**CSC3060 AIDA – Assignment 3**

##### Dewei Liu

##### 40216004

##### Tuesday, 24 December 2019

# Contents

[Contents 2](#_Toc28102422)

[Introduction 3](#_Toc28102423)

[Section 1 4](#_Toc28102424)

[Overview 4](#_Toc28102425)

[R script 4](#_Toc28102426)

[Task 1.1 6](#_Toc28102427)

[Task 1.2 9](#_Toc28102428)

[Task 1.3 12](#_Toc28102429)

[Task 1.4 15](#_Toc28102430)

[Task 1.5 16](#_Toc28102431)

[Section 2 18](#_Toc28102432)

[Overview 18](#_Toc28102433)

[R script 18](#_Toc28102434)

[Task 2.1 20](#_Toc28102435)

[Task 2.2 22](#_Toc28102436)

[Task 2.3 25](#_Toc28102437)

[Section 3 28](#_Toc28102438)

[Task 3.1 29](#_Toc28102439)

[Task 3.2 32](#_Toc28102440)

[Task 3.3 36](#_Toc28102441)

[Task 3.4 38](#_Toc28102442)

[Conclusions 41](#_Toc28102443)

[References 42](#_Toc28102444)

# Introduction

In this document, please replace [StudentNumber], [FNAME], etc, with the appropriate values. The FirstNme and LastName on the report should match your first name and lastname as it appears on QOL and QSIS.

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

# Section 1

## Overview

In this section, feature data from assignment 2 (Liu, 2019) will be used for analysis. It contains data of 20 features regarding 80 living objects and 80 non-living objects. Hypothesis test, building classifier with different features will be performed in this section.

The description of each task in section 1 includes objective, assumption (if applicable), reasoning, implementation and result.

## R script

### Description

The tasks were implemented in R (version 3.6).

The R script for this section is divided into six parts.

The first part is the function setups which defines some basic functions and loads essential R libraries.

Each of the rest five parts represents a task in section 1, representatively. At the beginning of each task, the function *start\_task <- function(task\_number)* is called, where data are loaded and environment variables (e.g. feature data, seed value for randomness, output directory) are set up. At the end of each task, the function *finish\_task <- function(task\_number, reserved\_varialbes = c())* is called, where environment variables are destroyed except those stated in the parameter *reserved\_varialbes*. This ensures that each task is isolated to others, so that the implementation of any task will not affect the implementation and output of its following tasks (especially for procedures with elements of randomness). It also helps the evaluation of the R script and this report easier.

### Usage

#### Prerequisites

1. R (<https://www.r-project.org/>) is installed on your machine
2. R libraries (Team, 2019) yaml, e1071, caret and ggplot2 are installed.

#### Configuration

A YAML configuration file is placed at , which includes the default configuration.

#### Execution

The current working directory should be the same as the R script

Execute the following command.

#### Outputs

All outputs (e.g. figures, tables) of the execution will be saved at where it contains subfolders named as (e.g. ).

## Task 1.1

### Objective

The objective of this task is to differentiate living and non-living things using the feature verticalness.

### Assumption

The critical p-value is set as

### Reasoning

Logistic Regression (LR) will be used as a method in the analysis. LR uses the Sigmoid function (Equation 1), and as a result, it produces values between 0 and 1 (Chandrayan, 2019).

Equation Logistic Regression (Devereux, 2019)

Since the objects are needed to be classifies into two classes, we can set a cut-off value. If the LR model produces a value which is greater than the cut-off value, the object will be identified as a class. Otherwise, it will be identified as the opposite class.

### Implementation

The data frame with two columns verticalness and living is constructed. The values in the column living are Boolean values indicating if the observation is a living thing.

The data then is fit into the LR models. By interpreting the result of the trained model, we can decide if the feature verticalness is a sufficient feature to differentiate living and non-living things.

### Result

The summary of the data fitted into the model is as Table 1.

verticalness living

Min. :0.07534 Min. :0.0

1st Qu.:0.36631 1st Qu.:0.0

Median :0.50616 Median :0.5

Mean :0.51907 Mean :0.5

3rd Qu.:0.61048 3rd Qu.:1.0

Max. :1.27027 Max. :1.0

Table Summary of verticalness ~ logistic living value data

After fitting the model, the result of the model is as Table 2.

Call:

glm(formula = living ~ verticalness, family = "binomial", data = data)

Deviance Residuals:

Min 1Q Median 3Q Max

-1.23105 -1.16944 0.00095 1.17767 1.19216

Coefficients:

Estimate Std. Error z value Pr(>|z|)

(Intercept) -0.08656 0.37516 -0.231 0.818

verticalness 0.16676 0.65547 0.254 0.799

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 221.81 on 159 degrees of freedom

Residual deviance: 221.74 on 158 degrees of freedom

AIC: 225.74

Number of Fisher Scoring iterations: 3

Table Result of Linear Regression Model

For the Intercept value in the table, the estimate is , which means the model predicts the value of living is given the verticalness value is 0. The z-score is , which is calculated as . It shows the estimate is standard error away from 0. According to the z-score and the degrees of freedom value, p-value of this variable is calculated to be which is larger than the critical p-value. We consider rejecting the hypothesis that intercept value is differ from 0.

For the verticalness value in the table, the estimate is , which means if the verticalness value increases by 1 unit, the predicted value of living will be increased by 0.16676 unit. The z-score is , which is calculated as . It shows the estimate is standard error away from 0. According to the z-score and the degrees of freedom value, p-value of this variable is calculated to be which is larger than the critical p-value. Thus, for the hypothesis, we stick on the hypothesis that the slope value of verticalness value is equal to 0.

According to the result of the model, the coefficient of the estimates of the intercept and verticalness is and , respectively, which derives the Equation 2 and Figure 1.

Equation LR Model

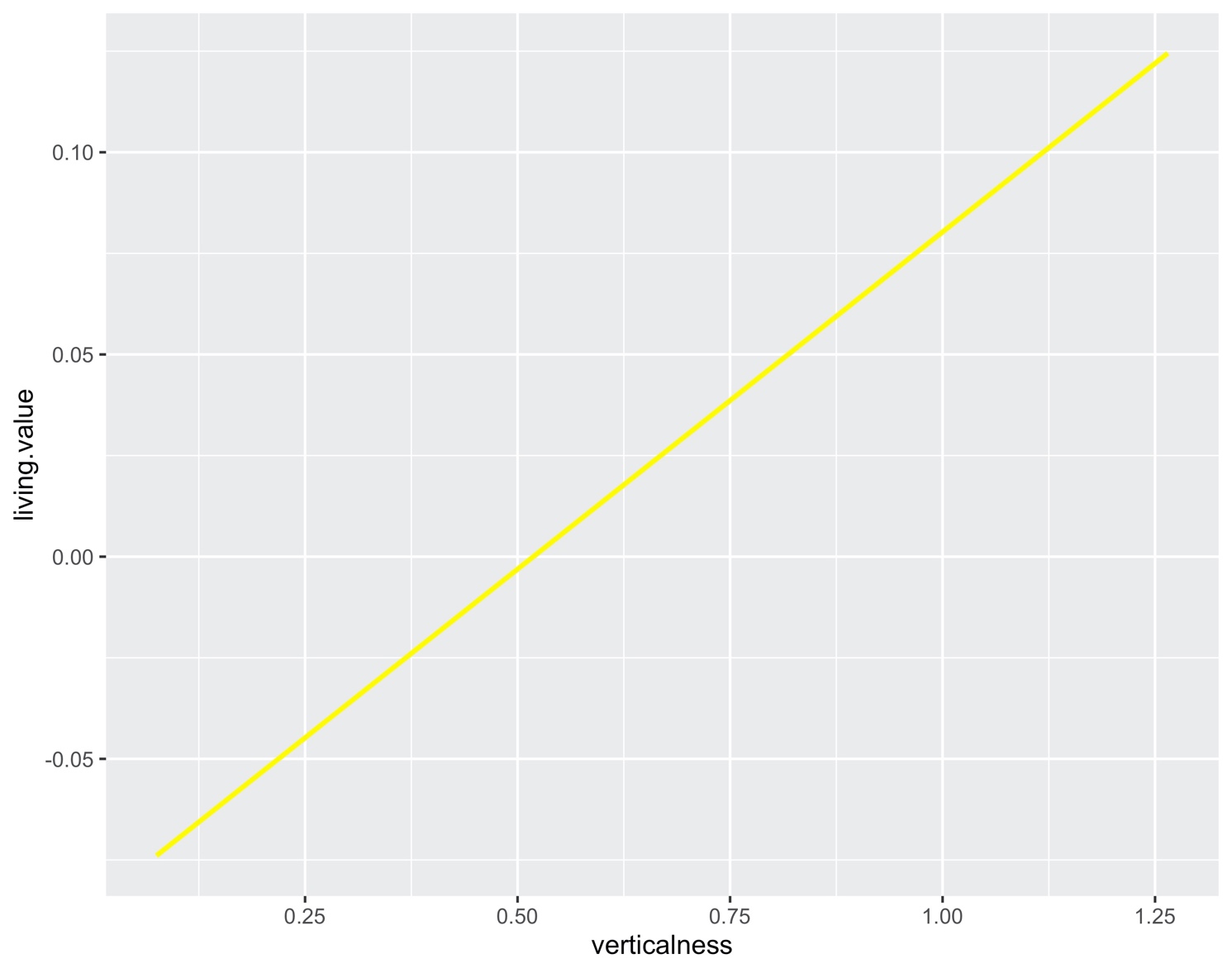


Figure Regression Line living.value ~ verticalness value

## Task 1.2

### Objective

The objective of this task is to create a classifier to differentiate living objects.

### Reasoning

To create the classifier, we need to draw plots to visualise the data, and see how they are distributed according to the verticalness values.

### Implementation

We first draw a histogram to visualise the verticalness distributions of living and non-living objects (Figure 2). We can see there are more living objects with verticalness values between 0.375 and 0.75. However, they do not have clear separation on the feature verticalness.

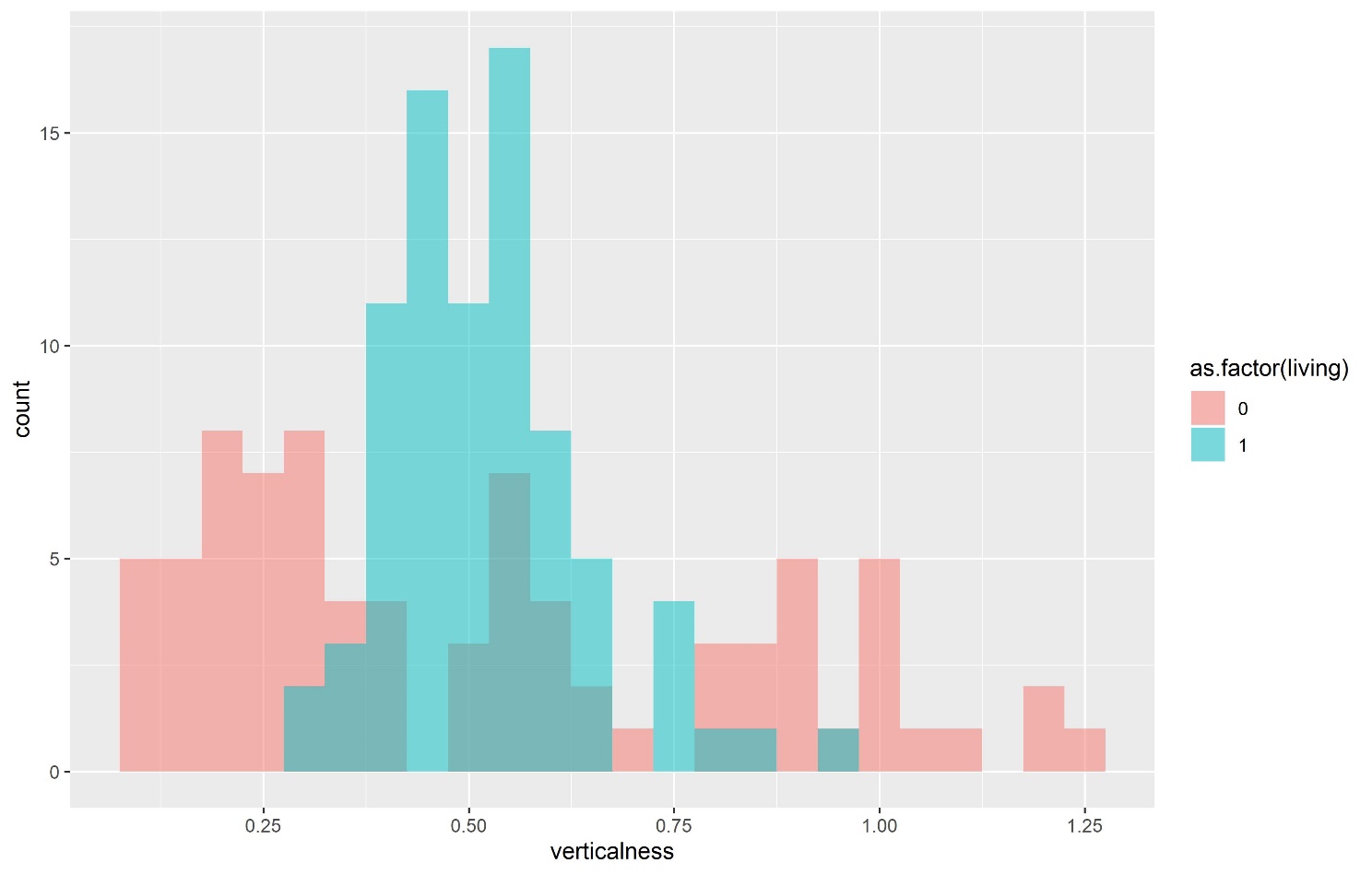


Figure Histogram verticalness distribution

We then plot the training data points and a fitted curve (Figure 3). We can see the slop of the fitted curve is too horizontal, which means there is no strong correlation between these two variables, and it is hard to get valuable information from this figure.

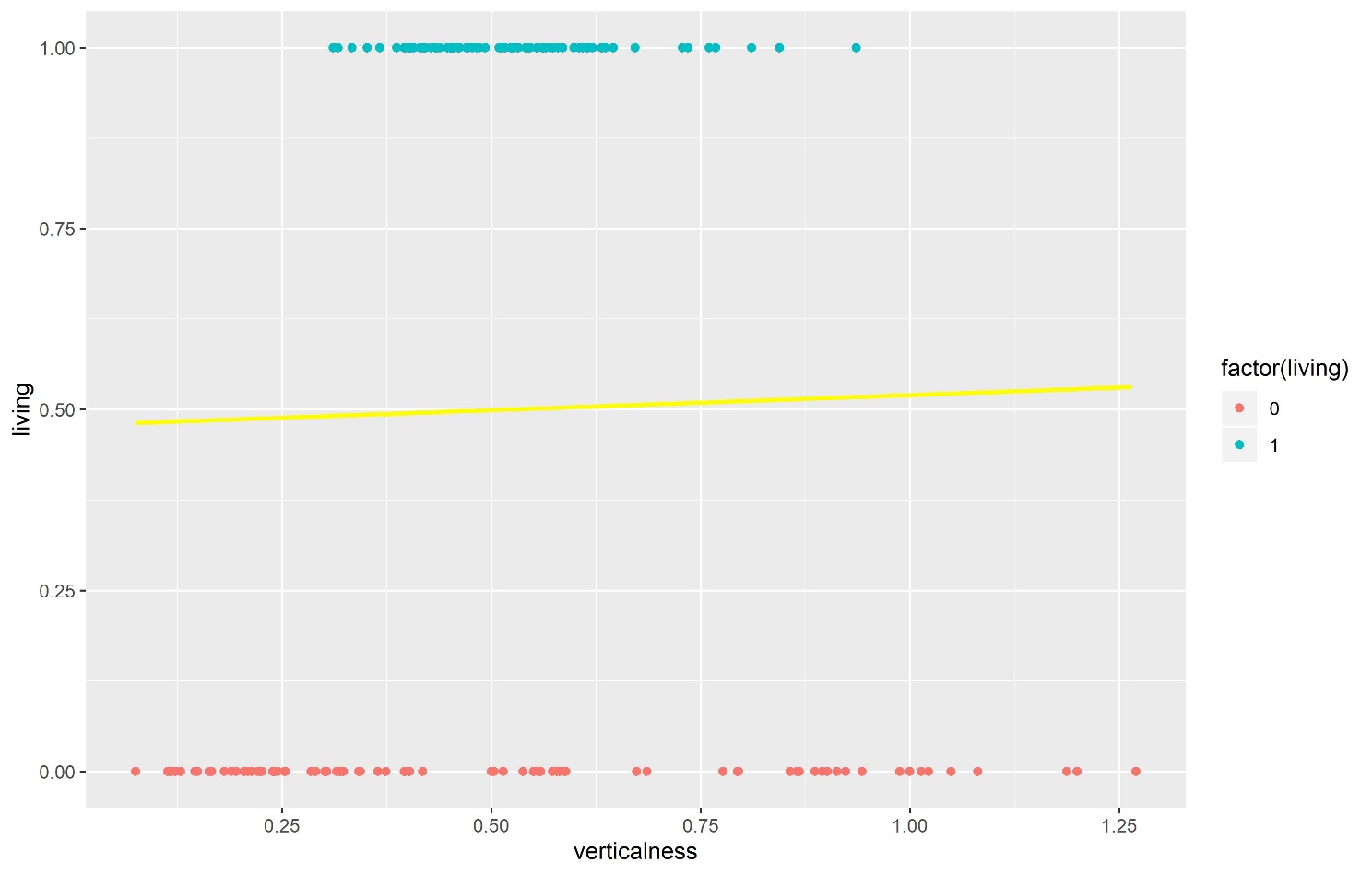


Figure Veticalness ~ living: training data and fitted curve

It is hard to observe in Figure 3, but from Figure 2, we consider observations with verticalness value greater than 0.375 or higher to be living objects. We calculate the cut-off point of the P(X) using the formula.

And we got the cut-off value.

The observations with predicted value higher than the cut-off value are classified as living objects.

### Result

The result of the prediction is Table 3 showing 112 predictions were correct while 48 were incorrect. Thus, we got 70% correctness for this classifier for the data provided.

Mode FALSE TRUE

logical 48 112

Table Summary of Correct Predictions

However, the accuracy of 70% is valid only on the dataset of 160 objects provided. The accuracy does not represent the performance of the model on other objects that were not included in the model.

## Task 1.3

### Objective

The objective of this task is to find three features to build a classifier for living and non-living objects, using logistic regression and cross-fold validation.

### Reasoning

To choose three best features for the prediction, we need to analyse the correlated between each feature and the dependant valuable . We decide to use Backward Elimination of p-value approach (Geeksforgeeks, 2019) to select these three features.

Then we use 5-fold cross validation (Drakos, 2018) to fit the data of these three features into the training model and validate the result.

### Implementation

As we only need three features as training data, we need to drop 17 features from the dataset. First, the whole dataset with 20 features is used to fit a linear regression model, and we the model as Table 4.

Call:

lm(formula = living ~ ., data = data)

Residuals:

Min 1Q Median 3Q Max

-0.48298 -0.12907 -0.03498 0.13070 0.62125

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) 2.525808 0.399713 6.319 3.34e-09 \*\*\*

nr\_pix -0.006279 0.004907 -1.280 0.20274

height 0.031369 0.009815 3.196 0.00173 \*\*

width 0.002565 0.010945 0.234 0.81506

span -0.040148 0.012969 -3.096 0.00237 \*\*

rows\_with\_5 0.007338 0.011688 0.628 0.53116

cols\_with\_5 0.032590 0.013372 2.437 0.01606 \*

neigh1 0.028496 0.025410 1.121 0.26403

neigh5 -0.007424 0.003995 -1.859 0.06521 .

left2tile -0.005431 0.009683 -0.561 0.57576

right2tile 0.024086 0.009498 2.536 0.01232 \*

verticalness -2.232424 0.375564 -5.944 2.13e-08 \*\*\*

top2tile -0.019631 0.009364 -2.096 0.03785 \*

bottom2tile 0.004558 0.008000 0.570 0.56977

horizontalness 0.659220 0.595643 1.107 0.27032

concentration 0.004855 0.003972 1.222 0.22362

crossness -0.009686 0.004555 -2.127 0.03522 \*

nr\_regions -0.350727 0.067510 -5.195 7.13e-07 \*\*\*

nr\_eyes -0.008960 0.019314 -0.464 0.64343

hollowness 0.128303 0.022039 5.822 3.86e-08 \*\*\*

straightness 0.002304 0.006160 0.374 0.70891

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.2255 on 139 degrees of freedom

Multiple R-squared: 0.8232, Adjusted R-squared: 0.7978

F-statistic: 32.36 on 20 and 139 DF, p-value: < 2.2e-16

Table Twenty feature linear model

As we notice, the feature width has the highest p-value. Thus, the feature width is dropped in the first round.

This process is repeated 17 times, where 17 features will be dropped. The final model has three features left (i.e. height, span and hollowness) as Table 5.

Call:

lm(formula = living ~ ., data = data)

Residuals:

Min 1Q Median 3Q Max

-0.78708 -0.24913 -0.03218 0.22489 0.81976

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) 0.460969 0.327675 1.407 0.161

height 0.048335 0.004889 9.887 < 2e-16 \*\*\*

span -0.047716 0.006482 -7.361 9.8e-12 \*\*\*

hollowness 0.143571 0.012947 11.089 < 2e-16 \*\*\*

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.3303 on 156 degrees of freedom

Multiple R-squared: 0.5746, Adjusted R-squared: 0.5664

F-statistic: 70.24 on 3 and 156 DF, p-value: < 2.2e-16

Table Three feature linear model

To perform 5-fold cross validation, data is randomly split into five parts.

For the first fold, the data of parts 2, 3, 4 and 5 is used to fit the logistic regression model. The model then will be used to produce a predicted living value using the data in part 1. Setting cut-off point as 0.5, all objects with predicted living value greater than 0.5 is predicted as living objects, and vice-versa. The predictions are compared with the actually classification, so the correct rate is calculated for this fold.

This process is repeated five times, and the correct rates of these five folds are 0.84375, 0.9375, 0.90625, 0.90625 and 0.84375, respectively.

### Result

The overall correct rate is the average value of all of the above rates which is . It is considered as the good result as the model is correct in 88.75% of times on the dataset of 160 objects provided.

## Task 1.4

### Objective

The objective of this task is to judge if the model is effective compared with a random model.

### Assumption

It is assumed that the value of living satisfies binomial distribution, the signification p-value is 0.05.

### Reasoning

Since the value of living is either TRUE of FALSE, so that it satisfies the binomial distribution. The number of correct predictions of the model, the number of prediction and the correctness rate of a random model can be simulated, so we can perform the above hypothesis.

### Implementation

A random model is simulated that randomly predicts 50% of objects as living objects and the other 50% as non-living objects. The accuracy of this random model is calculated, which is 47.5% based on the dataset is provided.

Given 160 objects in the dataset, our model with accuracy of 88.75% can correctly predict objects.

Assuming , the possibility value (p-value) that we can observe the given correctness or higher is

### Result

The p-value is calculated as 0, which is less than the critical p-value, we reject and accept .

We can conclude that our model is more effective than a random model.

## Task 1.5

### Objective

Beyond the accuracy of the model, more details (i.e. how the model incorrectly classifies objects) will be analysed.

### Reasoning

Confusion Matrix (Dataschool, 2019) will be used for the analysis. It provides a detailed description of the performance of a classifier.

### Implementation

Confusion Matrix function is applied to the cross-validation result of 160 predictions saved in Task 1.3

### Output

The output is as Table 6.

Confusion Matrix and Statistics

Reference

Prediction 0 1

0 75 13

1 5 67

Accuracy : 0.8875

95% CI : (0.828, 0.9319)

No Information Rate : 0.5

P-Value [Acc > NIR] : < 2e-16

Kappa : 0.775

Mcnemar's Test P-Value : 0.09896

Sensitivity : 0.9375

Specificity : 0.8375

Pos Pred Value : 0.8523

Neg Pred Value : 0.9306

Prevalence : 0.5000

Detection Rate : 0.4688

Detection Prevalence : 0.5500

Balanced Accuracy : 0.8875

'Positive' Class : 0

Table Confusion Matrix

(a) How often instances of each of the 4 living thing doodle types are incorrectly classified as “non-living”?

(b) How often instances of each of the 4 non-living thing doodle types are incorrectly classified as “living”?

The chance that the model incorrectly classifies living objects is much higher than that for non-living objects.

Also, it is spotted that if the model classifies an object as a living object, the chance that the object is actually living is higher than the average accuracy. Thus, if we are interested in finding out living objects among a group of objects, the model provides higher value than its accuracy of 88.75%.

# Section 2

## Overview

The description of each task in section 1 includes objective, assumption (if applicable), reasoning, implementation and result.

## R script

### Description

The tasks were implemented in R (version 3.6).

The R script for this section is divided into six parts.

The first part is the function setups which defines some basic functions and loads essential R libraries.

Each of the rest five parts represents a task in section 1, representatively. At the beginning of each task, the function *start\_task <- function(task\_number)* is called, where data are loaded and environment variables (e.g. feature data, seed value for randomness, output directory) are set up. At the end of each task, the function *finish\_task <- function(task\_number, reserved\_varialbes = c())* is called, where environment variables are destroyed except those stated in the parameter *reserved\_varialbes*. This ensures that each task is isolated to others, so that the implementation of any task will not affect the implementation and output of its following tasks (especially for procedures with elements of randomness). It also helps the evaluation of the R script and this report easier.

### Usage

#### Prerequisites

1. R (<https://www.r-project.org/>) is installed on your machine
2. The following R libraries are installed.
   1. yaml (<https://cran.r-project.org/web/packages/yaml/index.html>)
   2. e1071 (<https://cran.r-project.org/web/packages/e1071/index.html>)
   3. caret (<https://cran.r-project.org/web/packages/caret/index.html>)
   4. ggplot2 (<https://cran.r-project.org/web/packages/ggplot2/index.html>)

#### Execution

The current working directory should be the same as the R script

Execute the following command.

#### Outputs

All outputs (e.g. figures, tables) of the execution will be saved at where it contains subfolders named as (e.g. ).

## Task 2.1

### Objective

The objective of this task is to perform k-nearest-neighbour classification on the training dataset provided using the only first eight features, and test it using the same training data.

### Reasoning

Since the values of eight predictor features and the true label are given for 4000 objects, KNN (Ripley, 2019) can be applied to these data.

### Implementation

First, data are fitted into the KNN model, with .

A list of predicted labels are returned and compared with the true labels, so that the accuracy is calculated for .

The process is repeated for each odd numbers of k between 1 and 59.

### Result

Table 7 was produced which includes the accuracy on the training set for each k.

k = 1 accuracy = 1

k = 3 accuracy = 0.89

k = 5 accuracy = 0.85675

k = 7 accuracy = 0.84375

k = 9 accuracy = 0.82725

k = 11 accuracy = 0.81625

k = 13 accuracy = 0.809

k = 15 accuracy = 0.80225

k = 17 accuracy = 0.796

k = 19 accuracy = 0.79025

k = 21 accuracy = 0.78675

k = 23 accuracy = 0.77975

k = 25 accuracy = 0.77725

k = 27 accuracy = 0.7765

k = 29 accuracy = 0.77325

k = 31 accuracy = 0.77175

k = 33 accuracy = 0.7715

k = 35 accuracy = 0.769

k = 37 accuracy = 0.7685

k = 39 accuracy = 0.76825

k = 41 accuracy = 0.7645

k = 43 accuracy = 0.759

k = 45 accuracy = 0.7565

k = 47 accuracy = 0.75425

k = 49 accuracy = 0.7525

k = 51 accuracy = 0.7485

k = 53 accuracy = 0.74675

k = 55 accuracy = 0.7465

k = 57 accuracy = 0.7415

k = 59 accuracy = 0.7405

Table KNN result

## Task 2.2

### Objective

The objective of this task is to perform k-nearest-neighbour classification using the only first eight features. Cross Validation (Irizarry & Love, 2019) shall be used to evaluate the training result.

### Reasoning

Cross validation is a better way to evaluate the training result than what we did in Task 2.1, because it test the model with dataset that is not used for training (Picard & Cook, 2012).

### Implementation

Number of fold for the cross validation is 5 in this implementation, and five rounds of validation will be performed for each number of k.

First, the data is randomly shuffled and assign fold numbers to each observation. For the first round, we define the data in the first fold is validation data, and fit the data in the rest four fold into the KNN model with

The predictions of the validation set are returned and compared with the true label, so that the accuracy is calculated for the first round of .

For the rest four round of , the accuracies are calculated using the same technique. Then the overall accuracy of is calculated as the average value across all accuracies.

The above process is repeated for each odd numbers of k between 1 and 59.

### Result

Table 8 was produced which includes the overall accuracy on the validation for each k and the k value of the best performance.

k = 1 accuracy = 0.763

k = 3 accuracy = 0.768

k = 5 accuracy = 0.78

k = 7 accuracy = 0.77525

k = 9 accuracy = 0.777

k = 11 accuracy = 0.777

k = 13 accuracy = 0.7705

k = 15 accuracy = 0.76125

k = 17 accuracy = 0.761

k = 19 accuracy = 0.758

k = 21 accuracy = 0.7565

k = 23 accuracy = 0.75225

k = 25 accuracy = 0.75375

k = 27 accuracy = 0.7475

k = 29 accuracy = 0.74825

k = 31 accuracy = 0.74625

k = 33 accuracy = 0.74575

k = 35 accuracy = 0.74425

k = 37 accuracy = 0.732

k = 39 accuracy = 0.7385

k = 41 accuracy = 0.73525

k = 43 accuracy = 0.73475

k = 45 accuracy = 0.7315

k = 47 accuracy = 0.72175

k = 49 accuracy = 0.7265

k = 51 accuracy = 0.72375

k = 53 accuracy = 0.72125

k = 55 accuracy = 0.7205

k = 57 accuracy = 0.7205

k = 59 accuracy = 0.7185

Best Performance: k = 5 accuracy = 0.78

Table Cross Validation Result

According to the results of Task 2.1 and Task 2.2, Table 9 shows the summary of the error rates in the result.

k training.errors cross.validated.errors inversed.k

Min. : 1.0 Min. :0.0000 Min. :0.2200 Min. :0.01695

1st Qu.:15.5 1st Qu.:0.1993 1st Qu.:0.2388 1st Qu.:0.02248

Median :30.0 Median :0.2275 Median :0.2531 Median :0.03337

Mean :30.0 Mean :0.2105 Mean :0.2530 Mean :0.08941

3rd Qu.:44.5 3rd Qu.:0.2429 3rd Qu.:0.2684 3rd Qu.:0.06471

Max. :59.0 Max. :0.2595 Max. :0.2815 Max. :1.00000

Table Error rates summary

To visualise the result, Figure 4 illustrates the error rates of training dataset (orange line) and the cross validation dataset (green line).

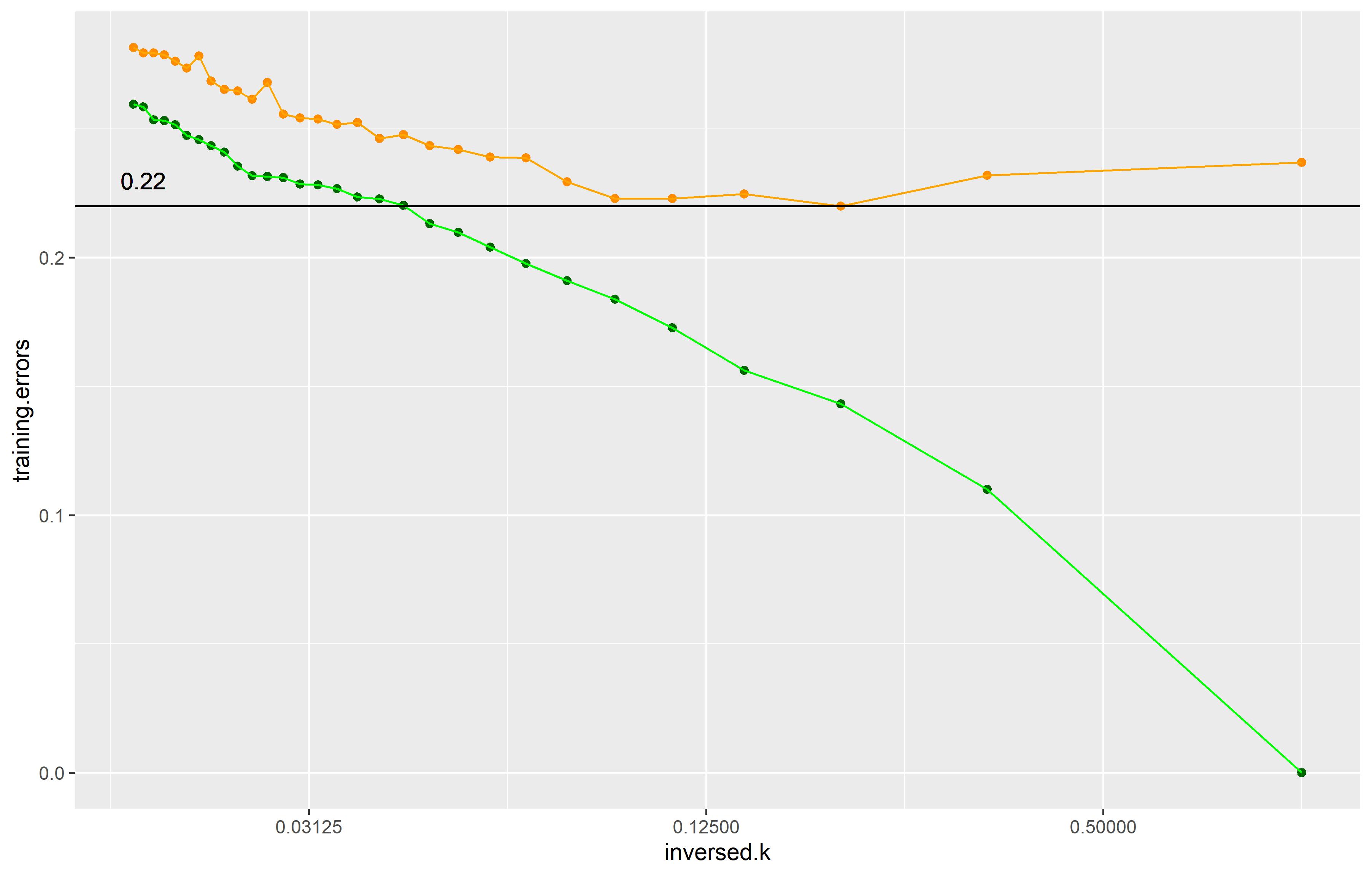


Figure Error rates - training set VS cross validation

According the above result, we can see that the model has best performance on testing data when, where its error rate reaches the lowest point of .

When, the model is under-fitted that it does not capture the underline correlations between the predictors and the response value, so the error rates for both training data and the testing data are high. When , the model is over-fitted that it captures too much noise in the training data, so it performs worse on the testing data, even it gives a good result on the training data (Koehrsen, 2018).

## Task 2.3

### Objective

The objective of this task is to analyse the performance of the model created in Task 2.2.

### Reasoning

Confusion Matrix (Dataschool, 2019) will be used for the analysis. It provides a detailed description of the performance of a classifier including how the classifier incorrectly classifies an observation.

### Implementation

In Task 2.2, we concluded that 5 is the best value of k to perform the KNN classification.

KNN classification is re-performed with , and a data frame in constructed containing the prediction and true label for each object. The function Confusion Matrix is applied to the data frame

### Result

The confusion matrix is produced (Table 10).

Using k = 5 to perform knn cross validation test

Confusion Matrix and Statistics

Reference

Prediction cherry flower banana pear envelope golfclub pencil wineglass

cherry 361 2 15 130 1 15 3 11

flower 0 487 15 0 7 0 8 5

banana 14 3 346 22 0 48 73 15

pear 109 1 36 319 0 9 10 19

envelope 2 4 1 0 489 0 0 0

golfclub 1 0 11 1 1 329 12 25

pencil 2 3 66 7 2 65 383 9

wineglass 11 0 10 21 0 34 11 416

Overall Statistics

Accuracy : 0.7825

95% CI : (0.7694, 0.7952)

No Information Rate : 0.125

P-Value [Acc > NIR] : < 2.2e-16

Kappa : 0.7514

Mcnemar's Test P-Value : NA

Statistics by Class:

Class: cherry Class: flower Class: banana Class: pear Class: envelope Class: golfclub Class: pencil

Sensitivity 0.72200 0.9740 0.6920 0.63800 0.9780 0.65800 0.76600

Specificity 0.94943 0.9900 0.9500 0.94743 0.9980 0.98543 0.95600

Pos Pred Value 0.67100 0.9330 0.6641 0.63419 0.9859 0.86579 0.71322

Neg Pred Value 0.95985 0.9963 0.9557 0.94824 0.9969 0.95276 0.96621

Prevalence 0.12500 0.1250 0.1250 0.12500 0.1250 0.12500 0.12500

Detection Rate 0.09025 0.1217 0.0865 0.07975 0.1222 0.08225 0.09575

Detection Prevalence 0.13450 0.1305 0.1303 0.12575 0.1240 0.09500 0.13425

Balanced Accuracy 0.83571 0.9820 0.8210 0.79271 0.9880 0.82171 0.86100

Class: wineglass

Sensitivity 0.8320

Specificity 0.9751

Pos Pred Value 0.8270

Neg Pred Value 0.9760

Prevalence 0.1250

Detection Rate 0.1040

Detection Prevalence 0.1258

Balanced Accuracy 0.9036

Table Confusion Matrix - KNN Cross validation

According to the matrix, the overall accuracy across all objects is

The accuracies (sensitivity in the table) of objects of cherry, flower, banana, pear, envelope, golf club, pencil and wineglass are , respectively. It shows that the model performs better than the average when classifying the objects of flower, envelop and wineglass, while it performs worse on other objects.

Especially, the model poorly performed on the objects of cherry and pear, as it seems to be confused by these two classes. It incorrectly classified 109 out of 500 cherries as pears (21.8%) and classified 130 out of 500 pears as cherries (26%).

Similarly, the model is confused among the objects of pencil, banana and golf club, and it classified 9.6% and 13% of golf clubs as banana and pencils, respectively, making it performed worst on the objects of golf clubs among all classes.

# Section 3

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

## Task 3.1

### Objective

The objective of this task is to use bagging of decision trees to perform classification using the first eight features, and evaluate the model with out-of-bag estimation (Breiman, 1996) and cross validation respectively.

### Reasoning

For the out-of-bag estimation, the R function bagging (Peters, 2019) will be used, as it performs out-of-bag estimation be default.

For the cross validation, a more complex package caret (Kuhn, 2019) will be used. The package provides functions for classification and regression training and evaluation, and it is flexible so we can configure the training method, tuning grid and training control to perform cross validation.

### Implementation

First, the data including the first eight features and the labels is extracted from the whole set of feature data, and they are fitted into the models of out-of-bag estimation and cross validation will be implemented separately.

#### Out-of-bag Estimation

The data is fitted into the bagging model with bag size of 25. A summary of the model similar Table 11 to is produced, showing the out-of-bag misclassification error. The accuracy of the model is calculated as

Bagging classification trees with 800 bootstrap replications

Call: bagging.data.frame(formula = label ~ ., data = data, nbagg = bag\_size,

coob = TRUE, control = rpart.control(minsplit = 2))

Out-of-bag estimate of misclassification error: 0.2925

Table Example output of bagging function

The process is repeated for each bag size of 25, 50, 200, 400 and 800. The accuracy is stored in a data frame.

#### Cross validation

The data is fitted into the bagging model (named as “treebag” and caret package) with parameters bag size of 25, method of cross validation and k-fold of 5. The output of the model is similar to Table 12, indicating the accuracy of the model.

Bagged CART

4000 samples

8 predictor

8 classes: 'cherry', 'flower', 'banana', 'pear', 'envelope', 'golfclub', 'pencil', 'wineglass'

No pre-processing

Resampling: Cross-Validated (5 fold)

Summary of sample sizes: 3200, 3200, 3200, 3200, 3200

Resampling results:

Accuracy Kappa

0.71325 0.6722857

Table Example result of Cross validation

The process is repeated for each bag size of 25, 50, 200, 400 and 800. The accuracy is stored in the same data frame as the out-of-bag estimation.

### Result

After the above two types of validations, the data frame of the accuracies for each model can be generated (Table 13).

evaluation.method accuracy bag.size

Out of Bag Estimate 0.71300 25

Out of Bag Estimate 0.70775 50

Out of Bag Estimate 0.71025 200

Out of Bag Estimate 0.71175 400

Out of Bag Estimate 0.71175 800

Cross Validation 0.71400 25

Cross Validation 0.71075 50

Cross Validation 0.70625 200

Cross Validation 0.72125 400

Cross Validation 0.72175 800

Table Data frame of accuracies

To better visualise the result, Figure 5 illustrate the accuracies of these two types of models with different values of bag size.

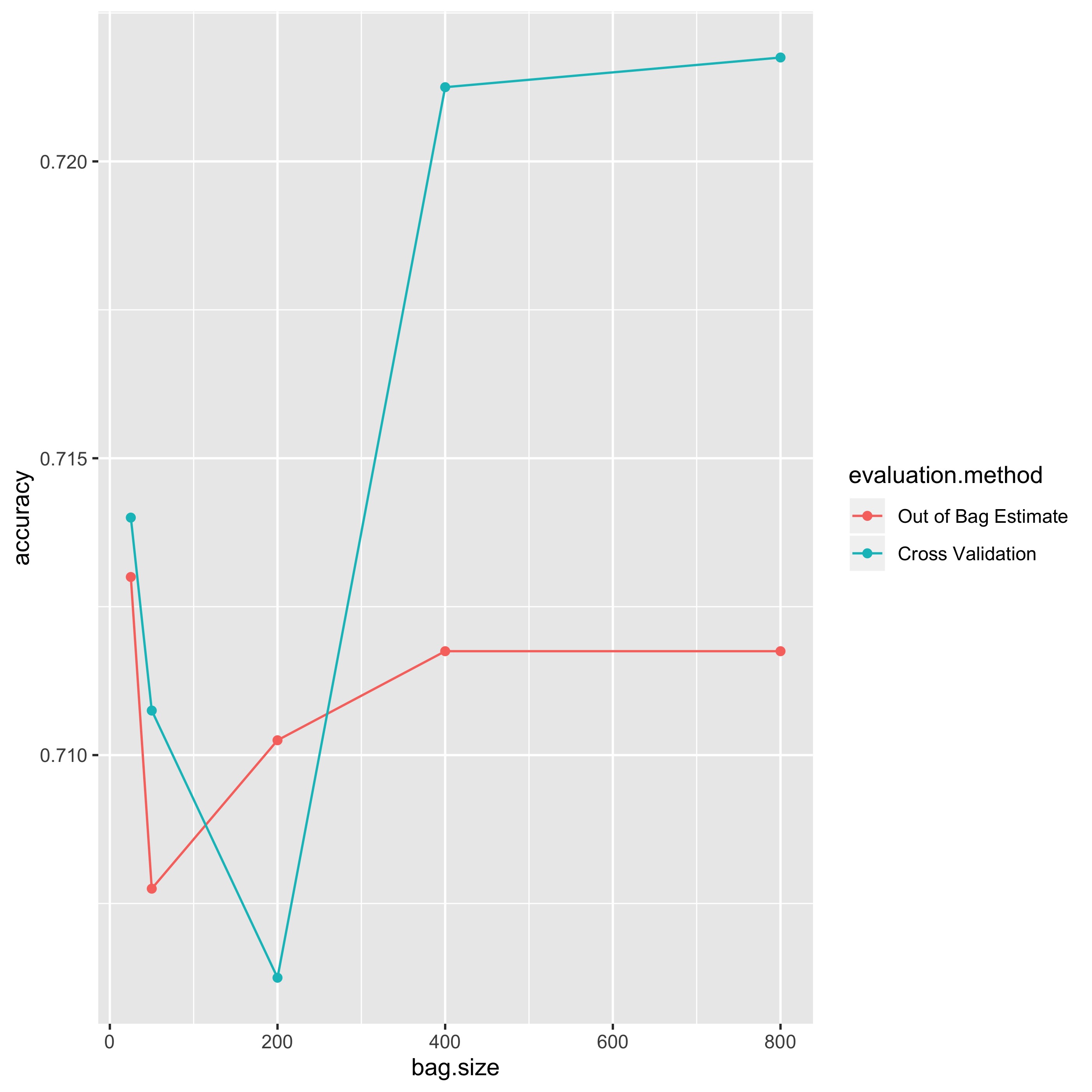


Figure Plot of accuracies of cross validation and out-of-bag estimation

According to the figure, we can see the accuracies of these models are between 70% and 73% in our experiment. The accuracies and their means evaluated by these two types are similar, but the values of cross validation are higher in most cases. Also, it is noticed that the variance of the accuracy values in the cross validation is greater than that in out-of-bag estimation.

## Task 3.2

### Objective

The objective of this task is to find the best parameters of number of trees () and number of predictors () using cross validation.

### Reasoning

The package Caret is used in this task. It is highly configurable and helps tune the model by providing grid search functionality. The package will be used to control the training process of random forest.

### Implementation

First, we extract the eight features we need from the whole set of feature data. Then the training control is set up using the following parameters.

We then use Caret to control and fit the model of random forest. For the first round, with the hyper-parameter number of trees of 25, Caret grid searches the accuracies of the models with number of predictors of 2, 4, 6 and 8. The result of the grid search is produced (similar to Table 14). The accuracies are recorded in a data frame along with the values of and .

Random Forest

4000 samples

8 predictor

8 classes: 'cherry', 'flower', 'banana', 'pear', 'envelope', 'golfclub', 'pencil', 'wineglass'

No pre-processing

Resampling: Cross-Validated (5 fold)

Summary of sample sizes: 3200, 3200, 3200, 3200, 3200

Resampling results across tuning parameters:

mtry Accuracy Kappa

2 0.84600 0.8240000

4 0.84450 0.8222857

6 0.84100 0.8182857

8 0.83275 0.8088571

Accuracy was used to select the optimal model using the largest value.

The final value used for the model was mtry = 2.

Table Example output of grid search

The process is performed repeatedly for each value of number of trees between 25 and 400 (increments of 25), and the accuracies for each performance are saved in a data frame.

### Result

Table 15 shows the accuracies for each value of and . The best hyper-parameters with the highest accuracy are calculated that where

mtry Accuracy ntree

2 0.83625 25

4 0.83650 25

6 0.83275 25

8 0.82975 25

2 0.84200 50

4 0.83625 50

6 0.83175 50

8 0.82750 50

2 0.84375 75

4 0.83675 75

6 0.83100 75

8 0.83250 75

2 0.84075 100

4 0.84075 100

6 0.83525 100

8 0.83250 100

2 0.84125 125

4 0.83650 125

6 0.83000 125

8 0.82850 125

2 0.84550 150

4 0.83875 150

6 0.83750 150

8 0.83325 150

2 0.84325 175

4 0.84100 175

6 0.83775 175

8 0.83125 175

2 0.84650 200

4 0.83775 200

6 0.83475 200

8 0.83050 200

2 0.84700 225

4 0.84250 225

6 0.83925 225

8 0.83125 225

mtry Accuracy ntree

2 0.84775 250

4 0.83925 250

6 0.83800 250

8 0.83075 250

2 0.84625 275

4 0.83600 275

6 0.83825 275

8 0.82975 275

2 0.84650 300

4 0.84500 300

6 0.83750 300

8 0.83575 300

2 0.84400 325

4 0.84425 325

6 0.83700 325

8 0.83050 325

2 0.84250 350

4 0.83825 350

6 0.83300 350

8 0.82575 350

2 0.84000 375

4 0.83700 375

6 0.83350 375

8 0.82475 375

2 0.84600 400

4 0.84450 400

6 0.83800 400

8 0.83775 400

Best Tune:

mtry Accuracy ntree

2 0.84775 250

Table Grid search result

To better interpret the result, Figure 6 illustrates the trends of accuracy values across various numbers of trees, grouped by the numbers of predictors.

As it shows, the model has best performance when in almost any value of . Conventionally, it is believed that the optimal number of predictor considered in each node is equal to the squared root of number of total predictors. In our case, the optimal value is . Our experiment verified this by showing the value of 2 is the best tuning value is our set of .

Also, for the value of , the model fluctuates in all groups, so it is hard to tell the general best value of according to our model.

Generally, the accuracies across our experiments are between 82.5% and 85%.

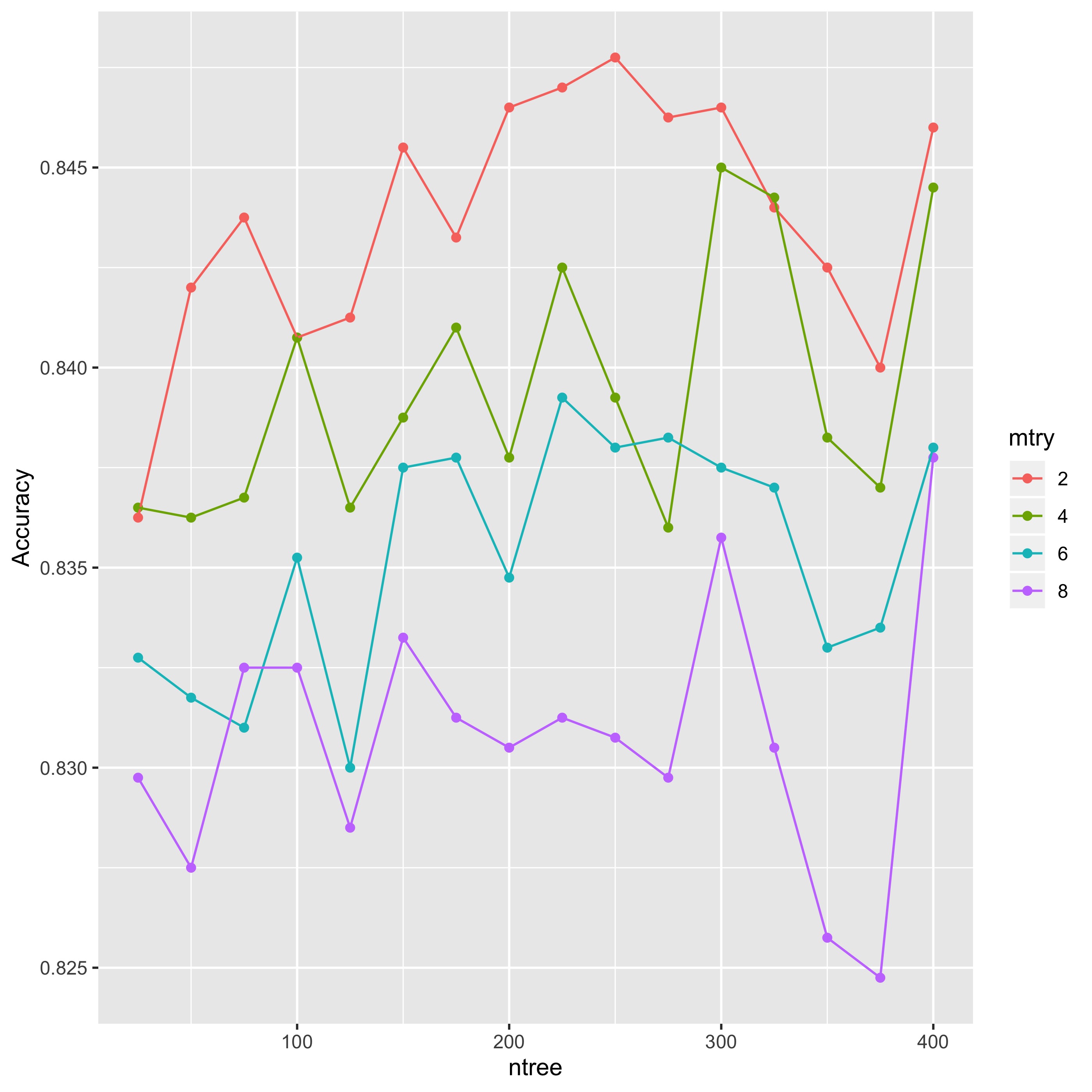


Figure Grid search result

## Task 3.3

### Objective

The objective of this task is to find out the variance and mean value of the accuracy of cross validated random forest with 20 times of experiments.

### Reasoning

The best values of hyper-parameters and from Task 3.2 will be used, and the experiment is performed 20 times.

### Implementation

From Task 3.2, the best tuning values are , the model of cross validated random forest is fitted by the data of first eight features. A table similar to Table 16 is produced. The accuracy value is collected and stored in a vector.

Random Forest

4000 samples

8 predictor

8 classes: 'cherry', 'flower', 'banana', 'pear', 'envelope', 'golfclub', 'pencil', 'wineglass'

No pre-processing

Resampling: Cross-Validated (5 fold)

Summary of sample sizes: 3200, 3200, 3200, 3200, 3200

Resampling results:

Accuracy Kappa

0.84125 0.8185714

Tuning parameter 'mtry' was held constant at a value of 2

Table Example output of cross validated random forest

The same process is performed 20 times, and a vector of 20 accuracy values are generated.

### Result

20 accuracy values are as Table 17.

0.84825 0.84325 0.84375 0.84625 0.845 0.84725 0.84725 0.83775 0.8455 0.84175 0.84675 0.846 0.85075 0.845 0.85025 0.842 0.8455 0.842 0.84575 0.84875

Table 20 accuracy values

The mean of the accuracy is

The standard deviation of the accuracy is

## Task 3.4

### Objective

The objective of this task is to construct a 7-feature model by removing one feature from the 8-feature model in Task 3.3. Then compare the accuracies of these two models.

### Assumption

The significant probability value (p-value) in the hypothesis test is set to

### Reasoning

The removed feature should has contributes the least to the classification. The R function Vip (Cao, 2019) will be used to visualise the importance values across all eight features and decide the feature to be removed.

Since the number of values in each group is less than 30 (considered as a small sample), hypothesis T test (R-core, 2019) is performed, based on the Assumption.

### Implementation

First, we fit the data of eight features into the model of bagging trees with 5-fold cross validation. The function Vip is used to interpret the model by visualising the importance values of these eight features (Figure 7).

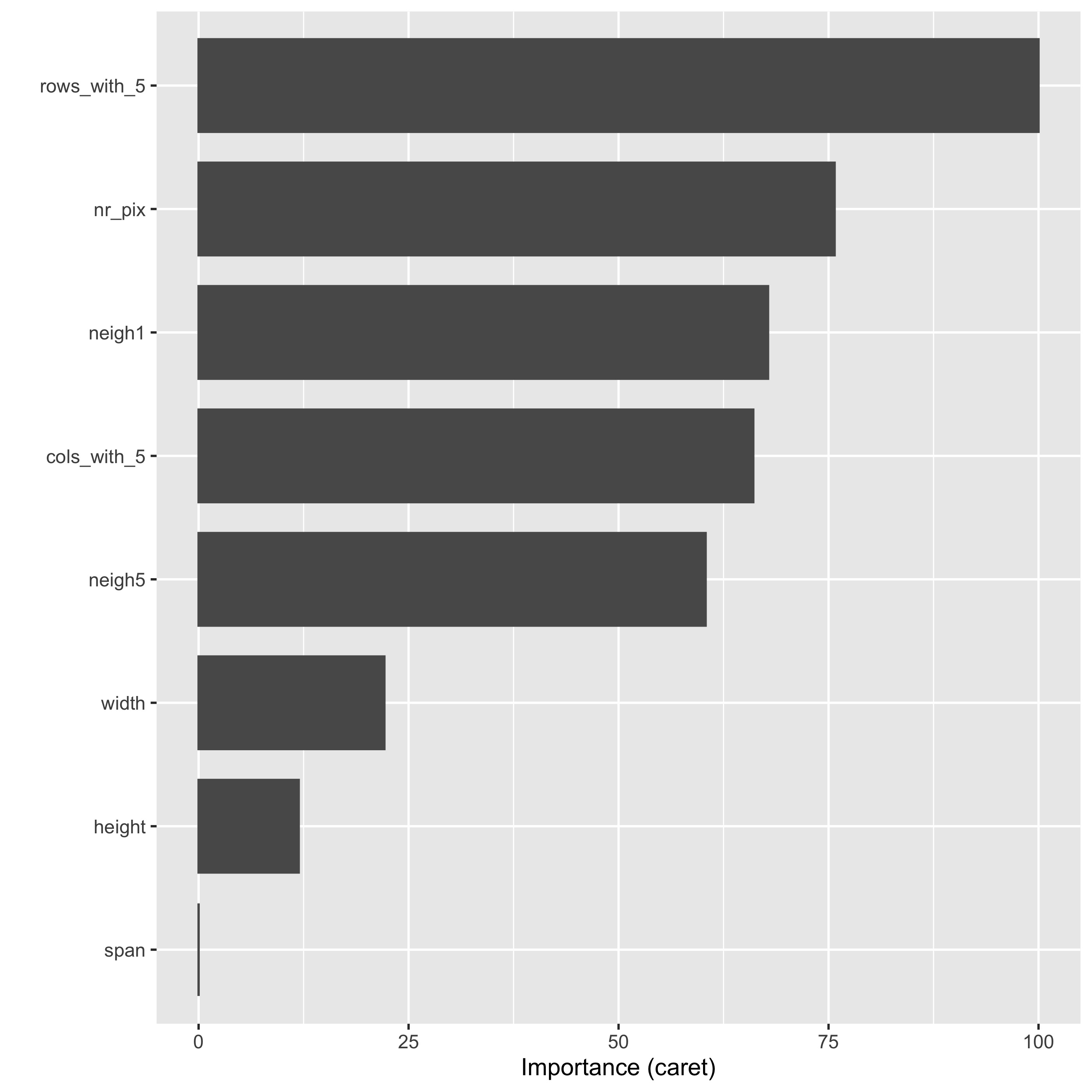


Figure Importance values of the eight feature model

As the figure illustrates, the feature span contribute the least (0 importance) to the classification model in our experiment. Thus, the 7-feature model is built is excluded the feature span.

According to Task 3.3, the best hyper-parameters are. The data of the rest seven features is used to fit the random forest model, evaluated by 5-fold cross validation. The evaluation is repeated 20 times, and 20 accuracy values for this 7-feature model is produced.

The total 40 accuracy values are stored in a data frame, split into two groups. To compare the difference in these two groups, T test is performed.

### Result

The summary of the accuracy values in these two models is as Table 18.

seven.feature eight.feature

Min. :0.8343 Min. :0.8377

1st Qu.:0.8363 1st Qu.:0.8436

Median :0.8381 Median :0.8456

Mean :0.8380 Mean :0.8454

3rd Qu.:0.8393 3rd Qu.:0.8472

Max. :0.8425 Max. :0.8508

Table Summary of accuracy values

As the table demonstrates, the mean value of 7-feature model is less than the eight feature model. Also, all measurements (e.g. minimum/maximum values, median) in the table for 7-feature model are slightly less than that for 8-feature model.

The result of the T test is as Table 19.

Welch Two Sample t-test

data: accuracies$seven.feature and accuracies$eight.feature

t = -8.6108, df = 34.646, p-value = 1.983e-10

alternative hypothesis: true difference in means is less than 0

95 percent confidence interval:

-Inf -0.005947609

sample estimates:

mean of x mean of y

0.8380375 0.8454375

Table T test - 7 features vs 8 features

It is noticed that the p-value is less than the significant p-value of 0.05. Thus, we reject the null hypothesis, and stick on . Thus, we can conclude that the true mean of accuracy of the 7 feature model is significantly less than that of the 8 feature model.

# Conclusions

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

# References

Breiman, L., 1996. *OUT-OF-BAG ESTIMATION,* s.l.: s.n.

Cao, K.-A., 2019. *vip function | R Documentation.* [Online]   
Available at: https://www.rdocumentation.org/packages/mixOmics/versions/6.3.2/topics/vip  
[Accessed 24 December 2019].

Chandrayan, P., 2019. *Logistic Regression For Dummies: A Detailed Explanation.* [Online]   
Available at: https://towardsdatascience.com/logistic-regression-for-dummies-a-detailed-explanation-9597f76edf46

Dataschool, 2019. *Simple guide to confusion matrix terminology.* [Online]   
Available at: https://www.dataschool.io/simple-guide-to-confusion-matrix-terminology/

Devereux, B., 2019. *160 Topic 16 - Intro to Machine Learning.pptx.* [Online]   
Available at: https://canvas.qub.ac.uk/courses/8433/files/487509?module\_item\_id=205271

Drakos, G., 2018. *Cross-Validation.* [Online]   
Available at: https://towardsdatascience.com/cross-validation-70289113a072

Geeksforgeeks, 2019. *ML | Multiple Linear Regression (Backward Elimination Technique).* [Online]   
Available at: https://www.geeksforgeeks.org/ml-multiple-linear-regression-backward-elimination-technique/

Irizarry, R. & Love, M., 2019. *Cross-validation.* [Online]   
Available at: https://genomicsclass.github.io/book/pages/crossvalidation.html  
[Accessed 22 December 2019].

Koehrsen, W., 2018. *Overfitting vs. Underfitting: A Complete Example.* [Online]   
Available at: https://towardsdatascience.com/overfitting-vs-underfitting-a-complete-example-d05dd7e19765  
[Accessed 22 December 2019].

Kuhn, M., 2019. *CRAN - Package caret.* [Online]   
Available at: https://cran.r-project.org/web/packages/caret/index.html  
[Accessed 24 December 2019].

Liu, D., 2019. *CSC3060 AIDA – Assignment 2,* Belfast: s.n.

Peters, A., 2019. *bagging function | R Documentation.* [Online]   
Available at: https://www.rdocumentation.org/packages/ipred/versions/0.4-0/topics/bagging  
[Accessed 24 December 2019].

Picard, R. R. & Cook, R. D., 2012. Cross-Validation of Regression Models. *Journal of the American Statistical Association,* 79(387), pp. 575-583.

R-core, 2019. *t.test function | R Documentation.* [Online]   
Available at: https://www.rdocumentation.org/packages/stats/versions/3.6.1/topics/t.test  
[Accessed 24 December 2019].

R. C. T., 2019. *R Download.* [Online]   
Available at: https://cran.r-project.org/bin/

Ripley, B., 2019. *K-Nearest Neighbour Classification.* [Online]   
Available at: https://www.rdocumentation.org/packages/class/versions/7.3-15/topics/knn  
[Accessed 22 December 2019].

Team, R. C., 2019. *Available CRAN Packages By Name.* [Online]   
Available at: https://cran.r-project.org/web/packages/available\_packages\_by\_name.html