Summary of the performance 10](#_Toc104941756)

# 1 Introduction:

The text classification is one of the most widely used parts of the Natural Language Processing often called the NLP. Text classification is the process where we categorize the text into different organized groups, In in-text classification, we extract generic tags from the unstructured data. The goal of our project is to perform the text classification and define the text into defined categories. In this project, we are utilizing the dataset which is called “the 20 newsgroup data”, this dataset has the collection of almost 20,000 newsgroup documents. The dataset is divided into 20 different newsgroup classes. In this project, we will use different libraries like Pandas, NumPy, Sci-kit learns, etc.

The whole process of the project is shown below:



Figure 1 Outline of the project

We will start by importing the data into the Python environment and perform the required advanced data preprocessing, tokenization, cleansing of the data, and perform the stemming, lemmatization, then feature extraction and machine, learning models.

# 2. The Exploratory Data Analysis:

The EDA often called Exploratory Data Analysis is the process to analyze the dataset and extract useful insights from the dataset. We have the data of 20 newsgroups articles which are categorized into 20 different classes, in the EDA part, we have processed the data into two parts, some of the preprocessing techniques are common across all the models and vectorizers and some of them are kept optional for later use. The purpose of doing so is to check the models and vectorizers with and without those optional preprocessing steps. In the end, we have compared the results of all the models and vectorizers, the comparison is based on the accuracies of the models that the higher the accuracy, the best and more accurate the model on testing data.

In the early data preprocessing, we have dealt with some basic steps, these steps are very important as without them, we can’t train models. All the steps we took in the early preprocessing phase are given below:

* We converted the texts into Lower-case
* We removed the stop words
* There are alphanumeric keywords that need to be removed
* Some punctuations are removed
* We performed the Vectorization, we used the most commonly used vectorizer called “TfIdfVectorizer”, we trained machine learning models using this vectorizer and compared the accuracies with all the models without TfIdfVectorizer. After the comparison, we got to know that all the models with TfIdfVectorizer performed very well on the testing data and hence chose the best vectorizer for this project.

After performing the above preprocessing steps, we kept some steps optional, the purpose of doing so is to check how the performance of the machine learning models changes with and without these steps. These steps are given below:

* Performing the Stemming
* Performing the Lemmatization
* The use of Bigrams and Unigrams

We examined the spread of result marks in the preparation information to check whether it was slanted towards any of the classes. Figure 1 underneath shows this class circulation. As found in the figure, the spread of factors is sensibly even in this way showing a close rise to the conveyance of each result variable, consequently, the grouping models will not be one-sided towards a specific class, and accordingly, no resampling was required. Additionally, from Figure 2 we see the class dissemination in the test.

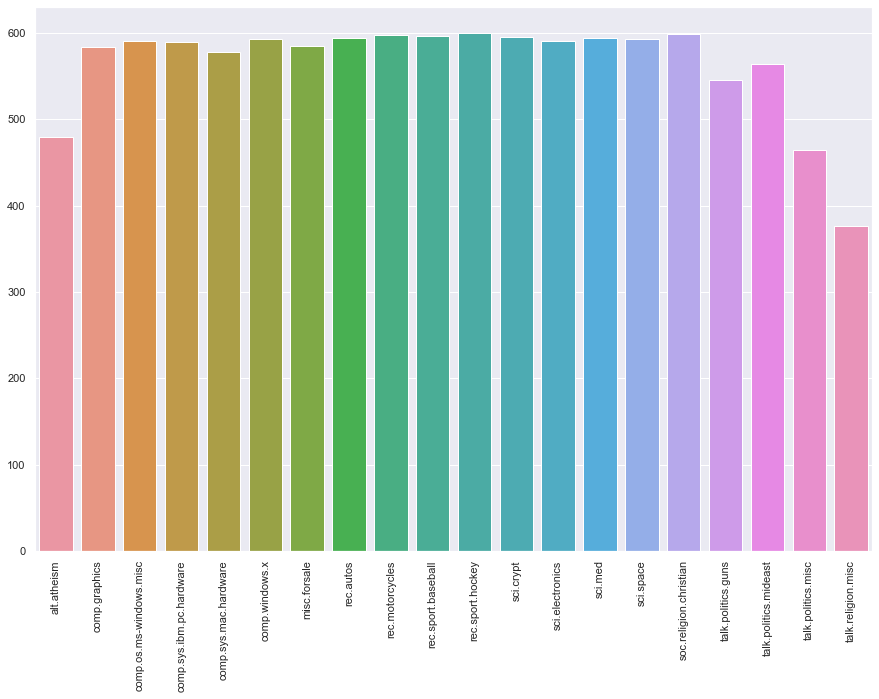


Figure 2 Distribution of class in training set

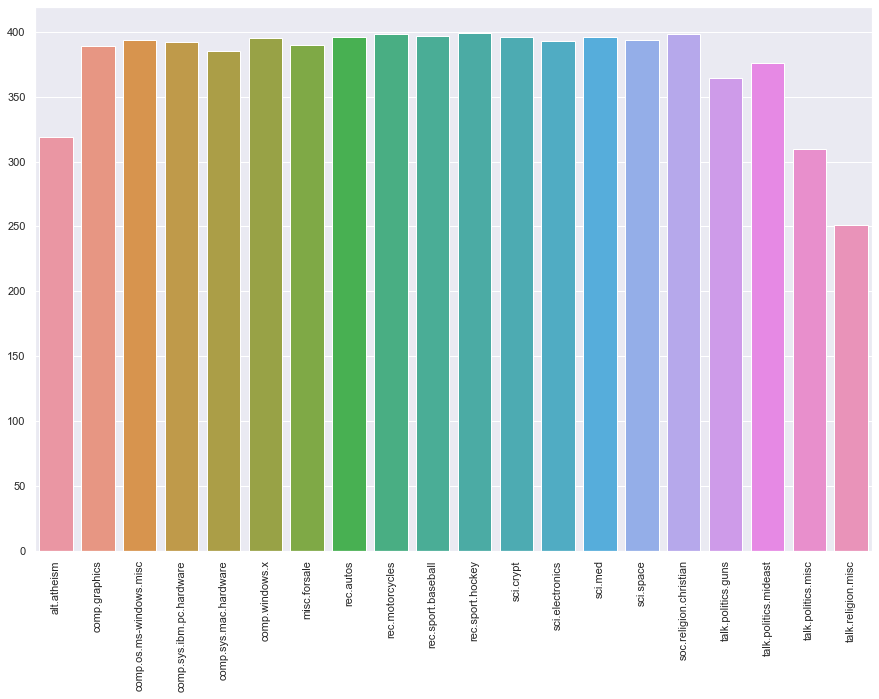


Figure 3 Distribution of class in the testing set

Upon going into more details, we found out that there is a huge number of unwanted and garbage texts in the data, these include Footers, Quotes, Headers, etc. To tackle this problem, we tried to extract the main body of the texts by removing those headers, footers, etc.

# 3. Methodology:

The goal of this project is to perform advanced text classification under natural language processing, for this purpose, we have used different TF-IDF vectorizers, the purpose of using different vectorizers is to see how different machine learning models perform best. In phase one, we started the model procedure by not taking lemmatization and stemming in the vectorization process, in the next step, we will apply the stemming only in the vectorizers and after that, in the end, we will use the lemmatization in our vectorizer. We used the unigrams as the resulting features and then we compared them with both unigrams and bigrams in the features set. After this combination, we proposed four different machine learning classification models to see how these classification models perform on the testing data. The four different machine learning classification models are given below:

* Naïve Bayes
* Logistic Regression
* Stochastic Gradient
* Descent Classifier
* K-Nearest Neighbors

All the analysis, data cleaning, and data preprocessing is done using Python and the modules used include Pandas, NumPy, Sci-kit learn, etc.

## 3.1 TF-IDF vector without Lemmatization/Stemming:

We will first use the TF-IDF vector by not considering the Lemmatization and stemming. The TF-IDF vectorizer is mainly used to extract the bag of words from the texts, we will apply this vectorizer to deal with the bag of words. After the analysis, we got to know that the vectorizer with scaling as *tf (1+log(tf))* got very good outcomes. After applying this vectorizer to our data, we generated a new document of all the text having 11,000 documents and 68,000 words without the lemmatization and stemming.

Before going to train and test our classifiers, we extracted the unigrams from the data only, after extracting the unigrams, we then trained our proposed four classification models that are shown above. We have also performed the parameters tuning by using the GridSearchCV method for only two models (Logistic Regression and K-Nearest Neighbors). The parameters we used for both models are given below:

* Logistic Regression (we used the Penalty as 'l1', 'l2')
* K-Nearest Neighbor (with values {5, 10, 100, 200} and weights are uniform and distance)

After extracting the unigrams and training our classification models, the next goal is to extract the unigrams and bigrams both from the text data we have using the same TF-IDF vectorizer, after performing the extraction of unigrams and bigrams from the data, we again trained and tested our models, we used the same technique and parameters for our models as parameters tuning and we got the following results:

Table 1 Performance using the testing data

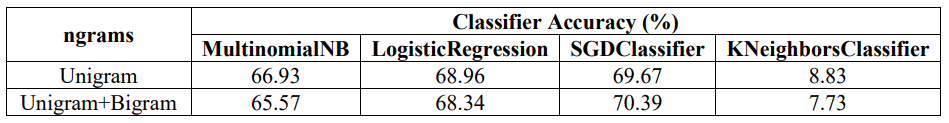
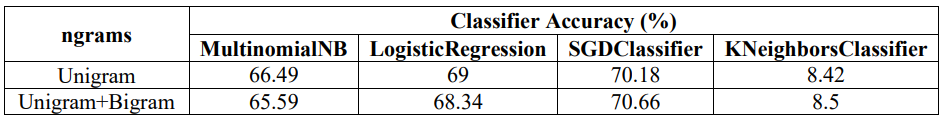


Table 1 shows the performance of the four different machine learning classification models we have utilized, the table also shows the comparison of the accuracies.

## 3.2 TF-IDF vector with Stemming:

Now, we will use the same TF-IDF vectorizer but this time we will use the stemming along with it. The stemming technique is used to cut the words into their roots. For the stemming technique, we have utilized a package which is known as the NLTK, from this package, we imported another model which is known as the “snowball English stemmer”, and we created a class which is named “Snowball”. After performing the stemming, we moved ahead to train and test our proposed models again on the data, we firstly trained and tested the models using the unigrams only after that did we train and tested the same models with the same parameters turning with unigrams and bigrams. The results are shown in the table below:

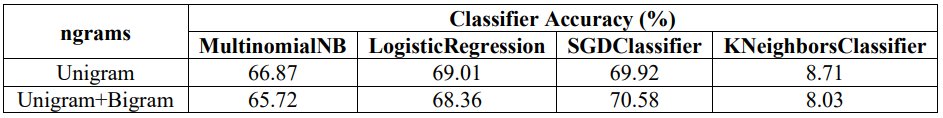
Table 2 Accuracies on testing data with stemming



## 3.3 TF-IDF vector with Lemmatization:

After applying the unigrams and bigrams along with stemming the dataset, the final step left is to perform the lemmatization on the data and trained our proposed models once again to see the whole accuracy and strength comparison of the models we have selected for this project. The lemmatization is often known as the morphology, this technique is used to grammatically correct the texts in the dataset, there is a possibility that the texts in the dataset have some grammatical mistakes. We have utilized the same NLTK package as we used for stemming, but this time we have imported the word net lemmatizer. After performing the lemmatization on the dataset, we again trained our models with bigrams and unigrams. Their results after lemmatization are shown below:

Table 3 Accuracies on the models after Lemmatization



# 4. Results:

From the start to the end of this project, we got the following results. The results are discussed in the next heading:

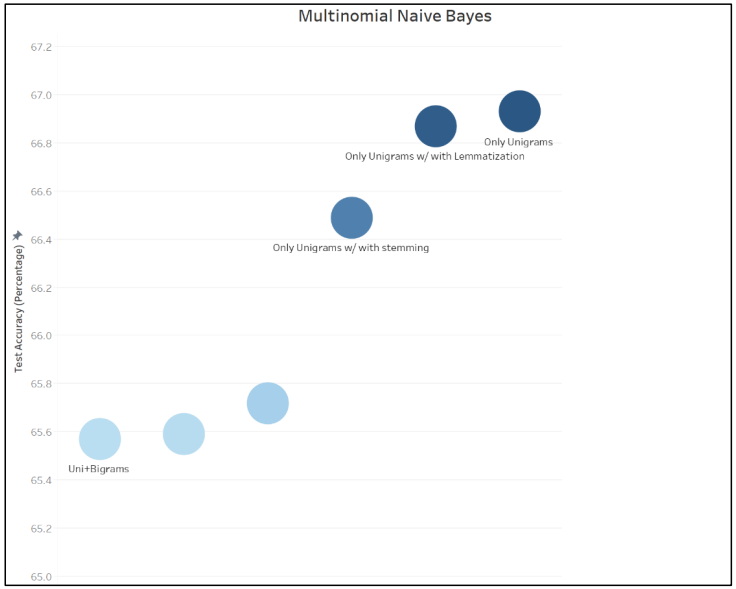


Figure 4 Naive Bayes Performance

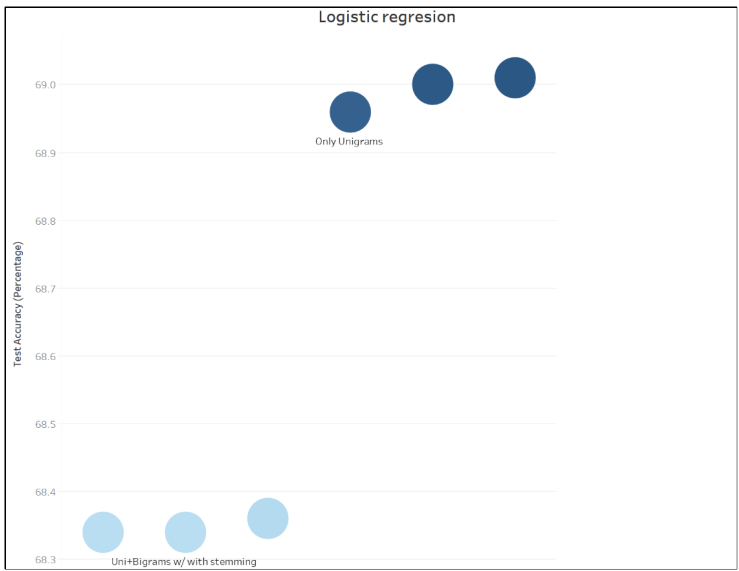


Figure 5 Logistic Regression Performance

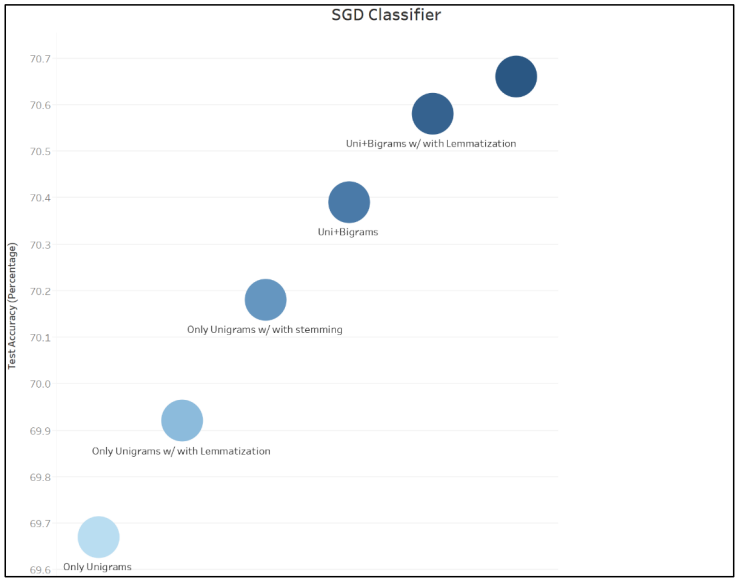


Figure 6 SGD Performance

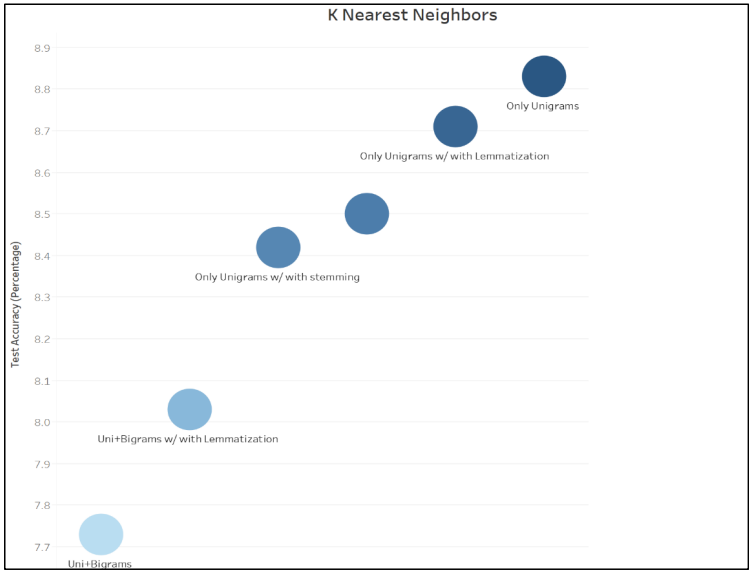
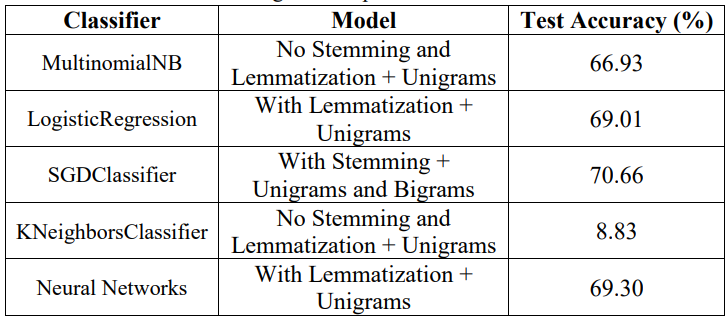


Figure 7 KNN Performance

Table 4 Summary of the performance



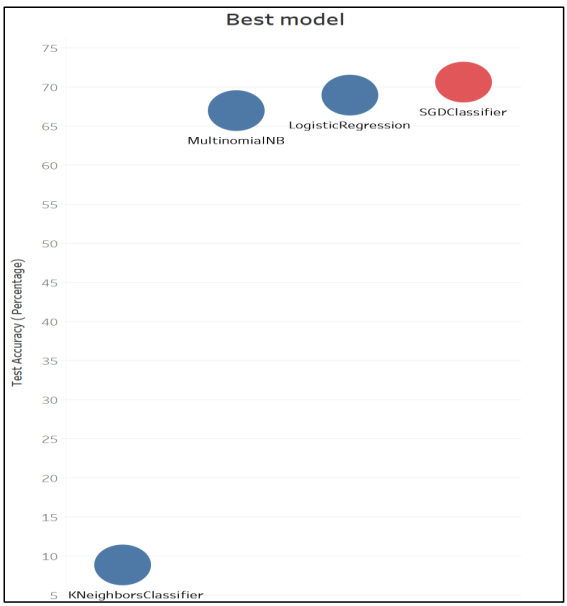


Figure 8 Model comparison

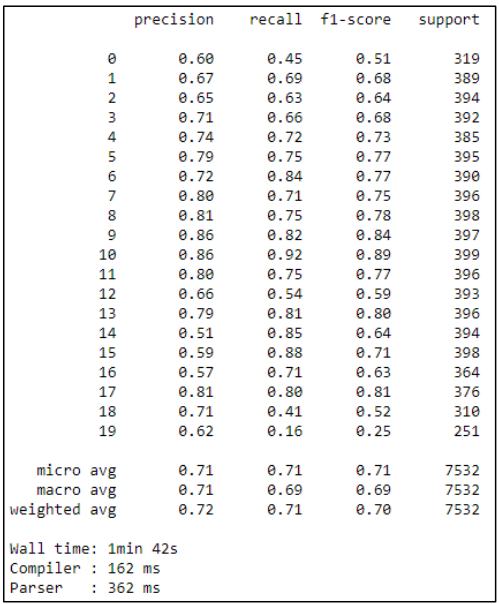


Figure 9 SGD Performance Metrics

# 5. Evaluation & Discussion:

The results and performance of the machine learning models that we have trained and tested are discussed in this section. In the results section, the first four figures from figure 4 to figure 7, the overall model accuracy and performance are shown in the graphical form. From the model performance figures (figure 4-7), we got to know that the best performance is given by Naïve Bayes and KNN without the stemming and lemmatization, the only thing performed is the extraction of unigrams by using the TF-IDF vectorizer. In Figure 4, we can see that the Logistic Regression performed very well with the lemmatization and unigrams only. In Figure 6, we can see that the SGD classifier has the best performance level with stemming and unigrams/bigrams both.

In the results section, a table is given having the best performance of the proposed model. We can see Table 4 with the high-performance level and Figure 8 have the comparison of the trained and tested models in this project. From the Table 4, we saw that the best performance is given by the Stochastic Gradient Descent (SGD) classifier which is 70.66%, on the second number we can see that the accuracy of the level of Logistic Regression is 69.01% and on the third number, we have the MultinomialNB with the accuracy of 66.93%. From Table 4, we can see that the worst results are given by the KNN, the reason can be because KNN can’t handle a huge number of features as it’s having a data dimensionality problem.

After the comparison of the performances of the models we have utilized in this project, we concluded that the SGD classifier performed very well by giving an accuracy of 70.66%. This is the best model for classifying the 20 newsgroup data into different classes. In Figure 9, we have drawn the performance metric of the SGD classifier which is showing the recall, precision, and f1 score for all the classes in the dataset.

# 6. Future Work:

The goal of this project was to classify the 20-newsgroup data into 20 different classes, for this purpose, we proposed four different machine learning classifiers. In the future, the following steps can be taken into consideration to get better results:

* Other machine learning classifiers can be used with proper parameter tuning to get the best results.
* Neural networks can be taken into consideration to improve the accuracy and performance level.
* Better model evaluations like overfitting and underfitting can be optimized.
* More features can be extracted like names, location, etc, and used as extra features by using the Named Entity Recognition.
* More word embedding features can be extracted from the data.
* Transfer learning can be proposed for better results.

# 7. Conclusion:

We have a dataset of 20-newsgroups further classified into different classes, we proposed four different machine learning classifiers which include Logistic Regression, SGD classifier, Naïve Bayes, and KNN. We noticed that the best performance is given by the SGD classifier by achieving an accuracy level of 70.66%. we also noticed that by using the TF-IDF vectorizer, we got good results than by using the count vectorizer. We saw that the best accuracy -archived by the SGD classifier is due to stemming and unigrams and bigrams both extracted in the TF-IDF vectorizer. The unigrams were seen to help the models achieve a better accuracy level. At the end of the project, we got the highest accuracy of 70.66% achieved by the SGD classifier for the twenty different classes.