Exercise 4 solution

Isabelle Cretton

Oct. 8th, 2024

```
# Set global code chunk options
knitr::opts_chunk$set(echo = TRUE, message = FALSE, warning = FALSE)
```

```
library(cluster)
library(stats)
library(dplyr)
library(ggplot2)
#install.packages("skimr")
library(skimr)
library(tidyverse)
library(ggrepel)
#library for LDA analysis
library(MASS)
#libraries for random forests and classification trees
library(rpart)        # For classification trees
library(rpart.plot)
library(randomForest)       # For data splitting and accuracy evaluation
library(caret)        # For data splitting and accuracy evaluation
```

Problem 1 (EC Dataset)

Load the txt file

```
# Load the data
ec_data <- read.csv("/Users/alimos313/Documents/studies/phd/university/courses/stat-modelling/StatModell
```

1.A

explore and visualize the data w/ plots and tables

```
# Summary of the data
skim(ec_data)
```

Table 1: Data summary

Name	ec_data
Number of rows	48
Number of columns	5
Column type frequency:	
character	2

numeric	3
Group variables	None

Variable type: character

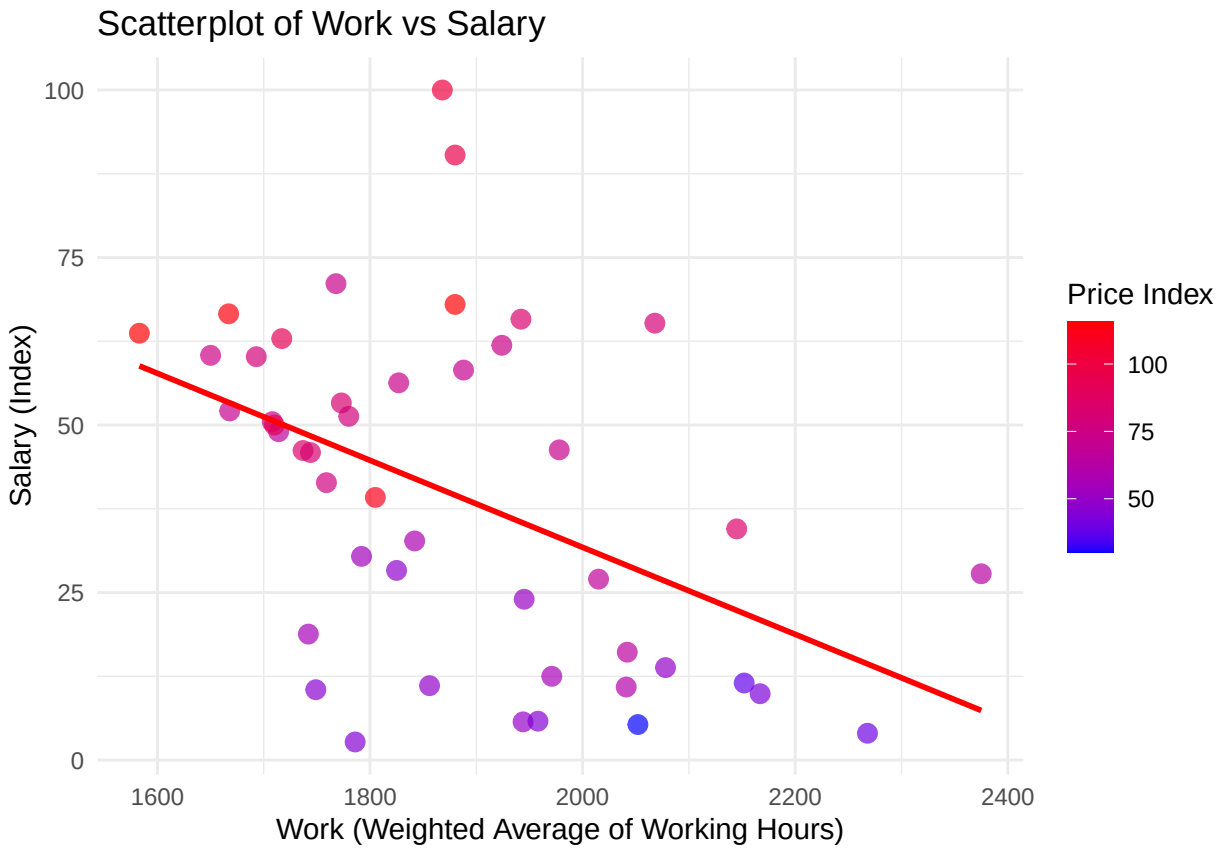
skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
City	0	1.00	4	14	0	48	0
SalaryCat	2	0.96	10	11	0	3	0

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
Work	2	0.96	1879.91	174.34	1583.0	1745.25	1849.00	1976.25	2375.0	
Price	0	1.00	68.86	21.78	30.3	49.65	70.50	81.70	115.5	
Salary	2	0.96	39.55	24.76	2.7	14.38	43.65	59.70	100.0	

```
ec_data_clean <- ec_data %>% na.omit()
```

```
# Scatterplot of the data
ggplot(ec_data, aes(x = Work, y = Salary, color = Price)) +
  geom_point(size = 3, alpha = 0.7) +
  geom_smooth(method = "lm", se = FALSE, color = "red") +
  scale_color_gradient(low = "blue", high = "red") +
  labs(title = "Scatterplot of Work vs Salary",
       x = "Work (Weighted Average of Working Hours)",
       y = "Salary (Index)",
       color = "Price Index") +
  theme_minimal()
```



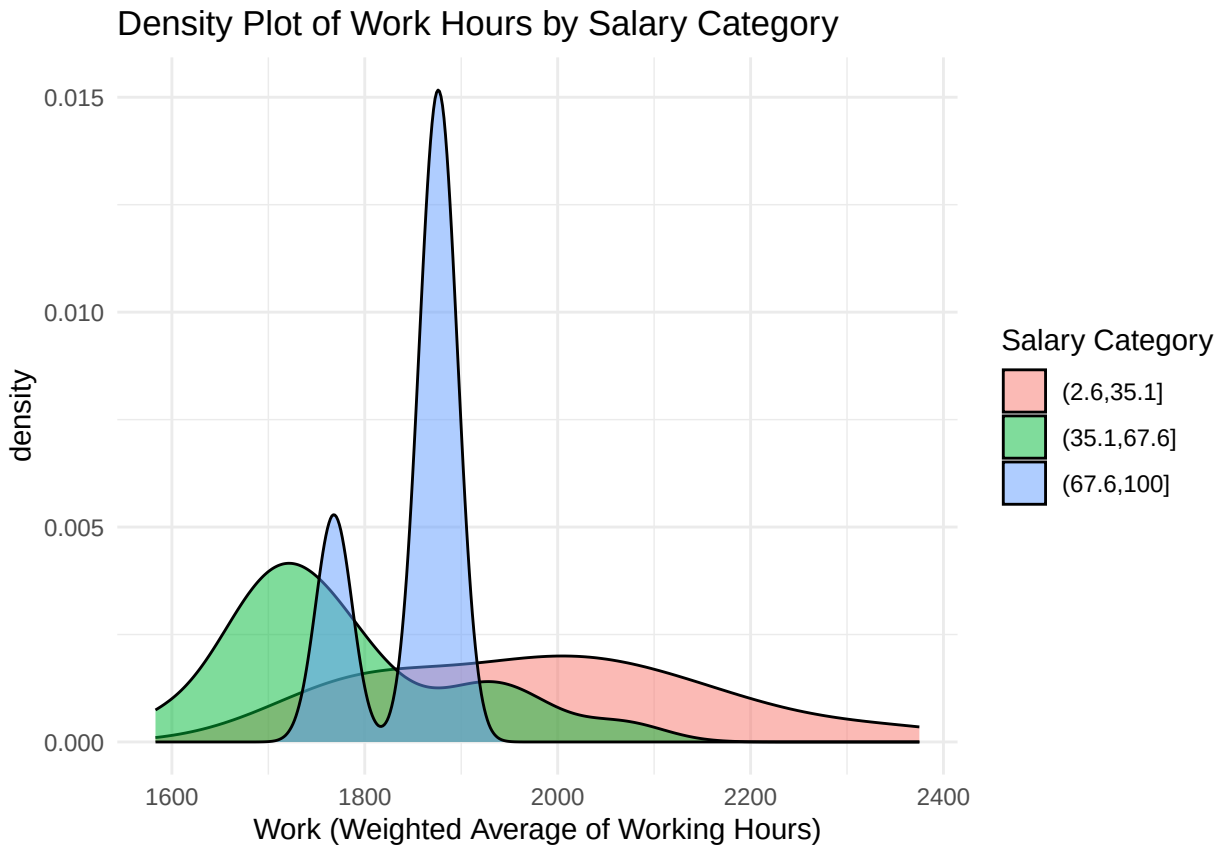
This plot shows a negative correlation between Work (average working hours) and Salary (index of earnings). Cities with higher working hours tend to have lower salaries. There seems to be a consistent trend where cities that require more work hours offer lower salaries, with Zurich (salary = 100) standing out as an exception with a lower work average and higher salary. Prices generally correlate positively with salaries, though not perfectly.

```
# Boxplot of the data
boxplot_data <- ec_data_clean %>%
  pivot_longer(cols = c(Price, Salary),
               names_to = "Variable",
               values_to = "Index")

# Plot the boxplot
ggplot(boxplot_data, aes(x = Variable, y = Index, fill = Variable)) +
  geom_boxplot() +
  scale_fill_manual(values = c("Price" = "#FF9999", "Salary" = "#9999FF")) +
  labs(title = "Boxplot of Price and Salary Indices") +
  theme_minimal()
```

The boxplot compares the distribution of Price and Salary. Price has a narrower spread than Salary, with a higher median value (~70) compared to the median salary (~40). The greater variability in Salary indicates that wage levels are more diverse across cities than price levels. Many cities face low salary levels despite high prices, emphasizing the cost of living challenges in certain cities.

```
# Density plot of the data
ggplot(ec_data_clean, aes(x = Work, fill = SalaryCat)) +
  geom_density(alpha = 0.5) +
  labs(title = "Density Plot of Work Hours by Salary Category",
       x = "Work (Weighted Average of Working Hours)",
       fill = "Salary Category") +
  theme_minimal()
```



This plot shows the distribution of Work (working hours) broken down by SalaryCat (salary categories). The highest salary category (67.6, 100] is associated with cities that have a narrow peak in work hours, with a median around 1900-2000 hours. Conversely, lower salary categories (2.6, 35.1] and (35.1, 67.6] are spread across a wider range of work hours, with some cities having very high work hours. Cities with higher salaries tend to have more standardized working hours, while cities with lower salaries show a broader spread in work hours, including cities with extremely high work burdens.

```
colnames(ec_data_clean)
```

```
## [1] "City"      "Work"      "Price"     "Salary"    "SalaryCat"
```

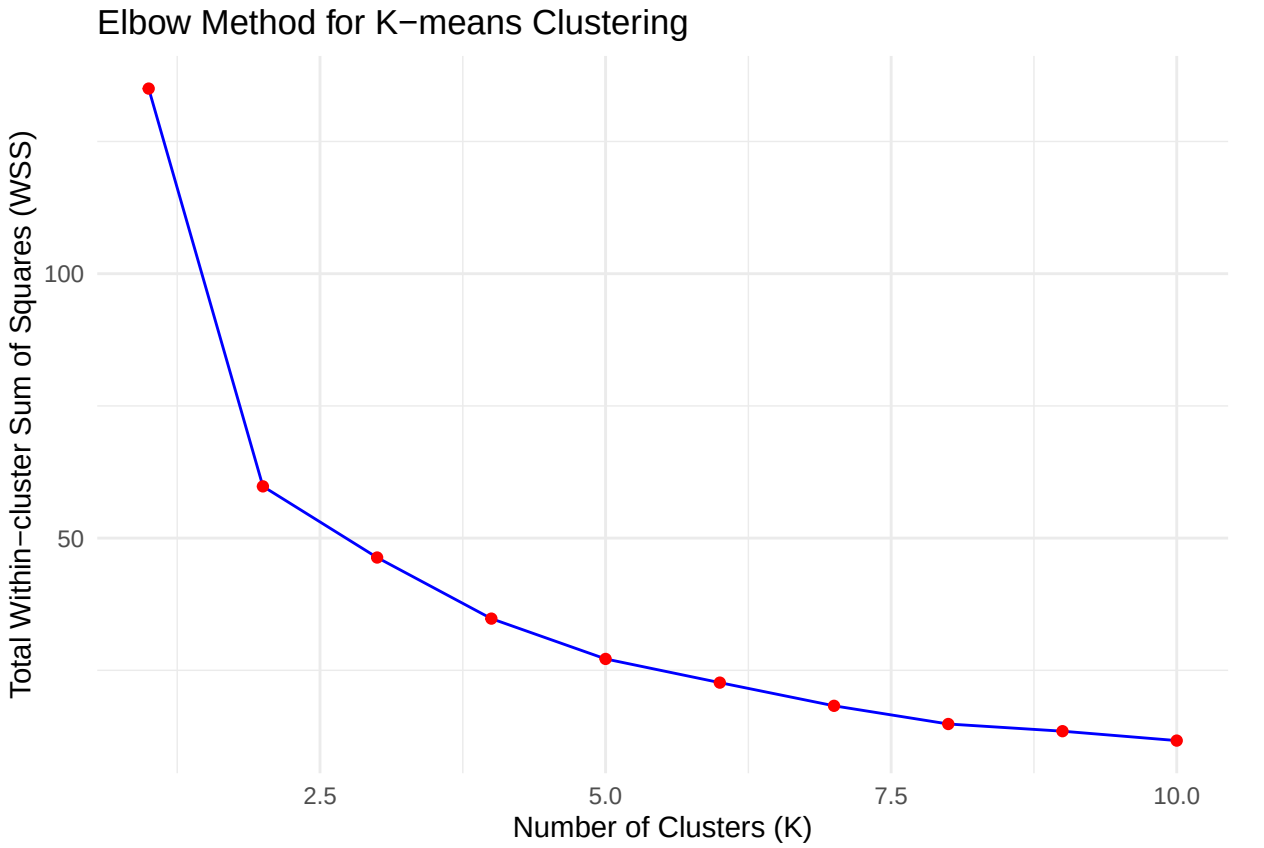
1.B

```
# (remove non-numeric columns and scale)
cluster_data <- ec_data_clean %>%
  dplyr::select(Work, Price, Salary) %>%
  scale()  # standardize the data to mean 0 and variance 1

set.seed(111)  # Set seed for reproducibility
wss <- sapply(1:10, function(k) { #within-cluster sum of squares
  kmeans(cluster_data, centers = k, nstart = 20)$tot.withinss
})

# Plot the elbow method
ggplot(data.frame(k = 1:10, WSS = wss), aes(x = k, y = WSS)) +
  geom_line(color = "blue") +
  geom_point(color = "red") +
  labs(title = "Elbow Method for K-means Clustering",
```

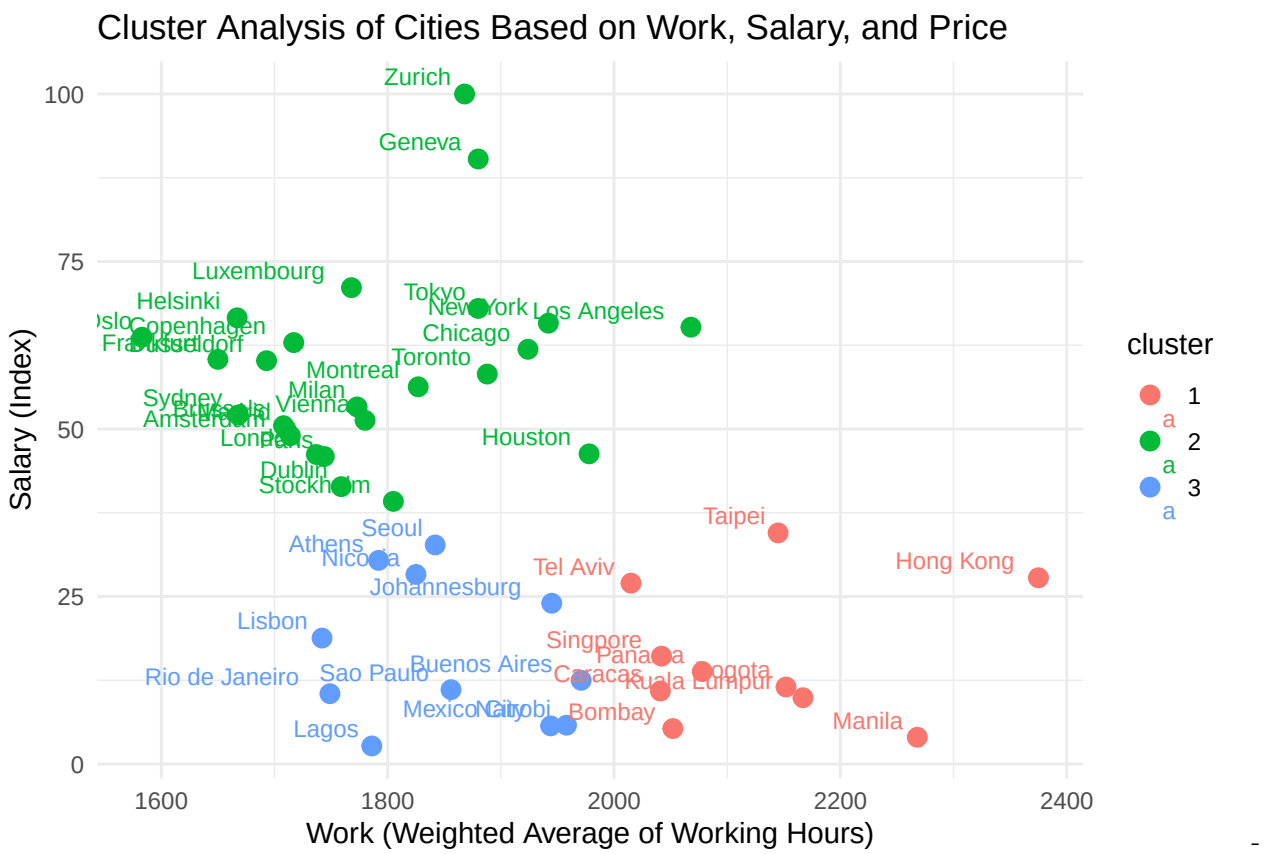
```
       x = "Number of Clusters (K)",
       y = "Total Within-cluster Sum of Squares (WSS)") +
  theme_minimal()
```




```
set.seed(111)
kmeans_result <- kmeans(cluster_data, centers = 3, nstart = 20)

# Add cluster assignment to the data
ec_data_clean$cluster <- as.factor(kmeans_result$cluster)

# Visualize clusters
ggplot(ec_data_clean, aes(x = Work, y = Salary, color = cluster)) +
  geom_point(size = 3) +
  geom_text(aes(label = City), hjust = 1.2, vjust = -0.5, size = 3) +
  labs(title = "Cluster Analysis of Cities Based on Work, Salary, and Price",
       x = "Work (Weighted Average of Working Hours)",
       y = "Salary (Index)") +
  theme_minimal()
```



Cluster 1 (Red): Cities in this cluster (e.g., Hong Kong, Manila, Bogota) are characterized by high work hours and low salaries. They might represent emerging markets where the cost of living and salaries are lower, but work demands are higher. - Cluster 2 (Blue): Cities like Mexico City, Seoul, and Athens are similar in that they have moderate working hours and moderate to low salaries. These cities often show a developing or transitional economic status. - Cluster 3 (Green): Cities like Zurich, New York, and Tokyo are similar in offering high salaries and lower working hours, suggesting that they are developed economies with better work-life balance and higher living standards.

1.C

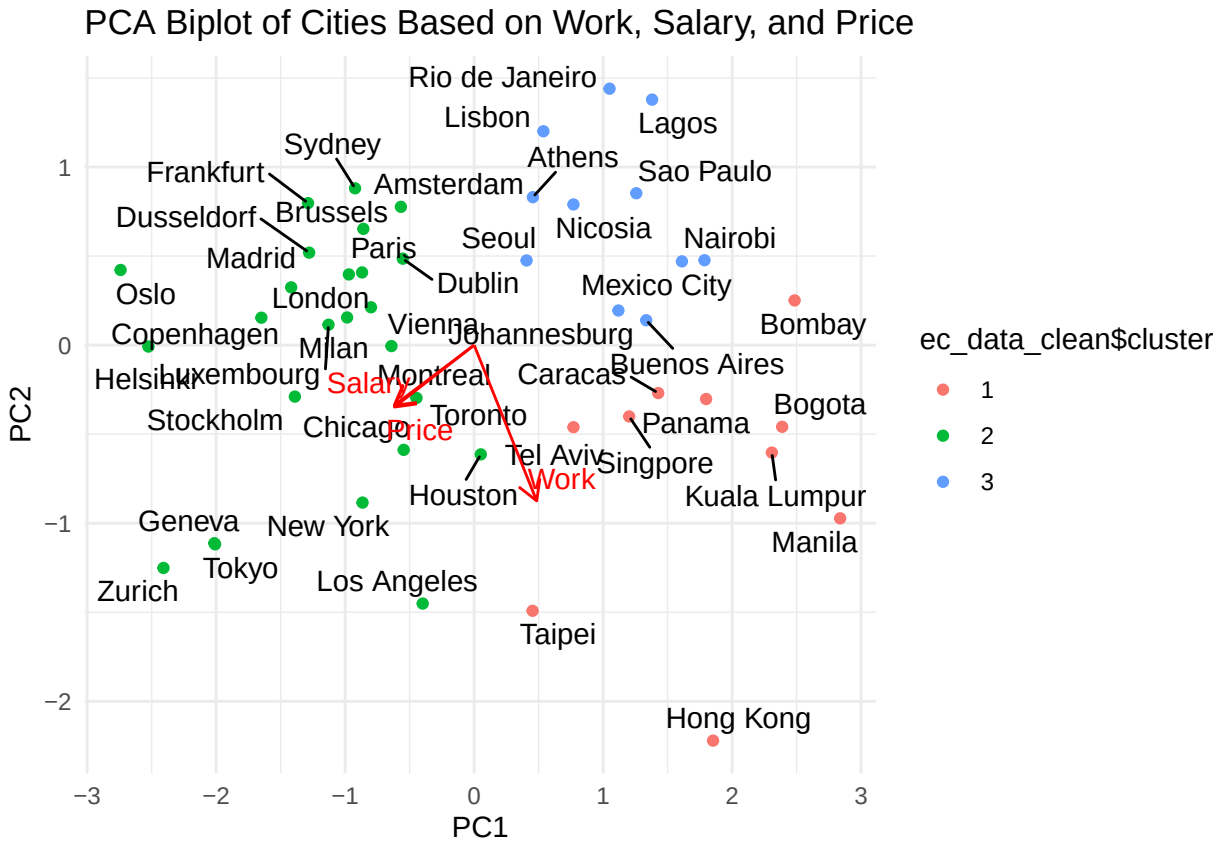
```
pca <- prcomp(cluster_data, scale = TRUE)

loadings <- pca$rotation
loadings_data <- data.frame(Variable = rownames(loadings), PC1 = loadings[, 1], PC2 = loadings[, 2])

# perform Dimensionality reduction
scores <- pca$x
pca_data <- data.frame(City = ec_data_clean$City, PC1 = scores[, 1], PC2 = scores[, 2])

# plot the PCA biplot
ggplot() +
  geom_point(data = pca_data, aes(x = PC1, y = PC2, color = ec_data_clean$cluster)) +
  geom_text_repel(data = pca_data, aes(x = PC1, y = PC2, label = City)) +
  geom_segment(data = loadings_data, aes(x = 0, y = 0, xend = PC1, yend = PC2),
               arrow = arrow(length = unit(0.3, "cm")), color = "red") +
  geom_text_repel(data = loadings_data, aes(x = PC1, y = PC2, label = Variable), color = "red") +
  labs(title = "PCA Biplot of Cities Based on Work, Salary, and Price",
```

```
       x = "PC1",
       y = "PC2") +
  theme_minimal()
```



The PCA biplot reveals how cities differ in terms of Work, Salary, and Price by projecting them onto two principal components (PC1 and PC2). PC1 primarily captures the variation in working hours, with cities like Hong Kong and Manila positioned on the right, indicating high working hours, while cities like Zurich and Geneva are on the left, showing fewer working hours. PC2 captures the variation in salary and price, with cities like Lisbon and Rio de Janeiro higher on the plot, indicating higher salary and price levels, while Bombay and Seoul show lower values. The biplot visually confirms the clustering pattern, where cities with similar work-life and economic profiles are grouped together, such as high-salary, low-work cities in Cluster 3 (green) and high-work, low-salary cities in Cluster 1 (red).

1.D (LDA & QDA)

```
data <- ec_data_clean %>%
  dplyr::select(Work, Price, SalaryCat) %>%
  na.omit()  # Remove rows with missing data

# Convert SalaryCat to a factor
data$SalaryCat <- as.factor(data$SalaryCat)

# LDA model
lda_model <- lda(SalaryCat ~ Work + Price, data = data)

# Create a grid of values for Work and Price to plot decision boundaries
x_min <- min(data$Work) - 1
```

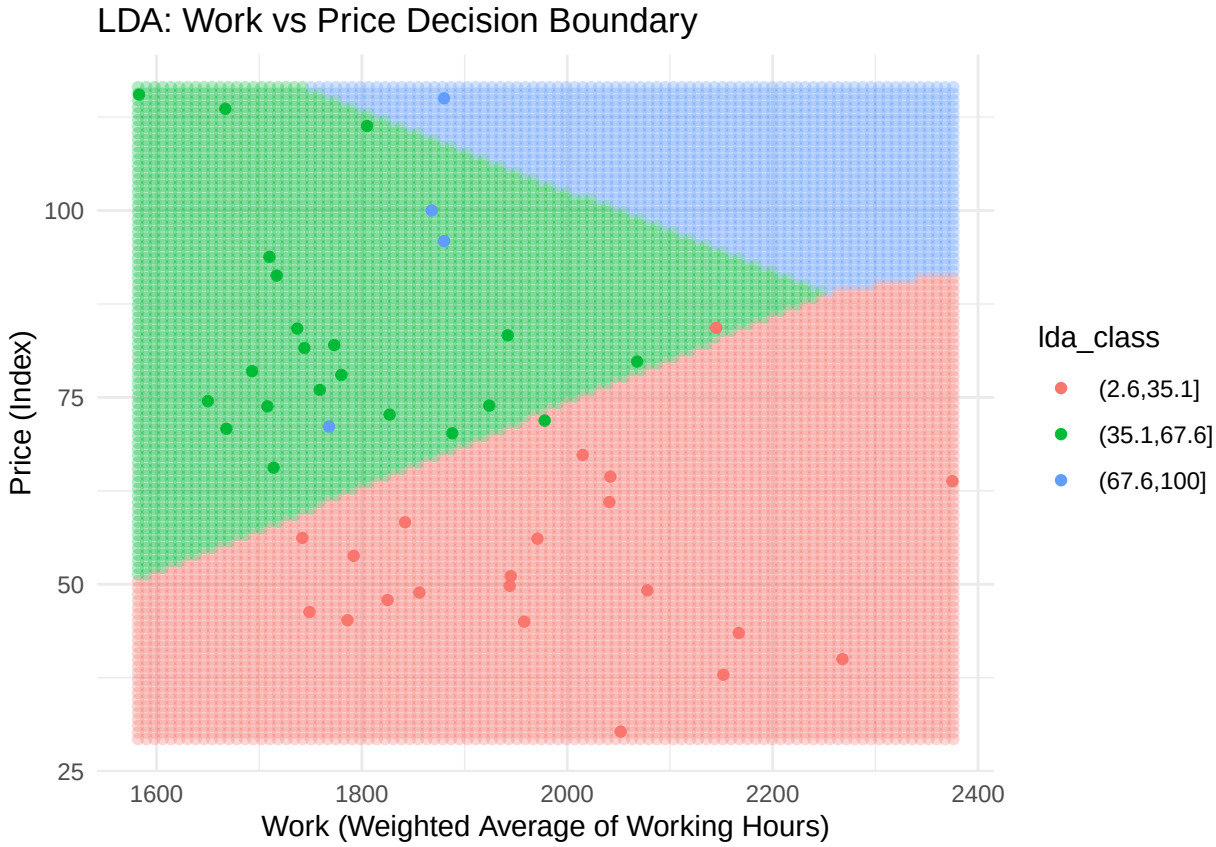
```r
x_max <- max(data$Work) + 1
y_min <- min(data$Price) - 1
y_max <- max(data$Price) + 1

grid <- expand.grid(Work = seq(x_min, x_max, length.out = 100),
                    Price = seq(y_min, y_max, length.out = 100))

# Predict class probabilities and categories for LDA and QDA
lda_pred <- predict(lda_model, grid)
grid$lda_class <- lda_pred$class

# Plot LDA decision boundaries
lda_plot <- ggplot() +
  geom_point(data = grid, aes(x = Work, y = Price, color = lda_class), alpha = 0.3) +
  geom_point(data = data, aes(x = Work, y = Price, color = SalaryCat)) +
  labs(title = "LDA: Work vs Price Decision Boundary",
       x = "Work (Weighted Average of Working Hours)",
       y = "Price (Index)") +
  theme_minimal()

lda_plot
```

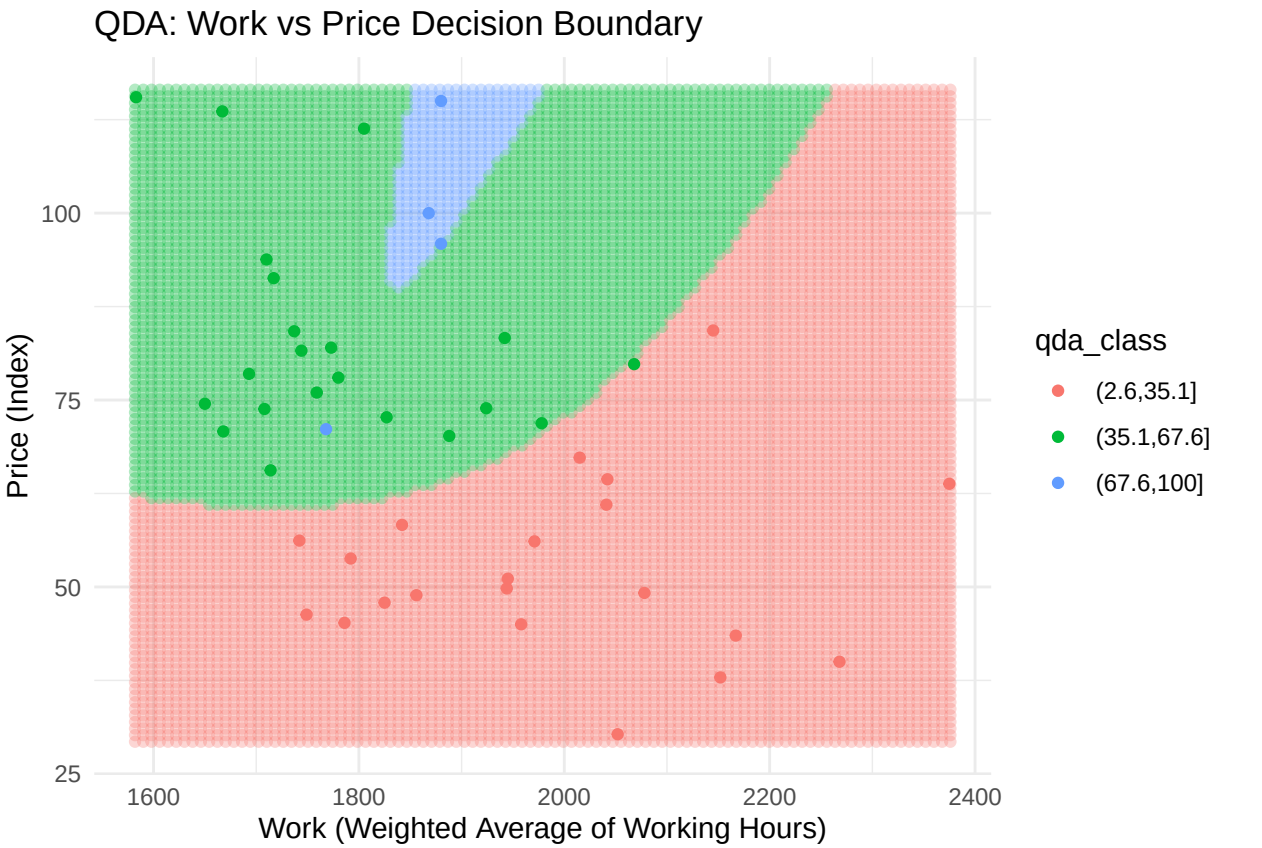



```r
#qda
qda_model <- qda(SalaryCat ~ Work + Price, data = data)

qda_pred <- predict(qda_model, grid)
grid$qda_class <- qda_pred$class
```

```r
# Plot QDA decision boundaries
qda_plot <- ggplot() +
  geom_point(data = grid, aes(x = Work, y = Price, color = qda_class), alpha = 0.3) +
  geom_point(data = data, aes(x = Work, y = Price, color = SalaryCat)) +
  labs(title = "QDA: Work vs Price Decision Boundary",
       x = "Work (Weighted Average of Working Hours)",
       y = "Price (Index)") +
  theme_minimal()

qda_plot
```



LDA vs QDA #### LDA The salary categories (35.1, 67.6] (green) and (67.6, 100] (blue) are more clearly separated. The linear boundary works well in this case, especially in the middle and upper categories. For example, the green region is formed between about 1700 and 2100 work hours, while blue is in the lower working hours and higher prices. However, the lower salary category (2.6, 35.1] (red) tends to dominate more at the extreme ranges of Work.

QDA The regions are similar to LDA, but QDA produces more curved and nuanced boundaries, especially in areas where there might be overlaps between salary categories. The flexibility of QDA allows for a more detailed separation in these regions. For example, the red region expands differently in QDA, accommodating some data points that were likely misclassified in LDA due to its rigid linear constraints.

Conclusion

For this data, QDA fits better. We can see that it effectively captures the non-linear relationships between Work and Price (this is especially marked for our Blue class, which LDA ineffectively captured), providing

more detailed decision boundaries that better separate the salary categories. LDA, while useful for linearly separable data, struggles with the complexity of this dataset, leading to some misclassifications and less distinct boundaries.

1.E

- EDA provided insights into the relationships between Work, Salary, and Price, showing clear patterns across cities.
- K-means clustering revealed three distinct groups of cities with different economic profiles based on their work hours and salary levels.
- PCA reduced the dimensionality of the data and visualized how these cities are distributed across the two main sources of variation (Work and Salary/Price), confirming the clustering results.
- LDA and QDA provided classification models for predicting salary categories based on Work and Price. QDA's quadratic boundaries offered superior classification accuracy, while LDA's linear approach struggled with the non-linear relationships in the data. QDA proved to be the most effective classification method for this dataset, while PCA and K-means clustering provided strong tools for understanding the structure of the data and identifying patterns among cities.

Problem 2 (classification trees & random forests)

Load the R data file

```
# Load the data
load("/Users/alimos313/Documents/studies/phd/university/courses/stat-modelling/StatModelEx/day4/data/wi
# Check the structure of the data
str(wine)
```

```
## 'data.frame':    178 obs. of  14 variables:
##  $ Type           : Factor w/ 3 levels "1","2","3": 1 1 1 1 1 1 1 1 1 1 ...
##  $ Alcohol        : num  14.2 13.2 13.2 14.4 13.2 ...
##  $ Malic          : num  1.71 1.78 2.36 1.95 2.59 1.76 1.87 2.15 1.64 1.35 ...
##  $ Ash            : num  2.43 2.14 2.67 2.5 2.87 2.45 2.45 2.61 2.17 2.27 ...
##  $ Alcalinity     : num  15.6 11.2 18.6 16.8 21 15.2 14.6 17.6 14 16 ...
##  $ Magnesium      : int  127 100 101 113 118 112 96 121 97 98 ...
##  $ Phenols        : num  2.8 2.65 2.8 3.85 2.8 3.27 2.5 2.6 2.8 2.98 ...
##  $ Flavanoids     : num  3.06 2.76 3.24 3.49 2.69 3.39 2.52 2.51 2.98 3.15 ...
##  $ Nonflavanoids  : num  0.28 0.26 0.3 0.24 0.39 0.34 0.3 0.31 0.29 0.22 ...
##  $ Proanthocyanins: num  2.29 1.28 2.81 2.18 1.82 1.97 1.98 1.25 1.98 1.85 ...
##  $ Color          : num  5.64 4.38 5.68 7.8 4.32 6.75 5.25 5.05 5.2 7.22 ...
##  $ Hue            : num  1.04 1.05 1.03 0.86 1.04 1.05 1.02 1.06 1.08 1.01 ...
##  $ Dilution       : num  3.92 3.4 3.17 3.45 2.93 2.85 3.58 3.58 2.85 3.55 ...
##  $ Proline        : int  1065 1050 1185 1480 735 1450 1290 1295 1045 1045 ...
```

2.A

```
set.seed(111)
# Split the data into training and testing sets
train_index <- sample(1:nrow(wine), 0.7 * nrow(wine))
train_data <- wine[train_index, ]
test_data <- wine[-train_index, ]
```

classification trees

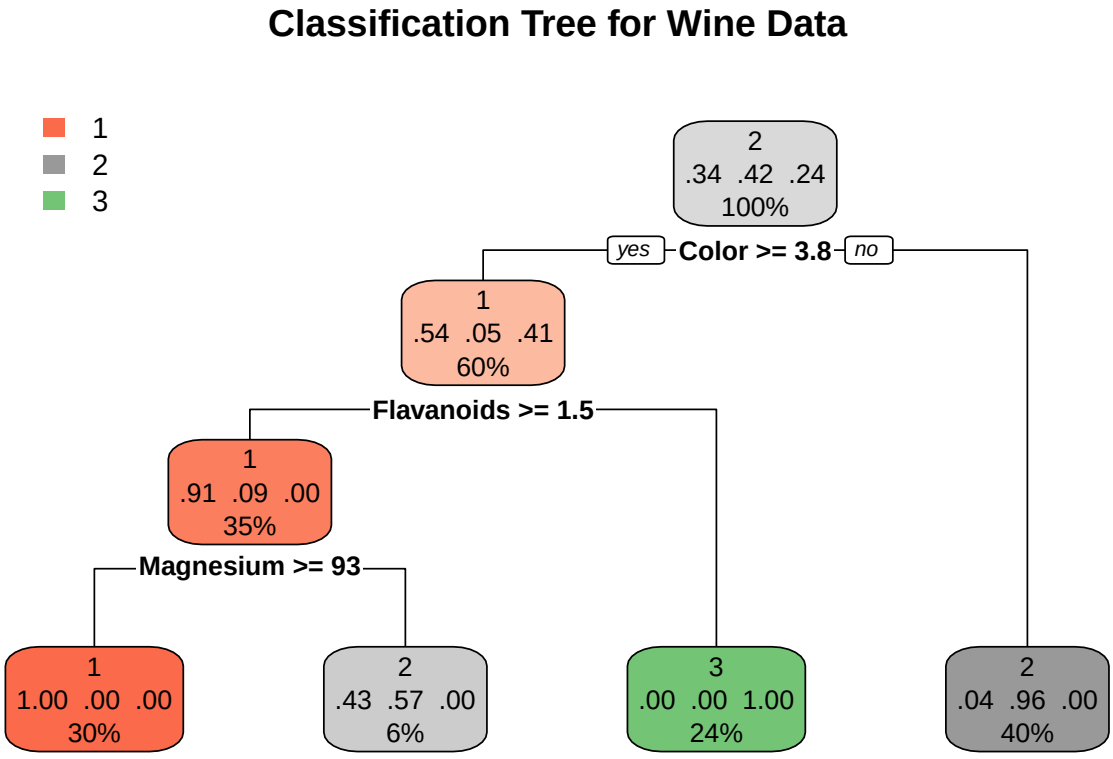
```
tree_model <- rpart(Type ~ ., data = train_data, method = "class")
tree_pred <- predict(tree_model, test_data, type = "class")
confusion_matrix_tree <- table(tree_pred, test_data$Type)
confusion_matrix_tree
```

```
##
## tree_pred  1  2  3
##         1 14  2  1
##         2  3 16  0
##         3  0  1 17
```

```
tree_accuracy <- confusionMatrix(tree_pred, test_data$Type)$overall["Accuracy"]
print(paste("Classification Tree Accuracy:", round(tree_accuracy * 100, 2), "%"))
```

```
## [1] "Classification Tree Accuracy: 87.04 %"
```

```
rpart.plot(tree_model, main = "Classification Tree for Wine Data")
```



Random Forest

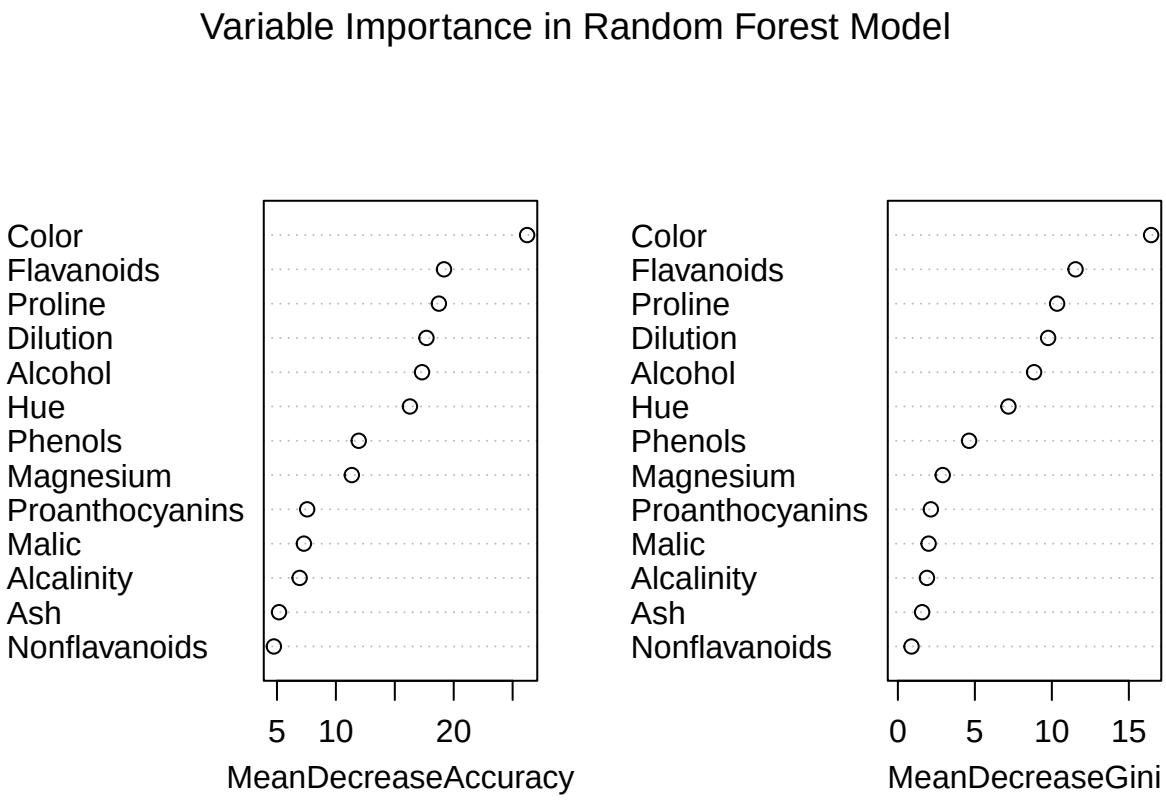
```
rf_model <- randomForest(Type ~ ., data = train_data, ntree = 500, mtry = 3, importance = TRUE)

# Predict on the test set
rf_pred <- predict(rf_model, test_data)

# eval
rf_accuracy <- confusionMatrix(rf_pred, test_data$Type)$overall["Accuracy"]
print(paste("Random Forest Accuracy:", round(rf_accuracy * 100, 2), "%"))
```

```
## [1] "Random Forest Accuracy: 98.15 %"
```

```
varImpPlot(rf_model, main = "Variable Importance in Random Forest Model")
```



Cross validation

```
# Define the cross-validation function
cross_validate_rf <- function(data, target_variable, n_folds = 10, model = "rf") {
  # Convert target variable to a factor
  data[[target_variable]] <- as.factor(data[[target_variable]])

  # Create a vector to store accuracy for each fold
  accuracies <- numeric(n_folds)

  # Create folds
  set.seed(123)  # For reproducibility
  fold_indices <- sample(1:n_folds, nrow(data), replace = TRUE)

  # Cross-validation loop
  for (fold in 1:n_folds) {
    # Split data into training and testing sets
    test_data <- data[fold_indices == fold, ]
    train_data <- data[fold_indices != fold, ]

    if (model == "rf"){
        # Train the Random Forest model
        classification_model <- randomForest(as.formula(paste(target_variable, "~ .")),
                                data = train_data,
                                importance = TRUE)
```

```
    } else if (model == "ctree"){
        classification_model <- ctree(Type ~ .,  data=wine,
                                      control = ctree_control(maxdepth = 5))
    }

    # Predict on the test data
    predictions <- predict(classification_model, test_data)

    # Calculate accuracy
    accuracies[fold] <- mean(predictions == test_data[[target_variable]])
  }

  # Calculate the average accuracy
  average_accuracy <- mean(accuracies)

  # Print results
  cat("Average accuracy from cross-validation: ", average_accuracy * 100, "%\n")

  # Return the average accuracy
  return(average_accuracy)
}


## run the function for randomforest model
cross_validate_rf(wine, "Type", 10, "rf")
```

```
## Average accuracy from cross-validation:  98.10714 %
```

```
## [1] 0.9810714
```

```
## run the function for classification tree model
cross_validate_rf(wine, "Type", 10, "ctree")
```

```
## Average accuracy from cross-validation:  94.96768 %
```

```
## [1] 0.9496768
```

Conclusion

plots

- Model Complexity: The classification tree is simpler and easier to interpret but sacrifices accuracy for simplicity. In contrast, random forest is more complex, but it captures more nuances in the data and provides better overall performance.
- Accuracy: Random forest outperforms the classification tree in terms of accuracy due to its ability to generalize better and handle the variance in the data.
- Variable Importance: Both models identify Color and Flavanoids as crucial variables, but random forest adds further insights into the importance of Proline and Dilution, which are not as evident in the single classification tree.

Cross-validation The classification tree performs reasonably well for predicting Type 1 wines, with an accuracy of over 85%. However, its performance drops for Type 2 wines, and more significantly for Type 3, where the accuracy is around 50%. This drop in accuracy for Type 3 suggests that the tree struggles to capture the patterns that distinguish Type 3 wines from others, likely due to the complexity of the data and the decision tree's limitations in handling more complex, non-linear relationships.

The Random Forest model significantly improves the accuracy of wine type prediction, achieving an overall accuracy of over 90%. The Random Forest model's ability to combine multiple decision trees and reduce overfitting helps capture the complex relationships in the data more effectively, leading to better predictions across all wine types.