ISLR (2nd Edition) Exercises - Solutions

Justin Tuyisenge (101002)

June 12, 2025

Contents


```r
library(ISLR2)      # For datasets
library(dplyr)      # For data manipulation
library(ggplot2)    # For plotting
library(MASS)       # For LDA, QDA
library(caret)      # For data partitioning
library(e1071)      # For Naive Bayes
library(class)      # For KNN
```

Chapter 2 - Exercise Questions

Exercise 4

I collect a set of data ($n = 100$ observations) containing a single predictor and a quantitative response. I then fit a linear regression model to the data, as well as a separate cubic regression, i.e., $Y = \beta_0+\beta_1X+\beta_2X^2+\beta_3X^3+\epsilon$.

(a) Suppose that the true relationship between X and Y is linear, i.e. $Y = \beta_0 + \beta_1X + \epsilon$. Consider the training residual sum of squares (RSS) for the linear regression, and also the training RSS for the cubic regression. Would we expect one to be lower than the other, would we expect them to be the same, or is there not enough information to tell? Justify your answer.

Answer: We would expect the training RSS for the cubic regression to be lower than or equal to the training RSS for the linear regression.

The cubic regression model is more flexible and includes the linear model as a special case (when $\beta_2 = 0$ and $\beta_3 = 0$). When fitting a model using least squares, the objective is to minimize the RSS on the training data. A more flexible model, by definition, can fit the training data at least as well as, and typically better than, a less flexible model. Therefore, its training RSS will be smaller or, in very specific cases, equal.

(b) Answer (a) using test rather than training RSS.

Answer: We would expect the test RSS for the linear regression to be lower than the test RSS for the cubic regression.

Since the true relationship is linear, the linear regression model has low bias and captures the underlying structure correctly. The cubic model, being more flexible, has higher variance. It will use its additional degrees of freedom to fit the random noise in the training data, leading to overfitting. This overfitting means it performs well on training data but poorly on unseen test data. The linear model, which correctly matches the complexity of the true relationship, will likely have a lower test RSS due to its lower variance.

(c) Suppose that the true relationship between X and Y is not linear, but we don't know how far it is from linear. Consider the training RSS for the linear regression, and also the training RSS for the cubic regression. Would we expect one to be lower than the other, would we expect them to be the same, or is there not enough information to tell? Justify your answer.

Answer: We would expect the training RSS for the cubic regression to be lower than or equal to the training RSS for the linear regression.

This reasoning is the same as in part (a). The cubic regression model is more flexible and contains the linear model as a special case. Therefore, it will always achieve a training RSS that is less than or equal to that of the less flexible linear model, regardless of the true non-linear relationship.

(d) Answer (c) using test rather than training RSS.

Answer: In this scenario, there is not enough information to tell which model's test RSS would be lower without knowing the true degree of non-linearity and the amount of noise in the data.

This is a classic bias-variance trade-off scenario:

- **Linear Regression:** If the true relationship is non-linear, the linear model will have high bias.
- **Cubic Regression:** The cubic model has the flexibility to capture non-linearity, so it will have lower bias than the linear model. However, it also has higher variance.

Whether the cubic regression's lower bias outweighs its higher variance, or vice-versa, depends on the specifics of the true non-linearity and noise level. If the non-linearity is mild, the linear model might still perform better due to lower variance. If the non-linearity is strong, the cubic model would likely perform better despite higher variance due to its much lower bias.

Exercise 7

It is claimed in the text that in the case of a regression of Y onto X, the R^2 coefficient (3.17) is equal to the square of the correlation between X and Y (3.18). Prove that this is indeed the case. For simplicity, you can assume that $\bar{x} = 0$ and $\bar{y} = 0$.

Proof:

We want to prove that for simple linear regression of Y onto X, $R^2 = (\text{Cor}(X, Y))^2$, given $\bar{x} = 0$ and $\bar{y} = 0$.

Definitions (with $\bar{x} = 0, \bar{y} = 0$):

- $\text{TSS} = \sum y_i^2$ (since $\bar{y} = 0$)
- $\text{Cor}(X, Y) = \frac{\sum x_i y_i}{\sqrt{\sum x_i^2 \sum y_i^2}}$
- Simple Linear Regression: $\hat{Y}_i = \hat{\beta}_0 + \hat{\beta}_1 X_i$. Given $\bar{x} = 0$, $\bar{y} = 0$, so $\hat{\beta}_0 = 0$. Thus, $\hat{Y}_i = \hat{\beta}_1 X_i$.
- Slope coefficient: $\hat{\beta}_1 = \frac{\sum x_i y_i}{\sum x_i^2}$
- $R^2 = 1 - \frac{\text{RSS}}{\text{TSS}}$

Derivation:

1. **Express RSS:**

$$\text{RSS} = \sum (y_i - \hat{y}_i)^2 = \sum (y_i - \hat{\beta}_1 x_i)^2$$

$$\text{RSS} = \sum (y_i^2 - 2\hat{\beta}_1 x_i y_i + \hat{\beta}_1^2 x_i^2)$$

$$\text{RSS} = \sum y_i^2 - 2\hat{\beta}_1 \sum x_i y_i + \hat{\beta}_1^2 \sum x_i^2$$

2. **Substitute** $\hat{\beta}_1$ into RSS:

$$\text{RSS} = \sum y_i^2 - 2\left(\frac{\sum x_i y_i}{\sum x_i^2}\right)\sum x_i y_i + \left(\frac{\sum x_i y_i}{\sum x_i^2}\right)^2 \sum x_i^2$$

$$\text{RSS} = \sum y_i^2 - 2\frac{(\sum x_i y_i)^2}{\sum x_i^2} + \frac{(\sum x_i y_i)^2}{\sum x_i^2}$$

$$\text{RSS} = \sum y_i^2 - \frac{(\sum x_i y_i)^2}{\sum x_i^2}$$

3. **Substitute RSS into** R^2 formula:

$$R^2 = 1 - \frac{\text{RSS}}{\text{TSS}} = 1 - \frac{\sum y_i^2 - \frac{(\sum x_i y_i)^2}{\sum x_i^2}}{\sum y_i^2}$$

$$R^2 = 1 - \left(1 - \frac{\frac{(\sum x_i y_i)^2}{\sum x_i^2}}{\sum y_i^2}\right)$$

$$R^2 = \frac{\frac{(\sum x_i y_i)^2}{\sum x_i^2}}{\sum y_i^2}$$

$$R^2 = \frac{(\sum x_i y_i)^2}{\sum x_i^2 \sum y_i^2}$$

4. **Square the correlation coefficient:**

$$(\text{Cor}(X,Y))^2 = \left(\frac{\sum x_i y_i}{\sqrt{\sum x_i^2 \sum y_i^2}}\right)^2$$

$$= \frac{(\sum x_i y_i)^2}{\sum x_i^2 \sum y_i^2}$$

Since both expressions are equivalent, $R^2 = (\text{Cor}(X,Y))^2$ is proven for simple linear regression under the assumption $\bar{x} = 0$ and $\bar{y} = 0$.

Chapter 3 - Exercise Questions

Exercise 4: The Curse of Dimensionality

When the number of predictors p is large, there tends to be a deterioration in the performance of KNN and other local approaches that perform prediction using only observations that are near the test observation for which a prediction must be made. This phenomenon is known as the curse of dimensionality, and it ties into why non-parametric approaches often perform poorly when p is large. We will now investigate this curse.

(a) Suppose we have a set of observations, each with measurements on $p = 1$ feature, X. We assume that X is uniformly (evenly) distributed on $[0, 1]$. Associated with each observation is a response value. Suppose we wish to predict a test observation's response using only observations that are within 10% of the range of X closest to that test observation. For instance, to predict the response for a test observation with $X = 0.6$, we will use observations in the range $[0.55, 0.65]$. On average, what fraction of observations will we use to make the prediction?

Answer: The range of X is $1 - 0 = 1$. A window of 10% of this range is 0.1.

For a uniform distribution on $[0, 1]$, the fraction of observations in an interval of length L is $L/(\text{Total Range})$.

The fraction is $0.1/1 = \mathbf{0.1}$.

On average, we will use 10% of the available observations.

(b) Now suppose we have a set of observations, each with measurements on $p = 2$ features, X_1 and X_2. We assume that (X_1, X_2) are uniformly distributed on $[0, 1] \times [0, 1]$. We wish to predict a test observation's response using only observations that are within 10% of the range of X_1 and within 10% of the range of X_2 closest to that test observation. For instance, to predict the response for a test observation with $X_1 = 0.6$ and $X_2 = 0.35$, we will use observations in the range $[0.55, 0.65]$ for X_1 and in the range $[0.3, 0.4]$ for X_2. On average, what fraction of the observations will we use to make the prediction?

Answer: The total space is a unit square with area $1 \times 1 = 1$.

The region of interest is a square with side lengths 0.1 for X_1 and 0.1 for X_2.

The volume of this region is $0.1 \times 0.1 = 0.1^2 = \mathbf{0.01}$.

On average, we will use 1% of the available observations.

(c) Now suppose we have a set of observations with $p = 100$ features. Again, the observations are uniformly distributed on each feature, and again, each feature ranges in value from 0 to 1. We wish to predict a test observation's response using observations within 10% of each feature's range closest to that test observation. What fraction of the observations will we use to make the prediction?

Answer: The total space is a 100-dimensional hypercube with volume $1^{100} = 1$.

The region of interest is a 100-dimensional hypercube with side lengths 0.1 in each dimension.

The volume of this region is $(0.1)^{100} = \mathbf{1 \times 10^{-100}}$.

On average, we will use an extremely small fraction of the available observations.

(d) Using your answers to parts (a)–(c), argue why there is a drawback of KNN when p is large that there are very few training observations "near" any given test observation.

Argument: As the number of dimensions (p) increases, the fraction of observations that fall within a fixed "local" neighborhood (defined as 10% of the range along each axis) around a test observation shrinks exponentially:

- For $p = 1$, 10% of observations are "near".
- For $p = 2$, only 1% are "near".
- For $p = 100$, an extremely small fraction (1×10^{-100}) is "near".

This illustrates the curse of dimensionality: in high-dimensional spaces, data points become extremely sparse. For KNN to find enough neighbors (e.g., K=5 or 10), the definition of "near" must be greatly relaxed, meaning the neighborhood radius has to expand to cover a large portion of the feature space. When neighbors are no longer truly "local," their response values may not represent the test observation's immediate vicinity, leading to poor prediction performance for KNN.

(e) Now suppose we wish to predict for a test observation by creating a p-dimensional hypercube centered around the test observation that contains, on average, 10% of the training observations. For $p = 1, 2$, and 100, what is the length of each side of the hypercube? Comment on your findings.

Answer: Let L be the length of each side of the p-dimensional hypercube. Its volume is L^p.

To contain 10% (0.1) of the observations (uniformly distributed in a unit hypercube), the volume must be 0.1.

So, $L^p = 0.1 \implies L = (0.1)^{1/p}$.

- For $p = 1$: $L = (0.1)^{1/1} = \mathbf{0.1}$
- For $p = 2$: $L = (0.1)^{1/2} = \sqrt{0.1} \approx \mathbf{0.316}$
- For $p = 100$: $L = (0.1)^{1/100} \approx \mathbf{0.977}$

Comment: This demonstrates that to maintain a constant fraction of neighbors (10%), the side length of the hypercube must increase significantly as the number of dimensions grows:

- In 1D, we only need to extend 0.1 units in each direction.
- In 2D, we need to extend about 0.316 units (31.6% of the range) in each direction.
- In 100D, we need to extend about 0.977 units (almost 97.7% of the range) in each direction to capture just 10% of the data.

This shows that in high dimensions, the "local" neighborhood needed to find enough neighbors becomes so large that it spans nearly the entire feature space. The observations within this "neighborhood" are no longer truly "near" or "similar" to the test point, undermining the core principle of local methods like KNN.

Exercise 9: Odds

This problem deals with odds.

(a) On average, what fraction of people with an odds of 0.37 of defaulting on their credit card payment will actually default?

Answer: Odds are defined as $\frac{p}{1-p}$, where p is the probability.

Given Odds = 0.37:

$$0.37 = \frac{p}{1-p}$$

$$0.37(1-p) = p$$

$$0.37 - 0.37p = p$$

$$0.37 = 1.37p$$

$$p = \frac{0.37}{1.37} \approx \mathbf{0.270}$$

Approximately 27.0% of people will default.

(b) Suppose an individual has a 16% probability of defaulting on their credit card payment. What are the odds that they will default?

Answer: Given probability $p = 0.16$.

Odds $= \frac{p}{1-p} = \frac{0.16}{1-0.16} = \frac{0.16}{0.84} \approx \mathbf{0.190}$

The odds that they will default are approximately 0.190.

Exercise 14: Predicting Gas Mileage

In this problem, you will create a model to predict whether a car gets high or low gas mileage based on the Auto dataset.

(a) Create a binary variable, mpg01, that contains a 1 if mpg is above its median, and a 0 if mpg is below its median.

```
# Load Auto dataset
data(Auto)

# Convert horsepower to numeric (it might be a factor)
Auto$horsepower <- as.numeric(as.character(Auto$horsepower))

# Remove rows with missing values
Auto_data <- na.omit(Auto)

# Calculate median MPG
```

```r
median_mpg <- median(Auto_data$mpg)
cat("Median MPG:", median_mpg, "\n")
```

```
## Median MPG: 22.75
```

```r
# Create binary variable mpg01
Auto_data <- Auto_data %>%
  mutate(mpg01 = ifelse(mpg > median_mpg, 1, 0))

# Display first few rows and summary
head(Auto_data)
```

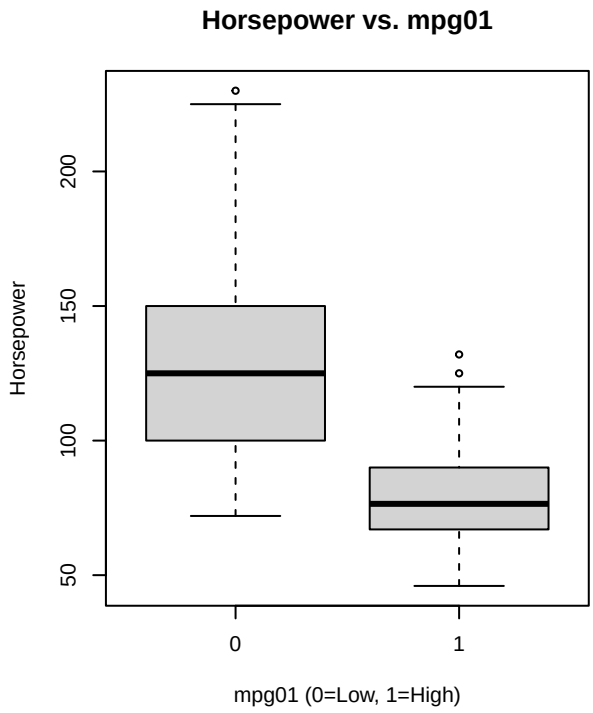
```
##   mpg cylinders displacement horsepower weight acceleration year origin
## 1  18         8          307        130   3504         12.0   70      1
## 2  15         8          350        165   3693         11.5   70      1
## 3  18         8          318        150   3436         11.0   70      1
## 4  16         8          304        150   3433         12.0   70      1
## 5  17         8          302        140   3449         10.5   70      1
## 6  15         8          429        198   4341         10.0   70      1
##                       name mpg01
## 1 chevrolet chevelle malibu     0
## 2         buick skylark 320     0
## 3        plymouth satellite     0
## 4             amc rebel sst     0
## 5               ford torino     0
## 6          ford galaxie 500     0
```

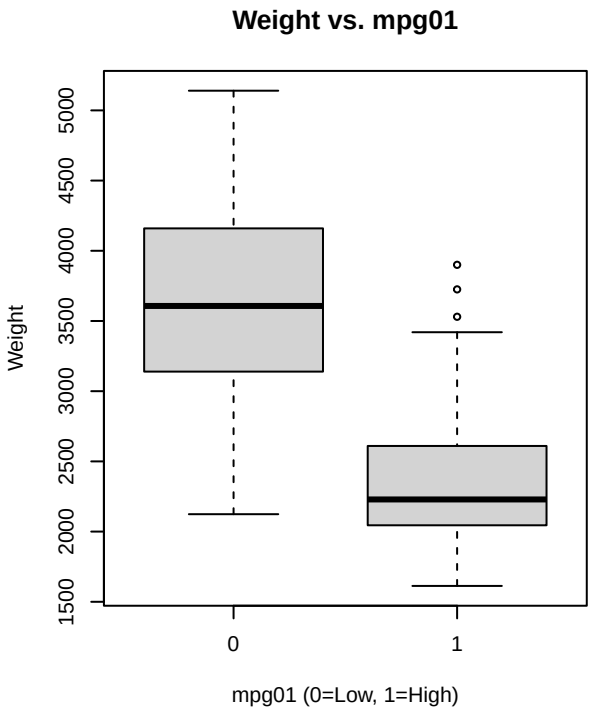
```r
table(Auto_data$mpg01)
```

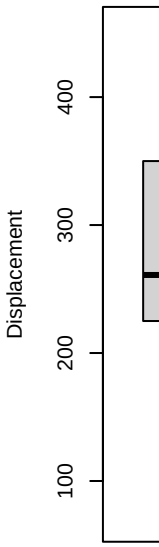
```
##
##   0   1
## 196 196
```

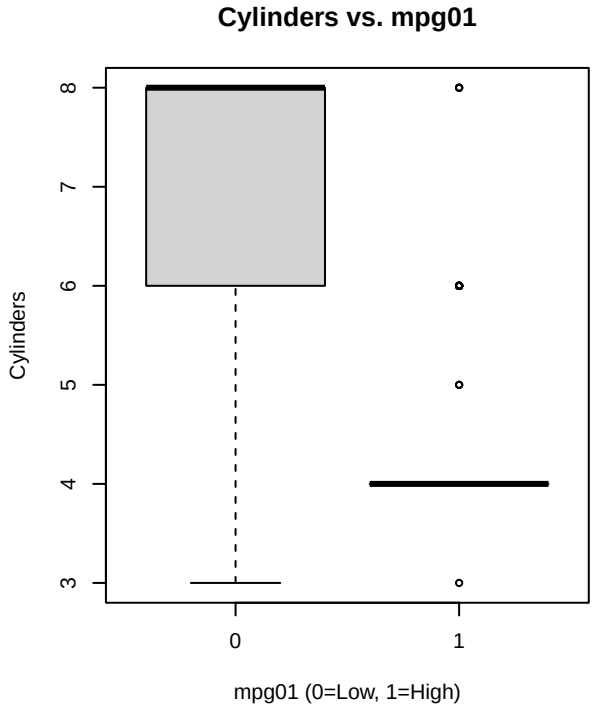
(b) Explore the data visually to investigate the association between mpg01 and other features. Which features seem most likely to be useful in predicting mpg01? Scatterplots and boxplots may help answer this question. Describe your observations.

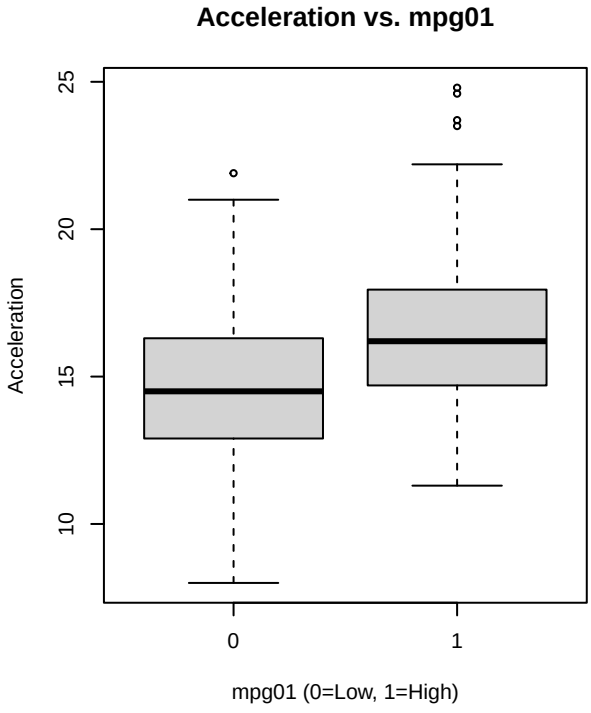
```r
# Create boxplots for continuous variables
par(mfrow = c(2, 3))
boxplot(horsepower ~ mpg01, data = Auto_data, main = "Horsepower vs. mpg01",
        xlab = "mpg01 (0=Low, 1=High)", ylab = "Horsepower")
boxplot(weight ~ mpg01, data = Auto_data, main = "Weight vs. mpg01",
        xlab = "mpg01 (0=Low, 1=High)", ylab = "Weight")
boxplot(displacement ~ mpg01, data = Auto_data, main = "Displacement vs. mpg01",
        xlab = "mpg01 (0=Low, 1=High)", ylab = "Displacement")
boxplot(cylinders ~ mpg01, data = Auto_data, main = "Cylinders vs. mpg01",
        xlab = "mpg01 (0=Low, 1=High)", ylab = "Cylinders")
boxplot(acceleration ~ mpg01, data = Auto_data, main = "Acceleration vs. mpg01",
        xlab = "mpg01 (0=Low, 1=High)", ylab = "Acceleration")
boxplot(year ~ mpg01, data = Auto_data, main = "Year vs. mpg01",
        xlab = "mpg01 (0=Low, 1=High)", ylab = "Year")
```

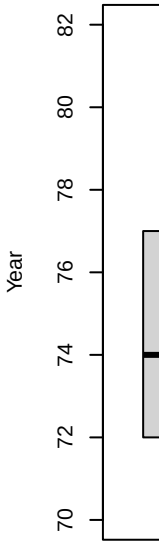










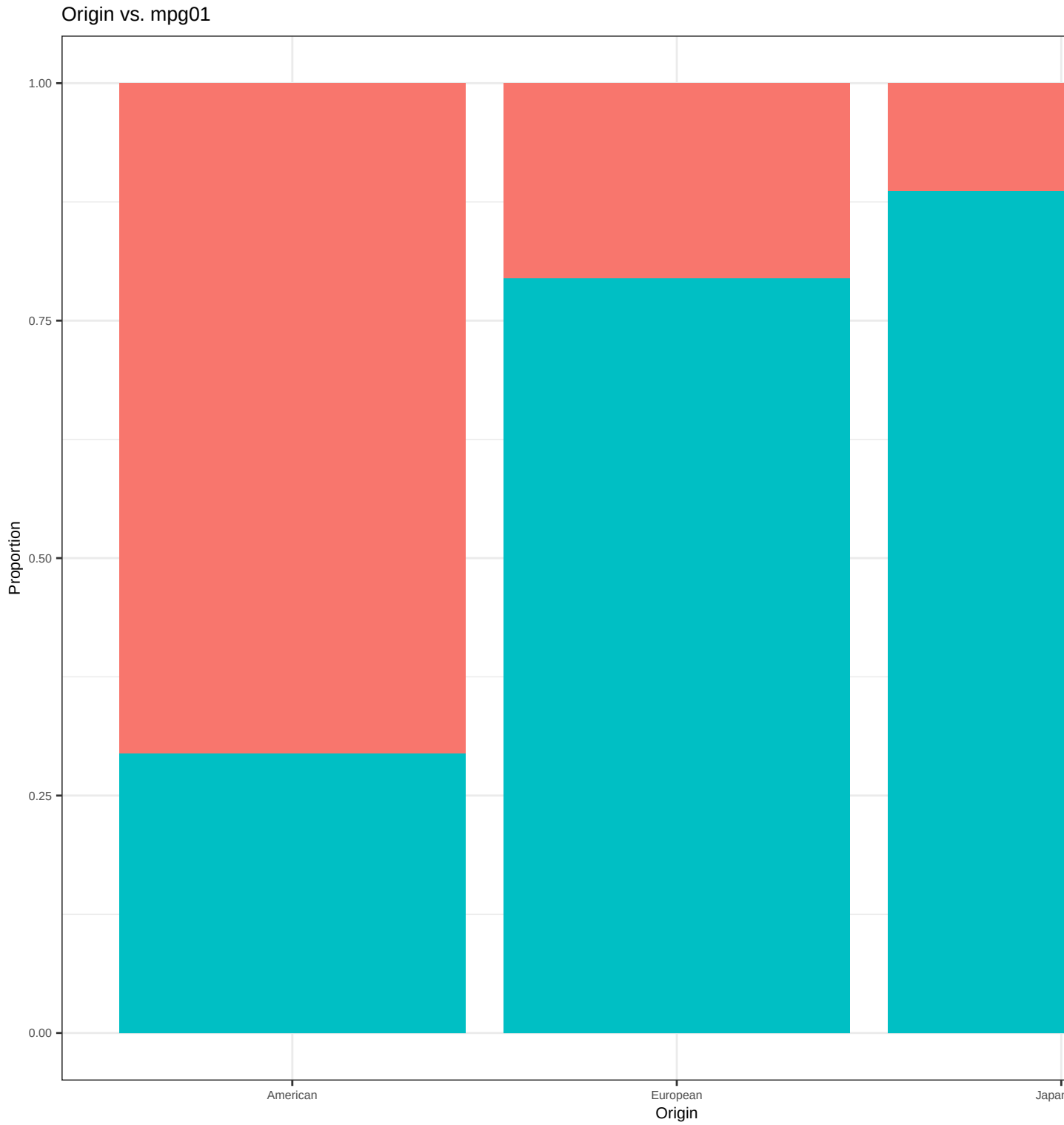



```r
par(mfrow = c(1, 1))

# Create factor for origin and plot
Auto_data$origin_factor <- factor(Auto_data$origin,
                                  levels = c(1, 2, 3),
```

```
                                  labels = c("American", "European", "Japanese"))

# Origin vs mpg01 plot
ggplot(Auto_data, aes(x = origin_factor, fill = factor(mpg01))) +
  geom_bar(position = "fill") +
  labs(title = "Origin vs. mpg01", x = "Origin", y = "Proportion") +
  theme_bw() +
  scale_fill_discrete(name = "mpg01", labels = c("Low", "High")) +
  theme(
    text = element_text(size = 8),
    plot.title = element_text(size = 10),
    axis.title = element_text(size = 8),
    legend.title = element_text(size = 8),
    legend.text = element_text(size = 7)
  )
```



Findings: - **horsepower**, **weight**, **displacement**, and **cylinders** show strong negative associations (higher values of these predictors are linked to low mpg01). - **year** shows a strong positive association (newer cars tend to have high mpg01). - **origin** also shows clear differences, with Japanese cars having a higher proportion of high mpg01. - **acceleration** appears less strongly associated.

Selected Predictors: horsepower, weight, displacement, cylinders, year, and origin.

(c) Split the data into training and test sets.

```
set.seed(42)
train_index <- createDataPartition(Auto_data$mpg01, p = 0.7, list = FALSE)
train_data <- Auto_data[train_index, ]
test_data <- Auto_data[-train_index, ]

cat("Training set dimensions:", dim(train_data), "\n")
```

```
## Training set dimensions: 276 11
```

```
cat("Test set dimensions:", dim(test_data), "\n")
```

```
## Test set dimensions: 116 11
```

(d) Perform LDA on the training data to predict mpg01 using the predictors that seemed most associated with mpg01 in (b). What is the test error of the model obtained?

```
# Ensure origin_factor is properly defined in both datasets
train_data$origin_factor <- factor(train_data$origin)
test_data$origin_factor <- factor(test_data$origin)

# Fit LDA model
lda_model <- lda(mpg01 ~ horsepower + weight + displacement + cylinders + year + origin_factor,
                 data = train_data)

# Make predictions
lda_pred <- predict(lda_model, newdata = test_data)
lda_class <- lda_pred$class

# Confusion matrix
confusion_matrix <- table(Predicted = lda_class, Actual = test_data$mpg01)
print(confusion_matrix)
```

```
##           Actual
## Predicted  0  1
##         0 48  1
##         1 10 57
```

```
# Calculate test error
lda_error <- mean(lda_class != test_data$mpg01)
cat("LDA Test Error:", round(lda_error, 4), "\n")
```

```
## LDA Test Error: 0.0948
```

(e) Perform QDA on the training data to predict mpg01 using the predictors that seemed most associated with mpg01 in (b). What is the test error rate of the model obtained?

```
# Fit QDA model
qda_model <- qda(mpg01 ~ horsepower + weight + displacement + cylinders + year + origin_factor,
                 data = train_data)

# Make predictions
qda_pred <- predict(qda_model, newdata = test_data)
qda_class <- qda_pred$class

# Confusion matrix
confusion_matrix <- table(Predicted = qda_class, Actual = test_data$mpg01)
print(confusion_matrix)
```

```
##          Actual
## Predicted  0  1
##         0 49  4
##         1  9 54
```

```
# Calculate test error
qda_error <- mean(qda_class != test_data$mpg01)
cat("QDA Test Error:", round(qda_error, 4), "\n")
```

```
## QDA Test Error: 0.1121
```

(f) Perform logistic regression on the training data to predict mpg01 using the predictors that seemed most associated with mpg01 in (b). What is the test error rate of the model obtained?

```
# Fit logistic regression model
logistic_model <- glm(mpg01 ~ horsepower + weight + displacement + cylinders + year + origin_factor,
                      data = train_data, family = binomial)

# Make predictions
logistic_probs <- predict(logistic_model, newdata = test_data, type = "response")
logistic_class <- ifelse(logistic_probs > 0.5, 1, 0)

# Confusion matrix
confusion_matrix <- table(Predicted = logistic_class, Actual = test_data$mpg01)
print(confusion_matrix)
```

```
##          Actual
## Predicted  0  1
##         0 49  3
##         1  9 55
```

```
# Calculate test error
logistic_error <- mean(logistic_class != test_data$mpg01)
cat("Logistic Regression Test Error:", round(logistic_error, 4), "\n")
```

```
## Logistic Regression Test Error: 0.1034
```

(g) Perform Naive Bayes on the training data to predict mpg01 using the predictors that seemed most associated with mpg01 in (b). What is the test error rate of the model obtained?

```
# Fit Naive Bayes model
naive_bayes_model <- naiveBayes(mpg01 ~ horsepower + weight + displacement + cylinders + year + origin_

# Make predictions
naive_bayes_class <- predict(naive_bayes_model, newdata = test_data, type = "class")

# Confusion matrix
confusion_matrix <- table(Predicted = naive_bayes_class, Actual = test_data$mpg01)
print(confusion_matrix)
```

```
##          Actual
## Predicted  0  1
##         0 49  3
##         1  9 55
```

```
# Calculate test error
naive_bayes_error <- mean(naive_bayes_class != test_data$mpg01)
```

```r
cat("Naive Bayes Test Error:", round(naive_bayes_error, 4), "\n")
```

```
## Naive Bayes Test Error: 0.1034
```

(h) Perform KNN on the training data, with several values of K, to predict mpg01 using only the predictors that seemed most associated with mpg01 in (b). What test errors do you get? Which value of K seems to perform best on this data set?

```r
# Define the selected predictors (excluding origin_factor for simplicity with KNN)
selected_predictors <- c("horsepower", "weight", "displacement", "cylinders", "year")

# Extract predictor matrices
train_predictors <- train_data[, selected_predictors]
test_predictors <- test_data[, selected_predictors]

# Scale the predictors
train_scaled <- scale(train_predictors)
test_scaled <- scale(test_predictors,
                     center = attr(train_scaled, "scaled:center"),
                     scale = attr(train_scaled, "scaled:scale"))

# Test different values of K
k_values <- c(1, 3, 5, 10, 15, 20, 25)
knn_errors <- numeric(length(k_values))

for (i in seq_along(k_values)) {
  k <- k_values[i]
  knn_pred <- knn(train = train_scaled,
                  test = test_scaled,
                  cl = train_data$mpg01,
                  k = k)
  knn_errors[i] <- mean(knn_pred != test_data$mpg01)
}

# Display results
results_df <- data.frame(K = k_values, Test_Error = round(knn_errors, 4))
print(results_df)
```

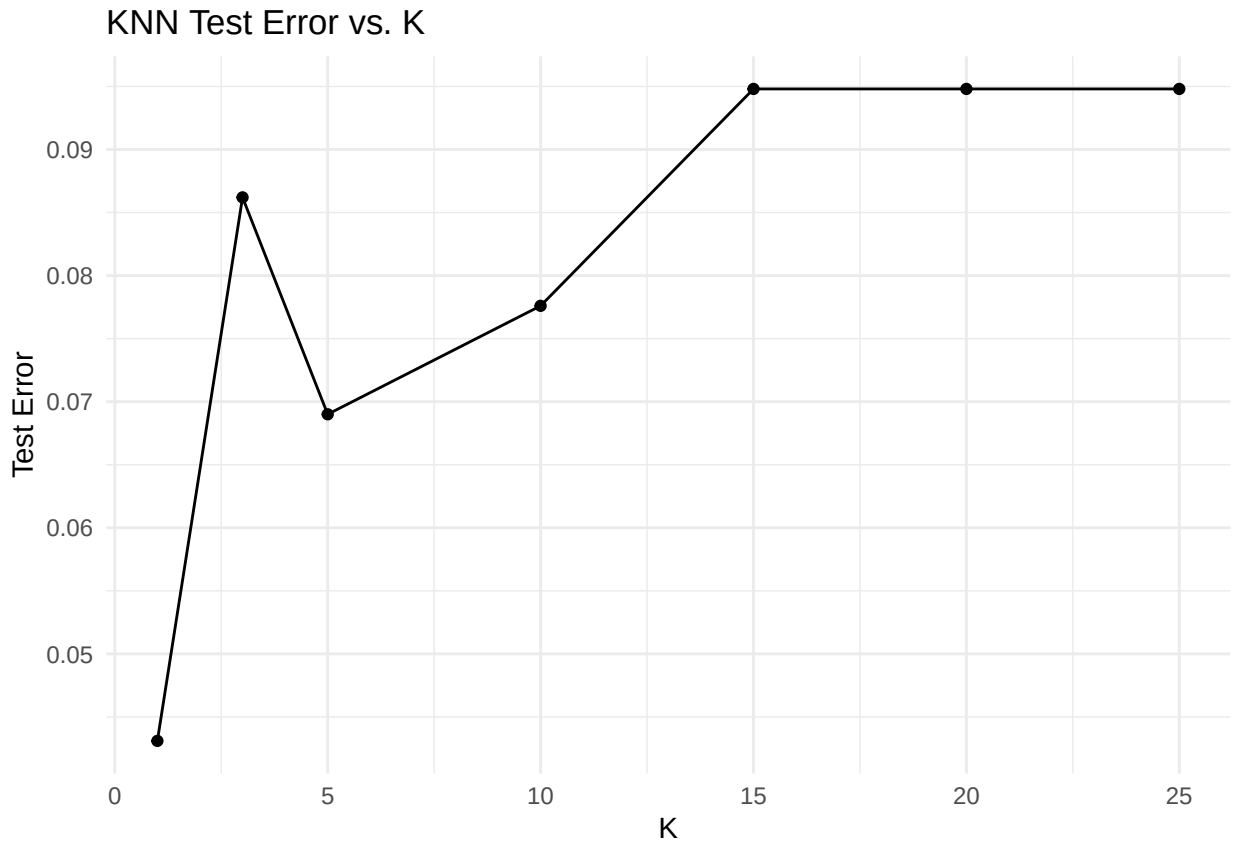
```
##    K Test_Error
## 1  1     0.0431
## 2  3     0.0862
## 3  5     0.0690
## 4 10     0.0776
## 5 15     0.0948
## 6 20     0.0948
## 7 25     0.0948
```

```r
# Find best K
best_k <- k_values[which.min(knn_errors)]
cat("Best K:", best_k, "with test error:", round(min(knn_errors), 4), "\n")
```

```
## Best K: 1 with test error: 0.0431
```

```r
# Plot the results
ggplot(results_df, aes(x = K, y = Test_Error)) +
  geom_line() +
```

```
  geom_point() +
  labs(title = "KNN Test Error vs. K", x = "K", y = "Test Error") +
  theme_minimal()
```



Summary of Results

```
# Create summary table of all model performances
model_results <- data.frame(
  Model = c("LDA", "QDA", "Logistic Regression", "Naive Bayes", paste("KNN (K=", best_k, ")", sep="")),
  Test_Error = c(lda_error, qda_error, logistic_error, naive_bayes_error, min(knn_errors))
)

model_results$Test_Error <- round(model_results$Test_Error, 4)
print(model_results)
```

```
##                 Model Test_Error
## 1                 LDA     0.0948
## 2                 QDA     0.1121
## 3 Logistic Regression     0.1034
## 4         Naive Bayes     0.1034
## 5           KNN (K=1)     0.0431
```

```
# Find best performing model
best_model <- model_results[which.min(model_results$Test_Error), ]
cat("\nBest performing model:", best_model$Model, "with test error:", best_model$Test_Error, "\n")
```

```
##
## Best performing model: KNN (K=1) with test error: 0.0431
```

The analysis shows the performance of different classification methods on the Auto dataset for predicting high vs. low gas mileage. The results demonstrate the trade-offs between different approaches and how model complexity affects performance on this particular dataset.