EXPERIMENT 11

Taniya Ahmed

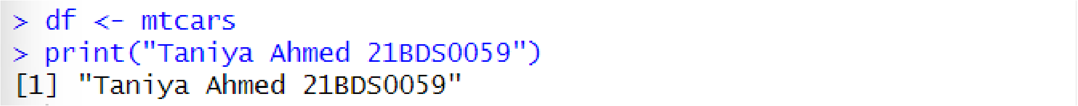
21BDS0059

1. **Loading mtcars dataset.**

CODE:

```
df <- mtcars

print("Taniya Ahmed 21BDS0059")
```

OUTPUT:



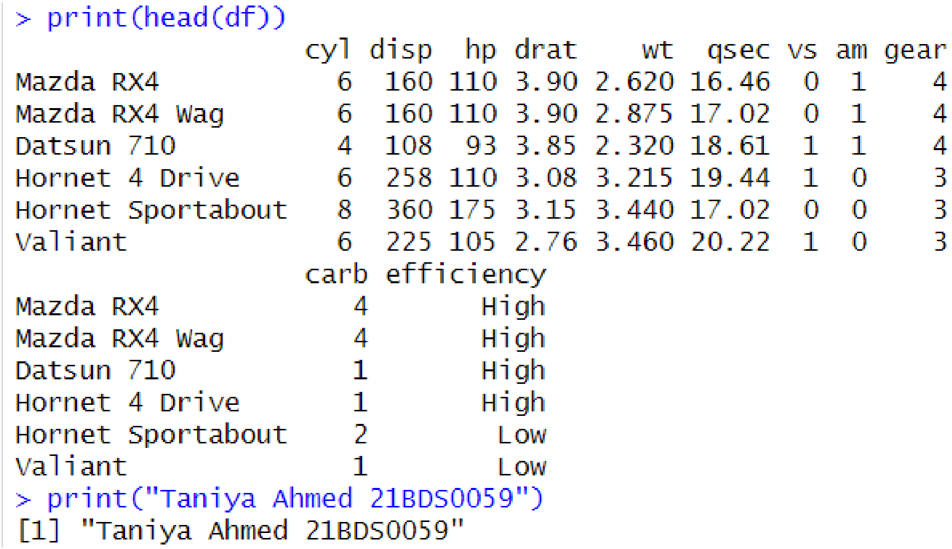
```
> df <- mtcars
> print("Taniya Ahmed 21BDS0059")
[1] "Taniya Ahmed 21BDS0059"
```

2. **Calculate labels for efficiency column and handle exception where the column doesn't exist, recheck the data frame.**

CODE:

```
if("mpg" %in% names(df)) {

  df$efficiency <- ifelse(df$mpg > median(df$mpg), "High", "Low")

  df <- df[, -which(names(df) == "mpg")]  # Remove 'mpg' column

} else {

  stop("The 'mpg' column does not exist in the dataset.")

}


print(head(df))

print("Taniya Ahmed 21BDS0059")
```

OUTPUT:



```
> print(head(df))
                  cyl disp  hp drat    wt  qsec vs am gear
Mazda RX4           6  160 110 3.90 2.620 16.46  0  1    4
Mazda RX4 Wag       6  160 110 3.90 2.875 17.02  0  1    4
Datsun 710          4  108  93 3.85 2.320 18.61  1  1    4
Hornet 4 Drive      6  258 110 3.08 3.215 19.44  1  0    3
Hornet Sportabout   8  360 175 3.15 3.440 17.02  0  0    3
Valiant             6  225 105 2.76 3.460 20.22  1  0    3
                  carb efficiency
Mazda RX4            4       High
Mazda RX4 Wag        4       High
Datsun 710           1       High
Hornet 4 Drive       1       High
Hornet Sportabout    2        Low
Valiant              1        Low
> print("Taniya Ahmed 21BDS0059")
[1] "Taniya Ahmed 21BDS0059"
```

3. **Crafting functions for calculating Euclidean distance, finding neighbours for an instance, predicting class of neighbours.**

CODE:

```
euclid_dist = function(a, b){
 return(sqrt(sum((a - b) ^ 2)))
}


calc_neighbours = function(train_data, test_data, k){
 distance = numeric(nrow(train_data))


 for(i in 1:nrow(train_data)){
  distance[i] = euclid_dist(as.numeric(train_data[i, -ncol(train_data)]), as.numeric(test_data))
 }


 distance_labels = data.frame(Distance = distance, Class = train_data[, ncol(train_data)])
 neighbours = distance_labels[order(distance_labels$Distance), ][1:k, ]


 return(neighbours)
```

}

predict_class = function(neighbours){

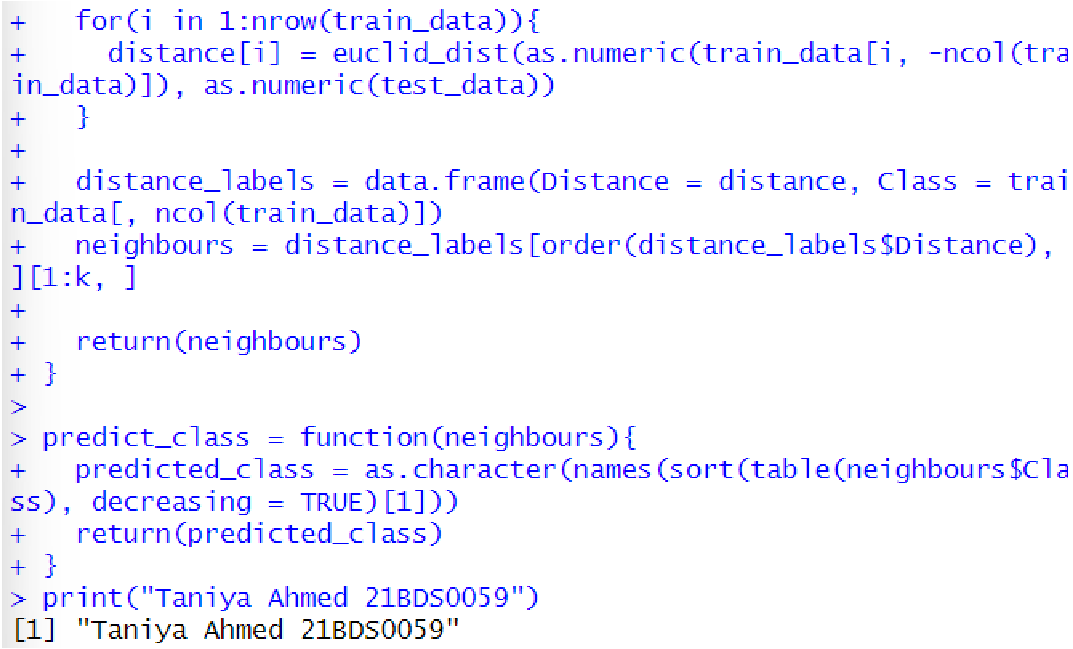
predicted_class = as.character(names(sort(table(neighbours$Class), decreasing = TRUE)[1]))

return(predicted_class)

}

print("Taniya Ahmed 21BDS0059")

OUTPUT:



```
+   for(i in 1:nrow(train_data)){
+     distance[i] = euclid_dist(as.numeric(train_data[i, -ncol(tra
in_data)]), as.numeric(test_data))
+   }
+
+   distance_labels = data.frame(Distance = distance, Class = trai
n_data[, ncol(train_data)])
+   neighbours = distance_labels[order(distance_labels$Distance),
][1:k, ]
+
+   return(neighbours)
+ }
>
> predict_class = function(neighbours){
+   predicted_class = as.character(names(sort(table(neighbours$Cla
ss), decreasing = TRUE)[1]))
+   return(predicted_class)
+ }
> print("Taniya Ahmed 21BDS0059")
[1] "Taniya Ahmed 21BDS0059"
```

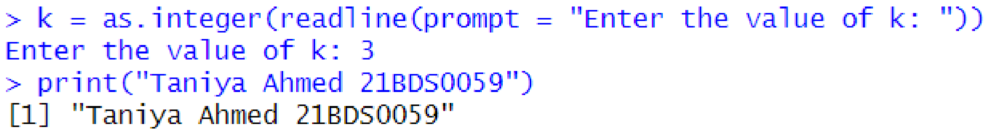
4. **Taking input value for k.**

CODE:

k = as.integer(readline(prompt = "Enter the value of k: "))

print("Taniya Ahmed 21BDS0059")

OUTPUT:



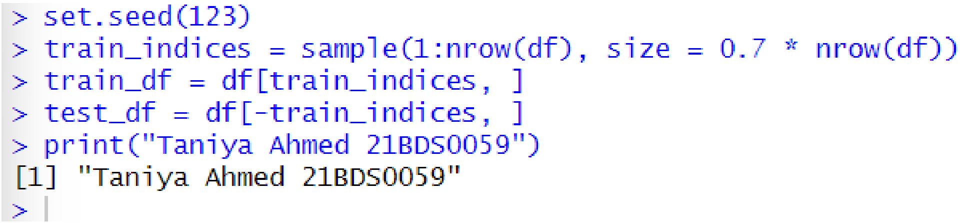
```
> k = as.integer(readline(prompt = "Enter the value of k: "))
Enter the value of k: 3
> print("Taniya Ahmed 21BDS0059")
[1] "Taniya Ahmed 21BDS0059"
```

5. **Splitting the training and testing dataset.**

CODE:

```
set.seed(123)

train_indices = sample(1:nrow(df), size = 0.7 * nrow(df))

train_df = df[train_indices, ]

test_df = df[-train_indices, ]

print("Taniya Ahmed 21BDS0059")
```

OUTPUT:



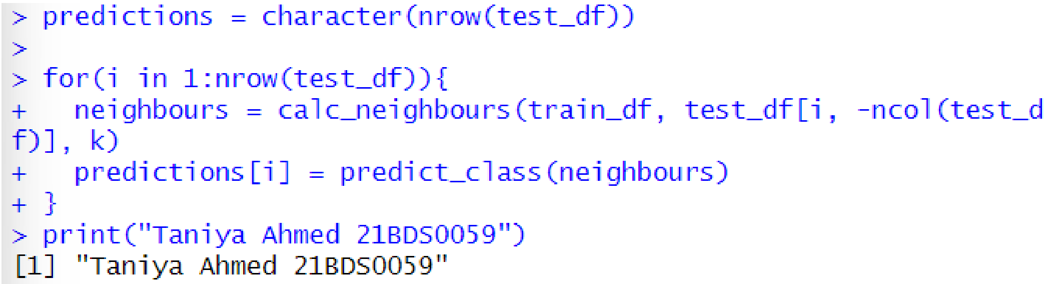
```
> set.seed(123)
> train_indices = sample(1:nrow(df), size = 0.7 * nrow(df))
> train_df = df[train_indices, ]
> test_df = df[-train_indices, ]
> print("Taniya Ahmed 21BDS0059")
[1] "Taniya Ahmed 21BDS0059"
> 
```

6. **Making a vector to store predictions.**

CODE:

```
predictions = character(nrow(test_df))


for(i in 1:nrow(test_df)){

 neighbours = calc_neighbours(train_df, test_df[i, -ncol(test_df)], k)

 predictions[i] = predict_class(neighbours)

}

print("Taniya Ahmed 21BDS0059")
```

OUTPUT:



```
> predictions = character(nrow(test_df))
> 
> for(i in 1:nrow(test_df)){
+   neighbours = calc_neighbours(train_df, test_df[i, -ncol(test_d
f)], k)
+   predictions[i] = predict_class(neighbours)
+ }
> print("Taniya Ahmed 21BDS0059")
[1] "Taniya Ahmed 21BDS0059"
```

7. **Creating a dataframe of the predicted and the actual classes.**

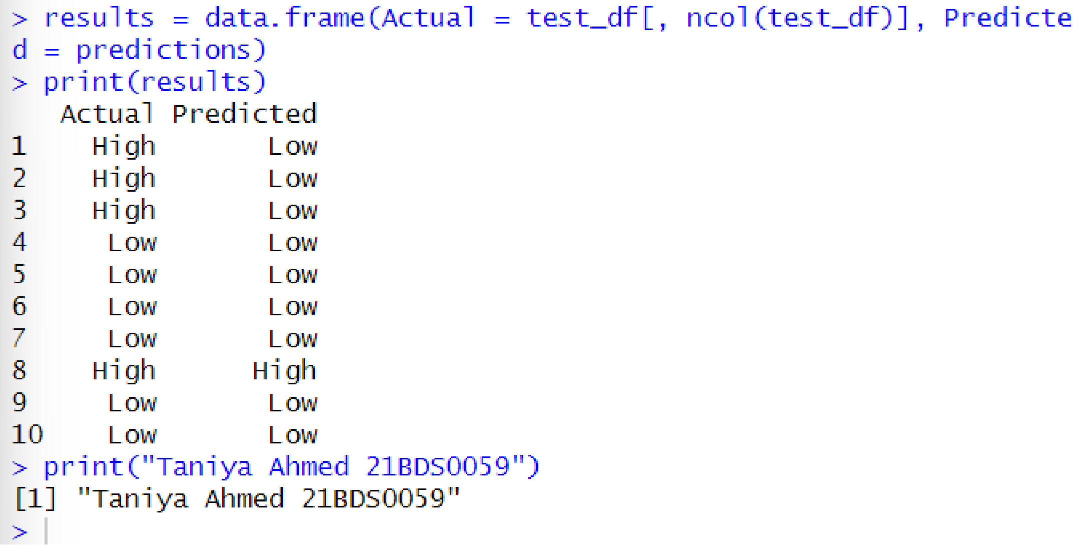
CODE:

results = data.frame(Actual = test_df[, ncol(test_df)], Predicted = predictions)

print(results)

print("Taniya Ahmed 21BDS0059")

OUTPUT:



```
> results = data.frame(Actual = test_df[, ncol(test_df)], Predicte
d = predictions)
> print(results)
   Actual Predicted
1    High       Low
2    High       Low
3    High       Low
4     Low       Low
5     Low       Low
6     Low       Low
7     Low       Low
8    High      High
9     Low       Low
10    Low       Low
> print("Taniya Ahmed 21BDS0059")
[1] "Taniya Ahmed 21BDS0059"
> 
```

8. **Calculating the accuracy.**

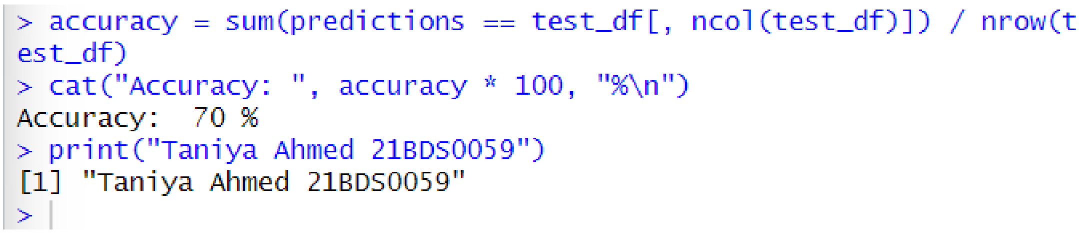
CODE:

accuracy = sum(predictions == test_df[, ncol(test_df)]) / nrow(test_df)

cat("Accuracy: ", accuracy * 100, "%\n")

print("Taniya Ahmed 21BDS0059")

OUTPUT:



```
> accuracy = sum(predictions == test_df[, ncol(test_df)]) / nrow(t
est_df)
> cat("Accuracy: ", accuracy * 100, "%\n")
Accuracy:  70 %
> print("Taniya Ahmed 21BDS0059")
[1] "Taniya Ahmed 21BDS0059"
> 
```