COMP551 Assignment 2

Alif Naufal Farrashady (261121584) Calvin Chan (261121702)
Katherine Meyers (260987652)

November 8, 2022

Abstract
In this paper, we examined two different datasets, IMDB reviews dataset and 20 newsgroup dataset comprising 18,000 different posts on 20 topics - which was narrowed down to 4 specific categories (hardware, hockey, space, mid-east).

Different models were tested with each of the datasets, thereafter, comparing each model accuracy against KNN. For the IMDB dataset, as we processed the reviews as binary data (1 for positive and 0 for negative), we used logistic regression for prediction. The best result was achieved with minibatch stochastic gradient descent, learning rate of 0.1 and batch size of 20, at a testing accuracy of 81.4%. Whereas KNN achieved a 77.8% testing accuracy at k-value of 14. For the 20 newsgroup dataset, as it was multi-categorical, we implemented multiclass logistic regression, with the best accuracy of 77% achieved by also using minibatch stochastic gradient descent and learning rate of 0.1. KNN proved to be much weaker in multi-categorical classification, as it only achieved a testing accuracy of 55.5% at k-value of 3.

Introduction
In this paper, we explore two classification techniques, Logistic Regression, as well as Multiclass Regression. There are similarities between these two techniques but the key difference is that Logistic Regression is meant for binary classification problems while Multiclass Regression can handle multiclass classification problems. We will be developing a custom implementation of these two techniques and applying them on two different datasets, one binary and one multiclass. In order to benchmark the performance of our custom implementations, we will be comparing the results against Scikit-learn's implementation of K-Nearest Neighbours (KNN).

We found that our models were able to outperform KNN for both datasets. The difference in performance was large for the multiclass problem, with more than 20 percentage points difference, while for the binary classification problem, both models achieved a good accuracy, although Logistic Regression performed marginally better.

Methods
We have come up with a custom implementation of the Logistic Regression and Multiclass Regression models, along with Stochastic Gradient Descent and Minibatch Stochastic Gradient Descent as the gradient descent methods. Both models use 2 stopping criteria for gradient descent, when the maximum iterations is reached, as well as if the change in gradients are small.

For the gradient descent methods, they support momentum and schedule parameters which we will experiment with and study how they affect model performance. Minibatch Stochastic Gradient Descent also has 2 extra parameters, batch size and whether or not to shuffle the dataset before generating batches. Additionally, we will also experiment with different values

of learning rates, maximum iterations and epsilon (threshold to compare changes in gradients against) to observe which combinations of hyperparameters will produce the best model performance. In order to verify that the gradient descent methods are correct, we will plot a graph that shows cross-entropy loss as a function of the number of iterations.

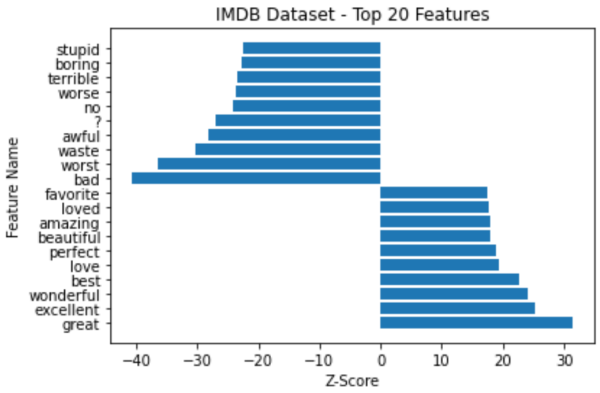
Datasets

The first dataset that we processed was the IMDB reviews dataset, which contained 25000 reviews, mapping the rating to the words used in the corresponding review. This dataset contained 50% positive and 50% negative reviews. We first converted the target from consisting of categorical values ranging from 1 to 10 to consisting of binary values (1 for a positive review and 0 for a negative review), using 5 as the threshold between positive and negative. Then, since this dataset had 89526 features to begin with, one for each word, we first filtered out the words occurring in less than 1% of reviews and greater than 50% of reviews, since these are unlikely to have an impact on the predictions. This process reduced the dataset to 1744 features. From there, we used linear regression on each remaining feature with the target and calculated its z-score. We then further narrowed down the features by selecting the 250 words with the highest absolute z-scores. We selected the top 250 words because beyond this point we noticed a drop off in words that were obviously correlated to positive or negative reviews.

The second dataset we processed was the 20 newsgroups dataset, which contained 2347 reviews with each mapping to a category it is under. 4 distinct categories were chosen for easier vectorization of words (0: 'comp.sys.ibm.pc.hardware', 1: 'sci.space', 2: 'rec.sport.hockey', 3: 'talk.politics.mideast'). Distribution of the categories were quite even (0: 590, 1:600, 2:593, 3:564). Using CountVectorizer from sklearn, we preprocessed, tokenized and filtered the stopwords for those that occurred in less than 1% of the documents and those which appeared in all documents. Additionally, stopwords are also removed using NLTK library's English stopwords. This reduced the words to a csr matrix of 2347 rows x 1600 columns - with the columns representing 1 feature word each, thus there are 1600 unique words. The data was then normalised using TfidfTransformer which would make it easier for the model to learn. Then the top 100 feature words of each class were chosen based on the highest mutual information scores. Before combining them into a dataframe and removing the duplicates which left us with 212 feature words chosen, and 4 one-hot-encoded y categories.

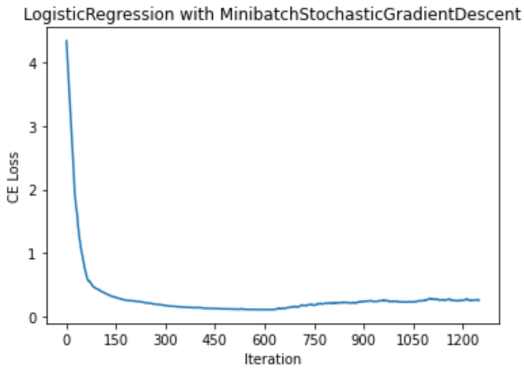
Results

Consider the following horizontal bar plot of the top 20 features in the IMDB reviews dataset.



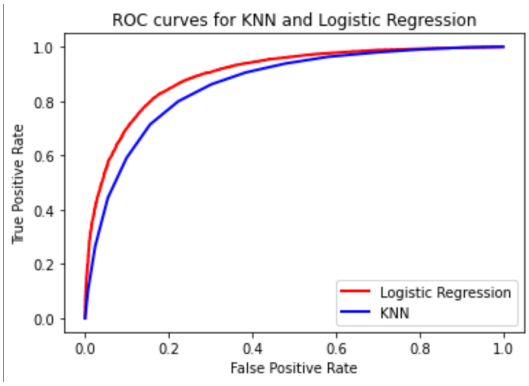
From this figure, it is clear that the 10 features with the lowest z-scores are strongly correlated to negative reviews and that the 10 features with the highest z-scores are strongly correlated to positive reviews. The only exception is the "?" feature, which has a very low z-score, despite not carrying any obvious negative connotation.

When conducting logistic regression on the IMDB dataset, we obtained a testing accuracy of 81.4%. We conducted the logistic regression using minibatch stochastic gradient descent, with a learning rate of 0.1 and a batch size of 20. The convergence of this model can be seen in the following convergence plot.

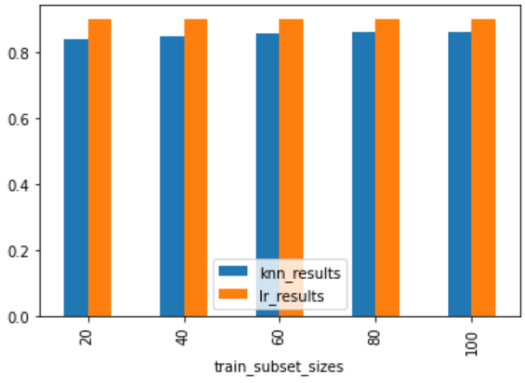


When conducting KNN on the IMDB dataset, we obtained a testing accuracy of 77.8%. This was achieved using k = 14, which was determined to be the optimal k-value.

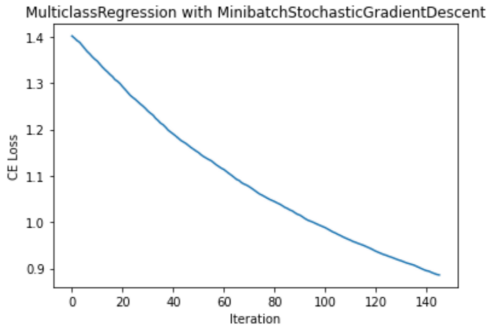
The ROC curves for logistic regression and KNN on the IMDB dataset were as follows.



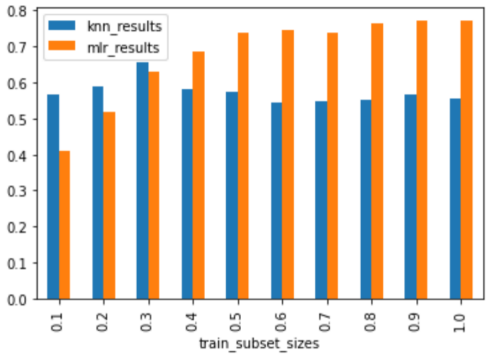
When comparing logistic regression to KNN for different percentages of the training data, we noticed that the accuracy was relatively constant for logistic regression, but increased with increasing percentage of training data for KNN.



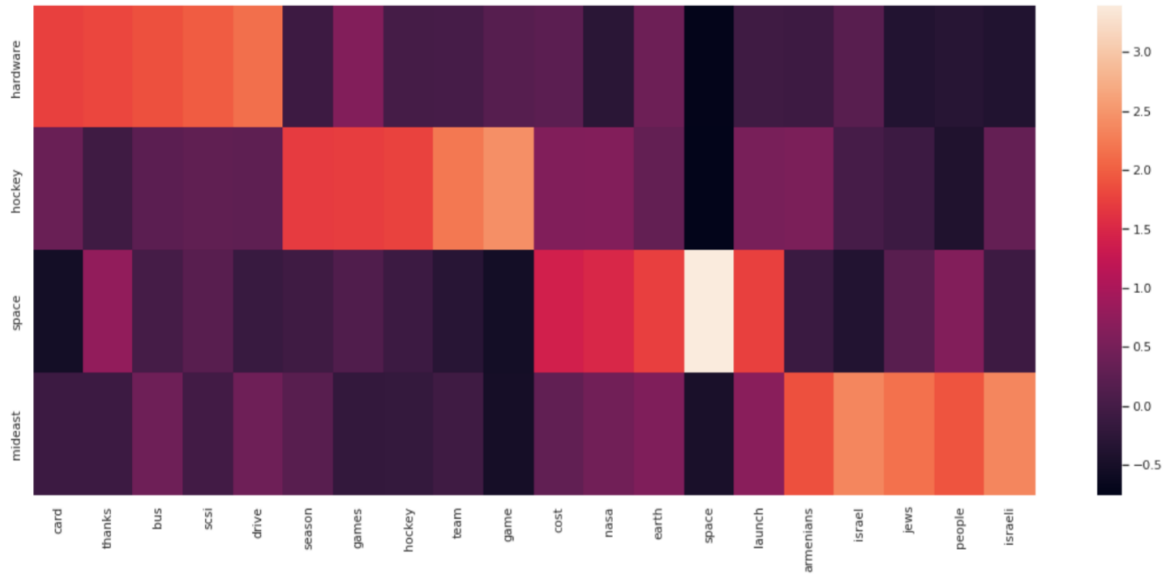
With Multiclass Logistic Regression, we were able to achieve a test set accuracy of around 77%. The model was also able to converge relatively quickly in less than 200 iterations at a learning rate of 0.1.



Comparing KNN and Multiclass Logistic Regression (MLR) at different training set sizes shows that KNN performs better at lower training sizes. However, when more training data is available, MLR is able to perform better and achieve higher test set accuracies while KNN decreases in accuracy.



Based on the weights for multiclass logistic regression, we can see that the top words in each class are quite distinct. The heavily weighted words in each category: 'hardware', 'space', 'hockey' and 'mideast', are only heavily weighted for their own category. Thus, this could also be an indicator as to why multiclass logistic regression had performed quite well in separating the categories with an accuracy of around 77% since each category likely had their own distinct set of words.



Additionally, we experimented with various hyperparameters on the gradient descent method, the gradient descent method as well as the learning rate of both models. As mentioned earlier, we will experiment with the choice of gradient descent method (Stochastic Gradient Descent vs Minibatch Stochastic Gradient Descent), learning rate, momentum, as well as batch size.

For the IMDB reviews dataset with Logistic Regression, the models that performed the best utilised Minibatch Stochastic Gradient Descent, with a batch size of 100, as well as learning rate of 0.1. However, momentum of 0 and momentum of 0.1 were able to achieve the same accuracy, at 0.81.

```
['Logistic, gd: MbSGD, lr: 0.1, m: 0.0, batch: 100, s: False', 0.81]
['Logistic, gd: MbSGD, lr: 0.1, m: 0.1, batch: 100, s: False', 0.81]
```

For the newsgroups dataset with Multiclass Regression, the model that performed the best used Minibatch Stochastic Gradient Descent, with a batch size of 1000, learning rate of 0.1 and momentum of 0.1, achieving 78% accuracy. There were also other models that were close, at 77% accuracy. The common hyperparameters amongst these models were the usage of Minibatch Stochastic Gradient Descent, as well as a learning rate of 0.1. This suggests that these 2 hyperparameters are the relatively more important determinants of model performance. This is further supported by the results on the IMDB reviews dataset, where again, Minibatch Stochastic Gradient Descent with learning rate of 0.1 achieved the best performance.

As such, our additional experiments on these hyperparameters revealed that choice of gradient descent method and learning rate are most likely to have the largest effect on model performance. This generally fits with what we have learnt so far, in that choice of learning rate is important, as a large learning rate may lead to overshooting past the optimal solution during gradient descent, while a small learning rate will result in longer times to converge, possibly requiring more iterations than the maximum threshold set in our models. In our case, Minibatch Stochastic Gradient Descent outperformed regular Stochastic Gradient Descent.

```
['Multiclass, gd: MbSGD, lr: 0.1, m: 0.1, batch: 1000, s: False', 0.78]
['Multiclass, gd: MbSGD, lr: 0.1, m: 0.0, batch: 100, s: False', 0.77]
['Multiclass, gd: MbSGD, lr: 0.1, m: 0.0, batch: 1000, s: False', 0.77]
['Multiclass, gd: MbSGD, lr: 0.1, m: 0.1, batch: 100, s: False', 0.77]
```

Discussion and Conclusion

Our results indicate that logistic regression predicts with roughly the same accuracy regardless of the size of the training set, whereas the accuracy of KNN increases with an increasing amount of training data. Although, it is important to note that the accuracy was consistently higher for logistic regression, indicating that logistic regression is a better choice of model for this task. Our results also indicate that for a limited training set, KNN may work better in classifying multi-categorical data. However, as the amount of training data increases, one may opt to use multiclass regression instead to achieve better results.

However, for both logistic regression and multiclass regression, these trends should be investigated further on a large number of datasets to confirm that they are true in general, and not simply trends associated with these specific datasets.

Statement of Contributions

Alif implemented logistic regression and multiclass regression, and experimented with different learning rates and forms of gradient descent. Katherine processed and ran experiments on the IMDB dataset and Calvin processed and ran experiments on the 20 News Groups dataset. All members contributed to the report.