**CSE 6363: Machine Learning**

**Project 2: Report**

**About the dataset:**

The dataset used is the 20 Newsgroup Dataset, which can be downloaded for free at https://www.cs.cmu.edu/afs/cs/project/theo-20/www/data/news20.html. The most well-known samples for text clustering and document classification are those used in this process.

20,000 messages, distributed (almost) evenly across 20 distinct newsgroups, were collected from 20 different netnews newsgroups for this dataset. Twenty newsgroups were divided into 1000 messages each, and the names of the newsgroups were used to determine the partitioning.

The 20 different newsgroups that the documents are partitioned into are:

1. alt.atheism
2. talk.politics.guns
3. talk.politics.mideast
4. talk.politics.misc
5. talk.religion.misc
6. soc.religion.christian
7. comp.sys.ibm.pc.hardware
8. comp.graphics
9. comp.os.ms-windows.misc
10. comp.sys.mac.hardware
11. comp.windows.x
12. rec.autos
13. rec.motorcycles
14. rec.sport.baseball
15. rec.sport.hockey
16. sci.crypt
17. sci.electronics
18. sci.space
19. sci.med
20. misc.forsale

**Method:**

The algorithm that is applied to this is the **Naïve Bayes Classifier**.

Naïve Bayes Classifier is a linear classifier based on the Bayes Probability Theorem. Mathematically, Bayes Theorem states that:

A screenshot of a cell phone

Description automatically generated

Naive Bayes is a classification algorithm for binary and multiclass classification problems. Often, probabilities are used for prediction and that is why Naïve Bayes Classifier is used for document classification.

**Implementation:**

This model has been implemented using Python 3, which categorizes the data into one of the twenty available newsgroups. The dataset has been classified into the two catagories one for the Training and other for the Testing.

* **Splitting Data set into train and test 50-50:** 
  + First, the dataset – 20 Newsgroup should be downloaded and unzipped.
  + Once the dataset has been unzipped, this file can be run.
  + Creating a empty dataset to collect all the folders and sub-folders.
* **Extracting words:**
  + After cleaning the data and replacing the special characters and extra spaces the next step we are loading the words using a forloop and adding the words to sets.
  + In the Next step we are finding the probability of the each step.
* **Training:**
  + I first counted the words in each lesson, after which I made a dictionary of every word along with its frequency.
  + I then deleted the less common terms from the dictionary for words with a frequency count of less than 3.
  + After that, the classes were trained by figuring out how likely each phrase was to fall into each category.
  + If a word in the dictionary does not appear in any of the training courses, the default count is set to 2, which is the lowest possible count because the counts of other words are at least 3.
* **Testing:**
  + The other half of the data is used for assessment after the training is finished.
  + I determined the exponential value of the likelihood that each word in a class is present. These numbers are taken into account as a sum. The outcomes are stored in a dictionary after this is completed for all courses.
  + For the test output, the class with the greatest probability for a given file is taken into account, and the actual class is then compared with it. If they line up, the accuracy rises, lowering the total error and boosting the model's accuracy.
  + After comparing the right instances to all of the instances, we can determine the accuracy for each class the mean of all the clasess.

**Note:**

Python libraries used in this model are:

* os
* nltk
* word\_tokenize
* Counter
* tqdm
* random

**Results**

I have then taken the mean of the accuracies obtained for each class. This comes up to 78.2956%.

