COMP 551 – APPLIED MACHINE LEARNING – WINTER 2022

# MINIPROJECT 2:
# CLASSIFICATION OF TEXTUAL DATA

*Frida-Cecilia Acosta-Parenteau – 260870444*
*Karl Michel Koerich – 260870321*
*Simon Nakane Marcil – 260926522*

McGill University, Montréal, QC, Canada

**ABSTRACT**

Social media platforms have become a primary point for consuming content related to current trends, political news, science discoveries, and overall population sentiment towards daily events. Therefore, creating annotated datasets containing enormous amounts of text written by web users makes sense. Using machine learning predictive models to classify this data can be helpful for many applications, such as public health, security, economics, and politics. This paper presents two algorithms trained on two of these datasets. We use Gaussian Naive Bayes and Logistic Regression to perform classifications tasks on the Sentiment140 and the 20NewsGroup datasets. In the first, perform binary classification between positive and negative sentiment, while in the second, we try to predict which of the 20 newsgroups wrote the corresponding documents. We use K-fold Cross-Validation for hyperparameter tuning, experiment with different training sizes, and compare the models' accuracies. Finally, we achieved 65% and 75% accuracy using Naive Bayes and Softmax Regression on 20NewsGroup, and 60% and 81% accuracy using Naive Bayes and Logistic Regression on 140 Sentiment.

***Index Terms*—** K-fold Cross-Validation, Naive Bayes, Logistic and Softmax Regression.

## 1. INTRODUCTION

Naive Bayes and Softmax regression are both machine learning (ML) models used in text data analysis. This project will compare the two models based on two textual datasets. As we are working with text, we first go over the feature extraction procedure used. Then, the Naive Bayes model is implemented and the logistic regression package is used from scikit-learn. In addition to familiarizing ourselves with these algorithms, K-fold cross-validation is implemented to tune the hyper-parameters.

The two datasets are the 20 newsgroups dataset and the sentiment 140 data sets. The 20 newsgroups data is a collection of documents that will be multi-classed into one of the 20 news group that wrote the documents. Likewise, the sentiment 140 dataset are twitter posts classified into either a positive or negative sentiment.

This project compares two types of classification models, the generative and the discriminative classification. Naive Bayes is a generative classification algorithm revolving around Bayes' theorem [1]. Generative classification models classes by predicting which class is most likely to have a given observation in its features. Naive Bayes achieves this by learning the joint probability distribution of the data. Then, it can use Bayes' rule to calculate conditional probability to make a prediction of its label [1]. On the other hand, logistic or softmax regression is a discriminative classification which learns the input features that are the most useful to classify. Logistic regression does this by calculating the posterior probability to draw decision boundaries between the possible classes [1].

## 2. DATASETS

### 2.1. Text Data Feature Extraction

This project classifies two datasets where its data is text. Since ML algorithms do not take plain text as input, this data has to be turned into numerical features. The bags of words representation is used to convert the text into a numerical values. This representation works by tokenizing every word appearing in the dataset as a feature and counting its occurrences [2]. Using scikit-learn's CountVectorizer, the tokenizing process can be achieved to obtain the dataset's feature vectors. This function returns a sparse matrix where a feature's value is the occurrence of a word in the text [3]. The occurrence of a word is a numerical value that can be passed to the algorithm, but is not ideal to fit the model to. The issue is that the occurrence can be influenced by the length of the text where a higher count is seen because there are more

words in the text itself [2]. Alternatively, these features are normalized by using the frequencies of the word rather then the occurrence. A word's frequency can be obtained from the occurrence sparse matrix using the Tfidftransformer. It has the benefit of rescaling the frequency of words by how often they appear in all documents, so that the scores for frequent words like "the" that are also frequent across all documents are penalized.[4].

When vectorizing, there are other parameters to consider as a preprocessing procedure to achieve appropriate classification. Since all words are tokenized, the number of features augment really quickly [2]. To reduce the processing required by the model, we decided to reduce the number of features by excluding words to tokenize. This decision was made because not all words contributes towards the classification of an instance. For example, stop words such as 'the', 'I', 'and', etc. does not add meaning to the sentence which does not help classify by sentiment [5]. By defining stop_words, max_df, min_df, and max_features when using CountVectorizer, the feature size can be adjusted to some extent [3]. In contrast, the risk here is that you could also be excluding words greatly affecting the accuracy of the classification. For instance, in a sentiment classification, if a negation word is excluded the opposite sentiment could be predicted [5].

### 2.2. 20 Newsgroups Dataset

The 20 Newsgroups dataset is a collection of documents written by different newsgroups. The objective with this dataset is to do a multi-class classification determining which of the 20 newsgroup wrote the corresponding document. There is a total of 18846 instances where its data is text [6]. As mentioned in the previous section, the text feature is vectorized to be processed.

Scikit-learn does provide an already vectorized dataset to import, but this did not allow flexibility to adjust the size of features based on the words we would like to exclude [7]. Therefore, the text feature dataset was imported and respectively vectorized with the method mentioned above. To reduce the features to process, the built-in stop word list for English is used. If the word occurrence is in less than 2 documents, it disregarded by setting the min_df parameter to 2. These two constraints reduces the number of features from 101631 to 39115.

### 2.3. Sentiment 140 Dataset

The Sentiment 140 Dataset is a collection of posts from users of Twitter social media platform. It is divided into two different CSV documents, the training data contains 1 599 999 instances. And the testing data contains 358 instances. We imported them into pandas data frames. The only feature from the datasets that we use is the text of the tweet. The class corresponds to the polarity of the tweet, for which there were three classes originally but we only kept negative and positive sentiment. We one-hot encoded the class negative to 0, and the class positive to 1. As mentioned in the previous section, the text feature is vectorized to be processed. We reduced the number of features from 684 357 to 139 662 features. In the CountVectorizer function, the 'stop_words' where set to 'english', and the minimum word occurence to at least 3 different instances (min_df=3). Finally, we set the maximum frequency of the word presence in documents to 0.5 (max=0.5), since there are almost equal number of positive and negative instances.

## 3. RESULTS

### 3.1. Hyper-parameter Tuning

We tested two hyper-parameters to achieve the best accuracy for logistic or softmax regression. We used 5-fold cross validation to tune these hyper-parameters. This method evaluates the accuracy on five sub-samples of the training set. For each possible value of a hyper-parameter, five accuracies are calculated to then obtain an average accuracy for that hyper-parameter setting. These averages can then be compared to determine what parameter setting should be applied for the best overall performance.

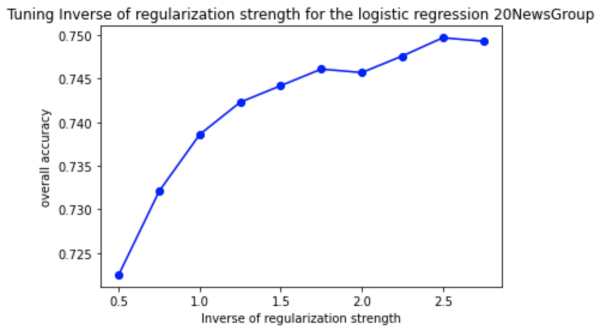
Gaussian Naive Bayes, did not have any hyper-parameters to tune. For logistic and softmax Regression, we decided to test two hyper-parameters: the regularization strength and the penalty cost functions. For regularization strength, value between 0.5 and 3 with increments of 0.25 was passed for cross validation. Then there were four possible penalty cost functions which are L1, L2, elasticnet, or none.

### 3.2. 20 Newsgroups Results

The following presents the results of the experiments on the 20 newsgroups dataset. Figure 1 and figure 2 shows the best hyper-parameter setting for softmax regression. Running 5-fold cross validation, the best regularization strength with the highest average accuracy of 74.97% is 2.5. As for the best penalty cost to use out of the four options is L2.

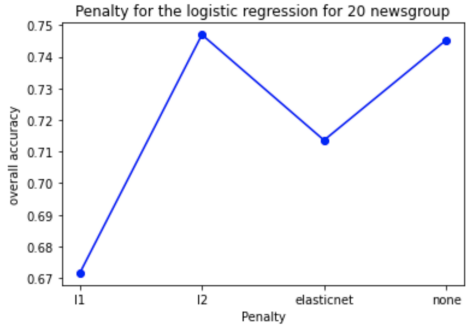
Applying these parameters in our next experiment, the performance between softmax regression and Gaussian naive Bayes are compared. As a result, softmax regression out performs naive Bayes by 10% with an accuracy of 74.81%. This performance difference does not change with the size of the training dataset comparing figure 3 and 4.

Furthermore, we expect for softmax regression to work better with larger datasets. Which complies with the plot in figure 3. In contrast, naive Bayes is expected to work better with smaller datasets. However, an increase in accuracy is seen with the training size in figure 4. This might be because the 20 newsgroup dataset does not get large enough to see a sudden plummet in accuracy like seen with the sentiment 140 dataset following this section.




```
The best C: 2.5 with accuracy of: 0.7497
```

**Fig. 1**. Accuracy plot with different regularization strengths on 20 newsgroups




```
The best penalty: l2 with accuracy of: 0.747
```

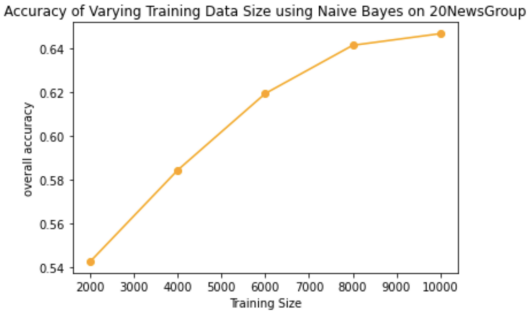
**Fig. 2**. Accuracy plot with different penalty cost on 20 newsgroups


```
The best split: 10000 with accuracy of: 0.7480974124809742
```

**Fig. 3**. Accuracy plot with different training size using softmax regression on 20 newsgroups

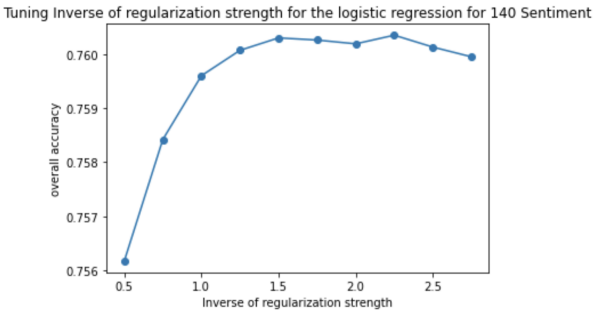
### 3.3. Sentiment 140 Results

The x-axis represents the inverse value of the regularization strength. For smaller values the strength is bigger. For the 140 sentiment dataset, we see the figure 5 that the value C=2.25 has the best accuracy.




```
The best split: 10000 with accuracy of: 0.6468797564687976
```

**Fig. 4**. Accuracy plot with different training size using naive Bayes on 20 newsgroups

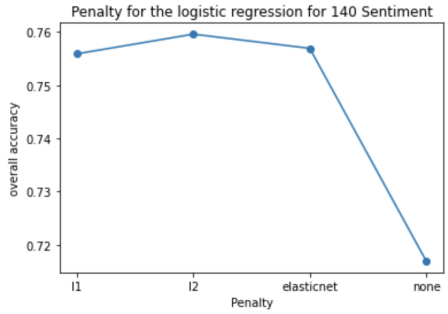



```
The best C: 2.25 with accuracy of: 0.76035
```

**Fig. 5**. Accuracy plot with different regularization strengths on Sentiment 140

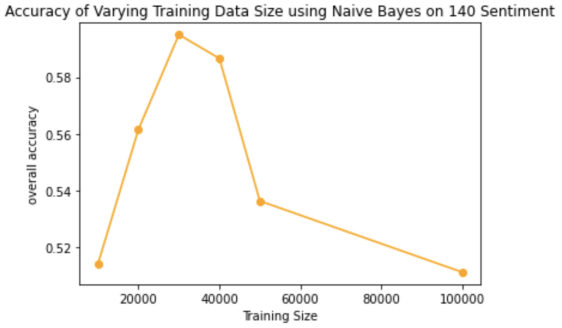
For the penalty, which we define as cost function in the course, we see in figure 6 that 'l1' , 'l2' and 'elasticnet' all have accuracies between 0.75 and 0.76. Therefore we can't confidently say that one is the best. Therefore, we can see that adding regularization does not decrease the accuracy thus means there is no overfitting. A logistic regression with 'l1' penalty yields sparse models, and can thus be used to perform feature selection which is beneficial in bag-of-words problems which have too many features [8].

Finally, using the best hyper-parameters (penalty='l2', C=2.25), we can compare the effects of the varying training set size on the accuracy for Naive Bayes and Logistic Regression. In the Naive Bayes figure 7, we see that the accuracy reaches a peak at 0.595 with 30 000 instances, and then decreases sharply as the size increases, which is expected. Indeed, this model assumes that all the features are independent, therefore, bag-of-words datasets that have a high number of features will have complex relationships between features that can cause the model to fail. In the Logistic Regression figure 8 we have the expected behavior, which is that the more instances we have the better the accuracy will be. The highest accuracy achieved is 0.815 at 100 000 instances.




```
The best penalty: l2 with accuracy of: 0.75959
```

**Fig. 6**. Accuracy plot with different penalty cost for on Sentiment 140




```
The best split: 30000 with accuracy of: 0.5949720670391061
```

**Fig. 7**. Accuracy plot with different training size using naive Bayes on Sentiment 140


```
The best split: 100000 with accuracy of: 0.8156424581005587
```

**Fig. 8**. Accuracy plot with different training size using logistic regression on Sentiment 140

## 4. DISCUSSION AND CONCLUSION

Overall, we were successful in conducting classification tasks for both datasets. We achieved 65% and 75% accuracy using Naive Bayes and Softmax Regression on 20NewsGroup, and 60% and 81% accuracy using Naive Bayes and Logistic Regression on 140 Sentiment. One of the main challenges faced while working with these datasets was their size. To better handle the data, we had to significantly reduce the number of instances analyzed, especially for Sentiment 140. However, due to it being a large and binary dataset, by selecting only a fraction of the dataset, we were still able to get evenly distributed representatives of each class, which in turn prevented us from performing poorly in our classification tasks. For possible directions, it would have been useful to do a grid-search for the hyper-parameters tuning. In our case we only proceeded by taking the best regularization strength first, using the default L2 penalty, and then tried to find the best penalty with the strength value found during the tuning. Therefore, L2 had a much better chance at being the best hyper-parameter in all the datasets. One of the main issues that prevented us from doing this was the large running time it would've taken.

## 5. STATEMENT OF CONTRIBUTIONS

Dataset cleaning: Simon and Frida-Cecilia; Write-up: Everyone; Implementation of models: Karl; Experiments: Everyone.

## 6. REFERENCES

[1] Tony Jebara, *Machine learning: discriminative and generative*, vol. 755, Springer Science & Business Media, 2012.

[2] Scikit Learn, "Working With Text Data," Accessed Mar. 07, 2022 [Online].

[3] Scikit Learn, "sklearn.feature_extraction.text.CountVectorizer," Accessed Mar. 07, 2022 [Online].

[4] Scikit Learn, "sklearn.feature_extraction.text.TfidfTransformer," Accessed Mar. 07, 2022 [Online].

[5] Joel Nothman, Hanmin Qin, and Roman Yurchak, "Stop word lists in free open-source software packages," in *Proceedings of Workshop for NLP Open Source Software (NLP-OSS)*, Melbourne, Australia, July 2018, pp. 7–12, Association for Computational Linguistics.

[6] Scikit Learn, "sklearn.datasets.fetch_20newsgroups," Accessed Mar. 07, 2022 [Online].

[7] Scikit Learn, "sklearn.datasets.fetch_20newsgroups_vectorized," Accessed Mar. 07, 2022 [Online].

[8] Scikit Learn, "1.13. Feature selection," Accessed Mar. 07, 2022 [Online].