CSC3060 AIDA – Assignment 2

Donal Mallon

40154387

January 11th 2018

Introduction

The aim of this assignment is to attempt to classify handwritten symbols using the twenty features developed in Assignment 1. The classification would classify an inputted symbol as one of the 21 symbol categories which include the first 7 digits, the first 7 numbers and 7 common math symbols. To do this I will fit different classifiers to training data sets of images and evaluate and train these models in an attempt to predict the symbol category of an unseen image. Throughout my project I have programmed in the language R through the development environment of R Studio.

Section 1

1.1

Looking back at Assignment 1 Section 3.8 I could deduce that the most significant feature for differentiating between digits and letters is the feature 'rows_with_1' (number of rows of pixels with only one pixel) because when I performed a t-test to compare the means between values for each feature for letters and digits this feature resulted in the smallest p value. The resulting p value for each of the t-tests are displayed below in Fig. 1.1.

```
> digit_letter_tt
      nr_pix    height     width  tallness  rows_with_1 cols_with_1 rows_with_5plus cols_with_5plus  one_neigh three_plus_neigh
1 0.005034004 0.4665697 0.8119077 0.4605976 5.898752e-07  0.01809153       0.6486667    0.0002025457 0.02693349        0.2983924
  none_below none_above  none_before   none_after nr_regions   nr_eyes       r5_c5   bd_decimal  r_max_pixel c_max_pixel
1 0.03070302 0.04870927 1.820807e-05 0.0001041142  0.2699283 0.8482283 0.002930145 2.709653e-06 3.971762e-06 0.001640662
>
```

Fig 1.1

The features file from Assignment 1 ("40154387_features.csv") consisted of the feature values calculated from 8 training images for each of the 21 symbol categories. For this section I performed binary classification using just the digits and letter training data in order to predict if a symbol was a digit or a letter.

I began by first visualising the count for each value of rows_with_1 by type of symbol (digit/letter) based on the data from the features file created in Assignment 1. From the histogram produced (Fig. 1.2) you can see a significant difference between digits and letters for the rows_with_1 value and that generally digits have a greater value for rows_with_1.

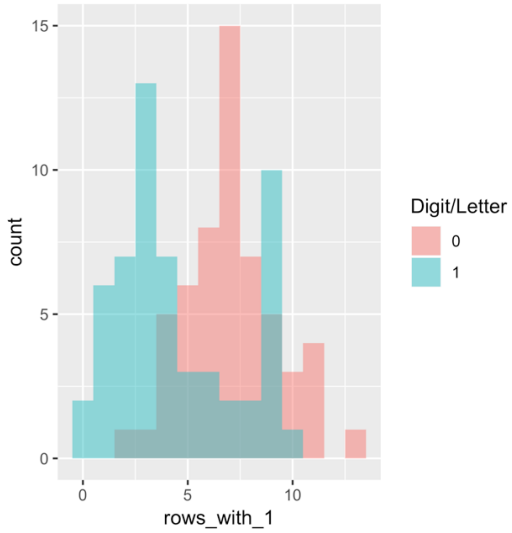

Fig 1.2

I then produced a logistic regression model which would classify an input symbol based on the rows_with_1 variable. In Fig. 1.3 you can see the curve of the logistic regression model plotted along with the training data along with the decision boundary for differentiating between digit and letter.

When the model is then fitted back on the training data, the model predicts an inputted variable as either digit or letter with 72.3% accuracy, correctly predicting 81 of the 112 symbols in the data and incorrectly predicting 31.

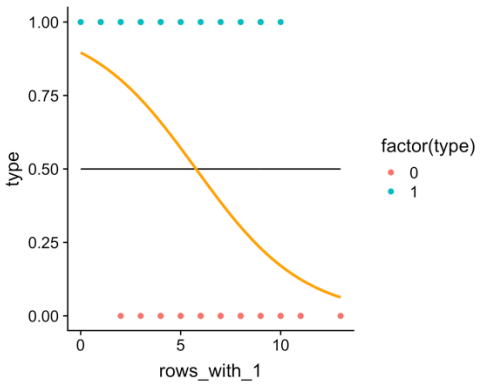

Fig 1.3

1.2

Next I created a logistic regression model using the first 8 features from the same features data as before. From the summary of the fit I could deduce that only two features, 'rows_with_1' and 'cols_with_5plus', were significant at predicting whether a symbol was a digit or letter at at 95% confidence interval. Overall this model had an accuracy of 0.750 which is only marginally more accurate than the single feature linear regression using just the rows_with_1 feature in Section 1.1 which had an accuracy of 0.723

1.3

To check the validity of both the previous models I performed 7-fold cross-validation on the models. The 7-fold cross-validation of the first model, the single feature logistic regression model, resulted in a cross-validated accuracy of 0.7857143. The 7 fold cross-validation of the second model, the eight feature logistic regression model, resulted in a cross-validated accuracy of 0.6964286. Although the eight feature model had an accuracy higher than that of the single feature model when the model was fitted back on the training data, the eight feature model had a significantly less cross-validated accuracy. This would suggest that the single feature model is a more robust model and is more valid for predicting new items drawn from the same population and that the eight feature model is over complicated and therefore over fits to the training data.

1.4

A trivial model that predicts digit every time will have an accuracy of 0.5 as will a random model that predicts letter 50% of the time and digit 50% of the time. All my accuracies for the four examples were above 0.5 and therefore I wanted to discover if these accuracies were significantly greater than 0.5. I wanted to check if the alternative hypothesis that that the accuracies obtained from before are significantly greater than 0.5 was true. I tested the null hypothesis that each of the accuracies were equal to 0.5 at a 95% confidence interval in the binomial distribution. The p values obtained were 4.661292e-07, 1.790152e-08, 1.189779e-10 and 8.178429e-06 and therefore for each of the four accuracies I could reject the null hypothesis and accept the alternative hypothesis at a 95% confidence interval and could therefore conclude that my models were significantly more accurate than trivial models.

1.5

After fitting the 8 feature logistic regression model to the training data I then produced a bar plot to show how many times each digit was incorrectly classified as a letter (Fig. 1.4) and another to show how many times each letter was incorrectly classified as a digit (Fig. 1.5). Interestingly the digit '6' was incorrectly classified as a letter more often than not and similarly letters 'c' and 'f' were incorrectly classified more often than not.

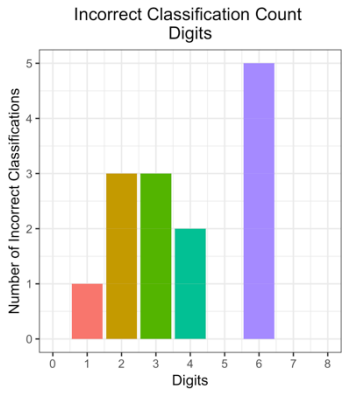

Fig 1.4

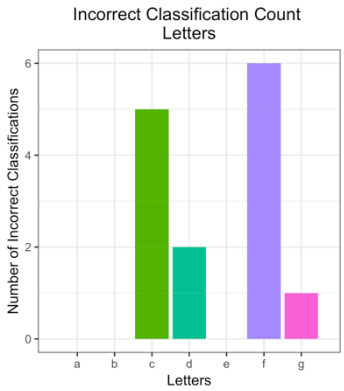

Fig 1.5

Section 2

For this section I was tasked with performing classification with respect to all 21 symbol categories, containing 7 digits, 7 letters and 7 maths symbols. The dataset I would use to train my model would be a set of features of 420 training items for each of the 21 symbols.

Section 2.1

I performed a k-nearest-neighbour classification using the first 10 standardised features from the training dataset for multiple values of k. I then produced a graph showing the accuracy of each the models at each value of k (Fig. 2.1). As expected the value of k=1 almost fits the data perfectly. Accuracy of model on training data then decreases at a steep gradient up to k=9 at which the gradient of the slope begins to ease off.

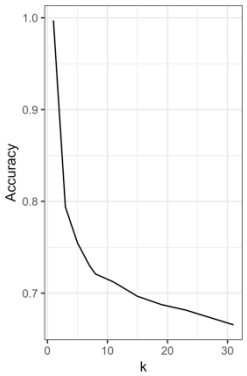

Fig 2.1

Section 2.2

I then performed the same k-nearest-neighbour classification as previous but this time using a 5-fold cross-validation in order to produce a valid accuracy for each value of k. I then plotted a graph of of the cross-validated accuracy for each value of of k (Fig 2.2). The cross-validated accuracy generally increases up until the two peak values of k=9 and k=15 and then accuracy begins to decrease as k is increased. The crossed-validated accuracy is at its highest value of 0.6332200 when k=15, therefore 15 is the optimal value to use for k in a model for predicting the correct labels for test data from the same population.

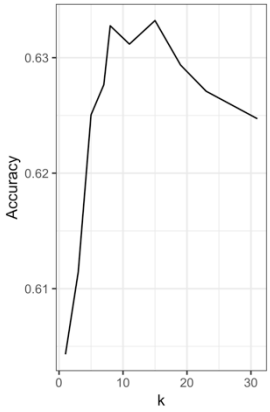

Fig 2.2

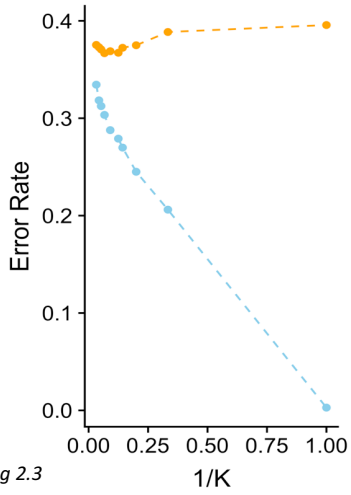

Fig 2.3

Figure 2.3 shows the k-nearest-neighbour classification error rate over the training data in blue and the cross-validated error-rate for each value of 1/k. As the level of flexibility of the the model increases error rate of the classification over the training data decreases and the cross validated accuracy also decreases to a trough and then increases. The two lines for the error rates never converge suggesting that even at a high value of k the model over fits the training data.

Section 2.3

Based on my cross-validated results a value k=15 would result in the most accurate k-nearest-neighbour model using the first 10 features of the training data. The graphs below show that based on each inputted symbol which label does the model predict and how frequently does it predict this label out of the 420 training items for each of the 21 symbols. I have excluded all symbols which were predicted less than 5% of the time for ease of visualisation. For example, when the value '1' was inputted, the model predicted 1 nearly all of the time but confused it as the letter 'f' quite a few times.

Notable confusions/miss predictions:

- 3 is not well predicted and is often confused for other numbers and letters especially 5.
- 6 is also not well predicted and is confused with b,d,g,e often.
- b and d are often confused with each other under the model
- Math symbols < and > are often confused with each other as are ≥ and ≤ but the model can tell each apart more often than not.

It is apparent from the graphs that the pair of digits that are most often confused are 3 and 5.

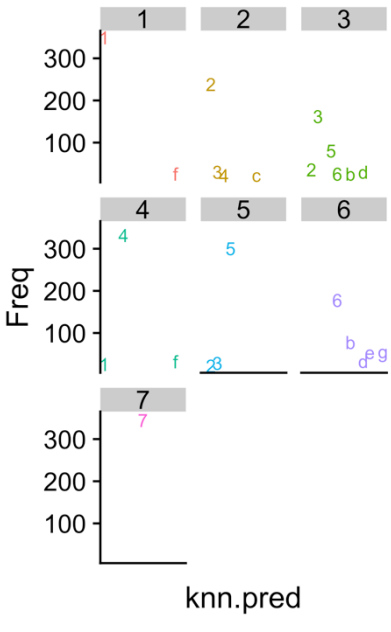

Fig 2.4

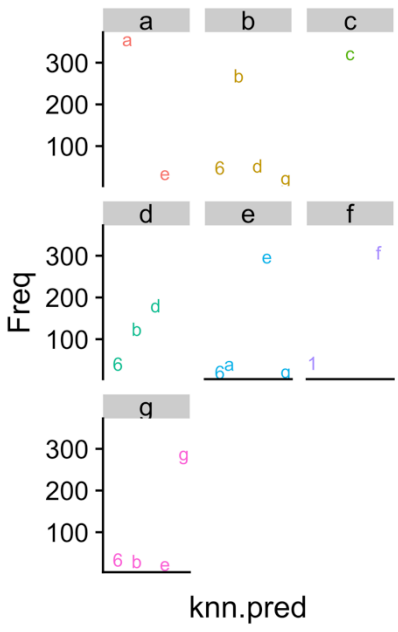

Fig 2.5

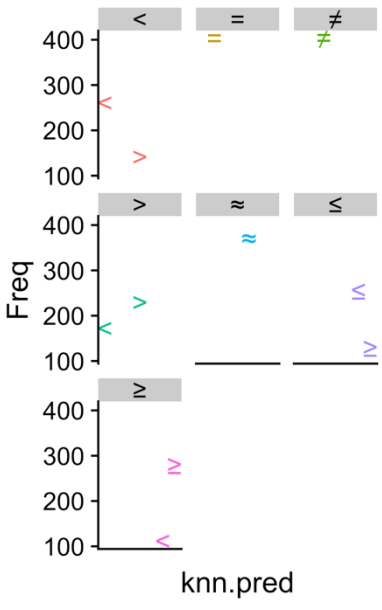

Fig 2.6

Section 3

For this section I was again tasked with the classification of the 21 symbols with the aim of predicting a symbol based on a handwritten test symbol but this time using decision trees and random forests.

Section 3.1

I was tasked with performing classification with respect to the 21 symbol categories using the bagging of decision trees. I could optimise the classification tree by using 5-fold cross-validation at each of the values for number of bags to discover which number of bags resulted in the highest cross-validated accuracy. From the graph of cross-validated accuracy for each number of bags (nbag) in Fig. 3.1 the accuracy of the model appears to increase with number of bags with the highest number of bags of 400 resulting in the highest accuracy of 0.8277778. However, accuracy fluctuates as number of bags increases and differences between accuracies are very small, when nbag=50 accuracy=0.8266440 which is only 0.0011338 of a difference from nbag=400 . Therefore, a more efficient and parsimonious tree bagging model would be to use the nbag=50 when classifying symbol categories.

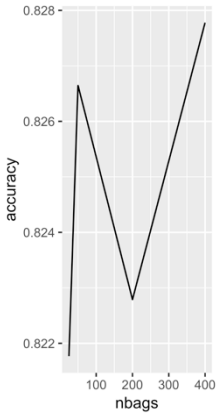

Fig 3.1

Section 3.2

For this section I wanted to create a random forest model for classification of the 21 symbol categories. I performed a 5 fold cross-validation grid search to gain an accuracy for each value of number of trees and number of predictors considered at each node. The cross-validated accuracy of the random forest model for each value of number of trees and number of predictor features considered at each node is displayed in Fig. 3.2.

From the graph you can clearly see that only using two features at each node results in a significantly less accurate model. Using 8 features at each node generally appears to give less accurate predictions based on 5-fold cross validation. Using 4 or 6 features at each node generally gives very similar accuracy values at each node of tree suggesting there is no information gained by using 6 nodes instead of 4. Also from the graph you can see that an increase in number of trees increases the cross-validated accuracy up to a point between 100-150 trees and then generally levels off with only a gentle positive slope at each of the values for nfeatures. This suggests that a more parsimonious model would use a ntrees value of between 100-150.

The combination of number of trees (ntrees) and number of predictors(nfeatures) at each node that produces the highest cross-validated accuracy after 5-fold cross-validation is ntrees= 330 and nfeatures = 4 with an accuracy of 0.8534014.

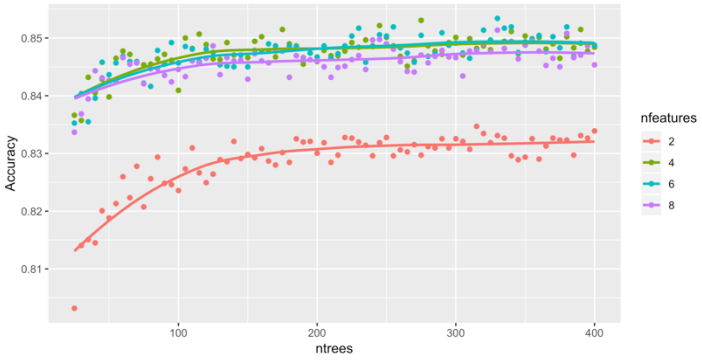

Fig 3.2

Section 4

For this Section I loaded in the feature data as before but this time I wanted to calculate the values for the three custom features I developed in Assignment 1 so that these features could also be utilised to build an efficient model. To do this I looped through the "*pixel.csv" and performed calculations as done in Assignment 1 to gain values for the custom features for each of these tests items.

As suggested by the evaluating the cross-validated accuracies of the multiple random forest models in Section 3.2, adding more predictor variables would result in little to no information gain and even a reduction in cross-validated accuracy of the model. Therefore, I used Principle Components Analysis (PCA) to reduce the dimensionality of the training data to produce the most parsimonious model. I first wanted to visualise the cumulative proportion of variance that was explained by each number of principle components in order to see, as a rule of thumb, how many components it would take to explain at least 80% percent of the variance. As you can see from the graph in Fig. 4.1, it would take 5 principle components to capture at least 70% percent of the variance.

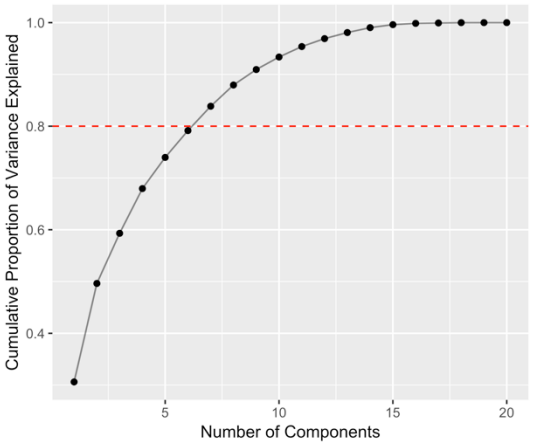

Fig 4.1

Now that I had visualised the explained variance of the PCA I could simply calculate the pre processing parameters for my random forests model using the preProcess() function, which would standardize the training data and use PCA on the data at a given threshold of 0.7. I could then include the pre-processed data in my final random forests model.

I would then need to decide on an ideal number of trees and number of features at each node of the random forests model. To do this I would perform a grid search on the pre-processed data using the two hyper-parameters of number of trees and number of predictors considered at each node and report the accuracy of each using 5-fold cross validation. I considered the number of trees as between 100-150 because, as suggested in Section 3.2, these values of trees would produce a more parsimonious model as adding more trees was unnecessary and did not result in a significant increase in accuracy. I looked at values for number of components considered at each node as between 2-5 to include all permutations of the five principle components.

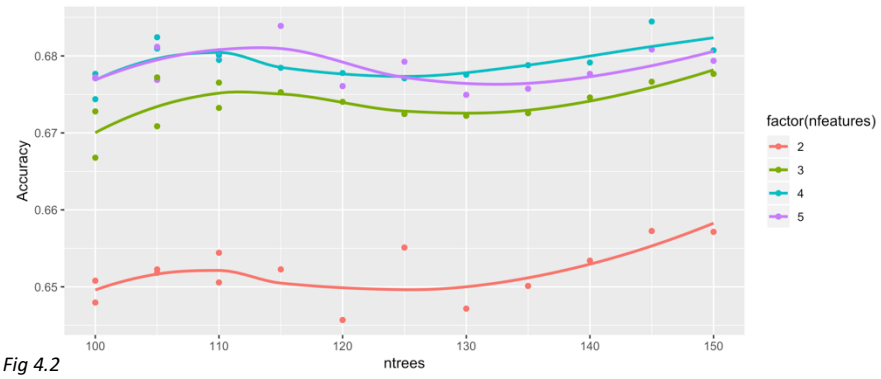

Fig 4.2

From the graph you can see that considering 4 or 5 features at each node resulted in the highest cross-validated accuracies. The combination of the two hyper-parameters that resulted in the highest cross-validated accuracy was ntree= 145 and nfeatures= 4 with an accuracy of 0.68932. This model is therefore less accurate on the training than the random forests model produced in Section 3.2 due to the dimensionality reduction performed in pre-processing using PCA, however, I believe this model provides a sufficient level of explanation with as few predictor features as possible resulting in a more parsimonious model.

Subsequently I could then create my final model which would be a random forests model, pre processed using PCA, with the number of trees being 145 and the number of components used at each node being 4. I loaded in the test features data, calculated my own custom features on the test pixel data and added these features to the data set as was done for for the training dataset. I then created a data frame which included

the index of the test item along with the label that my model predicted for that item and saved this information to the file “40154387_section4_predictions.csv”.

Conclusions

In Section 1 of this assignment I used logistic regression in an attempt to differentiate between letters and digits based on inputted features data. Throughout the rest of the assignment I analysed and developed models based on a training data set of handwritten symbols with the aim of classifying an inputted symbol as one of 21 symbol categories. The final model was decided as a random forest model based on cross-validation accuracy results which was then used to predict the label for the unknown test data.