CSC3060 AIDA – Assignment 3

Niall Devine

40173800

Introduction

The purpose of this assignment is to use machine learning to build classifiers for image data, using data that was created from the previous assignment, as well as a large sample of unseen data used for Training. This report will detail the observations and accuracy of various predictor models such as logistic regression, k-nearest neighbour and decision trees using predetermined features from the data.

Section 1

The primary focus of this section is to use logistic regression on the feature data from assignment 2 to build classifiers for living and non-living objects. From the results of the fitted models, we will be looking at the accuracy over the data to determine which individual or combined features might be useful in correctly classifying a given object.

Before I could begin this Assignment, I first had to update the Data being carried over from the Previous Assignment. The mean reason behind this was due to Missing Data.

During the last Assignment, I was unable to correctly work out some features and so had to use random integers between 1:10 to fill these in. This would cause uncertainty in my attempt to distinguish between living and non-living objects, which would be a big problem since it means from the start all my Tests could be considered unfair. This would make it harder to spot any anomalies among my outputs.

Section 1.1 – Logistic Regression using the Verticalness feature

Upon loading the feature data into the script, I then needed to go about classifying each of the observations as either “living” or “non-living” for the purpose of the Task. In order to do this, I wrote a function that iterated through each observation of the feature data, which would then evaluate the value of the ‘label’ column.

If the value of the label was one of the living things (banana, cherry, flower, pear) then the classification for that observation would be assigned a 1. Likewise, for those observations which had labels belonging to non-living things (envelope, golfclub, pencil, wineglass), the classification would then be assigned a 0. It was from here, that classifications were made under the assumption that all living things would be represented by a value of 1 and the non-living things by a value of 0.

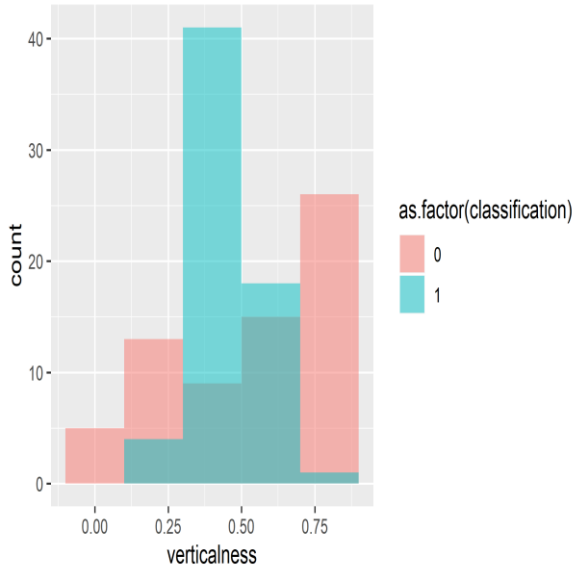


Figure 1 - A histogram of the verticalness feature, with non-living things coloured red and living things coloured blue

With all 160 observations now classified as either a living or non-living, I was then able to begin building the model. I shuffled a sample of the original dataset to a percentage of 80% to 20%, into a Training Dataset and a Test Dataset respectively. The histogram for the 'verticalness' feature shows a small overlap for both classifications between the values 0.1 and 0.6 (roughly). From a simple assessment of these results one might be able to suggest that the 'verticalness' feature could be seen as a good predictor of whether a doodle belongs to either the living or non-living category. However, more work would need to be done to ensure certainty.

The results of the model show significant p values ($p < 0.01$) for both the Intercept and the verticalness coefficients.

```
Call:
glm(formula = classification ~ verticalness, family = "binomial",
    data = training_data)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.5713  -1.0552  -0.8306   1.1499   1.3980

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)    0.9609     0.4650   2.067   0.0388 *
verticalness  -2.0848     0.8787  -2.373   0.0177 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 182.87  on 131  degrees of freedom
Residual deviance: 176.94  on 130  degrees of freedom
AIC: 180.94

Number of Fisher Scoring iterations: 4
```

Figure 3 - The results table of the logistic regression

The model was built with the training dataset and the results were close to what I would have expected. The model has a small chance that it could be certain of a doodle's classification for a given verticalness value, as Figure 2 shows.

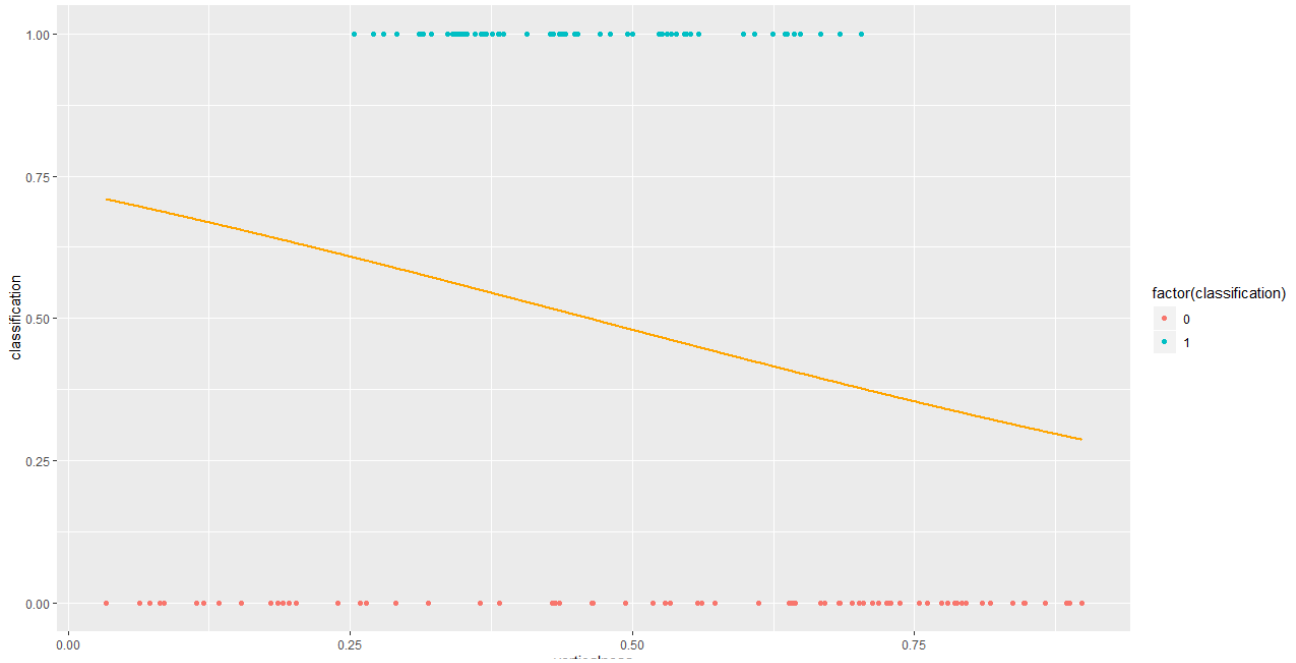


Figure 2 - The fitted curve for the model

Section 1.2 – Building a classifier using the model from 1.1

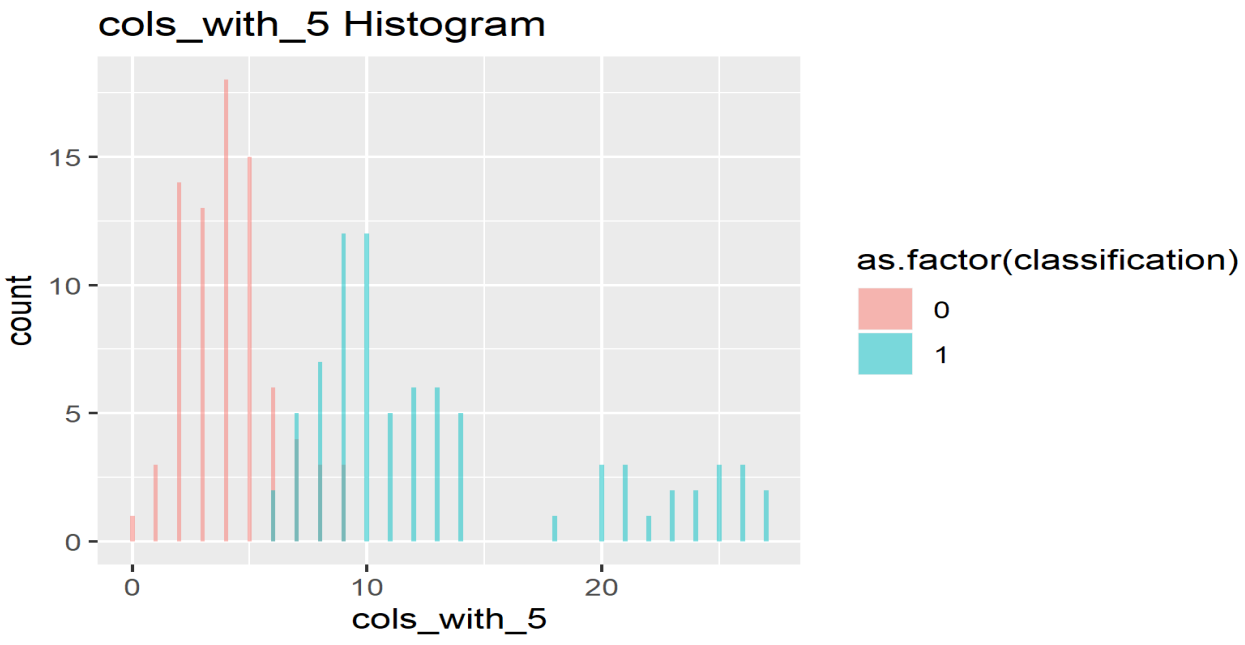
I thought finding a suitable cut-off value for p to provide the best accuracy for the model would be difficult, since there was some overlap between living and non-living for a given verticalness value. So, rather than try and guess what the best p-value would be, I decided to go out and check all possible values of p from 0.01 to 0.99 (inclusive) in increments of 0.01. These Results would then be stored into a Table (see '1.2_p_accuracy.csv' which can be found in "Images and Table Information/Section 1/1.2" for full table) showed that several p values shared the highest accuracy of 78.6%. What was quite interesting to see in these results, was that most of the p-values that obtained this accuracy were all consecutive (0.35 – 0.4) inclusive, except for 0.36 which shared similar results as the 0.41 – 0.43 range of 75%.

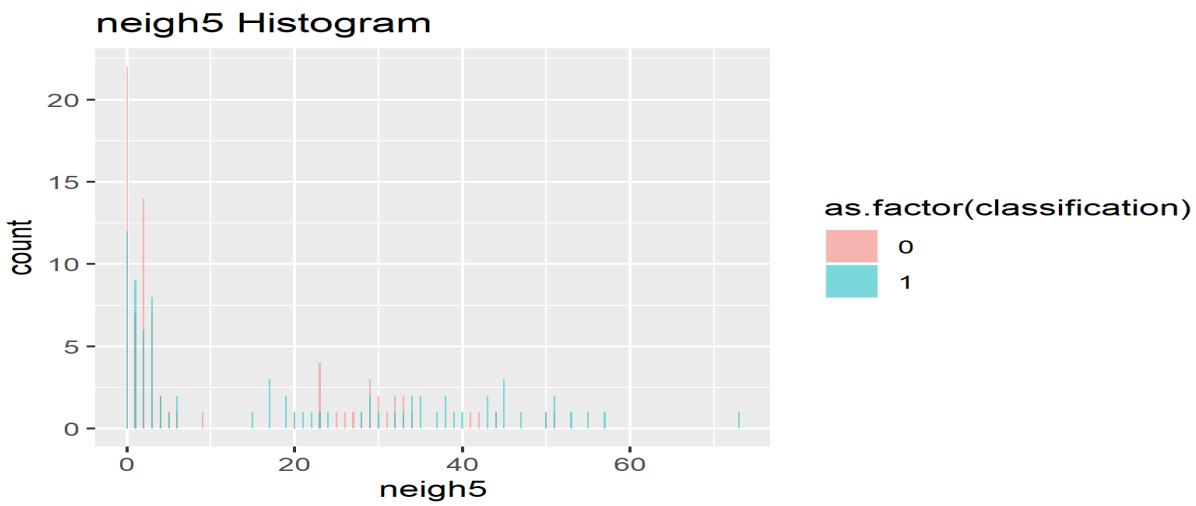
I would like to believe that the model would perform similarly for any future doodle samples if they were designed in a similar fashion to the sample used here, otherwise there would be a reasonable doubt that the code used to calculate the feature data might output different values for the verticalness and therefore result in a very different accuracy for this model.

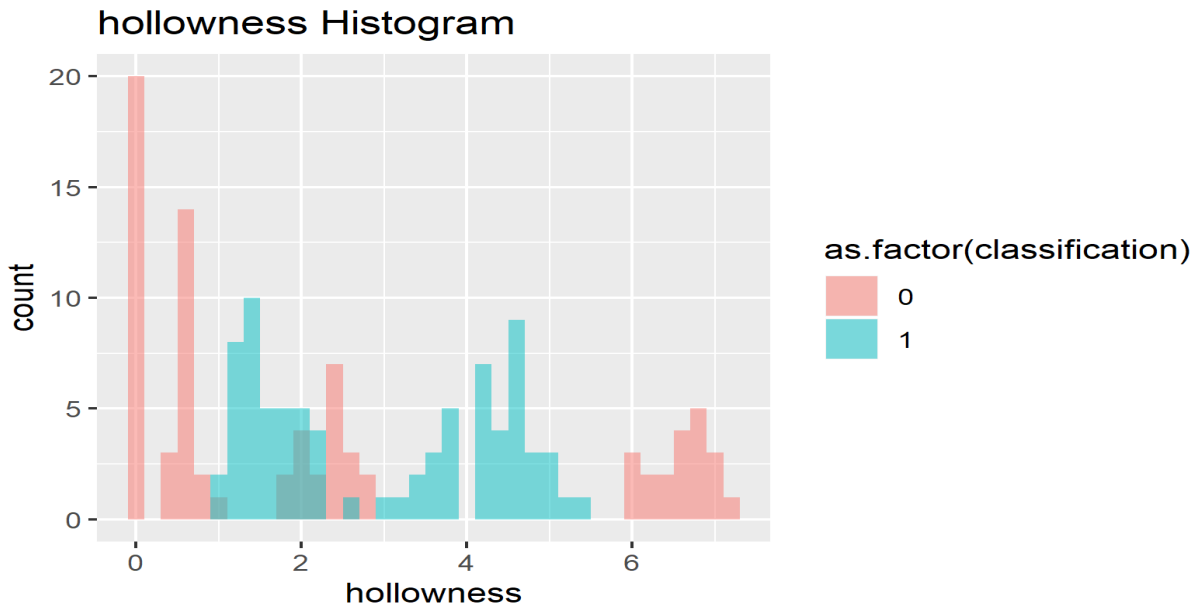
Section 1.3 – Custom Classifier using 3 Features

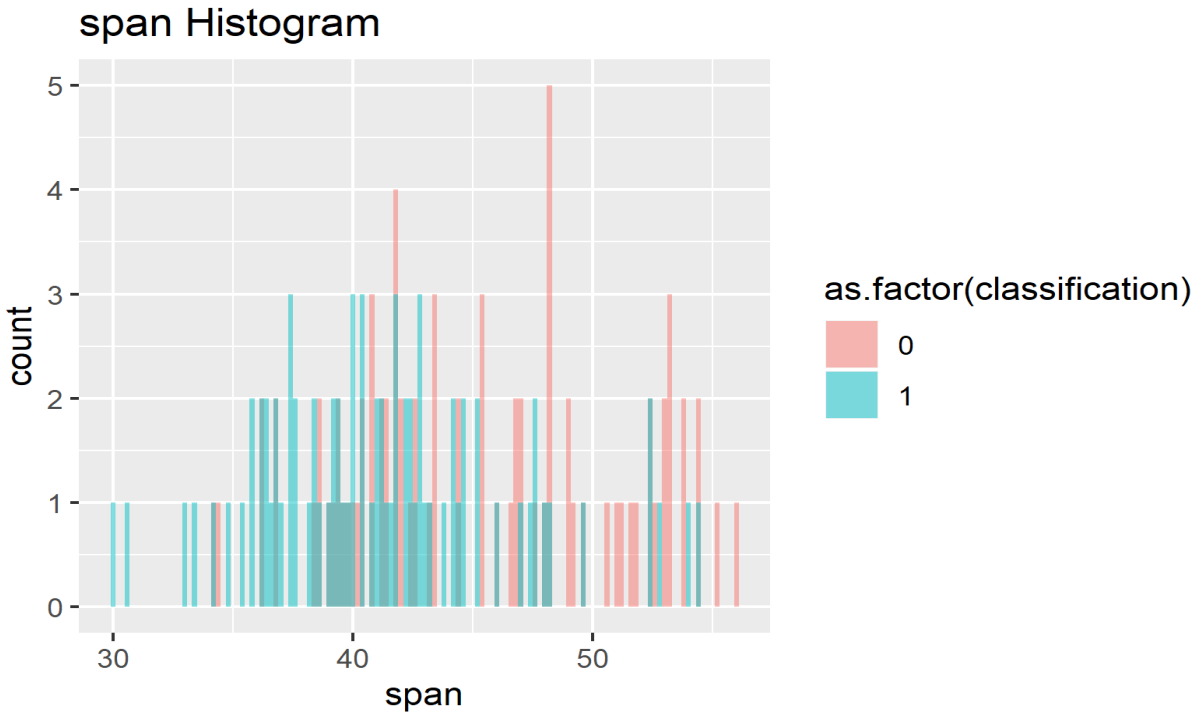
I felt that assessing the feature data from the previous assignment, would be detrimental to finding the absolute best three features on which to build a classifier, since I was unsure about myself on being able to interpret such results in order to build a classifier with the highest accuracy. This means, that I could not verify from the start whether the chosen features were indeed the best 3 features to choose from.

However, after going through the Feature Data from the previous Assignment, I believed that the best 3 features would be Cols_with_5, neigh5 and Span. I was quite happy with the turnout for Cols_with_5 and Span as I believed they distributed living and non-living objects quite well. However, neigh 5 didn't hold up as well compared to these two. You can slightly see some individuality between living and non-living in neigh5 but most of it would be tough to distinguish. I decided to then use Hollowness as a substitute for neigh5 and I feel that the Results are a lot better.




neigh5 Histogram
count
neigh5
0
20
40
60
0
5
10
15
20
as.factor(classification)
0
1


hollowness Histogram
count
hollowness
0
2
4
6
0
5
10
15
20
as.factor(classification)
0
1


span Histogram
count
span
30
40
50
0
1
2
3
4
5
as.factor(classification)
0
1

Section 1.4 – Comparing the Model for 1.3 to a Random Model

Before even investigating the accuracy of a random model, I believe I could confidently say that such a model would be inferior to the model used in 1.3. Given that each observation has a 50% probability of being correct, I would expect a random model to have a very slim chance of attaining an accuracy greater than 96.25%. Figure 4 shows the probability density for the random model. The blue line represents the number of correct predictions needed to be considered more accurate than the model in 1.3 (154 to be exact). The probability of such an occurrence is basically zero, and we can disregard such a model in comparison to the one created in 1.3 since it would otherwise be impossible for a random model to have a 96% success rate with only a probability of success for a given trial being 50%.

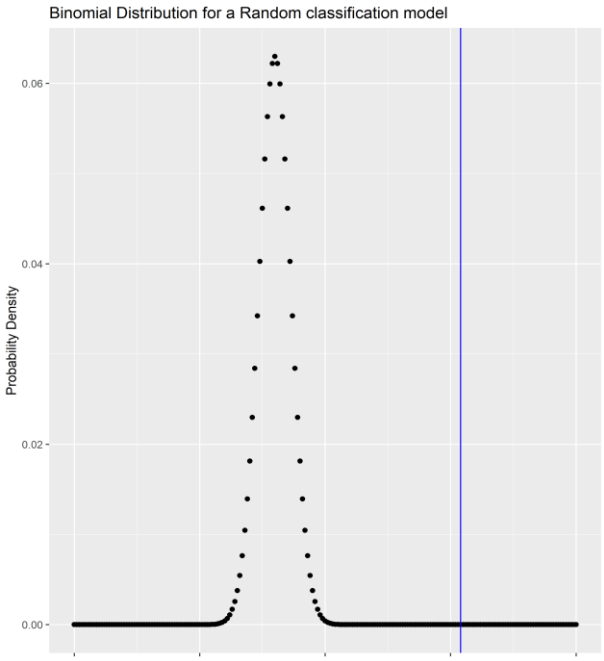


Figure 4 - The biomial distribution of a random classification model.

Section 1.5 - Analysing the Pattern of errors in the 1.3 Model

I decided to go back and gather the data whilst the code for 1.3 was executing, as the results wouldn't be different since the sample was unchanged at this point, and I thought it would be more performant than building the same model a second time. The following table shows the frequency of misclassifications made by the model in 1.3. It is a culmination of the predictions made over each validation fold.

Table 1 – The results for the model prediction over the validation sets for each fold

label	0 (non-living)	1 (living)	total
banana	3	17	20
cherry	0	20	20
envelope	20	0	20
flower	0	20	20
golfclub	16	4	20
pear	0	20	20
pencil	20	0	20
wineglass	19	1	20

The results table above shows clearly the impressive accuracy of the model; all but 3 observation of the Banana doodles from the entire sample of living things, was classified incorrectly. This is in comparison to the 5 misclassifications among the non-living objects (4 Golf clubs and 1 Wineglass were classed as living). Without going into detail fully examining the validation samples in each fold, I think it would be accurate to put the misclassifications down to the distribution of objects in each fold sample, as the cross-tables printed to the console revealed a relatively large imbalance some samples. In the first fold for example, observations of wineglass objects accounted for 6.2% of that fold's observations, in comparison to what should be around 12.5%.

Total Observations in Table: 32

test_data$label	test_data$predicted_class 0	1	Row Total
banana	2	2	4
	0.500	0.500	0.125
cherry	0	3	3
	0.000	1.000	0.094
envelope	5	0	5
	1.000	0.000	0.156
flower	0	4	4
	0.000	1.000	0.125
golfclub	2	2	4
	0.500	0.500	0.125
pear	0	4	4
	0.000	1.000	0.125
pencil	6	0	6
	1.000	0.000	0.188
wineglass	2	0	2
	1.000	0.000	0.062
Column Total	17	15	32

Figure 5 - The cross table for the first fold of validation items and the model's predictions.

Determining which features might improve the accuracy of the model in 1.3 was interesting, as I believed that the addition of a fourth feature to the model could yield an improvement, but as I experienced in 1.3, finding a good fourth feature on the basis of visual assessment was difficult, mainly because I would need to find a feature that can be seen to have distinct values for living and non-living objects. The horizontalness feature, for example, showed strong visual variation between the groups, as well as a near normal distribution, so I would have thought it to be the likely fourth candidate.

Section 2

In this section I will be investigating the accuracy of models built using k-nearest neighbour to classify individual images. I will be evaluating the difference between models built with and without 5-fold cross validation, to assess the effect that overfitted data might have on the accuracy of the model.

Section 2.1 - KNN Classification For All Odd Values 1-59

Using the supplied training data of 4000 items, I decided to create training and test set split at a 80 to 20 percent ration. I found that the quickest way to get odd k values between 1 and 59 was to iterate through every number in that range and build a model when the value modulo 2 resulted in a remainder of 1 (i.e. an odd number). I stored the accuracy and complimentary error rates for each value of k in a table.

Initially I had made the mistake of testing the model's accuracy over the full sample of 4000 items, which yielded accuracies in low accuracies, which at first, I had begun to believe was an indicator of how poor the features were when combined. However, I eventually realised that I had to test the accuracy over the labels for the test data I created for the model, since those labels were unseen by the model.

When this mistake had then been corrected, I then begun to see far greater accuracies, which made much more sense given that the model included the best predictor features (for logistic regression) from 1.3 and 1.5. which I had the results of each iteration appended to a data frame for future reference.

Additionally, I decided to sample training and test datasets only once before iterating through the values for k, since I thought it would be better to compare the accuracy for different values of k over the same sample.

Section 2.2 – 5 Fold Cross Validation KNN for all Odd Values 1-59

In my initial attempts to complete this section, I made the same mistake as I had done in 2.1, which was testing model accuracy over the full sample, resulting in the same terrible accuracies. When fixed however, the results showed a slight improvement in the range of accuracies.

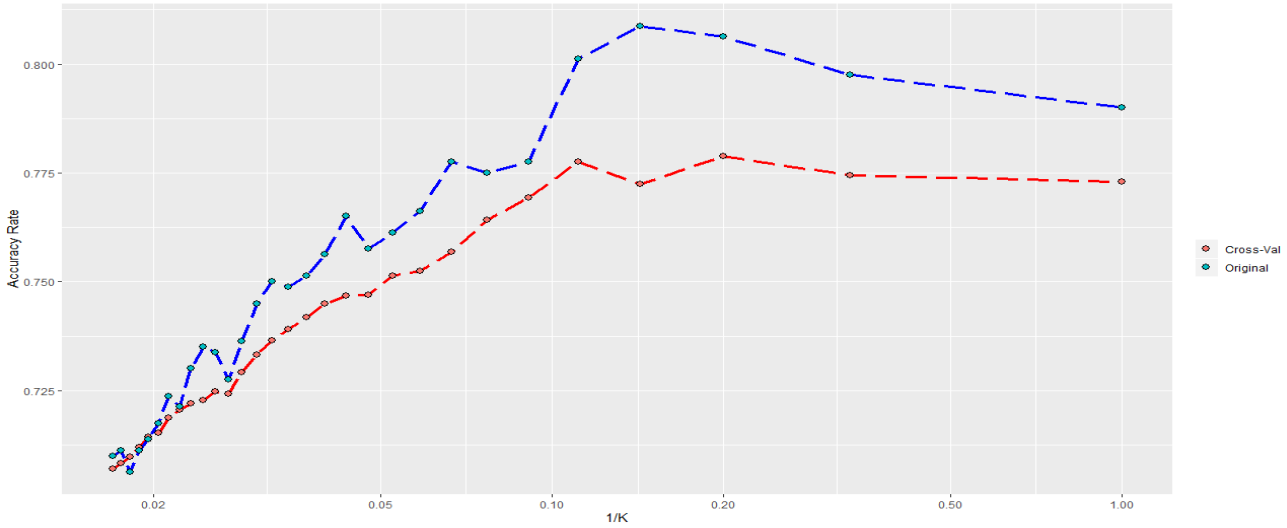


Figure 6 - The cross-validated accuracies and the sample accuracies for 1/k.

When it came to plot the accuracies against 1/k, it took a lot longer than I had thought. Despite having the results for both methods tabled against k and 1/k, the actual plot looked terrible. By default, the plotted graphs were scaled realistically, such that the units used to space the values of 1/k were constant. This led to a situation where most of the values for 1/k were too coalesced to make any meaningful interpretation of the results. After a couple of Tries and failed attempts, I then decided the solution was to scale the x axis logarithmically.

The graph shows a slight contrast in how the accuracies change for each method as the number of k neighbours decreases (i.e. 1/k increases). The method that used a random sample from the beginning seemed to have displayed a stronger fluctuation in accuracy as the value of 1/k changed. However, as the number of k-neighbours decreased, both modelling methods showed a trend emerging with a peak accuracy between 0.1 and 0.2, with both then showing a slight decline until reaching k = 1.

Section 2.3 – Confusion Between Doodles

Using the value k = 5 to build the knn model, I iterated through each fold and stored the cross tabulated results for each fold's test dataset into one large table that gives an overview of how often the model confused a given doodle with another.

Table 2 - Overall Contingency Table for the KNN model. The columns are the model predictions and the rows are the actual test data labels.

label	banana	cherry	envelope	flower	golfclub	pear	pencil	wineglass	total
banana	337	23	0	14	15	29	74	8	500
cherry	17	339	1	0	6	122	5	10	500
envelope	0	0	489	6	1	3	1	0	500
flower	0	2	2	493	0	2	1	0	500
golfclub	45	14	0	0	345	9	54	33	500
pear	25	153	0	1	3	299	2	17	500
pencil	68	2	1	3	16	8	391	11	500
wineglass	18	10	0	7	24	15	9	417	500

As Table 2 shows, the model had varying accuracy for any given doodle. From here we are able to determine which objects are getting confused with another the most. What is an interesting observation is the confusion rate between the living banana and the non-living pencil/golfclub doodles which have barely any confusion between themselves (Suggesting that Bananas have not 1 but 2 features similar to non-living objects) Though this confusion is tiny compared to the confusion between the living objects Pear and Cherry, which is a little unsurprising if you were to look at the image data since you would notice that some instances of pears strongly resemble cherries and vice versa.

The model was most accurate in identifying Flower doodles (98.6% accuracy) with envelopes coming in close second (97.8%). This does not surprise me given their unique shapes relative to the other doodles which often have similar shapes to each other (i.e. pears and cherries or pencils, golfclubs and bananas).

Section 3 – Working with Decision Tree Models

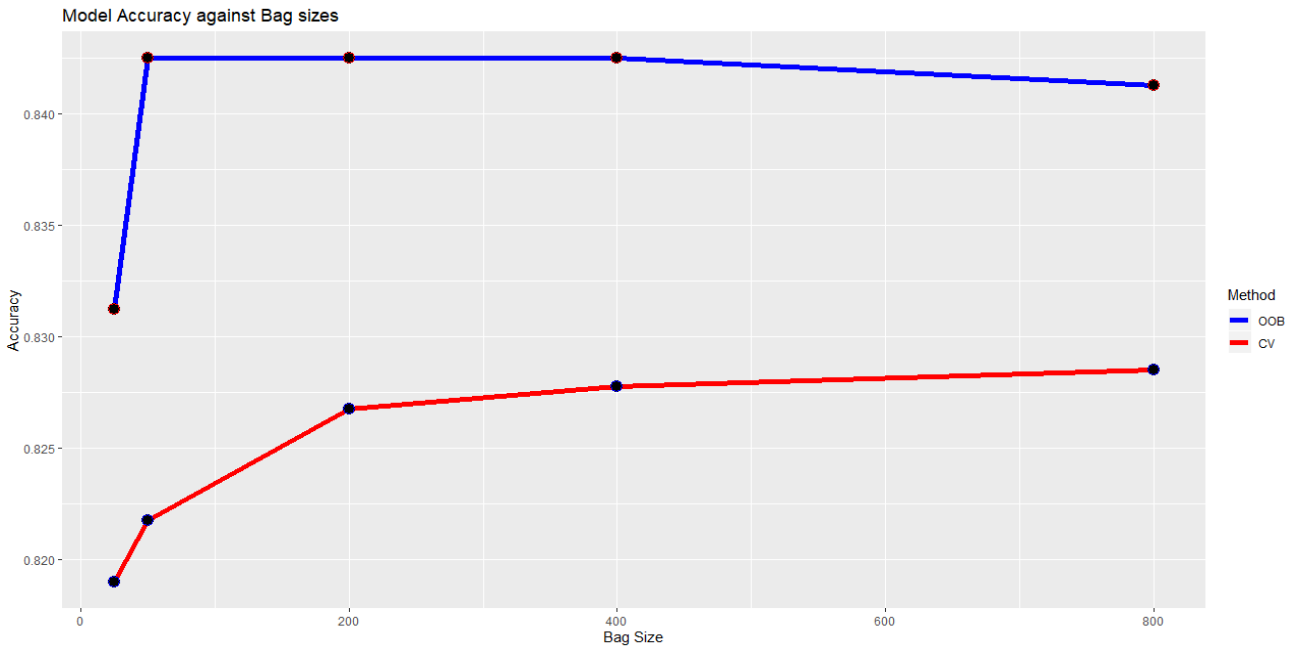
This section will evaluate the accuracy of decision trees and random forests, comparing the accuracies of both 5-fold cross validation and out-of-bag estimation as well as investigating the results of constructing the same random forest 20 times and evaluating the performance of a model once a feature has been removed.

Section 3.1 – Cross-Validated Bagging versus Out-Of-Bag Estimation Bagging

For this subsection, I had to spend a considerable amount of time looking on the web for examples of decision tree bagging in R. The ipred library has a built-in bagging function which seemed easy to use but had several parameters which I didn't quite understand. However, giving the thought that of creating the bagging function at a decent level would be a lot more time consuming than just attempting to use the unfamiliar built-in function to build the models.

bag.size	oob.accuracy	cv.accuracy
25	0.83125	0.819
50	0.8425	0.82175
200	0.8425	0.82675
400	0.8425	0.82775
800	0.84125	0.8285

The results table showed both models performing well.



An interesting observation here is the slight drop in accuracy going from 400 to 800 Bag size for the out-of-bag results, though this change in accuracy isn't monumental. It was also interesting to point out that after that 50 Bag size mark that Out-of-Bag was able to hold a continuous accuracy until the 800 Bag size.

It is interesting to note that Cross-Validation has seem to have taken a different approach compared to OOB. Cross Validation slowly increases its accuracy as the Bag Size increases. Unlike the Out-Of-

Bag observation, at no point does the Cross-validation continue at a steady accuracy or drop off near the end.

Overall it is interesting to see that out-of-bag outperformed the cross-validation estimation for all bag sizes. However, I thought I should expect this due to fundamental differences in how each method builds their respective models.

Section 3.2 – Random Forests using various hyperparameters

This section was the most computationally expensive (in terms of execution time) of all sections of the assignment, since I had to fit models using 16 different values for the number of trees, combined with 4 options of predictor and building a model for each of these combinations using 5-fold cross-validation. The results showed very little variance ranging from 82% to 84.5%, which I found to be a little surprising given the broad range of variables used to create each random forest.

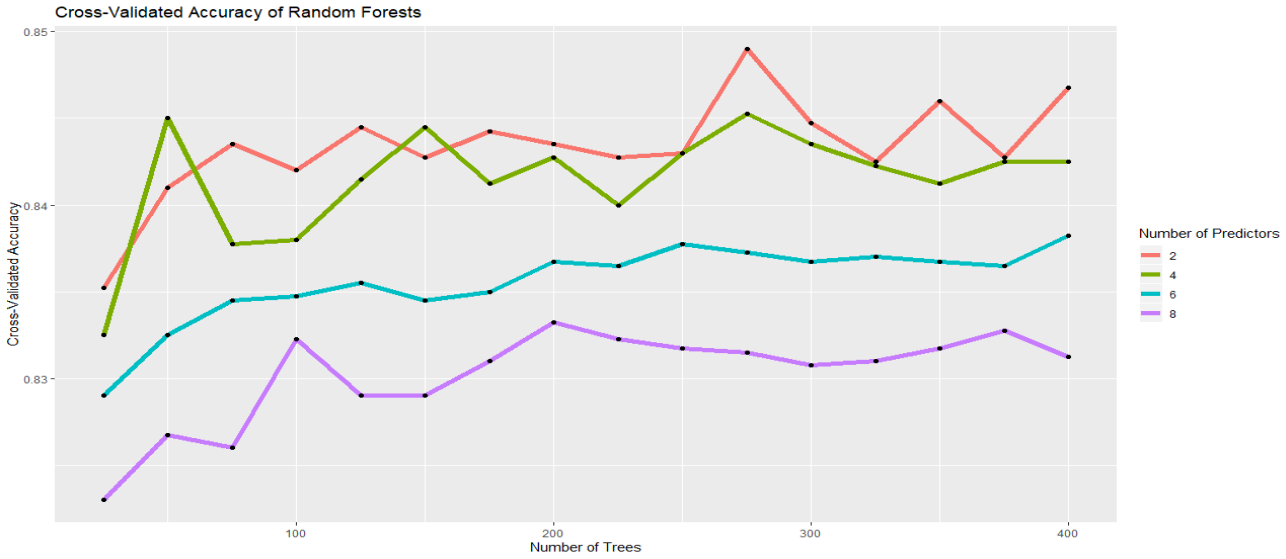


Figure 7 - The accuracy of the random forests using various hyperparameters.

As Figure 7 shows above, as the number of predictors increased, the accuracy of the model continued to worsen. Interestingly, forests built using more predictors experienced more consistent accuracies as the number of trees increased. Take npredictor = 8 for example. It performed at its worst initially (a common theme for all predictors), until it got to 200 Trees and then its accuracy stayed around the 83.4% - 83%, at a range of ~ 0.4%.

Amazingly as well the output that was best was when npredictor = 6. Looking at the graph you could clearly see the accuracy steadily increase with no sudden jumps. It also had an impressive accuracy range after passing the 100 Mark of ~ 0.3%. While it would be desirable to have more consistent accuracy rates, this is not always economically feasible given it requires much more execution time for larger npredictors and ntrees. In hindsight, I would have calculated the cost-accuracy ratio for each configuration to find which would be more optimal.

Section 3.3 – Repeated Cross-Validation on the best model from 3.2

Using the table that I constructed in 3.2, I found the entries which had cross-validated accuracies equal to the highest accuracy of the whole table. The highest cross-validated accuracy was obtained using ntrees = 275 and npredictors = 2, for an accuracy of 84.8%. However, as I mentioned in 3.2, a smaller number of predictors had higher accuracies overall, but would then experience a higher fluctuation as ntrees changed.

The mean of the 20 cross-validated model accuracies was 84%, with a standard deviation of 0.00257 (0.226%). While a 0.5% difference might not seem like much, keep in mind that models using 8 predictors in 3.2 only had accuracies differing by a max of about 0.4%. Which begs the question on how well this model would perform if it had been refitted 100 times.

Section 3.4 – Analysing the Accuracy of the 3.2 model with a feature removed

I found the best way to evaluate which feature should be removed was to check each combination, like previous tasks. I generated each unique 7 feature subset (8 in total) from the original 8 features and built random forests for each using 5-fold cross-validation, with the number of trees and predictors considered equal to the best result obtained in 3.3 (275 and 2 respectively). I then found the difference between the cross-validated accuracy for the given subset and the accuracy obtained using the above parameters.

The results showed that removing any feature reduced the accuracy of the model, with the 'Cols_wth_5' feature resulting in the smallest decrease in accuracy (1.3%) to be exact.

removed.feature	cv.accuracy	percent.difference
neigh5	0.776	-0.073
neigh1	0.787	-0.062
cols_with_5	0.836	-0.013
rows_with_5	0.828	-0.021
span	0.832	-0.017
width	0.832	-0.017
height	0.83	-0.019
nr_pix	0.823	-0.026

The results of the 20 cross-validated models showed a mean of 83.4% accuracy. Compared to the accuracy of the model in 3.3, this is a difference of ~ 1.1% (0.01061). Such a small difference in accuracy might suggest that the feature was not so useful in building the model, since the accuracy difference is rather insignificant. However, the model was restricted to using the best hyperparameter values from 3.3 (ntrees = 275 and npredictors = 2), and so one might ask how the model might perform without the feature for the other hyperparameter values that were used earlier.

Conclusions

As the Results from my Report have shown, in section 1 it is possible to classify living and non-living objects using logistic regression by using only 3 features to build the model. While not fully shown, the idea of adding a fourth feature would help improve the model also existed.

Section 2 demonstrated that KNN models could correctly identify doodles most of the time with a small exception range. It also helped show information that models will often mistake similar (but with deferring classification) doodles for each other quite often, raising the question on how would you go about trying to improve the accuracy of such models without having to radically alter the features that were used to build them.

Section 3 demonstrated how using classification decision tress we could also have some impressive accuracy rates when it comes to identify doodles. It also showed us, how changing the hyperparameter values might alter the accuracy rates of a model, providing more consistent accuracy rates but at the cost of proportionally increasing performances costs.

Overall, I thought Assignment 3 was quite a challenging Tasks, one which required a good bit of online research. It was however quite interesting developing and testing the different classification models to see their accuracy.