STAT462_Ass3

Chris Chang

2025-10-01

Part A

Classifying wine samples (classification trees)

In this question we will train tree-based classification algorithms to classify samples according to the variable Class, which states the cultivar used for the wine.

Q1.

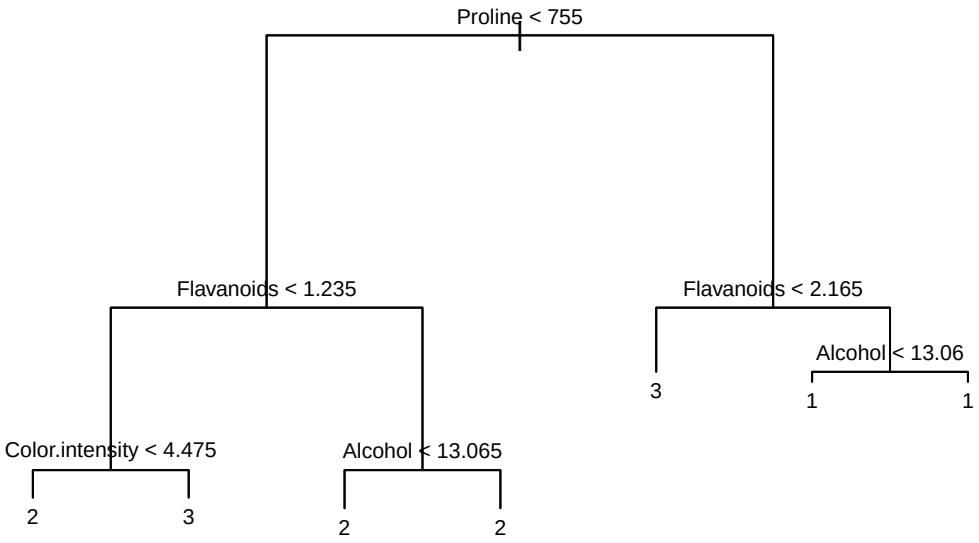
Train a single tree-based classifier on the training set. Use cross-validation to prune this tree suitably. Visualise the classification tree.

```
# Read in data to train and test sets
train <- read.csv("Data for assignment 3-20251002\\wine_train.csv")
test <- read.csv("Data for assignment 3-20251002\\wine_test.csv")
```

```
# Ensure Class is categorical
train$Class <- as.factor(train$Class)
test$Class <- as.factor(test$Class)

# train a tree to classify
classification.tree <- tree(Class ~ ., train)
summary(classification.tree)
```

```
##
## Classification tree:
## tree(formula = Class ~ ., data = train)
## Variables actually used in tree construction:
## [1] "Proline"         "Flavanoids"      "Color.intensity" "Alcohol"
## Number of terminal nodes:  7
## Residual mean deviance:  0.2472 = 33.37 / 135
## Misclassification error rate: 0.05634 = 8 / 142
```

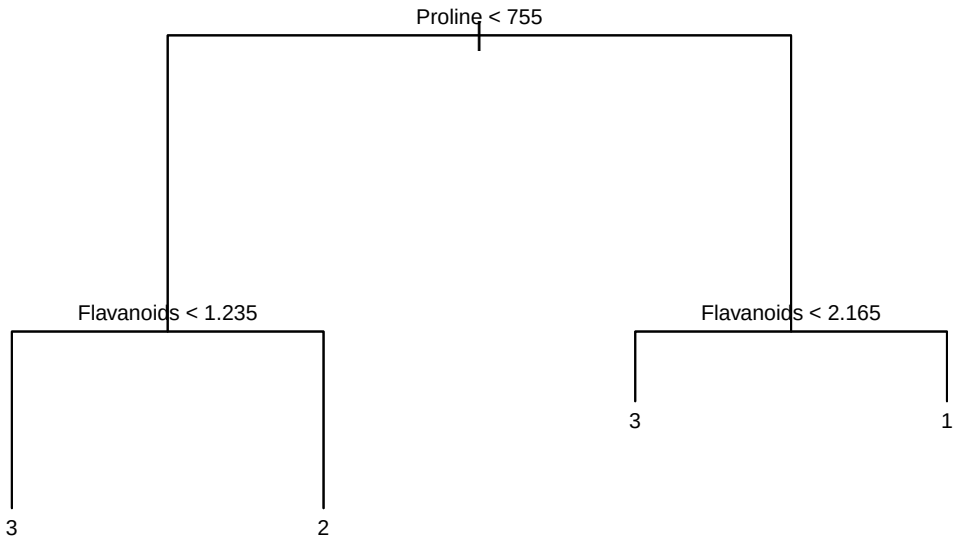
```
# Plot the intital tree
plot(classification.tree)
text(classification.tree, pretty = 0, cex = 0.7)
```




```
cv.classification <- cv.tree(classification.tree, FUN = prune.misclass)

# Find out the best tree size that corresponds to the lowest deviance
best.size <- cv.classification$size[which.min(cv.classification$dev)]

# Use the best size to prune the tree
pruned.tree_best <- prune.misclass(classification.tree, best = best.size)
best.size
```

```
## [1] 4
```

```
# Plot the tree after pruning
plot(pruned.tree_best)
text(pruned.tree_best, pretty = 0, cex = 0.7)
```



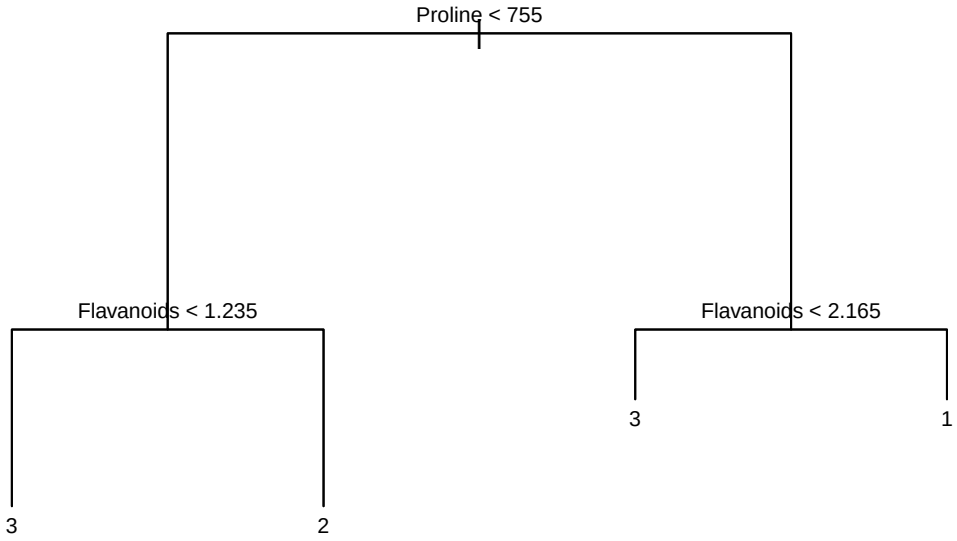

```
# Make predictions on test data
pruned.test.pred <- predict(pruned.tree_best, newdata = test, type = "class")

# Compute the error rate
pruned.er <- mean(pruned.test.pred != test$Class)

pruned.er
```

```
## [1] 0.1666667
```

Q2.

Prune your tree enough so that you only need two features to make predictions. Visualise your data in these two dimensions, and illustrate the decision tree of your classifier graphically.

```
# pruning to two branches
pruned.tree_2 <- prune.misclass(classification.tree, best = 4)

# plot pruned tree with only 2 branches
plot(pruned.tree_2)
text(pruned.tree_2, pretty = 0, cex = 0.7)
```



Q3.

Fit a bagged classification tree model and/or a random forest to see whether you can improve on your single tree's performance.

```
# Count the number of predictors. Class needs to be excluded.
num_predictors <- ncol(train) -1

# Fit a bagged tree
bag.tree <- randomForest(Class ~ . , data = train, mtry = num_predictors, importance=TRUE)

bag.tree
```

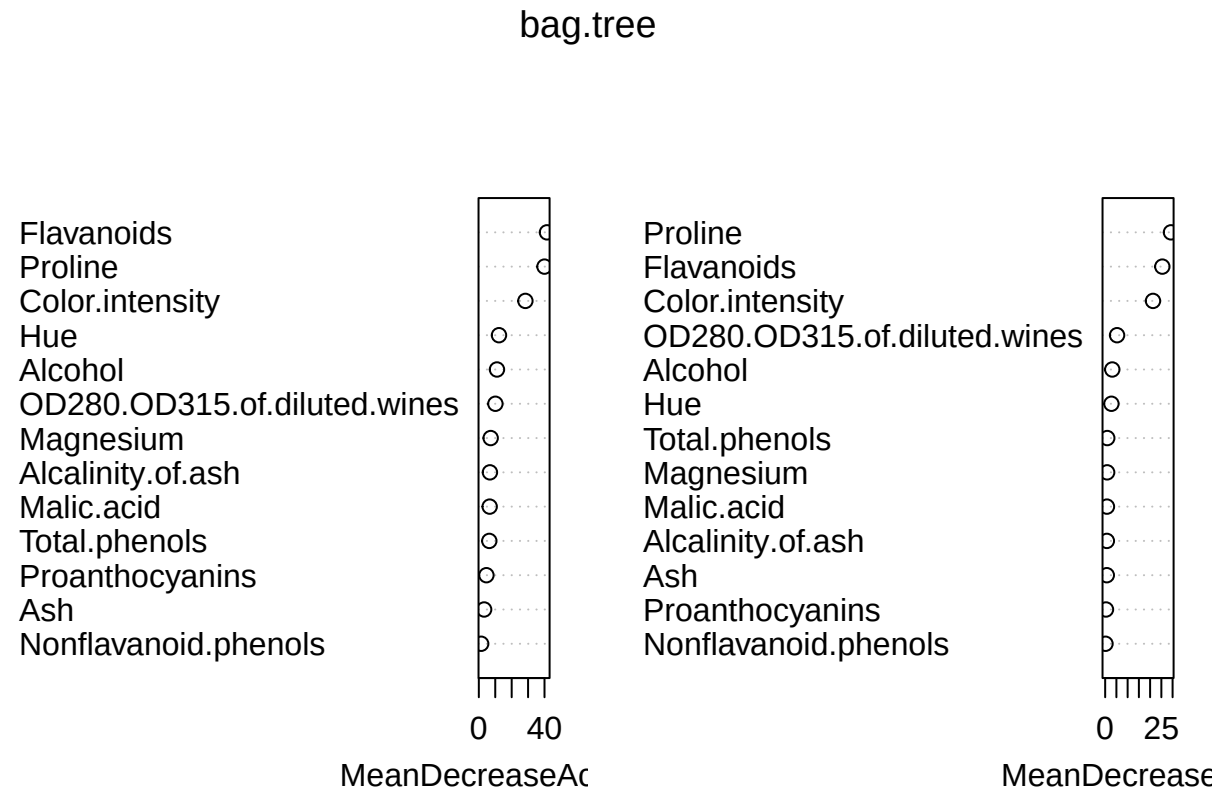
```
##
## Call:
##  randomForest(formula = Class ~ ., data = train, mtry = num_predictors,      importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 13
##
##         OOB estimate of  error rate: 4.23%
## Confusion matrix:
##    1  2  3 class.error
## 1 48  1  0  0.02040816
## 2  2 55  2  0.06779661
```

```
## 3  0  1 33  0.02941176
```

```
bag.pred <- predict(bag.tree, newdata = test)

bag.er <- mean(bag.pred != test$Class)

bag.er
```

```
## [1] 0
```

```
# Report most important features
varImpPlot(bag.tree)
```



Part B

Clustering the wine dataset (Hierarchical clustering and k-means)

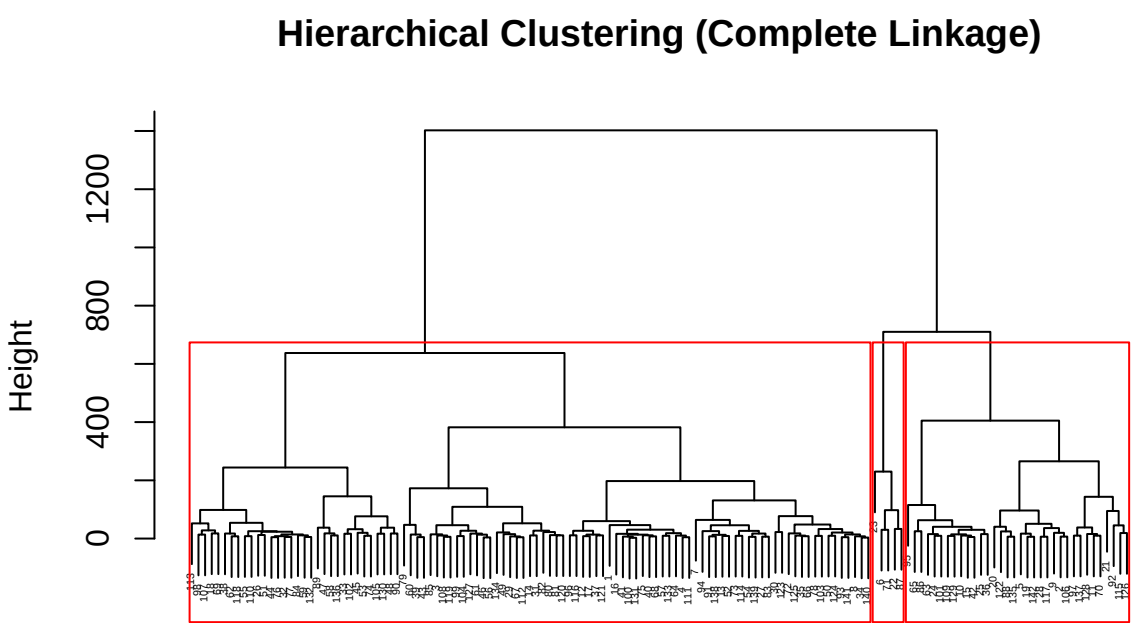
```
plot_func <- function(data, label, title = "") {

   # Ensure labels are factor
  labels <- factor(labels)

  # Convert to data frame if needed
```

```
  data <- as.data.frame(data)

  # PCA for 2D reduction
  pca <- prcomp(data, scale. = TRUE)
  pca_df <- as.data.frame(pca$x[, 1:2])
  colnames(pca_df) <- c("PC1", "PC2")

  # Add cluster/class labels
  pca_df$label <- labels


  # Plot
  p <- ggplot(pca_df, aes(x = PC1, y = PC2, color = label)) +
    geom_point(size = 3.0, alpha = 0.7) +
    scale_color_manual(values = c("#00AFBB", "#FC4E07", "#e7b800",
                                  "#006400", "#9400D3", "#FF8C00","#008080")) +
    labs(title = title, x = "PC1", y = "PC2", color = "Cluster/Class") +
    theme_grey()
  return(p)
}
```

Q1.

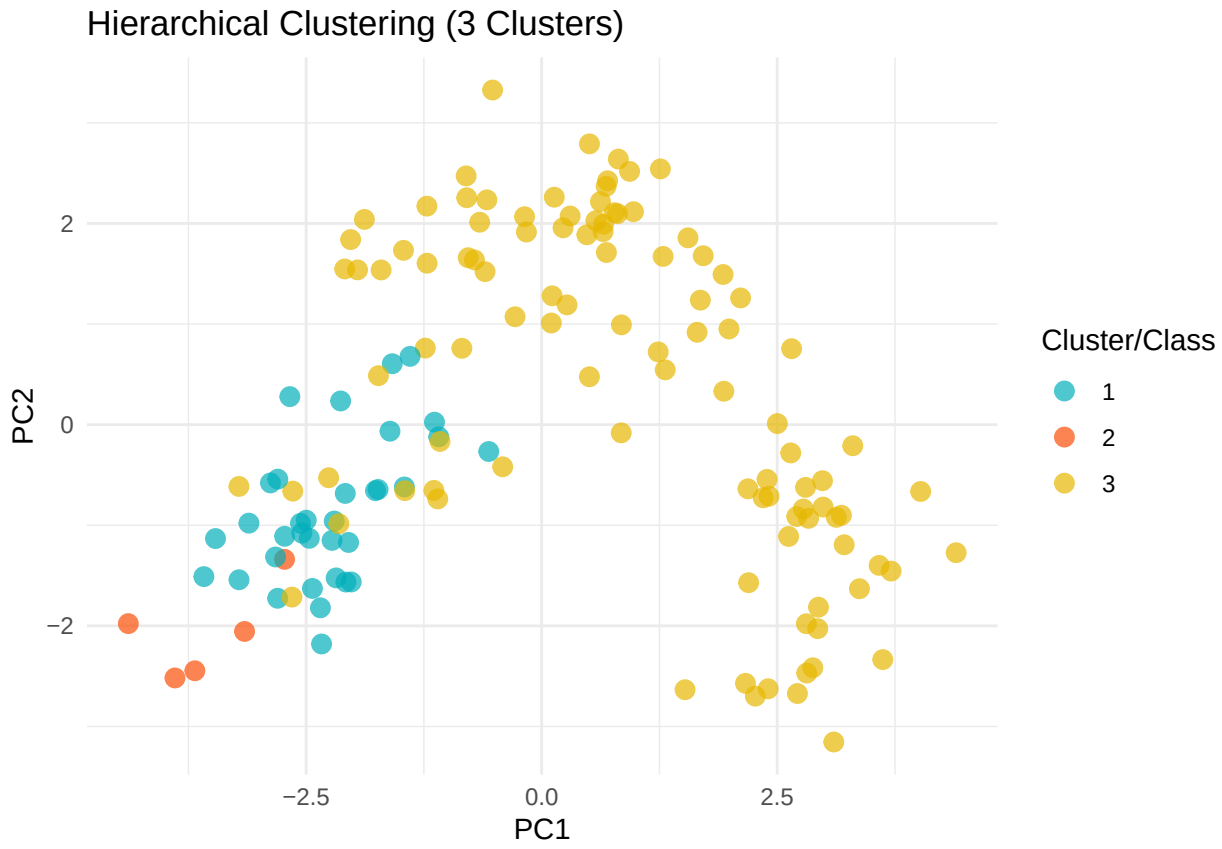
Perform hierarchical clustering on the wine dataset, but do not include the Class feature. Group the data into three clusters and check/visualise whether this is a good reconstruction of the (actual) classes recorded in the Class feature.

```
# Remove Class column for clustering
train_no_class <- train[, !(names(train) %in% "Class")]

# Distance matrix
dist_matrix <- dist(train_no_class)

# perform hierarchical clustering
hc.complete <- hclust(dist_matrix, method = "complete")
plot(hc.complete, main = "Hierarchical Clustering (Complete Linkage)", xlab = "", sub = "", cex = .3)

# Cut tree into 3 clusters
clusters <- cutree(hc.complete, k = 3)

plot(hc.complete, main = "Hierarchical Clustering (Complete Linkage)", xlab = "", sub = "", cex = .3)
# Sepcify a chosen number of clusters
rect.hclust(hc.complete, k = 3, border = "red")
```

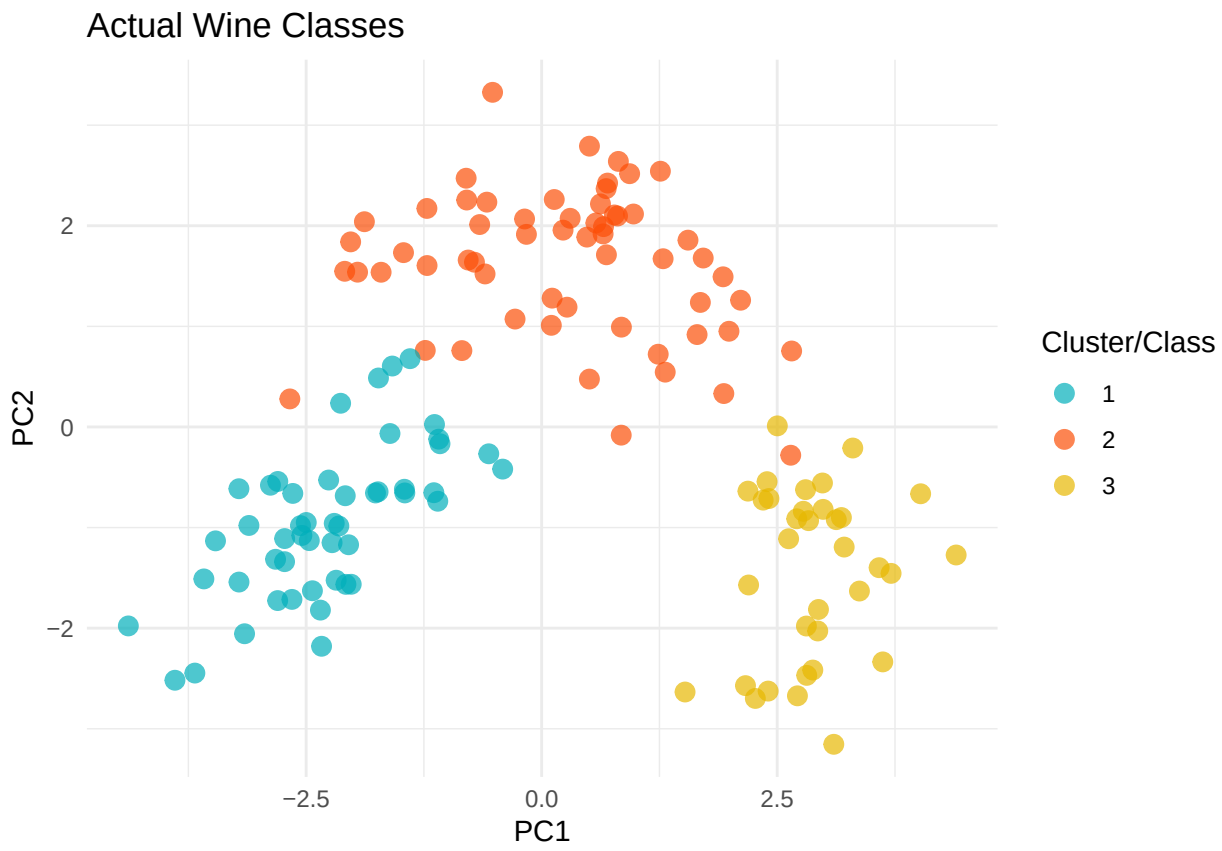



```r
# Cut tree into 3 clusters
clusters <- cutree(hc.complete, k = 3)

# 3. PCA-based plotting function
plot_func <- function(data, labels, title = "") {
  # Ensure labels is a vector and factor
  labels <- factor(as.vector(labels))

  # Convert data to data frame if needed
  data <- as.data.frame(data)

  # PCA to reduce to 2D
  pca <- prcomp(data, scale. = TRUE)
  pca_df <- as.data.frame(pca$x[, 1:2])
  colnames(pca_df) <- c("PC1", "PC2")

  # Add labels
  pca_df$label <- labels

  # Plot
  ggplot(pca_df, aes(x = PC1, y = PC2, color = label)) +
    geom_point(size = 3, alpha = 0.7) +
    scale_color_manual(values = c("#00AFBB", "#FC4E07", "#e7b800",
                                  "#006400", "#9400D3", "#FF8C00","#008080")) +
    labs(title = title, x = "PC1", y = "PC2", color = "Cluster/Class") +
    theme_minimal()
```

```
}

# Manual remapping - adjust these numbers based on what you see
# If cluster 1 should be class 2, cluster 2 should be class 1, cluster 3 should be class 3:
manual_map <- c(3, 1, 2)  # Adjust these based on your dendrogram
remapped_clusters <- manual_map[clusters]


# Plot hierarchical clustering result
plot_func(train_no_class, remapped_clusters, "Hierarchical Clustering (3 Clusters)")
```




```
# Plot actual Class labels
plot_func(train_no_class, train$Class, "Actual Wine Classes")
```

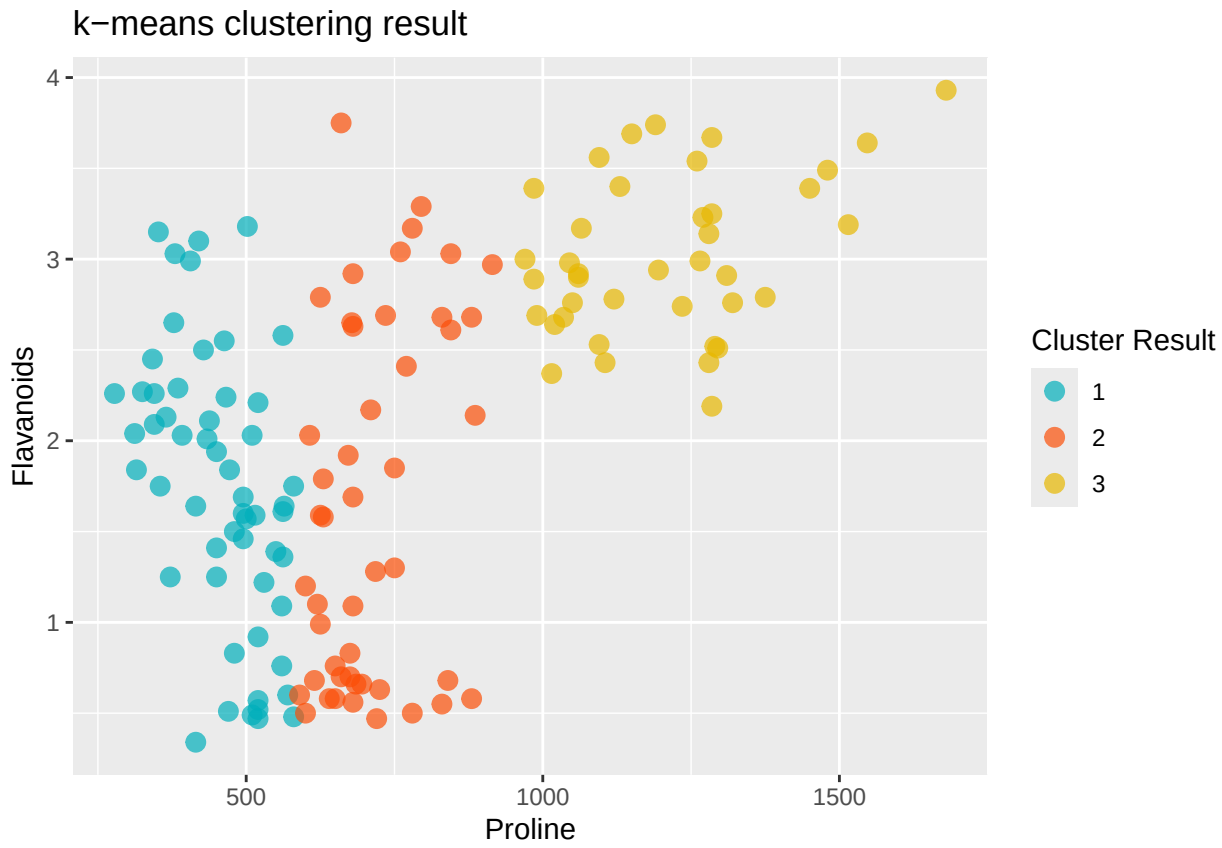


Q2.

Do the same using the k-means algorithm, for k=3. For visualisation, you can pick the two features that were sufficient for classification in question A, and plot datapoints in these two dimensions, comparing actual classes and predicted cluster labels.

```
dat <- train[,c("Proline", "Flavanoids")]

k.clus <- kmeans(dat, 3)

plot_func_clus <- function(data, label, title = "") {

  data$label <- factor(label)

  x_col <- colnames(data)[1]
  y_col <- colnames(data)[2]
  label <- colnames(data)[3]


  p <- data %>%
    ggplot(data = ., mapping = aes(x = .data[[x_col]], y = .data[[y_col]], color = .data[[label]])) +
    scale_color_manual(values = c("#00AFBB", "#FC4E07", "#e7b800", "#006400", "#9400D3", "#FF8C00","#008
    geom_point(size = 3.0, alpha = 0.7) +
    labs(
      title = title,
```

```
      x = x_col, y = y_col, color = "Cluster Result"
    ) +
    theme_grey()
  return(p)
}

# Manual remapping - adjust these numbers based on what you see
# If cluster 1 should be class 2, cluster 2 should be class 1, cluster 3 should be class 3:
manual_map <- c(3, 1, 2)  # Adjust these based on your dendrogram
remapped_clusters <- manual_map[k.clus$cluster]

plot_func_clus(dat, remapped_clusters, "k-means clustering result")
```



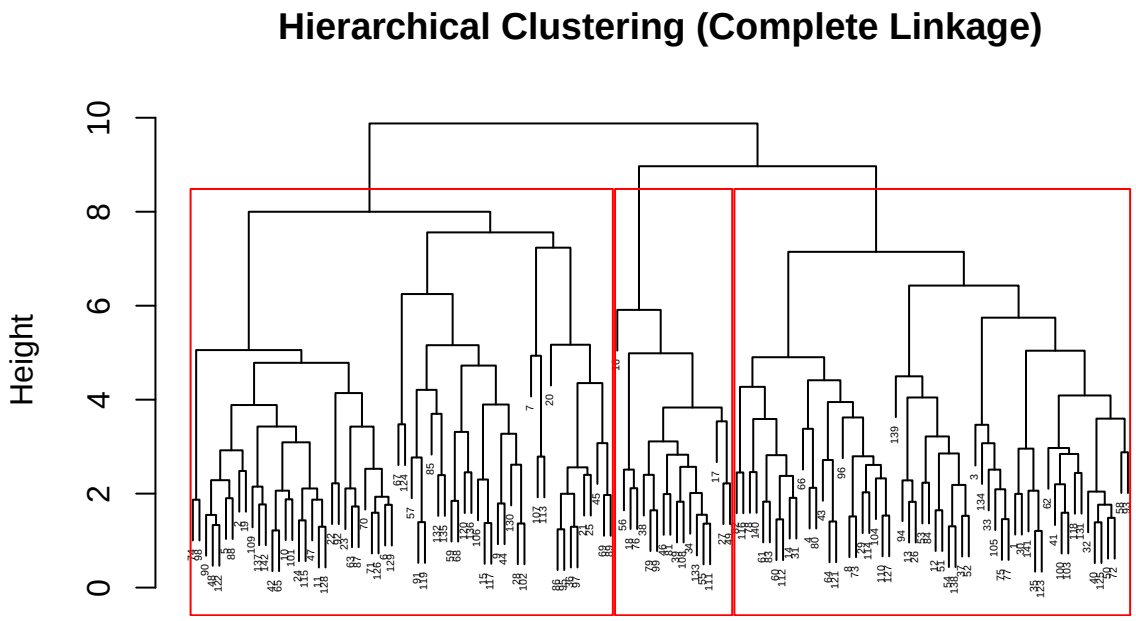

```
# Plot actual Class labels
plot_func_clus(dat, train$Class, "Actual Wine Classes")
```

Q3.

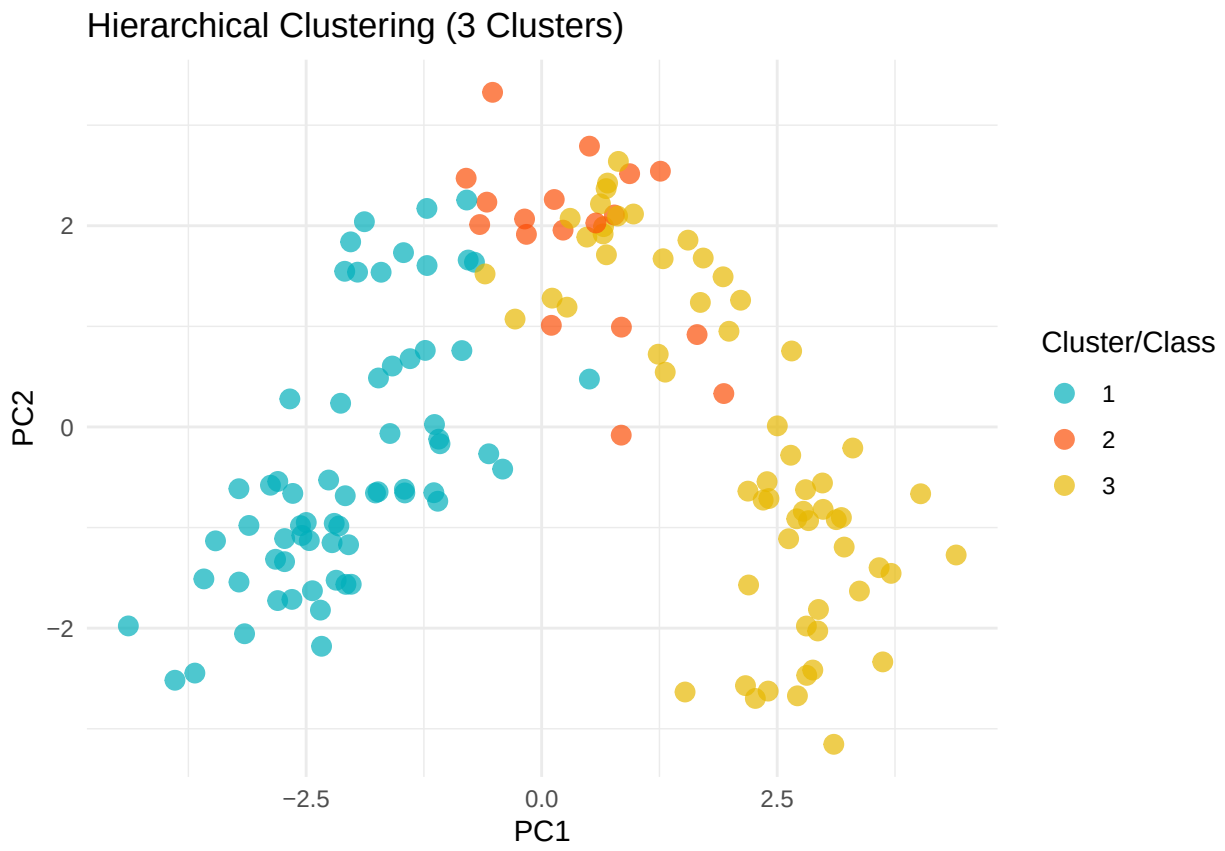
You will likely not get great results, because your features vary on very different orders of magnitude (for example, Nonflavanoid.phenols is mostly between 0 and 1, but Proline is in the 1000 range). Normalise all numerical features using either z-score transformation or min-max normalisation, which will bring them into comparable orders of magnitude. Then repeat parts 1. and 2., and see whether your results improve.

```
# Remove Class column for clustering
train_no_class <- train[, !(names(train) %in% "Class")]

# Scale all predictors
scaled_train <- scale(train_no_class)
scaled_train <- as.data.frame(scaled_train)

# Distance matrix
dist_matrix <- dist(scaled_train)

# perform hierarchical clustering
hc.complete <- hclust(dist_matrix, method = "complete")
plot(hc.complete, main = "Hierarchical Clustering (Complete Linkage)", xlab = "", sub = "", cex = .3)

# Cut tree into 3 clusters
clusters <- cutree(hc.complete, k = 3)

plot(hc.complete, main = "Hierarchical Clustering (Complete Linkage)", xlab = "", sub = "", cex = .3)
```

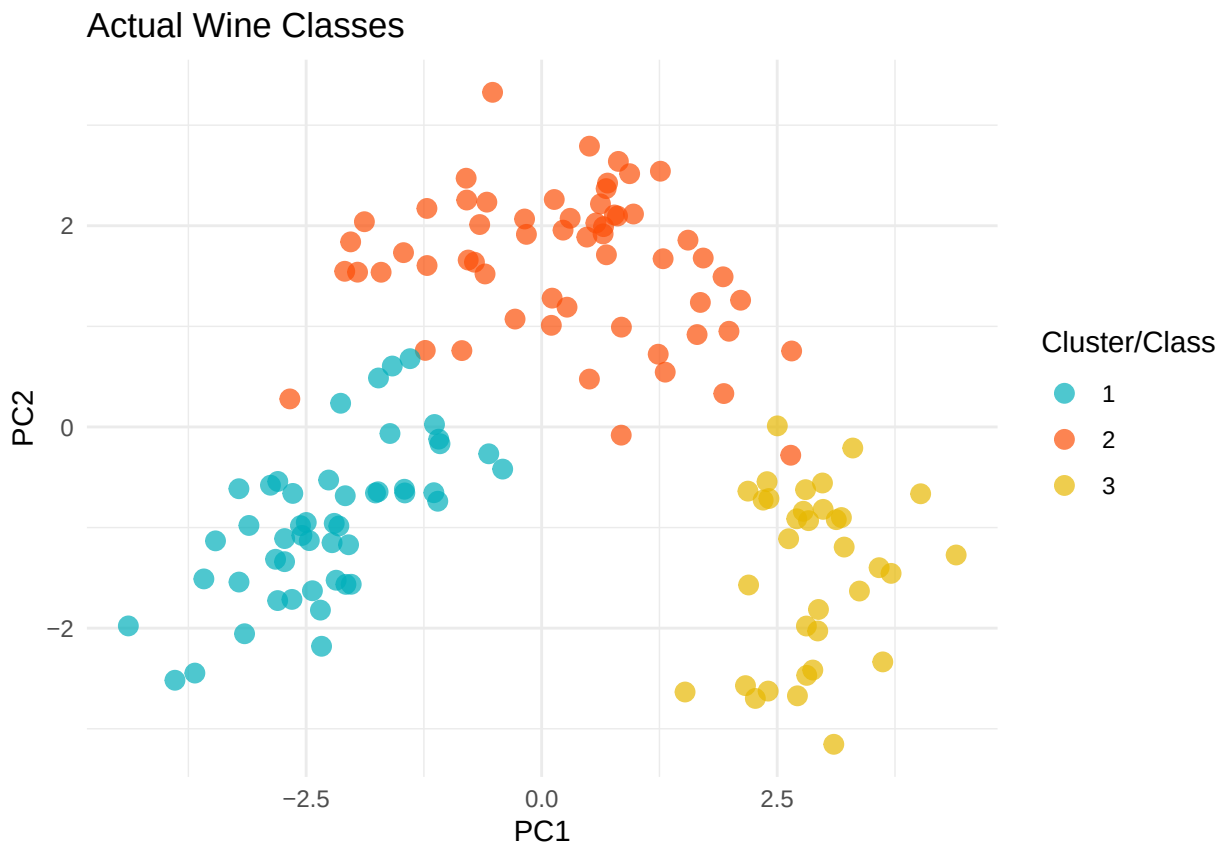
```r
# Sepcify a chosen number of clusters
rect.hclust(hc.complete, k = 3, border = "red")
```




```r
# Cut tree into 3 clusters
clusters <- cutree(hc.complete, k = 3)

# Manual remapping - adjust these numbers based on what you see
# If cluster 1 should be class 2, cluster 2 should be class 1, cluster 3 should be class 3:
manual_map <- c(3, 1, 2)  # Adjust these based on your dendrogram
remapped_clusters <- manual_map[clusters]

# Plot hierarchical clustering result
plot_func(scaled_train, remapped_clusters, "Hierarchical Clustering (3 Clusters)")
```




```
# Plot actual Class labels
plot_func(scaled_train, train$Class, "Actual Wine Classes")
```




```
dat <- train[,c("Proline", "Flavanoids")]
dat <- as.data.frame(scale(dat))

k.clus <- kmeans(dat, 3)

# This colour remapping code is from Claude
# Create a mapping from k-means clusters to actual classes
# Find which cluster number corresponds to which class
cluster_to_class <- sapply(1:3, function(cluster) {
  # Get the most common actual class for this cluster
  cluster_members <- train$Class[k.clus$cluster == cluster]
  as.numeric(names(sort(table(cluster_members), decreasing = TRUE)[1]))
})

# Remap cluster labels to match actual classes
remapped_clusters <- cluster_to_class[k.clus$cluster]

plot_func_clus(dat, k.clus$cluster, "k-means clustering result")
```


```
# Plot actual Class labels
plot_func_clus(dat, train$Class, "Actual Wine Classes")
```


Actual Wine Classes
Flavanoids
Proline
2
1
0
−1
−1
0
1
2
3
Cluster Result
1
2
3