CSC2062 AIDA – Assignment 3

[Thomas Reid]

[40263793]

[28/04/2021]

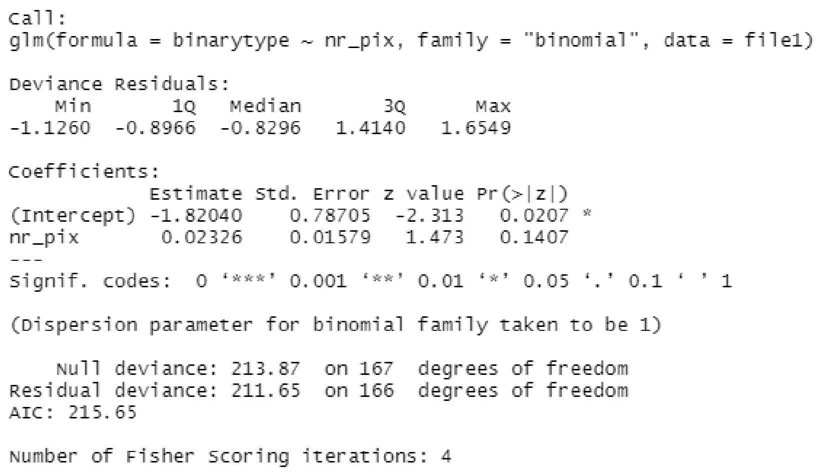
Introduction

In this assignment I will be using data analysis techniques that I developed in assignment 2, to preform classification techniques on a large dataset. In Sections 1 and 2 I will be using machine learning techniques such as K-nearest neighbour and logistic regression to preform classification between different groups of data, distinguishing if the data is a symbol, letter, or number. In section 3 I will be using a range of techniques to classify between 21 different image categories. In each section I will explain my reasoning for using different techniques and display figures and graphs to promote my findings.

Section 1

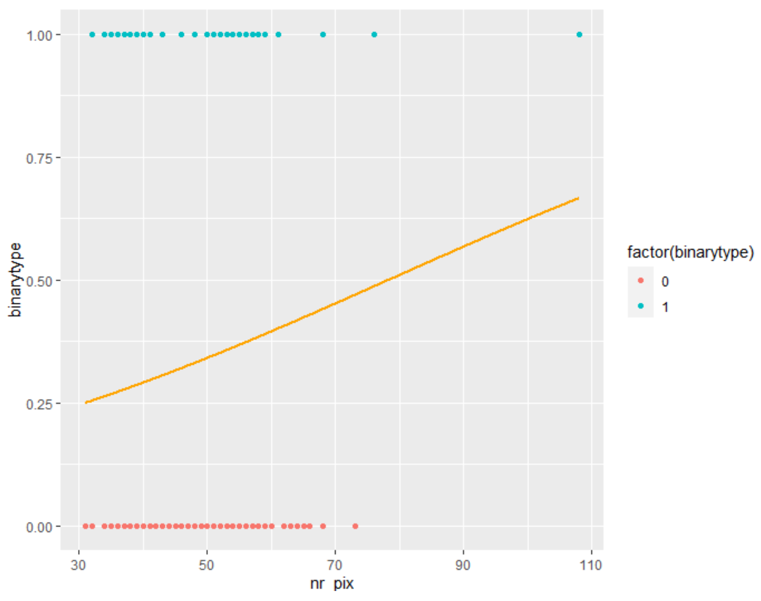
1.1

In this section I have been asked to use the features.csv file created in assignment 2 to fit a single logistic regression model to predict the probability of belonging to the “math symbol” category of images. To do this I imported the features.csv file and wrote code that would distinguish between the datasets based upon their label. If an image is a math symbol it will have the value of 1, whereas if the image belongs to the letter or symbol dataset, it will be given the value of 0; I called this column the binarytype. This will allow me to differentiate between their predicted category and their label to calculate the model’s accuracy. I created a logistic regression model that will predict the “binarytype” value based upon only the Nr_Pix value. Below is the results table for this regression model.



```
Call:
glm(formula = binarytype ~ nr_pix, family = "binomial", data = file1)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.1260  -0.8966  -0.8296   1.4140   1.6549

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -1.82040    0.78705  -2.313   0.0207 *
nr_pix       0.02326    0.01579   1.473   0.1407
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 213.87  on 167  degrees of freedom
Residual deviance: 211.65  on 166  degrees of freedom
AIC: 215.65

Number of Fisher Scoring iterations: 4
```

The coefficient estimate for nr_pix is 0.0233, meaning there is a slight positive correlation between the number of pixels in an image and whether or not that image is a math symbol. The Z-score for this linear regression is 1.47. The p-value is 0.141, this is greater than 0.05 meaning the result is not significant and we accept the null hypothesis, this correlation could have occurred by chance.



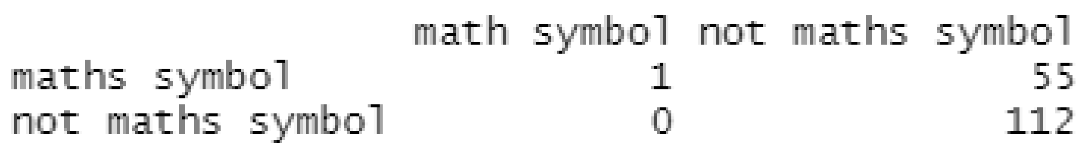
Above is a visualisation of the regression model. On the Y-axis is the binarytype, this allows us to discriminate between the two sets of data we are classifying against. on the x-axis is the number of pixels in each image. The orange line is a visualisation of what binarytype value an image would have been predicted based upon the number of pixels in the image. We can see a positive slope which would be expected since the summary of the model stated it would have a slight positive corelation.

1.2

I used the train function to create the linear regression model and used 5-fold cross validation as this should increase the model's accuracy as it splits the data into 5 different subsets of data, using 4 to train the model and 1 to test it, changing which subset is used to test the data each time. I then used this linear model to predict each items binarytype value based upon its nr_pix value. From these predicted values I used 0.5 as the threshold to distinguish if the image was a math symbol or not.

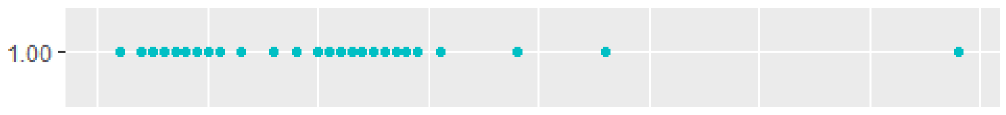
The accuracy of the model was 67.3%. The model's recall was 0.671 while the F1 score was 0.8. The precision of the Model was 1 when predicting non-maths symbols but only 0.0179 when predicting maths symbols; this is because it almost always predicted the image type was a non-maths symbol. It makes it very precise at predicting non-maths symbols but only predicts a math symbol correctly 1.8% of the time.

Below is the confusion matrix that was generated from the data.



```
                  math symbol not maths symbol
maths symbol                1               55
not maths symbol            0              112
```

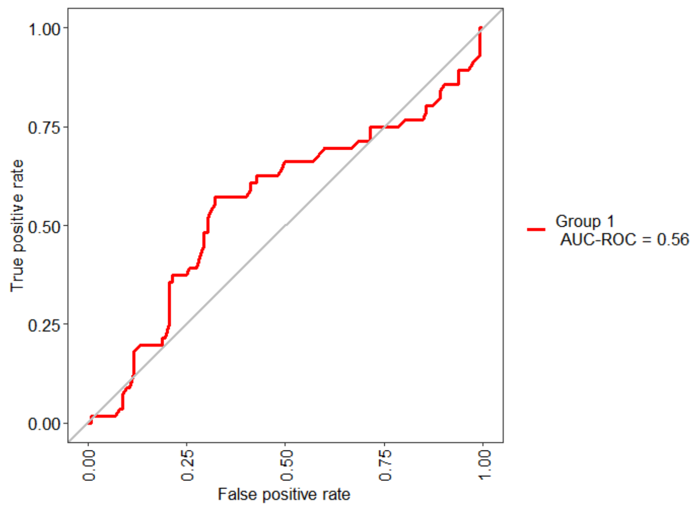
The one time it correctly predicted a math symbol, the nr_pix value it was using to predict was an outlier. All the values in the left cluster were classified as being non-maths symbols, however the value on the far right will have a large influence on the slope of the line as it is an influential outlier.



There is not a strong correlation between whether an image is a maths symbol and the number of pixels in the image. In an ideal scenario the relationship should produce a sigmoidal curve that would have a clear cut-off point where it would prove useful at differentiating between the two sets of data, however the lack of relationship causes the line to have a continuous positive slope that does not prove useful.

A way that we could improve this model would be changing the threshold that allows us to predict whether an image is a maths symbol or not. It is currently set at 0.5, but if it were lowered it could increase the model's precision for detecting maths symbols. It could also affect the model's overall accuracy. We can plot an ROC curve to assess what value would give us the best accuracy.

1.3



Looking at the ROC curve, it tells us what threshold would give us the best results as a classifier. The curve can give us an indication of what value to use in order for our model to maintain the highest specificity and the highest sensitivity. Using a decision threshold of 0.56 will give us the lowest false positive rate while having the highest true positive rate. To improve the model created, using a decision threshold of 0.56 would be best.

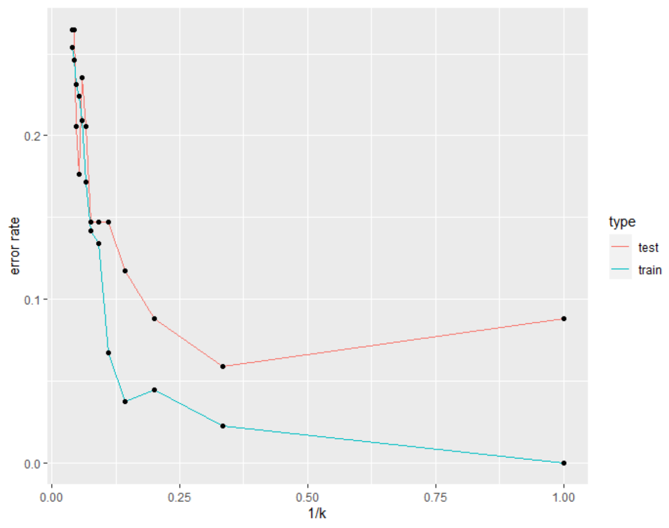
Section 2

To preform K-nearest neighbour classification on the feature's dataset, I first had to import the dataset then create a new column which groups the data into 3 categories - numbers, letters, and math symbols. To preform KNN classification with varying values of K I had to create an array containing all odd values from 1-25 and use a for-loop to cycle through these values. I created a trained model using KNN that was trained and tested on the same dataset of 168 items. The model will attempt to classify the data into one of the three categories based upon its first 6 columns in the feature's dataset. The accuracies are presented in pairs below.

```
[1] 1
[1] 1
[1] 3
[1] 0.9880952
[1] 5
[1] 0.9464286
[1] 7
[1] 0.952381
[1] 9
[1] 0.9166667
[1] 11
[1] 0.8869048
[1] 13
[1] 0.8630952
[1] 15
[1] 0.8452381
[1] 17
[1] 0.8392857
[1] 19
[1] 0.8214286
[1] 21
[1] 0.8154762
[1] 23
[1] 0.797619
[1] 25
[1] 0.8035714
```

The odd rows give the value for K that was used, and beneath gives the accuracy of the model for that value of K. Since the model was trained and tested on the same dataset it would only make sense that when K is equal to 1, the accuracy was 1, as each point would fit the model exactly, but as K increased the accuracy of the model decreased. When K is equal to 1 the model has low bias but high variability as the model is overfit whereas whenever K increases the model has higher bias but lower variability. If we were to test the model on a different dataset with K equal to 1 you could expect it to have a lower accuracy than for an increased value of K.

I then preformed the same K-nearest neighbour test although this time I preformed 5-fold cross validation. I tested the model on both the set testing dataset and also the training dataset and recorded their accuracies. Below is the graph of these results.



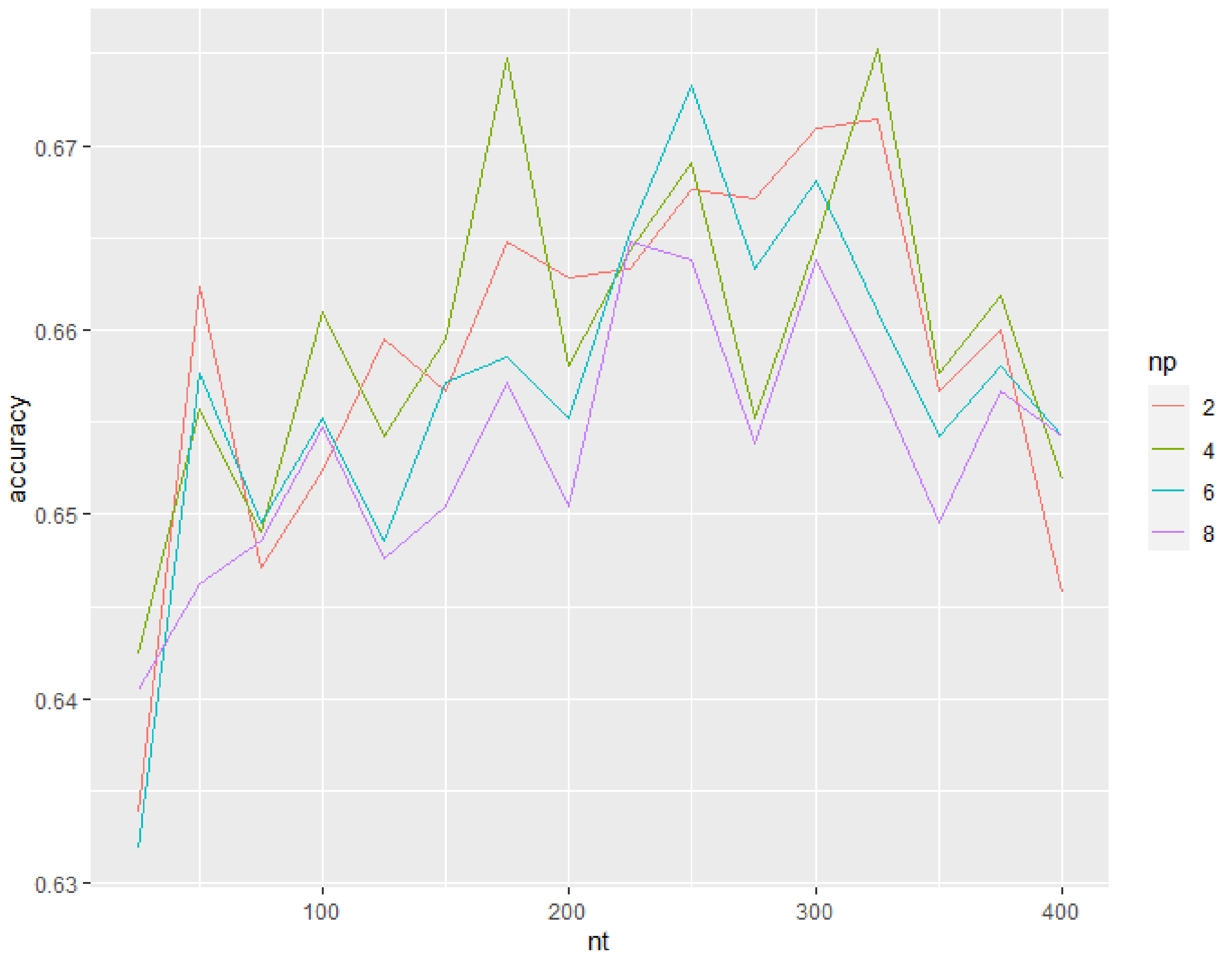
When testing the model on the training dataset whenever K was equal to 1 it gave us a 0% error rate, however whenever tested on the set testing dataset, it did not preform as well producing a higher error rate of 0.092. The error rate for the testing dataset then decreases as K increases, as the data was overfit when K was equal to 1, meaning the model created was too sensitive to noise in the training dataset. Therefore as K increased it allowed a better fit for the model. The best value of K was 3 as it gave the testing data the lowest error rate. As K increased above 3, the error rate for both datasets increased. This is because the model was underfit, meaning it was not flexible enough to fit the test data accurately enough.

Section 3

For this section I will no longer be using the features.csv dataset that has been used to this point in the report. Instead, I will be using the dataset that was provided, consisting of 21 different image categories with 100 training items each. I will also be preforming multinomial classification into the 21 different image categories.

Section 3.1

To preform random forest classification, the data was imported, and an array was created containing values from 25 to 400 in increments of 25. I also used a tune grid, containing the values 2,4,6,8 which represented the number of predictors that will be used to create the model. I used the train function to build the random forest model with 5-fold cross validation. All the results were stored in an array, to enable it to be plotted after the function was run. Below is the graph created.



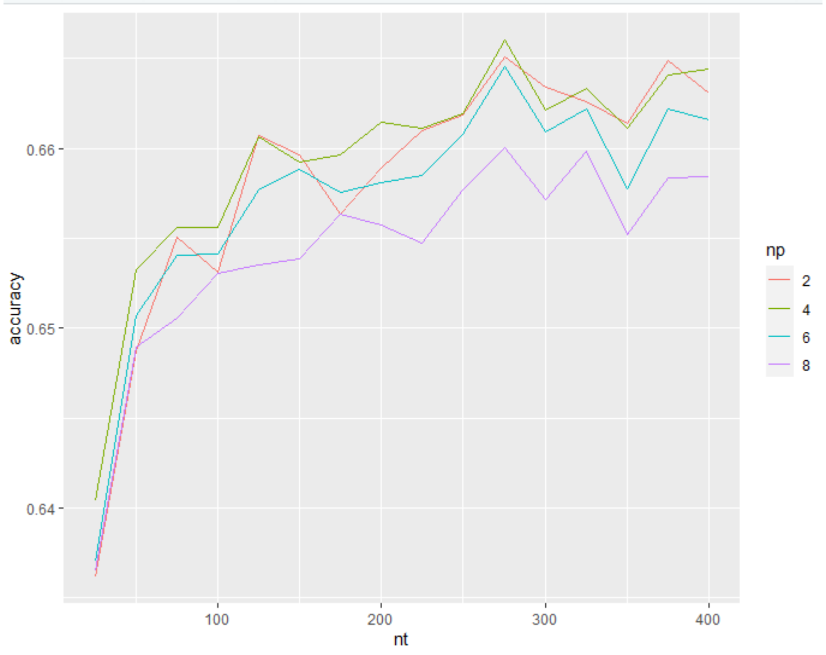
Examining the graph, we can arrive at several conclusions. Whenever there is too many predictors it can lower the model's accuracy; for example, whenever the model was generated using 8 predictors it was nearly always the least reliable compared to the models created using a smaller number of predictors. However, the models produced using only 2 predictors can still be unreliable compared to some of the models produced using more predictors.

When trying to assess the best number of predictors and trees to use when creating the model, we need to examine the peaks of all four plotted results. There is a slight trend we can observe in that, as the number of trees increased the overall accuracy of the models increased. However, the number of trees is still prone to underfitting. As the models approach 400 trees, there is a significant drop off in their accuracy, since the model has been too sensitive to the training data.

The overall best fit of the data is using 325 trees with 4 predictors.

Section 3.2

Using the above graph to fine tune for the best combination of number of predictors and number of trees still creates variability. I ran the model 15 times to obtain 15 different accuracies which will allow less variability.



There os similar patterns displayed across the two graphs. Both have an increase in accuracy as the number of trees increased. Another similarity between the two graphs is that using 8 predictors to generate the model was consistently less accurate than models created with a lower number of predictors.

However there is also several differences between the two graphs. Graph 2 contains less random interference as there is less of a range in the accuracy of the lines, whereas Graph 1 is more varied. In Graph 2 there is less of a drop off in accuracies for the models created using 400 trees. The overall trend seen in Graph 2 is an increase in accuracy as the trees increase but it begins to level off. The model which produced the best accuracy was the model created using 4 predictors and 275 trees.

Section 3.3

For this section I was to create a model that could preform multinomial classification on the 21 image categories with the highest accuracy. I wanted to use the two classification techniques I had already used, KNN and random forest classification, along with some methods I have never encountered: Naïve bayes, neural networks.

Random Forest

Initially with random Forests, I was able to reuse a lot of the code I had previously written, alongside knowledge of which combination of variables give the highest accuracy - 4 predictors with 275 trees.

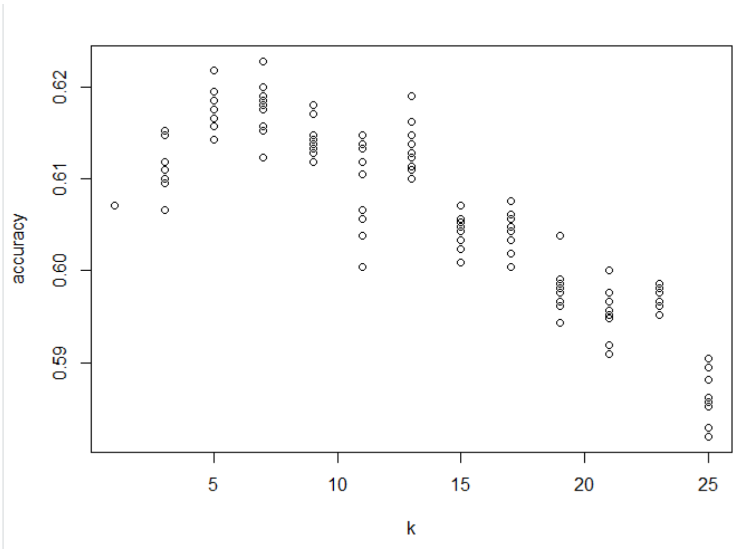
Inside a for-loop, which would repeat 10 times, I used a train function using the random forest method to create the model, using 5-fold cross validation to get an accuracy which was saved into a dataframe. To obtain the overall final accuracy I calculated an average of all the accuracy values saved in the data frame which gave the value 65.5%.

Using random forests is a good classifier but there is definitely room for improvement within this model. I could fine tune the number of trees and classifiers to produce a higher accuracy value but running these random forest classifiers takes a lot of time and processing power and with all the variables fine tuned to their best ability, the increase in accuracy might only be extremely minimal and focusing on different classification methods might be more worthwhile.

K-Nearest neighbour

Again, I was able to reuse a lot of the code for KNN that had been written for section 2. However, I had to make slight adjustments, as in section 2 I was preforming binomial classification whereas now I am preforming multinomial classification of the entire 21 label dataset.

I also need to fine tune the data to get the best value of K that will produce the most accurate model. To do this I preformed classification of the data for each odd value of K between 1 and 25 using 5-fold cross validation and repeated this 10 times to improve accuracy as it still has an element of randomness to it. These were the results.



From the data here we can see that when K=7, the accuracy is consistently the highest so when creating a KNN model with the highest accuracy this value for K will be used.

I ran the final KNN model 10 times using 5-fold cross validation and set K equal to 7. The average accuracy for the KNN model was 60.7%.

The model does not preform as well as the model created using random forests. This can be down to the fact that KNN does not work as well when there is a large number of classifiers. So far using a random forest model provides the highest accuracy.

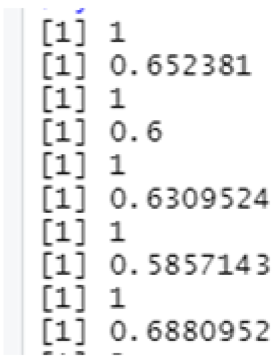
There is also several classification techniques that I had never used before which might improve the accuracy - Naïve Bayes and Neural networks. Research suggests that both of these techniques can be used for multinomial classification producing high accuracies.

Naïve Bayes

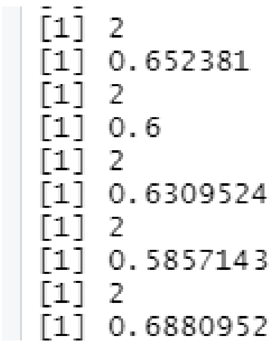
Multinomial naïve bayes is a useful classification technique that allows for the classification of data and works for both small and large datasets.

The train function that I used previously also allows for building naïve bayes classification models. So, writing the formula for this was not very different from any of the previous models.

I ran the train method using 5-fold cross validation 15 times to get the best accuracy for independent runs but each time the accuracies were identical.



```
[1] 1
[1] 0.652381
[1] 1
[1] 0.6
[1] 1
[1] 0.6309524
[1] 1
[1] 0.5857143
[1] 1
[1] 0.6880952
```



```
[1] 2
[1] 0.652381
[1] 2
[1] 0.6
[1] 2
[1] 0.6309524
[1] 2
[1] 0.5857143
[1] 2
[1] 0.6880952
```

For both independent runs the accuracies were the same

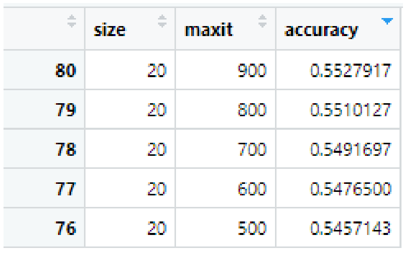
I got an average of these accuracies to achieve the overall accuracy of the naïve bayes classifier, which was 63.8%. This is a good method of differentiating between the different image labels, however it still under preformed when compared to the random forest model. There was still one method I wanted to try out however, before I work on fine tuning the best model for image classification.

Neural Networks

Neural networks are a classification technique that are included in the nnet library for R. I want to use the nnet function to build a model that will be useful for differentiating between the image labels. The nnet function has a lot of variables that allow for fine tuning of the model.

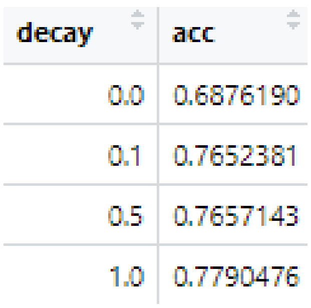
I wanted to test the accuracy of the neural network while varying the number of iterations and layers. I tested and trained the data on different datasets so that it would give a better indication of the overall accuracy of the model. Below in table* are the combinations that returned the best accuracies.

TABLE *



	size	maxit	accuracy
80	20	900	0.5527917
79	20	800	0.5510127
78	20	700	0.5491697
77	20	600	0.5476500
76	20	500	0.5457143

The combination that returned the highest accuracies occurred when we trained the model using large sizes and iterations. We can increase the accuracy of the model furthermore by changing the decay of the model. I trained and tested the model with different datasets, and below in TABLE * are the average accuracies for each decay value.



decay	acc
0.0	0.6876190
0.1	0.7652381
0.5	0.7657143
1.0	0.7790476

Having a larger decay value increased the overall accuracy of the model significantly. I also attempted to create a neural network model using scaled data. It produced an accuracy of 75%. There doesn't seem to be much of an improvement in accuracy whenever I used raw or scaled data.

Conclusions

The best model I was able to create, which would allow for multinomial classification, was using neural networks. They were able to predict the right image label 77.9% of the time after cross validation. This method of classification was significantly better than K-nearest neighbour, which worked 60.7% of the time, random forests, which worked 65.5% of the time, and naïve bayes, which worked 63.8%.

The reason Neural networks worked a lot better than the rest of the classification methods can be down to the fact that neural networks use lots of neurons that recognise patterns in a large dataset and create model complex relationships.