Assignment 3

qvns53

2024-11-19

Question 1

We are using a k-nearest neighbours model to predict the response variable y based on the predictor variables x_1 and x_2. We want to assign a value to the point with $x_1 = 1$ and $x_2 = 2$. Here are the nearest neighbours from the training set:

x_1	x_2	y
1	1	0.65
2	2	0.53
2	1	0.42
3	2	0.31
0	3	0.75

1(a)

What is the predicted value of y for the new point using the k-nearest neighbours model with $k = 3$ using the Euclidean distance?

```r
# Create a data frame with the training data
training_data <- data.frame("x_1" = c(1, 2, 2, 3, 0),
                            "x_2" = c(1, 2, 1, 2, 3),
                            "y" = c(0.65, 0.53, 0.42, 0.31, 0.75))

# Create a data frame with the new point
new_point <- data.frame("x_1" = 1,
                        "x_2" = 2)

# Calculate the Euclidean distance
training_data$distance <- sqrt((new_point$x_1 - training_data$x_1)^2 + (new_point$x_2 - training_data$x
```

Here I have used the 'tidyverse' package (seen in MATH2687) to predict y, however the 'caret' package would also work.

```r
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.0     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
```

```
# Estimate y for the new_point
k_nearest <- training_data |>
  arrange(desc(distance)) |>
  slice(1:3)

prediction <- mean(k_nearest$y)
print(paste0("The predicted value for x_1 = 1, x_2 = 2, is ", round(prediction, 2)))
```

```
## [1] "The predicted value for x_1 = 1, x_2 = 2, is 0.49"
```

1(b)

What is the predicted value of y for the new point using the k-nearest neighbours model with $k = 3$ using the Manhattan distance?

```
# Calculate the Manhattan distance
training_data$manhattan_distance <- abs(new_point$x_1 - training_data$x_1) + abs(new_point$x_2 - traini

# Estimate y for the new_point
k_nearest_manhattan <- training_data |>
  arrange(desc(manhattan_distance)) |>
  slice(1:3)

prediction_manhattan <- mean(k_nearest_manhattan$y)
print(paste0("The predicted value for x_1 = 1, x_2 = 2, is ", round(prediction_manhattan, 2)))
```

```
## [1] "The predicted value for x_1 = 1, x_2 = 2, is 0.49"
```

1(c)

Find the predicted value of y for the new point using the weighted variant of the k-nearest neighbours model with $k = 3$, Manhattan distance and the inverse distance weighting.

```
# Calculate the inverse distance weighting
training_data$weight <- 1/training_data$manhattan_distance

# Estimate y for the new_point
k_nearest_manhattan_weighted <- training_data |>
  arrange(desc(weight)) |>
  slice(1:3)

prediction_manhattan_weighted <- sum(k_nearest_manhattan_weighted$y * k_nearest_manhattan_weighted$weig
print(paste0("The predicted value for x_1 = 1, x_2 = 2, is ", round(prediction_manhattan_weighted, 2)))
```

```
## [1] "The predicted value for x_1 = 1, x_2 = 2, is 0.56"
```

Question 2

In this question, we will utilise the `wine` dataset in R. The dataset contains information about 1,599 Portuguese red wines.

```
wine <- read.csv("https://www.maths.dur.ac.uk/users/john.p.gosling/MATH3431_presentations/winequality-r
```

2(a)

The variable `quality` is listed as a numeric variable. Convert this variable to a binary factor with levels "bad" and "good". A wine is considered "good" if the quality is greater than or equal to 7.

```
# Add a new column to the data frame
wine$quality_binary <- ifelse(wine$quality >= 7, "good", "bad")

# Convert the new column to a factor
wine$quality_binary <- as.factor(wine$quality_binary)
```

2(b)

Perform a 60/40 split of the dataset into a training set and a test set.

```
# Set the seed for reproducibility
set.seed(124)

# Split the dataset
train_indices <- sample(1:nrow(wine), 0.6*nrow(wine))
wine_train <- wine[train_indices, ]
wine_test <-  wine[-train_indices, ]

str(wine_train)
```

```
## 'data.frame':    959 obs. of  13 variables:
##  $ fixed.acidity       : num  11.5 6.8 9.9 10.2 7.9 6.4 9.9 6.6 8.6 7 ...
##  $ volatile.acidity    : num  0.42 0.64 0.35 0.54 0.33 0.57 0.57 0.58 0.52 0.54 ...
##  $ citric.acid         : num  0.48 0.1 0.38 0.37 0.23 0.02 0.25 0.02 0.38 0 ...
##  $ residual.sugar      : num  2.6 2.1 1.5 15.4 1.7 1.8 2 2 1.5 2.1 ...
##  $ chlorides           : num  0.077 0.085 0.058 0.214 0.077 0.067 0.104 0.062 0.096 0.079 ...
##  $ free.sulfur.dioxide : num  8 18 31 55 18 4 12 37 5 39 ...
##  $ total.sulfur.dioxide: num  20 101 47 95 45 11 89 53 18 55 ...
##  $ density             : num  0.999 0.996 0.997 1.004 0.996 ...
##  $ pH                  : num  3.09 3.34 3.26 3.18 3.29 3.46 3.04 3.35 3.2 3.39 ...
##  $ sulphates           : num  0.53 0.52 0.82 0.77 0.65 0.68 0.9 0.76 0.52 0.84 ...
##  $ alcohol             : num  11 10.2 10.6 9 9.3 9.5 10.1 11.6 9.4 11.4 ...
##  $ quality             : int  5 5 7 6 5 5 5 7 5 6 ...
##  $ quality_binary      : Factor w/ 2 levels "bad","good": 1 1 2 1 1 1 1 2 1 1 ...
```

```
str(wine_test)
```

```
## 'data.frame':    640 obs. of  13 variables:
##  $ fixed.acidity       : num  7.4 7.8 7.4 7.9 7.8 7.5 6.7 7.5 7.8 8.9 ...
##  $ volatile.acidity    : num  0.7 0.76 0.7 0.6 0.58 0.5 0.58 0.5 0.61 0.22 ...
##  $ citric.acid         : num  0 0.04 0 0.06 0.02 0.36 0.08 0.36 0.29 0.48 ...
##  $ residual.sugar      : num  1.9 2.3 1.9 1.6 2 6.1 1.8 6.1 1.6 1.8 ...
##  $ chlorides           : num  0.076 0.092 0.076 0.069 0.073 0.071 0.097 0.071 0.114 0.077 ...
##  $ free.sulfur.dioxide : num  11 15 11 15 9 17 15 17 9 29 ...
##  $ total.sulfur.dioxide: num  34 54 34 59 18 102 65 102 29 60 ...
##  $ density             : num  0.998 0.997 0.998 0.996 0.997 ...
##  $ pH                  : num  3.51 3.26 3.51 3.3 3.36 3.35 3.28 3.35 3.26 3.39 ...
##  $ sulphates           : num  0.56 0.65 0.56 0.46 0.57 0.8 0.54 0.8 1.56 0.53 ...
##  $ alcohol             : num  9.4 9.8 9.4 9.4 9.5 10.5 9.2 10.5 9.1 9.4 ...
##  $ quality             : int  5 5 5 5 7 5 5 5 5 6 ...
##  $ quality_binary      : Factor w/ 2 levels "bad","good": 1 1 1 1 2 1 1 1 1 1 ...
```

2(c)

Using `rpart`, fit a decision tree model to the training set to predict the binary quality of the wine based on the 11 explanatory variables. What is the accuracy of the model on the test set? Provide a visualisation of the decision tree.

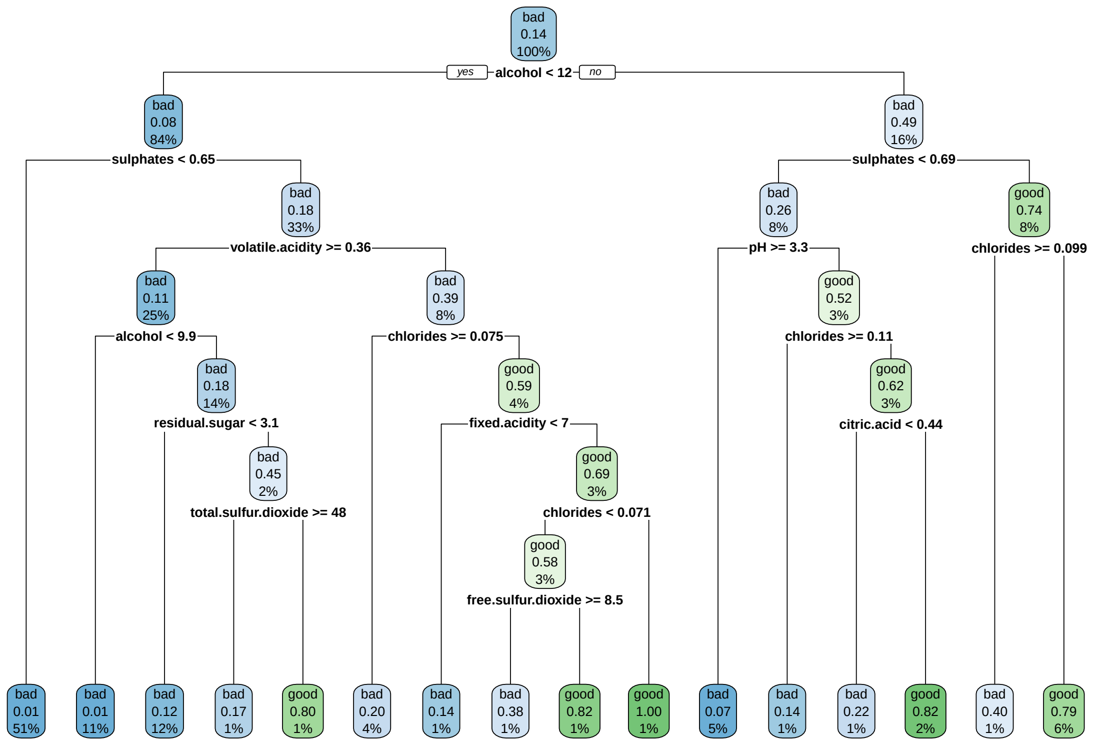
```r
library(rpart)

# Fit the decision tree model
wine_tree <- rpart(quality_binary ~ . -quality,
                   data = wine_train,
                   method = "class")

# Make predictions on the test set
predictions <- predict(wine_tree, wine_test, type="class")
# Calculate the accuracy
accuracy <- mean(predictions == wine_test$quality_binary)
print(accuracy)
```

```
## [1] 0.84375
```

```r
# Plot the decision tree
library(rpart.plot)
rpart.plot(wine_tree)
```



2(d)

Fit a k-nearest neighbours model to the training set using the `caret` and `class` packages. Use 15-fold cross-validation to determine the optimal value of k from the range 2 to 10. What is the optimal value of k and the associated accuracy of the model on the test set? Would you prefer to use the decision tree or the k-nearest neighbours model for this dataset?

```r
library(class)
library(caret)
```

```
## Loading required package: lattice

##
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':
##
##     lift
```

```r
# Fit the knn model using 15-fold cross validation whilst optimising k
knn_fit <- train(quality_binary ~ . - quality,
                 data = wine_train,
                 method = "knn",
                 tuneGrid = expand.grid(k = 2:10),
                 trControl = trainControl(method = "cv",
                                          number = 15))

# Display the knn_fit object
knn_fit
```

```
## k-Nearest Neighbors
##
## 959 samples
##  12 predictor
##   2 classes: 'bad', 'good'
##
## No pre-processing
## Resampling: Cross-Validated (15 fold)
## Summary of sample sizes: 895, 895, 895, 895, 895, 895, ...
## Resampling results across tuning parameters:
##
##   k   Accuracy   Kappa
##    2  0.8373347  0.3360288
##    3  0.8488095  0.2793274
##    4  0.8519015  0.2482927
##    5  0.8519015  0.2208523
##    6  0.8508763  0.1848856
##    7  0.8613095  0.1724622
##    8  0.8561012  0.1406675
##    9  0.8602679  0.1485673
##   10  0.8571594  0.1067945
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 7.
```

```r
# Predictions for the test set
test_pred <- predict(knn_fit, newdata = wine_test)

# Calculate the accuracy
accuracy_knn <- mean(test_pred == wine_test$quality_binary)
accuracy_knn
```

```
## [1] 0.846875
```

To compare the models, we can look at accuracy and complexity. The decision tree (DT) model has a rounded accuracy of 0.84375 and the kNN model achieved 0.846875. These are very similar accuracies, but it does suggest the kNN model to be marginally better.

However, on the side of computational complexity, the DT model would be preferrable as it requires less resources to make predictions than the kNN model. The DT model is also more interpretable.