Question 1:

Code :

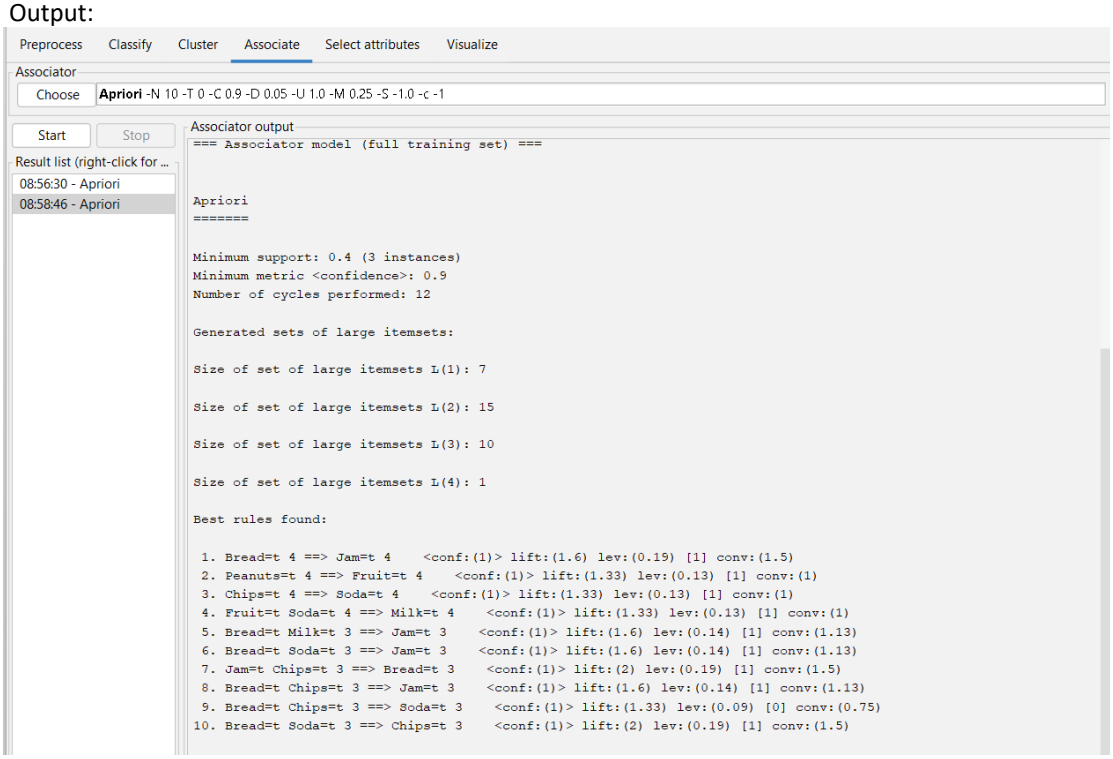
```r
sales_prices <- c(5, 10, 11, 13, 15, 35, 50, 55, 72, 92, 204, 215)

equal_freq <- function(data, num_bins) {
  n <- length(data)
  bin_size <- n / num_bins
  cut(data, breaks = unique(quantile(data, probs = seq(0, 1, 1/num_bins))),
      include.lowest = TRUE, labels = paste("Bin", 1:num_bins))
}

equal_width <- function(data, num_bins) {
  cut(data, breaks = num_bins, labels = paste("Bin", 1:num_bins))
}

clustering <- function(data, num_clusters) {
  kmeans_result <- kmeans(data, centers = num_clusters)
  factor(kmeans_result$cluster, labels = paste("Cluster", 1:num_clusters))
}

eq_freq_result <- equal_freq(sales_prices, 3)
eq_width_result <- equal_width(sales_prices, 3)
cluster_result <- clustering(sales_prices, 3)

cat("(a) Equal-frequency partitioning:\n")
print(data.frame(Price = sales_prices, Bin = eq_freq_result))

cat("\n(b) Equal-width partitioning:\n")
print(data.frame(Price = sales_prices, Bin = eq_width_result))

cat("\n(c) Clustering:\n")
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:



Question 4:

Code :

```r
data <- c(200, 300, 400, 600, 1000)

min_max_norm <- function(x) {
  (x - min(x)) / (max(x) - min(x))
}

z_score_norm <- function(x) {
  (x - mean(x)) / sd(x)
}

mad_norm <- function(x) {
  mad <- mean(abs(x - mean(x)))
  (x - mean(x)) / mad
}

decimal_scaling <- function(x) {
  j <- ceiling(log10(max(abs(x))))
  x / (10^j)
}

min_max_result <- min_max_norm(data)
z_score_result <- z_score_norm(data)
mad_result <- mad_norm(data)
decimal_result <- decimal_scaling(data)

cat("Original data:", data, "\n\n")
cat("(a) Min-Max Normalization:\n")
print(min_max_result)
cat("\n(b) Z-score Normalization:\n")
print(z_score_result)
cat("\n(c) Z-score Normalization using MAD:\n")
print(mad_result)
cat("\n(d) Normalization by Decimal Scaling:\n")
print(decimal_result)

results <- data.frame(
  Original = data,
  MinMax = min_max_result,
  ZScore = z_score_result,
  MAD = mad_result,
```

Output :

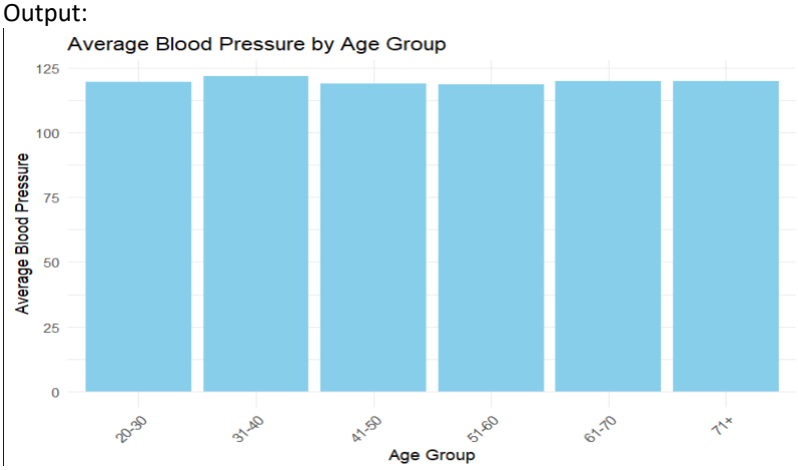
```
Comparison of all normalization methods:
> print(results)
  Original MinMax     ZScore        MAD DecimalScaling
1      200  0.000 -0.9486833 -1.2500000            0.2
2      300  0.125 -0.6324555 -0.8333333            0.3
3      400  0.250 -0.3162278 -0.4166667            0.4
4      600  0.500  0.3162278  0.4166667            0.6
5     1000  1.000  1.5811388  2.0833333            1.0
```

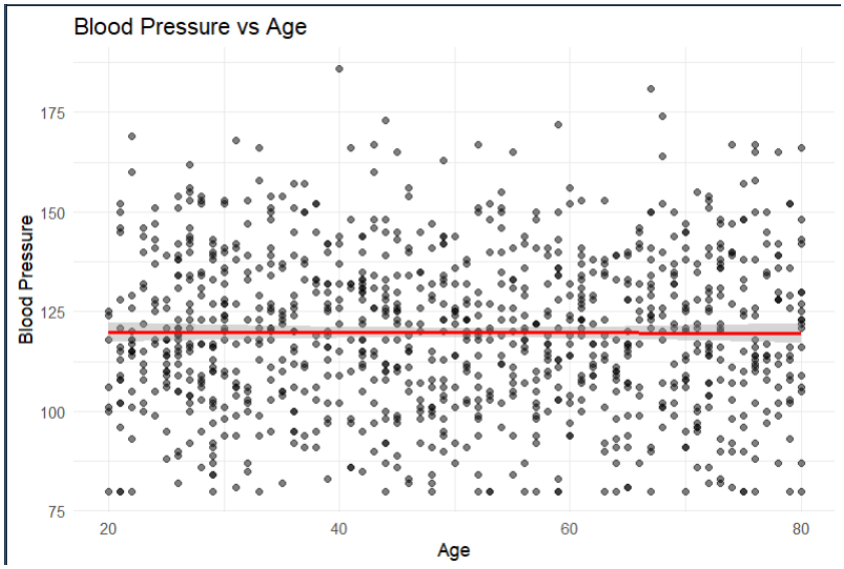
Question 5:

Code :

```r
library(ggplot2)

set.seed(123)
n <- 1000

diabetes <- data.frame(
  Age = sample(20:80, n, replace = TRUE),
  BloodPressure = round(rnorm(n, mean = 120, sd = 20)),
  Glucose = round(rnorm(n, mean = 100, sd = 25)),
  BMI = round(rnorm(n, mean = 25, sd = 5), 1),
  Diabetes = sample(c(0, 1), n, replace = TRUE, prob = c(0.7, 0.3))
)

diabetes$BloodPressure <- pmax(pmin(diabetes$BloodPressure, 200), 80)

diabetes$AgeGroup <- cut(diabetes$Age,
                         breaks = c(0, 30, 40, 50, 60, 70, Inf),
                         labels = c("20-30", "31-40", "41-50", "51-60", "61-70", "71+"))

avg_bp <- aggregate(BloodPressure ~ AgeGroup, data = diabetes, FUN = mean)

scatter_plot <- ggplot(diabetes, aes(x = Age, y = BloodPressure)) +
  geom_point(alpha = 0.5) +
  geom_smooth(method = "lm", color = "red") +
  labs(title = "Blood Pressure vs Age",
       x = "Age",
       y = "Blood Pressure") +
  theme_minimal()

bar_chart <- ggplot(avg_bp, aes(x = AgeGroup, y = BloodPressure)) +
  geom_bar(stat = "identity", fill = "skyblue") +
  labs(title = "Average Blood Pressure by Age Group",
       x = "Age Group",
       y = "Average Blood Pressure") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))

print(scatter_plot)
print(bar_chart)
```

Output:





Question 6:

Code :

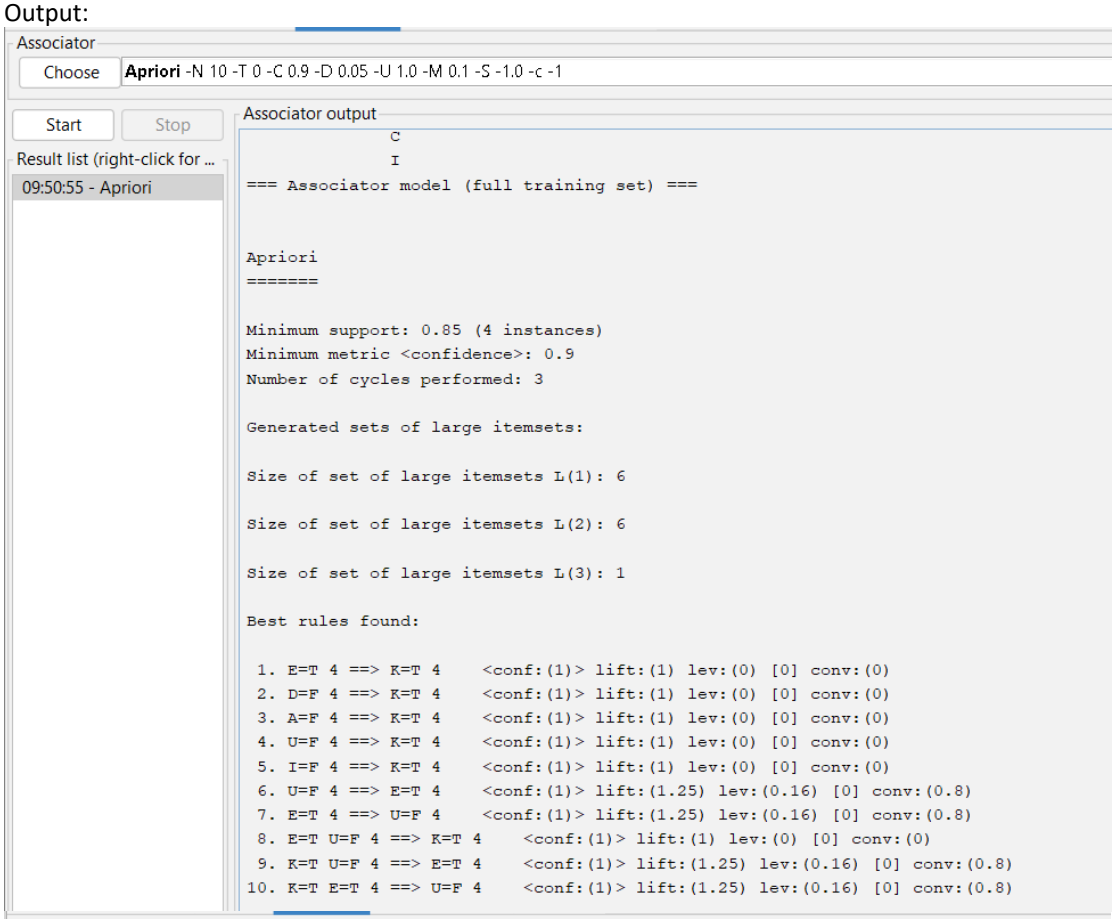
```r
library(ggplot2)

set.seed(42)

n <- 100
age <- sample(20:70, n, replace = TRUE)
blood_sugar <- rnorm(n, mean = 100, sd = 15) + 0.2 * age
diabetes_status <- ifelse(blood_sugar > 140, 1, 0)
diabetes_data <- data.frame(age = age, blood_sugar = blood_sugar, diabetes_status = diabetes_status)

ggplot(diabetes_data, aes(x = age, y = blood_sugar)) +
  geom_point() +
  geom_smooth(method = "lm", col = "blue") +
  labs(title = "Blood Sugar Level vs Age", x = "Age", y = "Blood Sugar Level")

linear_model <- lm(blood_sugar ~ age, data = diabetes_data)
print(summary(linear_model))

ggplot(diabetes_data, aes(x = age, y = blood_sugar)) +
  geom_point() +
  geom_smooth(method = "lm", col = "red") +
  labs(title = "Simple Linear Regression: Blood Sugar vs Age", x = "Age", y = "Blood Sugar Level")

multiple_model <- lm(diabetes_status ~ age + blood_sugar, data = diabetes_data)
print(summary(multiple_model))

par(mfrow = c(2, 2))
plot(multiple_model)

new_data <- data.frame(age = c(30, 50, 65), blood_sugar = c(130, 160, 145))
predictions <- predict(multiple_model, new_data, type = "response")
print(data.frame(new_data, predicted_diabetes_status = predictions))
```

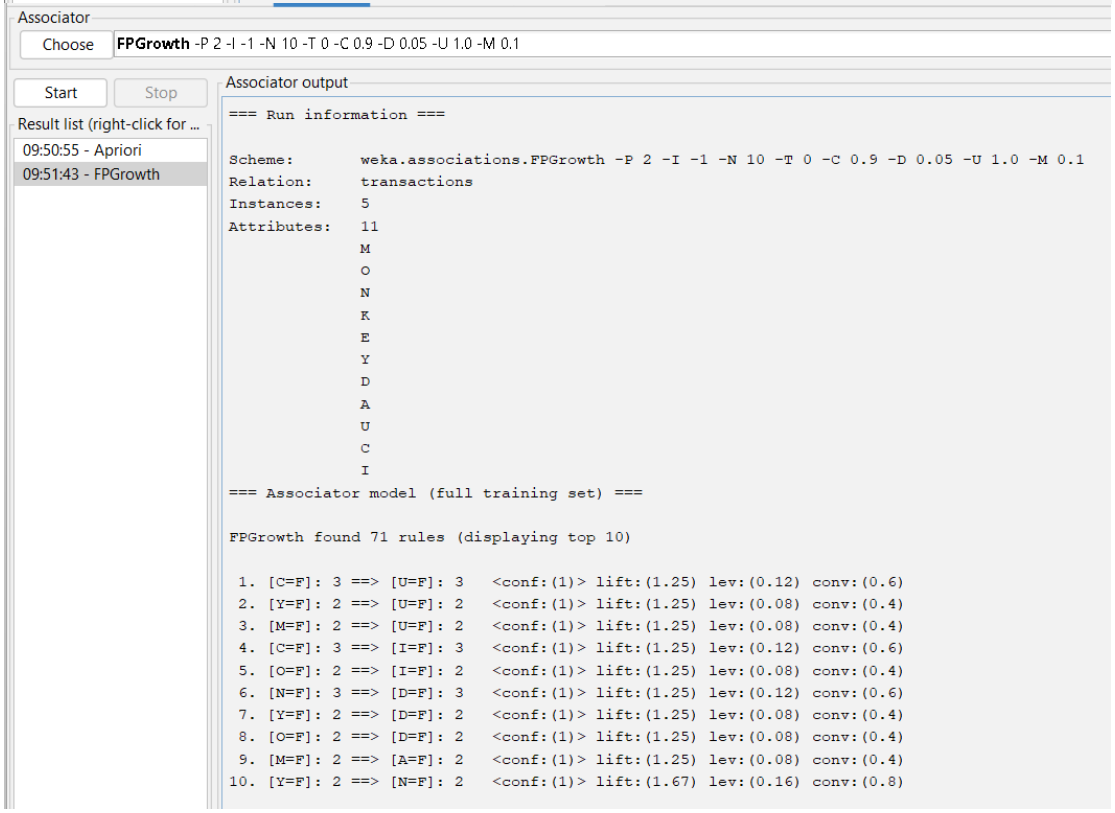
Output:

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

Question 7:

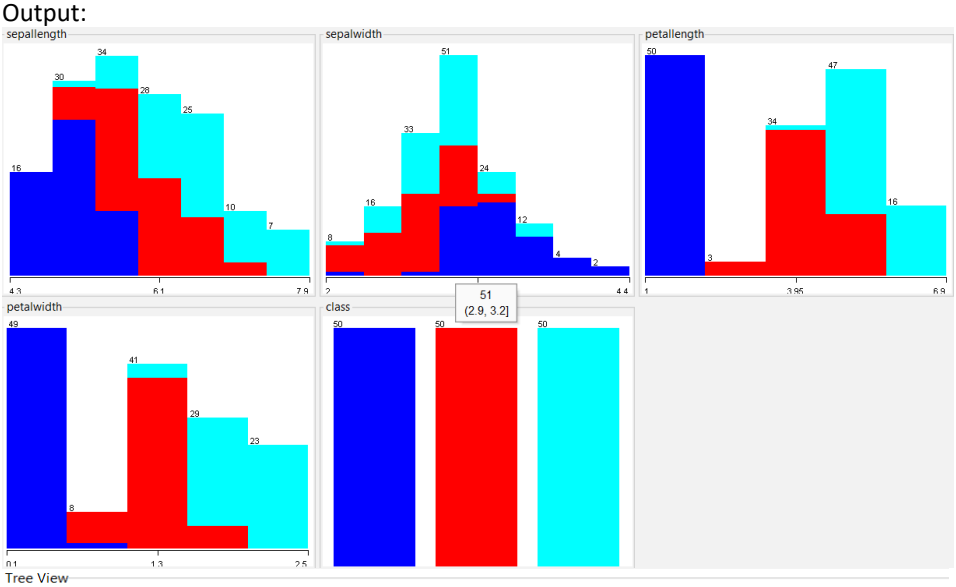
Output:

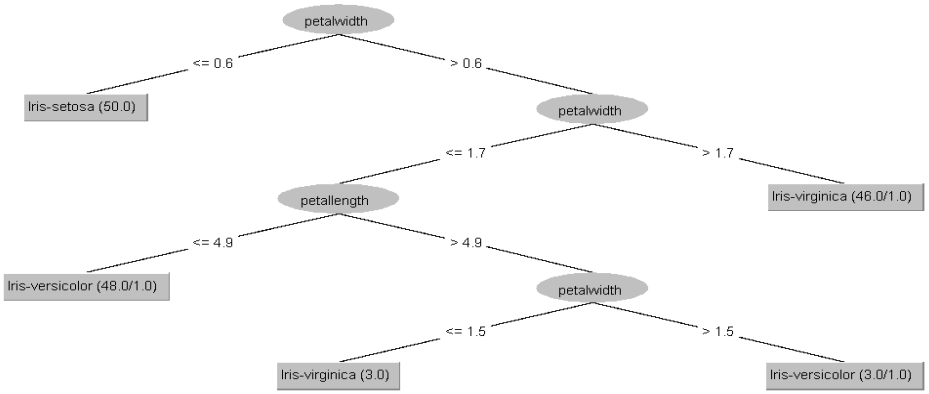




Question 8:

Output:





Classifier output

```
petallength                 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
  0 46  4 |  b = Iris-versicolor
  0  2 48 |  c = Iris-virginica
```

Question 9:

Code :

```r
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)
```

Output :

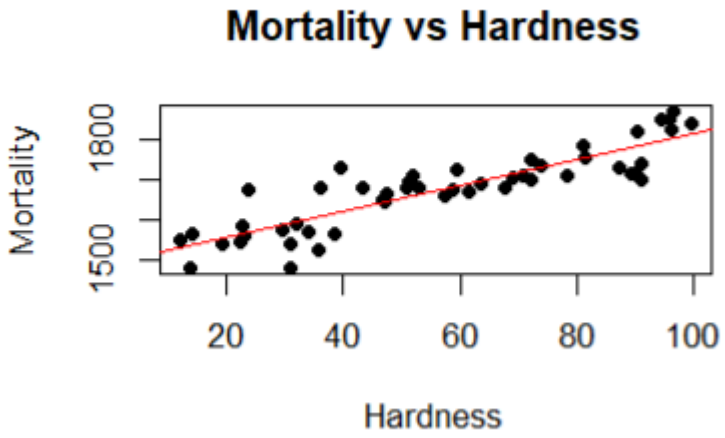
```
> ages <- c(13, 15, 16, 16, 19, 20, 20, 21, 22, 22, 25, 25, 25, 25, 30, 33,
+           33, 35, 35, 35, 35, 36, 40, 45, 46, 52, 70)
>
> quartiles <- quantile(ages, probs = c(0.25, 0.75))
>
> print(quartiles)
 25%  75%
20.5 35.0
```

Question 10:

Code :

```r
data("water")

head(water)

plot(water$hardness, water$mortality,
     main="Mortality vs Hardness",
     xlab="Hardness",
     ylab="Mortality",
     pch=19)

abline(lm(mortality ~ hardness, data=water), col="red")

model <- lm(mortality ~ hardness, data=water)

summary(model)

new_data <- data.frame(hardness = 88)
prediction <- predict(model, new_data)
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%

```
Size of the tree :      9


Time taken to build model: 0 seconds

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

Correctly Classified Instances         144               96      %
Incorrectly Classified Instances         6                4      %
Kappa statistic                          0.94
Mean absolute error                      0.035
Root mean squared error                  0.1586
Relative absolute error                  7.8705 %
Root relative squared error             33.6353 %
Total Number of Instances              150

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.980    0.000    1.000      0.980    0.990      0.985    0.990     0.987     Iris-setosa
                 0.940    0.030    0.940      0.940    0.940      0.910    0.952     0.880     Iris-versicolor
                 0.960    0.030    0.941      0.960    0.950      0.925    0.961     0.905     Iris-virginica
Weighted Avg.    0.960    0.020    0.960      0.960    0.960      0.940    0.968     0.924

=== Confusion Matrix ===

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

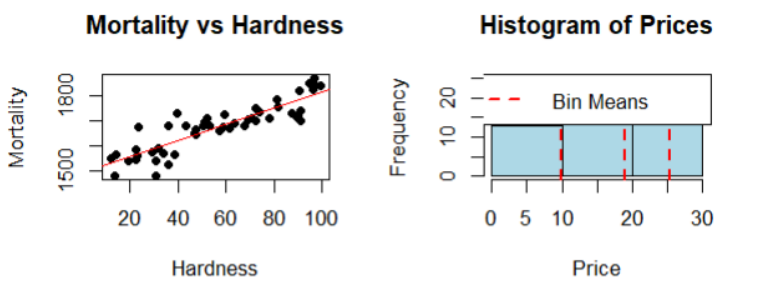
Accuracy : 96%

Question 13:

Code :

```r
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, 18, 20, 2

n_bins <- 3
bin_size <- length(prices) %/% n_bins
bins <- cut(prices, breaks = quantile(prices, probs = seq(0, 1, 1/n_bins)), include.lowest = TRUE)

bin_means <- tapply(prices, bins, mean)

bin_boundaries <- levels(bins)

cat("Bin means:\n")
print(bin_means)
cat("\nBin boundaries:\n")
print(bin_boundaries)

hist(prices, breaks = n_bins, main = "Histogram of Prices", xlab = "Price", ylab = "Frequency", col = "lightblue", border = "black")

abline(v = bin_means, col = "red", lwd = 2, lty = 2)

legend("topright", legend = c("Bin Means"), col = "red", lty = 2, lwd = 2)
```

Output:

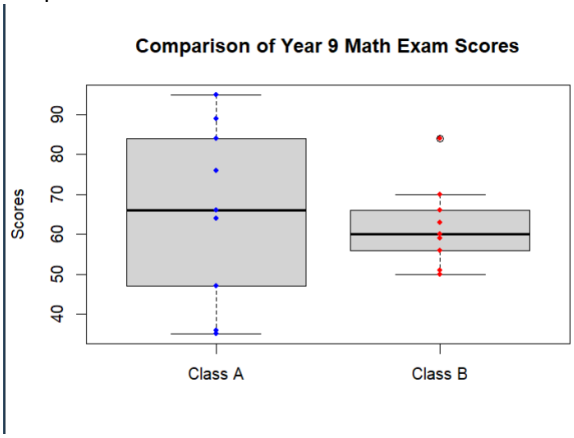


Question 14:

Code :

```r
class_A <- c(76, 35, 47, 64, 95, 66, 89, 36, 84)
class_B <- c(51, 56, 84, 60, 59, 70, 63, 66, 50)

mean_A <- mean(class_A)
mean_B <- mean(class_B)
median_A <- median(class_A)
median_B <- median(class_B)
range_A <- max(class_A) - min(class_A)
range_B <- max(class_B) - min(class_B)

cat("Class A:\n")
cat("Mean:", mean_A, "\n")
cat("Median:", median_A, "\n")
cat("Range:", range_A, "\n\n")

cat("Class B:\n")
cat("Mean:", mean_B, "\n")
cat("Median:", median_B, "\n")
cat("Range:", range_B, "\n\n")

boxplot(class_A, class_B, names = c("Class A", "Class B"),
        main = "Comparison of Year 9 Math Exam Scores",
        ylab = "Scores")

points(rep(1, length(class_A)), class_A, col = "blue", pch = 20)
points(rep(2, length(class_B)), class_B, col = "red", pch = 20)
```

Output :



Question 15: