**CSC3060 AIDA – Assignment 3**

Adam Coyle

40178464

December 2019 – January 2020

# Introduction

The purpose of this assignment is to use machine learning in order to build classifiers for image data, using data from the previous assignment as well as a large sample of unseen data. 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.

## Section 1.1 – Logistic Regression using the Verticalness feature

Upon loading the feature data into the script, I discovered that for the purpose of this task, I needed to classify each of the observations as either “living” or “non-living”. So, I wrote a function that iterated through each observation, evaluating 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 1. Likewise, for those observations which had labels belonging to non-living things (envelope, golfclub, pencil, wineglass), the classification would be 0. These classifications were made under the assumption that living things were represented by a value of 1 and non-living things by a value of 0.

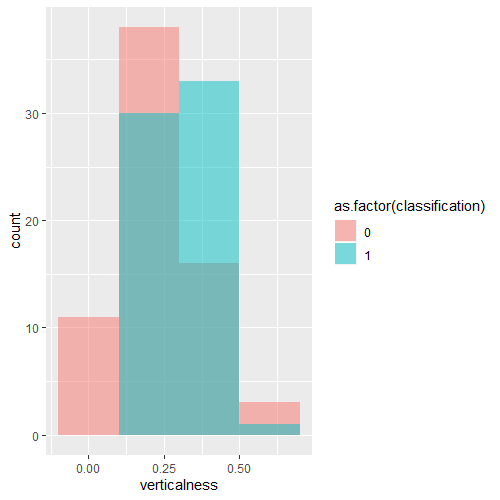
With all 160 observations classified as living or non-living, I was able to start building the model. I shuffled a sample of the original dataset and assigned the first 80% to the training dataset and the remaining 20% to the test dataset. The histogram for the ‘verticalness’ feature shows a significant overlap for both classifications between the values X and Y (roughly). From a simple assessment of these results one might already be able to suggest that the ‘verticalness’ feature is not a good predictor of whether a doodle belongs to either the living or non-living category.

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

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

A screen shot of a social media post

Description automatically generatedThe model was built with the training dataset and the results were close to what I had expected. The model can never truly classify an

Figure 2 - The results table of the logistic regression

## Section 1.2 – Building a classifier using the model from 1.1

Finding a suitable cut-off value for p to provide the best accuracy for the model was difficult, since there was no clear distinction between living and non-living for a given verticalness value (except for the range 0.5 ~ 0.14, to which the model was still not certain of the classification). Rather than guess the best p-value, I decided to check all possible values of p from 0.01 to 0.99 (inclusive) in increments of 0.01. The table of results (see ‘1.2\_p\_accuracy.csv’ for full table) showed that a p value of 0.46 had the highest classification accuracy of 68.1% (3 s.f)

## Section 1.3

I felt that assessing the feature data from assignment 2 (sample statistics, histograms etc.) would be detrimental to finding the absolute best three features on which to build a classifier. Initial attempts at this task (choosing the values that I thought were best) showed somewhat decent classification accuracy. Initially I chose to use the three features which had the least skew in the data, but after seeing the results fall in the range 60% ~ 70%, I began to wonder how would find the features with the highest accuracy.

I discovered the combn() function could return all the unique subsets of a given vector, which in the case of the feature data, was 1140. From there it was a similar model building procedure as before, except I built models for all 1140 feature combinations using 5-fold cross-validation and testing every p value between 0.01 and 0.99 for accuracy. After some 20 – 30 minutes, the code finished running and I had a table of ~11400 entries (see ‘combotable.csv’ for full results). The initial results showed that the combination of span, cols\_with\_5 and neigh5 had the highest cross-validated classification accuracy for the data.

## A close up of a white wall Description automatically generatedSection 1.4

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 3 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.

Figure 3 - The biomial distribution of a random classification model.

## Section 1.5

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.

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

|  |  |  |  |
| --- | --- | --- | --- |
| label | 0 (non-living) | 1 (living) | total |
| banana | 0 | 20 | 20 |
| cherry | 0 | 20 | 20 |
| envelope | 17 | 3 | 20 |
| flower | 0 | 20 | 20 |
| golfclub | 20 | 0 | 20 |
| pear | 1 | 19 | 20 |
| pencil | 20 | 0 | 20 |
| wineglass | 18 | 2 | 20 |

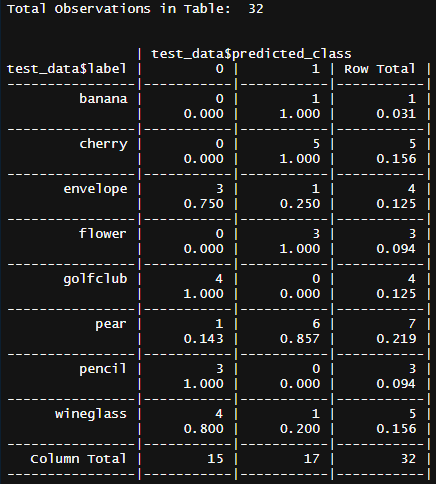
The results table above shows clearly the impressive accuracy of the model; all but 1 observation of the pear doodles from the entire sample of living things, was classified incorrectly. This is in comparison to the 5 misclassifications among the non-living objects (2 wineglasses and 3 envelopes were 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 pear objects accounted for 22% of that fold’s observations, in comparison to what should be around 12.5%.

Figure 4 - The cross table for the first fold of validation items and the models predictions.

Determining which features might improve the accuracy of the model in 1.3 was interesting, as I initially believed that the accuracy of 96%, I observed in 1.3 was as high as it might go. However, I realised 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, since I would need to find a feature which had distinct values for living and non-living objects. The height 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.

Although not explicitly required in the task, further model testing with a fourth feature from the remaining ones showed that height did in fact yield the highest improvement in accuracy (up to 97.5%).

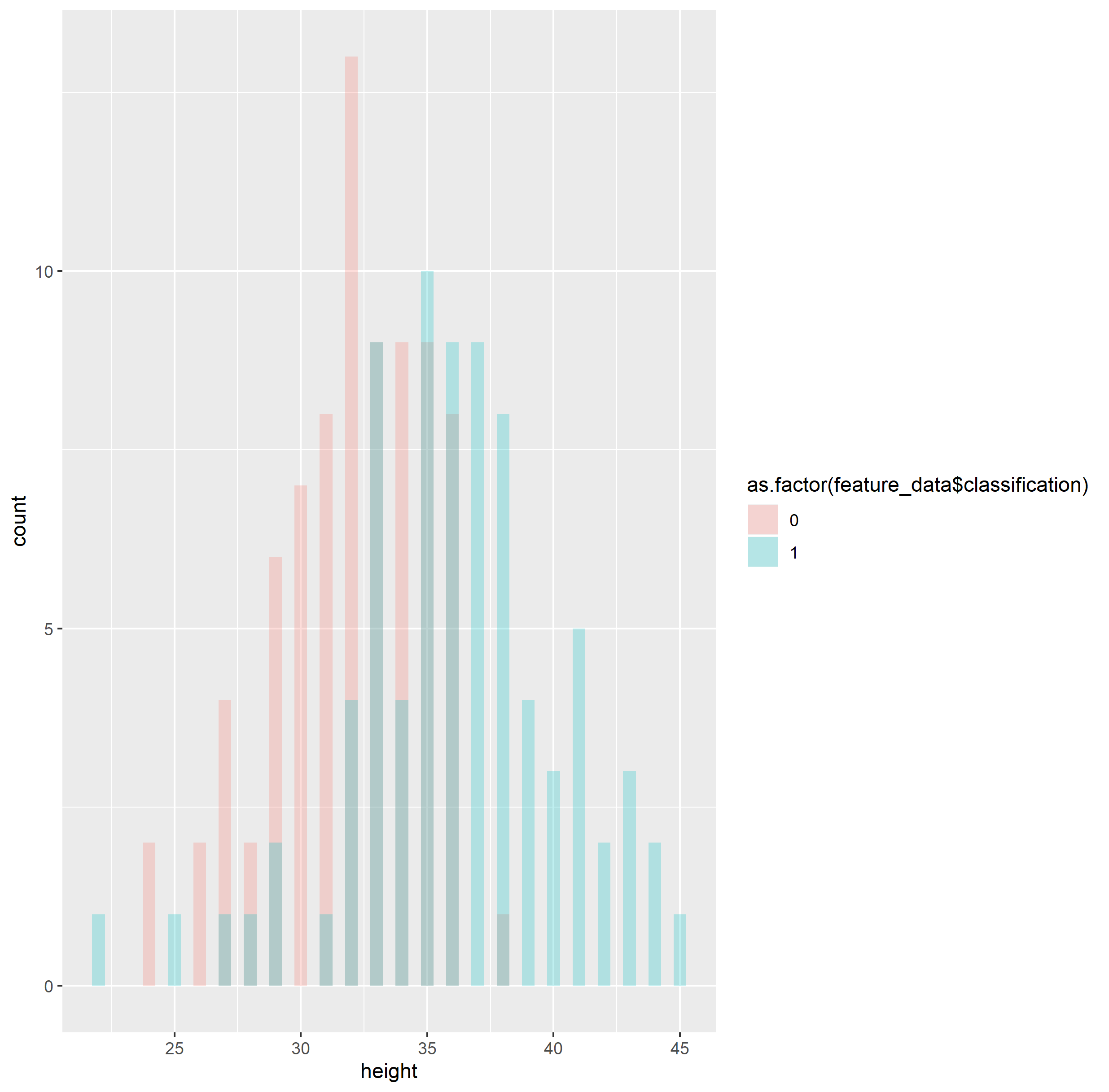


Figure 5 - The histogram for the height feature. Notice the distinction between classes on each side, with a modest overlap in the centre

# 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

Using the supplied training data of 4000 items, I decided to create training and test sets using an 80/20 split. 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 between 24% and 26%, and believed it 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. Once this mistake was rectified, I saw accuracies between 70% and 77.5%, which made much more sense given that the model included the best predictor features (for logistic regression) from 1.3 and 1.5.

## Section 2.2

In my initial attempts to complete this section, I made the same mistake as 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, that was 70.7% up to 78.5% (3 sig. figures).A picture containing kite, map, photo

Description automatically generated

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 much trial, error and frustration I discovered 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 initially displayed stronger fluctuations 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 around 0.2 (5 neighbours), and then showed a slight decline until reaching k = 1. Once the number of neighbours considered drops below 25, the cross-validated method becomes the dominant method with higher accuracies until reach k = 1.

## Section 2.3

I observed that the highest cross-validated accuracy from 2.2 (or lowest error rate in the case of my table), was obtained using a k value of 5. I performed 5-fold cross validation on the sample again using said value, and after building the model for each fold, I cross-tabulated the validation set labels with the model predictions. From observation of the results I thought there was something very wrong with my accuracy results for the previous two sections; instead of an error rate of 74% - 76%, the model was roughly **accurate** in each fold for those values.

# Section 3

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

## Section 3.1

## Section 3.2

## Section 3.3

## Section 3.4

# Conclusions