

Question 1:

Code :

```
1 sales_prices <- c(5, 10, 11, 13, 15, 35, 50, 55, 72, 92, 204, 215)
2
3 equal_freq <- function(data, num_bins) {
4   n <- length(data)
5   bin_size <- n / num_bins
6   cut(data, breaks = unique(quantile(data, probs = seq(0, 1, 1/num_bins))),
7     include.lowest = TRUE, labels = paste("Bin", 1:num_bins))
8 }
9
10 equal_width <- function(data, num_bins) {
11   cut(data, breaks = num_bins, labels = paste("Bin", 1:num_bins))
12 }
13
14 clustering <- function(data, num_clusters) {
15   kmeans_result <- kmeans(data, centers = num_clusters)
16   factor(kmeans_result$cluster, labels = paste("Cluster", 1:num_clusters))
17 }
18
19 eq_freq_result <- equal_freq(sales_prices, 3)
20 eq_width_result <- equal_width(sales_prices, 3)
21 cluster_result <- clustering(sales_prices, 3)
22
23 cat("(a) Equal-frequency partitioning:\n")
24 print(data.frame(Price = sales_prices, Bin = eq_freq_result))
25
26 cat("\n(b) Equal-width partitioning:\n")
27 print(data.frame(Price = sales_prices, Bin = eq_width_result))
28
29 cat("\n(c) Clustering:\n")
30 print(data.frame(Price = sales_prices, Cluster = cluster_result))
```

Output:

```
   Price  Bin
1      5 Bin 1
2     10 Bin 1
3     11 Bin 1
4     13 Bin 1
5     15 Bin 2
6     35 Bin 2
7     50 Bin 2
8     55 Bin 2
9     72 Bin 3
10    92 Bin 3
11   204 Bin 3
12   215 Bin 3
>
> cat("\n(b) Equal-width partitioning:\n")
(b) Equal-width partitioning:
> print(data.frame(Price = sales_prices, Bin = eq_width_result))
   Price  Bin
1      5 Bin 1
2     10 Bin 1
3     11 Bin 1
4     13 Bin 1
5     15 Bin 1
6     35 Bin 1
7     50 Bin 1
8     55 Bin 1
9     72 Bin 1
10    92 Bin 2
11   204 Bin 3
12   215 Bin 3
>
> cat("\n(c) Clustering:\n")
(c) Clustering:
> print(data.frame(Price = sales_prices, Cluster = cluster_result))
   Price Cluster
1      5 Cluster 2
2     10 Cluster 2
3     11 Cluster 2
```

Question 3:

Output:

The screenshot shows the Orange Data Mining software interface. The 'Associate' tab is selected. The 'Choose' dropdown is set to 'Apriori'. The command line shows: 'Apriori -N 10 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.25 -S -1.0 -c -1'. The 'Start' button is highlighted. The 'Result list' on the left shows two entries: '08:56:30 - Apriori' and '08:58:46 - Apriori', with the latter selected. The 'Associator output' pane displays the following text:

```
==== Associator model (full training set) ====

Apriori
=====

Minimum support: 0.4 (3 instances)
Minimum metric <confidence>: 0.9
Number of cycles performed: 12

Generated sets of large itemsets:

Size of set of large itemsets L(1): 7

Size of set of large itemsets L(2): 15

Size of set of large itemsets L(3): 10

Size of set of large itemsets L(4): 1

Best rules found:

1. Bread=t 4 ==> Jam=t 4    <conf: (1)> lift: (1.6) lev: (0.19) [1] conv: (1.5)
2. Peanuts=t 4 ==> Fruit=t 4  <conf: (1)> lift: (1.33) lev: (0.13) [1] conv: (1)
3. Chips=t 4 ==> Soda=t 4    <conf: (1)> lift: (1.33) lev: (0.13) [1] conv: (1)
4. Fruit=t Soda=t 4 ==> Milk=t 4  <conf: (1)> lift: (1.33) lev: (0.13) [1] conv: (1)
5. Bread=t Milk=t 3 ==> Jam=t 3  <conf: (1)> lift: (1.6) lev: (0.14) [1] conv: (1.13)
6. Bread=t Soda=t 3 ==> Jam=t 3  <conf: (1)> lift: (1.6) lev: (0.14) [1] conv: (1.13)
7. Jam=t Chips=t 3 ==> Bread=t 3  <conf: (1)> lift: (2) lev: (0.19) [1] conv: (1.5)
8. Bread=t Chips=t 3 ==> Jam=t 3  <conf: (1)> lift: (1.6) lev: (0.14) [1] conv: (1.13)
9. Bread=t Chips=t 3 ==> Soda=t 3  <conf: (1)> lift: (1.33) lev: (0.09) [0] conv: (0.75)
10. Bread=t Soda=t 3 ==> Chips=t 3  <conf: (1)> lift: (2) lev: (0.19) [1] conv: (1.5)
```

Question 4:

Code :

```
1 data <- c(200, 300, 400, 600, 1000)
2
3 min_max_norm <- function(x) {
4   (x - min(x)) / (max(x) - min(x))
5 }
6
7 z_score_norm <- function(x) {
8   (x - mean(x)) / sd(x)
9 }
10
11 mad_norm <- function(x) {
12   mad <- mean(abs(x - mean(x)))
13   (x - mean(x)) / mad
14 }
15
16 decimal_scaling <- function(x) {
17   j <- ceiling(log10(max(abs(x))))
18   x / (10^j)
19 }
20
21 min_max_result <- min_max_norm(data)
22 z_score_result <- z_score_norm(data)
23 mad_result <- mad_norm(data)
24 decimal_result <- decimal_scaling(data)
25
26 cat("Original data:", data, "\n\n")
27 cat("(a) Min-Max Normalization:\n")
28 print(min_max_result)
29 cat("\n(b) Z-score Normalization:\n")
30 print(z_score_result)
31 cat("\n(c) Z-score Normalization using MAD:\n")
32 print(mad_result)
33 cat("\n(d) Normalization by Decimal Scaling:\n")
34 print(decimal_result)
35
36 results <- data.frame(
37   Original = data,
38   MinMax = min_max_result,
39   ZScore = z_score_result,
40   MAD = mad_result,
```

Output :

Comparison of all normalization methods:

```
> print(results)
```

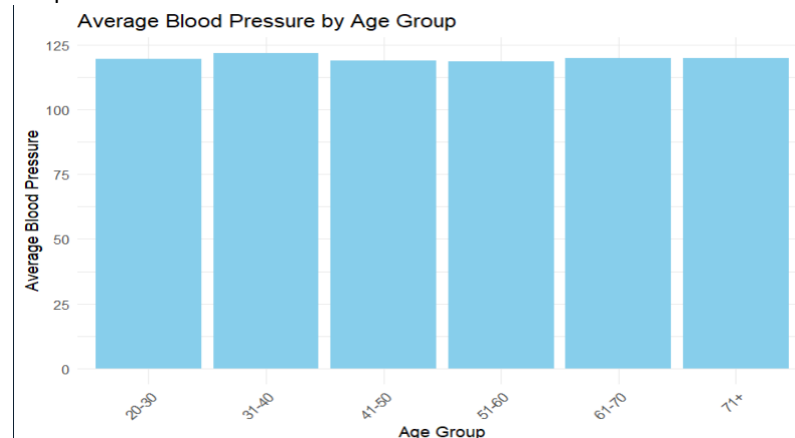
	Original	MinMax	ZScore	MAD	DecimalScaling
1	200	0.000	-0.9486833	-1.2500000	0.2
2	300	0.125	-0.6324555	-0.8333333	0.3
3	400	0.250	-0.3162278	-0.4166667	0.4
4	600	0.500	0.3162278	0.4166667	0.6
5	1000	1.000	1.5811388	2.0833333	1.0

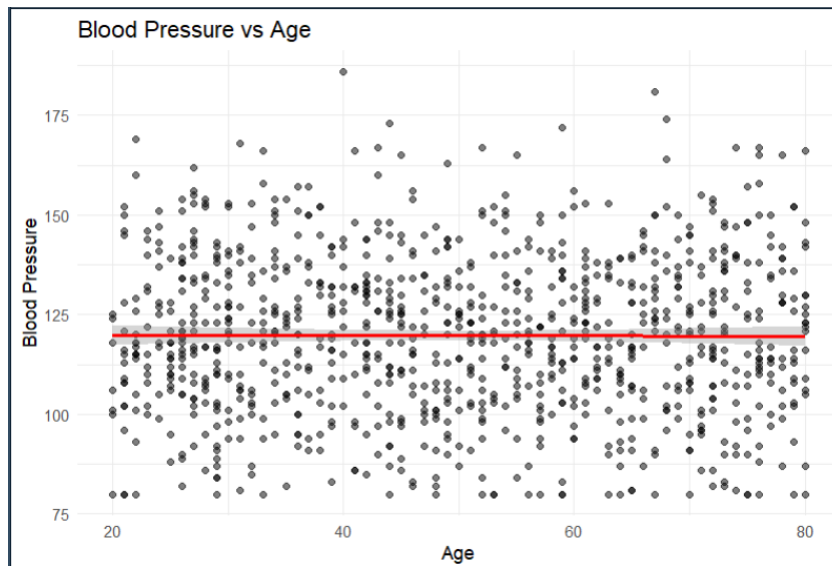
Question 5:

Code :

```
1 library(ggplot2)
2
3 set.seed(123)
4 n <- 1000
5
6 diabetes <- data.frame(
7   Age = sample(20:80, n, replace = TRUE),
8   BloodPressure = round(rnorm(n, mean = 120, sd = 20)),
9   Glucose = round(rnorm(n, mean = 100, sd = 25)),
10  BMI = round(rnorm(n, mean = 25, sd = 5), 1),
11  Diabetes = sample(c(0, 1), n, replace = TRUE, prob = c(0.7, 0.3))
12 )
13
14 diabetes$BloodPressure <- pmax(pmin(diabetes$BloodPressure, 200), 80)
15
16 diabetes$AgeGroup <- cut(diabetes$Age,
17   breaks = c(0, 30, 40, 50, 60, 70, Inf),
18   labels = c("20-30", "31-40", "41-50", "51-60", "61-70", "71+"))
19
20 avg_bp <- aggregate(BloodPressure ~ AgeGroup, data = diabetes, FUN = mean)
21
22 scatter_plot <- ggplot(diabetes, aes(x = Age, y = BloodPressure)) +
23   geom_point(alpha = 0.5) +
24   geom_smooth(method = "lm", color = "red") +
25   labs(title = "Blood Pressure vs Age",
26    x = "Age",
27    y = "Blood Pressure") +
28   theme_minimal()
29
30 bar_chart <- ggplot(avg_bp, aes(x = AgeGroup, y = BloodPressure)) +
31   geom_bar(stat = "identity", fill = "skyblue") +
32   labs(title = "Average Blood Pressure by Age Group",
33    x = "Age Group",
34    y = "Average Blood Pressure") +
35   theme_minimal() +
36   theme(axis.text.x = element_text(angle = 45, hjust = 1))
37
38 print(scatter_plot)
39 print(bar_chart)
```

Output:





Question 6:

Code :

```
1 library(ggplot2)
2
3 set.seed(42)
4
5 n <- 100
6 age <- sample(20:70, n, replace = TRUE)
7 blood_sugar <- rnorm(n, mean = 100, sd = 15) + 0.2 * age
8 diabetes_status <- ifelse(blood_sugar > 140, 1, 0)
9 diabetes_data <- data.frame(age = age, blood_sugar = blood_sugar, diabetes_status = diabetes_status)
10
11 ggplot(diabetes_data, aes(x = age, y = blood_sugar)) +
12   geom_point() +
13   geom_smooth(method = "lm", col = "blue") +
14   labs(title = "Blood Sugar Level vs Age", x = "Age", y = "Blood Sugar Level")
15
16 linear_model <- lm(blood_sugar ~ age, data = diabetes_data)
17 print(summary(linear_model))
18
19 ggplot(diabetes_data, aes(x = age, y = blood_sugar)) +
20   geom_point() +
21   geom_smooth(method = "lm", col = "red") +
22   labs(title = "Simple Linear Regression: Blood Sugar vs Age", x = "Age", y = "Blood Sugar Level")
23
24 multiple_model <- lm(diabetes_status ~ age + blood_sugar, data = diabetes_data)
25 print(summary(multiple_model))
26
27 par(mfrow = c(2, 2))
28 plot(multiple_model)
29
30 new_data <- data.frame(age = c(30, 50, 65), blood_sugar = c(130, 160, 145))
31 predictions <- predict(multiple_model, new_data, type = "response")
32 print(data.frame(new_data, predicted_diabetes_status = predictions))
```

Output:

	age	blood_sugar	predicted_diabetes_status
1	30	130	0.07526419
2	50	160	0.11063246
3	65	145	0.06105091

Question 7:

Output:

Associator

Choose **Apriori** -N 10 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.1 -S -1.0 -c -1

Start

Stop

Result list (right-click for ...)

09:50:55 - Apriori

Associator output

C
I

=== Associator model (full training set) ===

Apriori

=====

Minimum support: 0.85 (4 instances)

Minimum metric <confidence>: 0.9

Number of cycles performed: 3

Generated sets of large itemsets:

Size of set of large itemsets I(1): 6

Size of set of large itemsets I(2): 6

Size of set of large itemsets I(3): 1

Best rules found:

1. E=T 4 ==> K=T 4 <conf:(1)> lift:(1) lev:(0) [0] conv:(0)

2. D=F 4 ==> K=T 4 <conf:(1)> lift:(1) lev:(0) [0] conv:(0)

3. A=F 4 ==> K=T 4 <conf:(1)> lift:(1) lev:(0) [0] conv:(0)

4. U=F 4 ==> K=T 4 <conf:(1)> lift:(1) lev:(0) [0] conv:(0)

5. I=F 4 ==> K=T 4 <conf:(1)> lift:(1) lev:(0) [0] conv:(0)

6. U=F 4 ==> E=T 4 <conf:(1)> lift:(1.25) lev:(0.16) [0] conv:(0.8)

7. E=T 4 ==> U=F 4 <conf:(1)> lift:(1.25) lev:(0.16) [0] conv:(0.8)

8. E=T U=F 4 ==> K=T 4 <conf:(1)> lift:(1) lev:(0) [0] conv:(0)

9. K=T U=F 4 ==> E=T 4 <conf:(1)> lift:(1.25) lev:(0.16) [0] conv:(0.8)

10. K=T E=T 4 ==> U=F 4 <conf:(1)> lift:(1.25) lev:(0.16) [0] conv:(0.8)

Associator

Choose **FPGrowth** -P 2 -I -1 -N 10 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.1

Start

Stop

Result list (right-click for ...)

09:50:55 - Apriori

09:51:43 - FPGrowth

Associator output

=== Run information ===

Scheme: weka.associations.FPGrowth -P 2 -I -1 -N 10 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.1

Relation: transactions

Instances: 5

Attributes: 11

M

O

N

K

E

Y

D

A

U

C

I

=== Associator model (full training set) ===

FPGrowth found 71 rules (displaying top 10)

1. [C=F]: 3 ==> [U=F]: 3 <conf:(1)> lift:(1.25) lev:(0.12) conv:(0.6)

2. [Y=F]: 2 ==> [U=F]: 2 <conf:(1)> lift:(1.25) lev:(0.08) conv:(0.4)

3. [M=F]: 2 ==> [U=F]: 2 <conf:(1)> lift:(1.25) lev:(0.08) conv:(0.4)

4. [C=F]: 3 ==> [I=F]: 3 <conf:(1)> lift:(1.25) lev:(0.12) conv:(0.6)

5. [O=F]: 2 ==> [I=F]: 2 <conf:(1)> lift:(1.25) lev:(0.08) conv:(0.4)

6. [N=F]: 3 ==> [D=F]: 3 <conf:(1)> lift:(1.25) lev:(0.12) conv:(0.6)

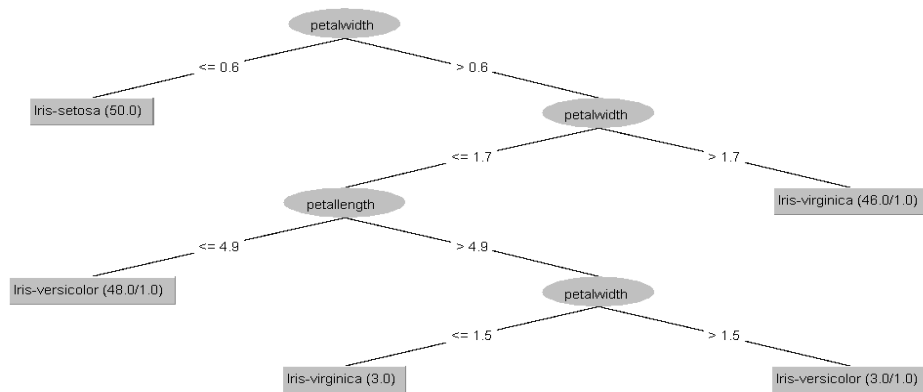
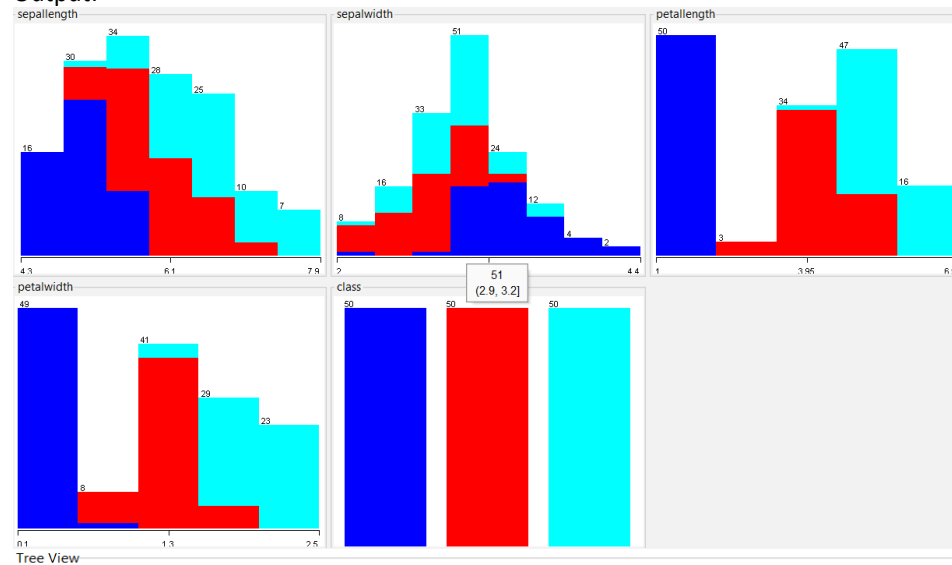
7. [Y=F]: 2 ==> [D=F]: 2 <conf:(1)> lift:(1.25) lev:(0.08) conv:(0.4)

8. [O=F]: 2 ==> [D=F]: 2 <conf:(1)> lift:(1.25) lev:(0.08) conv:(0.4)

9. [M=F]: 2 ==> [A=F]: 2 <conf:(1)> lift:(1.25) lev:(0.08) conv:(0.4)

10. [Y=F]: 2 ==> [N=F]: 2 <conf:(1)> lift:(1.67) lev:(0.16) conv:(0.8)

Output:



```

Classifier output
petalength          0          0.0001
petalwidth          0          0

Time taken to build model: 0.07 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances      144          96      %
Incorrectly Classified Instances    6          4      %
Kappa statistic                    0.94
Mean absolute error                 0.0287
Root mean squared error             0.1424
Relative absolute error              6.456 %
Root relative squared error          30.2139 %
Total Number of Instances          150

=== Detailed Accuracy By Class ===

      TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
1.000  0.000  1.000  1.000  1.000  1.000  1.000  1.000  Iris-setosa
0.920  0.020  0.958  0.920  0.939  0.910  0.972  0.934  Iris-versicolor
0.960  0.040  0.923  0.960  0.941  0.911  0.972  0.934  Iris-virginica
Weighted Avg.  0.960  0.020  0.960  0.960  0.960  0.940  0.981  0.956

=== Confusion Matrix ===

  a  b  c  <-- classified as
50  0  0 | a = Iris-setosa
  46  4 | b = Iris-versicolor
  0  2 48 | c = Iris-virginica

```

Question 9:

Code :

```
1 ages <- c(13, 15, 16, 16, 19, 20, 20, 21, 22, 22, 25, 25, 25, 25, 30, 33,
2           33, 35, 35, 35, 35, 36, 40, 45, 46, 52, 70)
3
4 quartiles <- quantile(ages, probs = c(0.25, 0.75))
5
6 print(quartiles)|
```

Output :

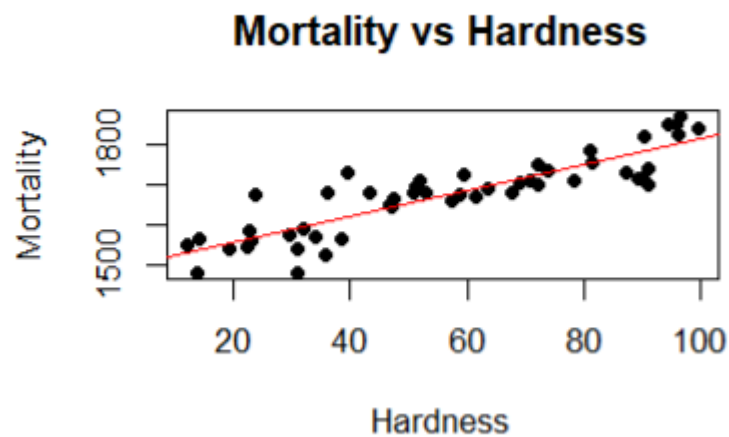
```
> ages <- c(13, 15, 16, 16, 19, 20, 20, 21, 22, 22, 25, 25, 25, 25, 30, 33,
+           33, 35, 35, 35, 35, 36, 40, 45, 46, 52, 70)
>
> quartiles <- quantile(ages, probs = c(0.25, 0.75))
>
> print(quartiles)
 25%  75%
20.5 35.0
```

Question 10:

Code :

```
1 data("water")
2
3 head(water)
4
5 plot(water$hardness, water$mortality,
6       main="Mortality vs Hardness",
7       xlab="Hardness",
8       ylab="Mortality",
9       pch=19)
10
11 abline(lm(mortality ~ hardness, data=water), col="red")
12
13 model <- lm(mortality ~ hardness, data=water)
14
15 summary(model)
16
17 new_data <- data.frame(hardness = 88)
18 prediction <- predict(model, new_data)
19 print(paste("Predicted mortality for hardness 88:", round(prediction, 2)))|
```

Output :



Question 11:

Output:

```
=== Associator model (full training set) ===

Apriori
=====

Minimum support: 0.55 (3 instances)
Minimum metric <confidence>: 0.6
Number of cycles performed: 9

Generated sets of large itemsets:

Size of set of large itemsets L(1): 6

Size of set of large itemsets L(2): 7

Size of set of large itemsets L(3): 4

Size of set of large itemsets L(4): 1

Best rules found:

1. Chips=t 4 ==> Buns=f 4      <conf:(1)> lift:(1.5) lev:(0.22) [1] conv:(1.33)
2. Buns=f 4 ==> Chips=t 4      <conf:(1)> lift:(1.5) lev:(0.22) [1] conv:(1.33)
3. Coke=t 3 ==> Buns=f 3      <conf:(1)> lift:(1.5) lev:(0.17) [1] conv:(1)
4. Coke=t 3 ==> Ketchup=f 3    <conf:(1)> lift:(1.5) lev:(0.17) [1] conv:(1)
5. Coke=t 3 ==> Chips=t 3      <conf:(1)> lift:(1.5) lev:(0.17) [1] conv:(1)
6. Ketchup=f Coke=t 3 ==> Buns=f 3 <conf:(1)> lift:(1.5) lev:(0.17) [1] conv:(1)
7. Buns=f Coke=t 3 ==> Ketchup=f 3 <conf:(1)> lift:(1.5) lev:(0.17) [1] conv:(1)
8. Buns=f Ketchup=f 3 ==> Coke=t 3 <conf:(1)> lift:(2) lev:(0.25) [1] conv:(1.5)
9. Coke=t 3 ==> Buns=f Ketchup=f 3 <conf:(1)> lift:(2) lev:(0.25) [1] conv:(1.5)
10. Ketchup=f Chips=t 3 ==> Buns=f 3 <conf:(1)> lift:(1.5) lev:(0.17) [1] conv:(1)
```

Question 12:

Output :

```
Merit of best subset found: 96
Evaluation (for feature selection): CV (leave one out)
Feature set: 4,5

Time taken to build model: 0.01 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances      139           92.6667 %
Incorrectly Classified Instances    11           7.3333 %
Kappa statistic                     0.89
Mean absolute error                 0.092
Root mean squared error             0.2087
Relative absolute error             20.6978 %
Root relative squared error         44.2707 %
Total Number of Instances          150

=== Detailed Accuracy By Class ===

                TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                1.000    0.000    1.000     1.000    1.000     1.000    1.000    1.000    Iris-setosa
                0.880    0.050    0.898     0.880    0.889     0.834    0.946    0.861    Iris-versicolor
                0.900    0.060    0.882     0.900    0.891     0.836    0.947    0.869    Iris-virginica
Weighted Avg.   0.927    0.037    0.927     0.927    0.927     0.890    0.964    0.910

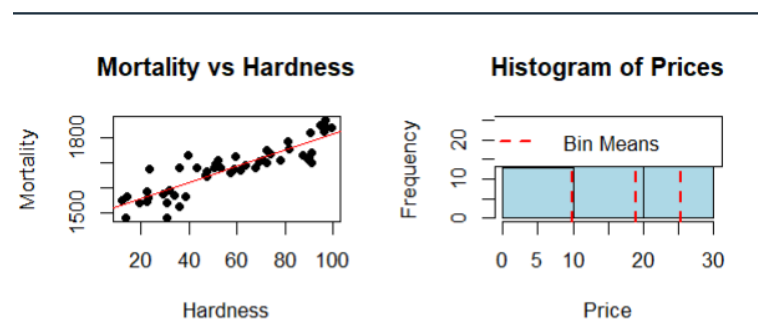
=== Confusion Matrix ===

  a  b  c  <-- classified as
50  0  0 | a = Iris-setosa
 0 44  6 | b = Iris-versicolor
 0  5 45 | c = Iris-virginica
```

Accuracy : 92.6%


```
a b c <-- classified as
49 1 0 | a = Iris-setosa
0 47 3 | b = Iris-versicolor
0 2 48 | c = Iris-virginica
```

```
1 prices <- c(1, 1, 5, 5, 5, 5, 5, 8, 8, 10, 10, 10, 10, 12, 14, 14, 14, 15, 15, 15, 15, 15, 15, 18, 18, 18, 18, 18, 18, 18, 20, 20)
2
3 n_bins <- 3
4 bin_size <- length(prices) %% n_bins
5 bins <- cut(prices, breaks = quantile(prices, probs = seq(0, 1, 1/n_bins)), include.lowest = TRUE)
6
7 bin_means <- tapply(prices, bins, mean)
8
9 bin_boundaries <- levels(bins)
10
11 cat("Bin means:\n")
12 print(bin_means)
13 cat("\nBin boundaries:\n")
14 print(bin_boundaries)
15
16 hist(prices, breaks = n_bins, main = "Histogram of Prices", xlab = "Price", ylab = "Frequency", col = "lightblue", border = "black")
17
18 abline(v = bin_means, col = "red", lwd = 2, lty = 2)
19
20 legend("topright", legend = c("Bin Means"), col = "red", lty = 2, lwd = 2)
```

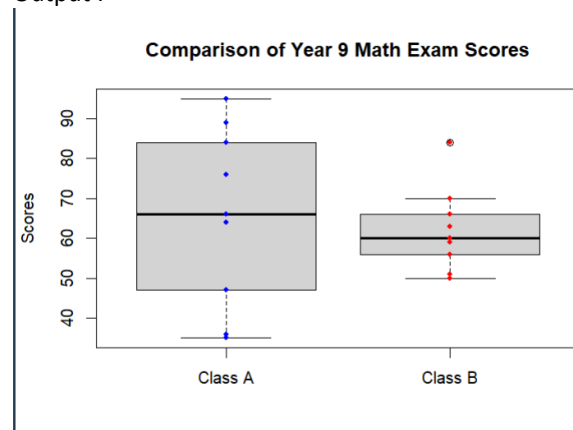


Question 14:

Code :

```
1 class_A <- c(76, 35, 47, 64, 95, 66, 89, 36, 84)
2 class_B <- c(51, 56, 84, 60, 59, 70, 63, 66, 50)
3
4 mean_A <- mean(class_A)
5 mean_B <- mean(class_B)
6 median_A <- median(class_A)
7 median_B <- median(class_B)
8 range_A <- max(class_A) - min(class_A)
9 range_B <- max(class_B) - min(class_B)
10
11 cat("Class A:\n")
12 cat("Mean:", mean_A, "\n")
13 cat("Median:", median_A, "\n")
14 cat("Range:", range_A, "\n\n")
15
16 cat("Class B:\n")
17 cat("Mean:", mean_B, "\n")
18 cat("Median:", median_B, "\n")
19 cat("Range:", range_B, "\n\n")
20
21 boxplot(class_A, class_B, names = c("Class A", "Class B"),
22         main = "Comparison of Year 9 Math Exam Scores",
23         ylab = "Scores")
24
25 points(rep(1, length(class_A)), class_A, col = "blue", pch = 20)
26 points(rep(2, length(class_B)), class_B, col = "red", pch = 20)
```

Output :



Question 15: