COMSATS UNIVERSITY ISLAMABAD

Data Science Fundamentals (DSC293) – BDS-2A

Department of Computer Science

Semester Group Project

- **Submitted By:**

Aleena Usman FA22-BDS-006
Omer Muhammadi FA22-BDS-029

- **Submitted To:**

Ma'am Hufsa Mohsin

- **Date of Submission:**
8th June 2023

COMSATS University Islamabad
Department of Computer Science
Data Science Fundamentals (DSC293) – BDS-2A
Project (CLO-5)

Table of Contents:

COMSATS University Islamabad
Department of Computer Science
Data Science Fundamentals (DSC293) – BDS-2A
Project (CLO-5)

Introduction:

This project documentation focuses on our analysis of the "**Exercise and Fitness Metrics Dataset**" obtained from **Kaggle.** We followed a series of steps to extract valuable insights and predictions from the dataset. We began by identifying a predictive problem and exploring the dataset thoroughly. Wrangling operations were applied to clean and transform the data into a tidy format. A suitable predictive algorithm was selected, and its predictions were visualized using multiple techniques. Classification and clustering algorithms were also employed, and their results were compared. We used box plots to identify the algorithm with the most stable outcomes. Finally, we built an interface to showcase the wrangling, classification, comparison, and results for an interactive experience. Through this project, we aimed to demonstrate the power of data analysis and predictive modeling using the available dataset.

Information About Dataset:

The "**Exercise and Fitness Metrics Dataset"** is a comprehensive dataset that captures various factors related to exercise, fitness, and weight management. The dataset includes a range of independent variables along with measurements of dream weight and actual weight. The exercise variable represents the type of exercise performed, while calories burned denotes the estimated number of calories burnt during the exercise session. Dream weight signifies the desired weight, and actual weight captures the measured weight with some natural variation.

Additional independent variables provide insights into the individuals performing the exercises. Age represents the age of the individuals, and gender indicates their gender (Male or Female). Duration records the length of each exercise session, and heart rate represents the average heart rate during the session. BMI, a commonly used health indicator, offers information about body composition. Weather conditions during exercise sessions are recorded, and exercise intensity provides a rating of the intensity level.

This dataset is valuable for analyzing relationships between exercise variables, calorie expenditure, weight-related measures, and other factors such as **age, gender, duration, heart rate, BMI, weather conditions, and exercise intensity**. It can be used for various purposes, including research in exercise science, fitness program development, weight management analysis, and correlation studies between exercise and health-related factors.

A Glimpse of Dataset:

	ID	Exercise	Calories.Burn	Dream.Weight	Actual.Weight	Age	Gender	Duration	Heart.Rate	BMI	Weather.Conditions	Exercise.Intensity
1	1	Exercise 2	286.9599	91.89253	96.30112	45	Male	37	170	29.42627	Rainy	5
2	2	Exercise 7	343.4530	64.16510	61.10467	25	Male	43	142	21.28635	Rainy	5
3	3	Exercise 4	261.2235	70.84622	71.76672	20	Male	20	148	27.89959	Cloudy	4
4	4	Exercise 5	127.1839	79.47701	82.98446	33	Male	39	170	33.72955	Sunny	10
5	5	Exercise 10	416.3184	89.96023	85.64317	29	Female	34	118	23.28611	Cloudy	3
6	6	Exercise 1	479.7227	78.88758	NA	60	Female	41	169	34.71934	Rainy	10
7	7	Exercise 9	457.6314	65.68113	61.81539	18	Male	53	103	34.59464	Cloudy	10
8	8	Exercise 4	272.9570	64.92956	62.80649	42	Male	25	104	22.05010	Cloudy	2
9	9	Exercise 10	195.0323	52.73107	54.53769	49	Male	37	161	30.94885	Sunny	1
10	10	Exercise 8	259.5311	95.16410	97.43683	NA	Male	55	103	31.22404	Cloudy	10
11	11	Exercise 5	248.5361	56.82978	54.14440	41	Male	52	151	34.01757	Cloudy	3

COMSATS University Islamabad
Department of Computer Science
Data Science Fundamentals (DSC293) – BDS-2A
Project (CLO-5)

Project Overview:

This project aimed to analyze a dataset by performing exploration, tidying, wrangling, and addressing predictive problems using graphing and plots, regression analysis, k-means clustering, and logistics regression. The dataset was thoroughly explored to understand its structure and variables, and visualizations were utilized for data analysis. Regression analysis was applied to establish relationships between variables and make predictions. K-means clustering was used to identify distinct groups in the dataset, while logistics regression was employed for classification tasks. The performance of these techniques was compared using evaluation metrics. The project findings were presented through visual representations, aiding in the interpretation of patterns and differences between the approaches. Overall, this project contributed to academic data analysis and predictive modeling research.

Libraries & Functions used:

We used the following libraries & packages in R-script to perform the analysis:

```
##installing packages required
install.packages("tidyverse")
install.packages("cluster")
install.packages("factoextra")
install.packages("randomForest")

##libraries
library(stats) ##clusters
library(gridExtra) ##displayplots
library(dplyr)
library(RColorBrewer)
library(randomForest)
library(tidyr)
library(tidytext)
library(stringr)
library(crayon)
library(DT)
library(ggthemes)
library(lubridate)
library(tidyselect)
library(scales)
library(ggplot2)
library(car)
library(tools)
library(tidyverse)  # data manipulation
library(cluster)    # clustering algorithms
library(factoextra) # clustering algorithms & visualization
```

Snapshots of code along with their output:

Exploring Dataset:

```
##exploring dataset
ncol(exercise_data)
nrow(exercise_data)
names(exercise_data)
dim(ex_data)
str(ex_data)
head(ex_data, 5)
tail(ex_data, 4)
summary(ex_data)
```

Output:

```
> ncol(exercise_data)
[1] 12
> nrow(exercise_data)
[1] 3864
> names(exercise_data)
 [1] "ID"                 "Exercise"           "Calories.Burn"      "Dream.Weight"       "Actual.Weight"      "Age"
 [7] "Gender"             "Duration"           "Heart.Rate"         "BMI"                "Weather.Conditions" "Exercise.Intensity"
>
```

Tail & Summary:

```
> tail(ex_data, 4)
       ID   Exercise Calories.Burn Dream.Weight Actual.Weight Age Gender Duration Heart.Rate      BMI Weather.Conditions
3861 3861 Exercise 4      486.3928     97.59896      92.70057  21 Female       49        160 26.60247              Rainy
3862 3862 Exercise 4      264.3077     94.94661      96.77894  57   Male       56        167 31.43535              Rainy
3863 3863 Exercise 9      185.9519     64.74391      68.66289  58 Female       60        128 19.77461              Rainy
3864 3864 Exercise 7      116.3604     56.75742      59.83340  35   Male       22        134 29.58133              Rainy
     Exercise.Intensity
3861                  5
3862                  9
3863                  1
3864                  1
> summary(ex_data)
       ID         Exercise         Calories.Burn    Dream.Weight    Actual.Weight         Age           Gender
 Min.   :   1   Length:3829        Min.   :100.0   Min.   :50.00   Min.   : 45.78   Min.   :18.00   Length:3829
 1st Qu.: 993   Class :character   1st Qu.:202.2   1st Qu.:62.38   1st Qu.: 62.51   1st Qu.:29.00   Class :character
 Median :1950   Mode  :character   Median :299.5   Median :75.52   Median : 75.50   Median :40.00   Mode  :character
 Mean   :1948                      Mean   :301.7   Mean   :75.12   Mean   : 75.18   Mean   :39.62
 3rd Qu.:2907                      3rd Qu.:403.7   3rd Qu.:87.69   3rd Qu.: 88.09   3rd Qu.:51.00
 Max.   :3864                      Max.   :499.9   Max.   :99.99   Max.   :104.31   Max.   :60.00
    Duration       Heart.Rate         BMI        Weather.Conditions Exercise.Intensity
 Min.   :20.00   Min.   :100.0   Min.   :18.50   Length:3829        Min.   : 1.000
 1st Qu.:30.00   1st Qu.:119.0   1st Qu.:22.69   Class :character   1st Qu.: 3.000
 Median :40.00   Median :140.0   Median :26.87   Mode  :character   Median : 5.000
 Mean   :40.21   Mean   :139.8   Mean   :26.80                      Mean   : 5.451
 3rd Qu.:51.00   3rd Qu.:160.0   3rd Qu.:30.94                      3rd Qu.: 8.000
 Max.   :60.00   Max.   :180.0   Max.   :35.00                      Max.   :10.000
>
```

Dimensions, Structure &head:

```
> dim(ex_data)
[1] 3829   12
> str(ex_data)
'data.frame':   3829 obs. of  12 variables:
 $ ID                : int  1 2 3 4 5 7 8 9 11 12 ...
 $ Exercise          : chr  "Exercise 2" "Exercise 7" "Exercise 4" "Exercise 5" ...
 $ Calories.Burn     : num  287 343 261 127 416 ...
 $ Dream.Weight      : num  91.9 64.2 70.8 79.5 90 ...
 $ Actual.Weight     : num  96.3 61.1 71.8 83 85.6 ...
 $ Age               : int  45 25 20 33 29 18 42 49 41 35 ...
 $ Gender            : chr  "Male" "Male" "Male" "Male" ...
 $ Duration          : int  37 43 20 39 34 53 25 37 52 28 ...
 $ Heart.Rate        : int  170 142 148 170 118 103 104 161 151 158 ...
 $ BMI               : num  29.4 21.3 27.9 33.7 23.3 ...
 $ Weather.Conditions: chr  "Rainy" "Rainy" "Cloudy" "Sunny" ...
 $ Exercise.Intensity: int  5 5 4 10 3 10 2 1 3 6 ...
 - attr(*, "na.action")= 'omit' Named int [1:35] 6 10 19 36 40 43 87 93 118 129 ...
  ..- attr(*, "names")= chr [1:35] "6" "10" "19" "36" ...
> head(ex_data, 5)
  ID    Exercise Calories.Burn Dream.Weight Actual.Weight Age Gender Duration Heart.Rate      BMI Weather.Conditions
1  1  Exercise 2      286.9599     91.89253      96.30112  45   Male       37        170 29.42627              Rainy
2  2  Exercise 7      343.4530     64.16510      61.10467  25   Male       43        142 21.28635              Rainy
3  3  Exercise 4      261.2235     70.84622      71.76672  20   Male       20        148 27.89959             Cloudy
4  4  Exercise 5      127.1839     79.47701      82.98446  33   Male       39        170 33.72955              Sunny
5  5 Exercise 10      416.3184     89.96023      85.64317  29 Female       34        118 23.28611             Cloudy
  Exercise.Intensity
1                  5
2                  5
3                  4
4                 10
5                  3
```

Identifying missing values of the Dataset:

```r
##tidy data operations
is.na(exercise_data)
```

Output:

True indicates missing values

```
> is.na(exercise_data)
         ID Exercise Calories.Burn Dream.Weight Actual.Weight   Age Gender Duration Heart.Rate   BMI Weather.Conditions
 [1,] FALSE    FALSE         FALSE        FALSE         FALSE FALSE  FALSE    FALSE      FALSE FALSE              FALSE
 [2,] FALSE    FALSE         FALSE        FALSE         FALSE FALSE  FALSE    FALSE      FALSE FALSE              FALSE
 [3,] FALSE    FALSE         FALSE        FALSE         FALSE FALSE  FALSE    FALSE      FALSE FALSE              FALSE
 [4,] FALSE    FALSE         FALSE        FALSE         FALSE FALSE  FALSE    FALSE      FALSE FALSE              FALSE
 [5,] FALSE    FALSE         FALSE        FALSE         FALSE FALSE  FALSE    FALSE      FALSE FALSE              FALSE
 [6,] FALSE    FALSE         FALSE        FALSE          TRUE FALSE  FALSE    FALSE      FALSE FALSE              FALSE
 [7,] FALSE    FALSE         FALSE        FALSE         FALSE FALSE  FALSE    FALSE      FALSE FALSE              FALSE
 [8,] FALSE    FALSE         FALSE        FALSE         FALSE FALSE  FALSE    FALSE      FALSE FALSE              FALSE
 [9,] FALSE    FALSE         FALSE        FALSE         FALSE FALSE  FALSE    FALSE      FALSE FALSE              FALSE
[10,] FALSE    FALSE         FALSE        FALSE         FALSE  TRUE  FALSE    FALSE      FALSE FALSE              FALSE
[11,] FALSE    FALSE         FALSE        FALSE         FALSE FALSE  FALSE    FALSE      FALSE FALSE              FALSE
[12,] FALSE    FALSE         FALSE        FALSE         FALSE FALSE  FALSE    FALSE      FALSE FALSE              FALSE
[13,] FALSE    FALSE         FALSE        FALSE         FALSE FALSE  FALSE    FALSE      FALSE FALSE              FALSE
[14,] FALSE    FALSE         FALSE        FALSE         FALSE FALSE  FALSE    FALSE      FALSE FALSE              FALSE
[15,] FALSE    FALSE         FALSE        FALSE         FALSE FALSE  FALSE    FALSE      FALSE FALSE              FALSE
[16,] FALSE    FALSE         FALSE        FALSE         FALSE FALSE  FALSE    FALSE      FALSE FALSE              FALSE
[17,] FALSE    FALSE         FALSE        FALSE         FALSE FALSE  FALSE    FALSE      FALSE FALSE              FALSE
[18,] FALSE    FALSE         FALSE        FALSE         FALSE FALSE  FALSE    FALSE      FALSE FALSE              FALSE
[19,] FALSE    FALSE         FALSE        FALSE         FALSE  TRUE  FALSE    FALSE      FALSE FALSE              FALSE
[20,] FALSE    FALSE         FALSE        FALSE         FALSE FALSE  FALSE    FALSE      FALSE FALSE              FALSE
[21,] FALSE    FALSE         FALSE        FALSE         FALSE FALSE  FALSE    FALSE      FALSE FALSE              FALSE
[22,] FALSE    FALSE         FALSE        FALSE         FALSE FALSE  FALSE    FALSE      FALSE FALSE              FALSE
[23,] FALSE    FALSE         FALSE        FALSE         FALSE FALSE  FALSE    FALSE      FALSE FALSE              FALSE
[24,] FALSE    FALSE         FALSE        FALSE         FALSE FALSE  FALSE    FALSE      FALSE FALSE              FALSE
[25,] FALSE    FALSE         FALSE        FALSE         FALSE FALSE  FALSE    FALSE      FALSE FALSE              FALSE
[26,] FALSE    FALSE         FALSE        FALSE         FALSE FALSE  FALSE    FALSE      FALSE FALSE              FALSE
[27,] FALSE    FALSE         FALSE        FALSE         FALSE FALSE  FALSE    FALSE      FALSE FALSE              FALSE
[28,] FALSE    FALSE         FALSE        FALSE         FALSE FALSE  FALSE    FALSE      FALSE FALSE              FALSE
[29,] FALSE    FALSE         FALSE        FALSE         FALSE FALSE  FALSE    FALSE      FALSE FALSE              FALSE
[30,] FALSE    FALSE         FALSE        FALSE         FALSE FALSE  FALSE    FALSE      FALSE FALSE              FALSE
[31,] FALSE    FALSE         FALSE        FALSE         FALSE FALSE  FALSE    FALSE      FALSE FALSE              FALSE
[32,] FALSE    FALSE         FALSE        FALSE         FALSE FALSE  FALSE    FALSE      FALSE FALSE              FALSE
[33,] FALSE    FALSE         FALSE        FALSE         FALSE FALSE  FALSE    FALSE      FALSE FALSE              FALSE
```

Count of missing values:

```r
sum(is.na(exercise_data))
```

Output:

```
> sum(is.na(exercise_data))
[1] 37
```

Removing missing values & displaying rows:

```
> ex_data <- na.omit(exercise_data)
> nrow(ex_data)
[1] 3829
```

Output:

```
ex_data <- na.omit(exercise_data)
nrow(ex_data)
```

After tidying the dataset, the number of rows are reduced from 3864 to 3829.

Data Wrangling:

```
##data Wrangling
ex_data %>%
  select(Age, Gender, Heart.Rate) %>% head(5)

ex_data %>%
  mutate(Mean_BMI = mean(BMI)) %>%
  select(Age, BMI, Mean_BMI) %>%
  head(5)

ex_data %>%
  filter(Age == "19")%>%
  select(Age, Gender, BMI)%>%
  head(5)


ex_data %>%
  rename(Weight = "Actual.Weight")%>%
  select(Gender, Age, Weight)%>%
  head(5)

ex_data %>%
  arrange(Calories.Burn)%>%
  head(5)

ex_data %>%
  group_by(Gender)%>%
  summarise(Mean_weight = mean(Actual.Weight),
            calories_burned = mean(Calories.Burn),age = mean(Age))
```

Output:

```
+   select(Age, Gender
+   head(5)
  Age Gender      BMI
1  19   Male 26.34012
2  19   Male 22.21857
3  19 Female 25.38978
4  19   Male 20.33647
5  19 Female 19.49576
```

```
+   rename(Weight = "Ac
+   select(Gender, Age,
+   head(5)
  Gender Age   Weight
1   Male  45 96.30112
2   Male  25 61.10467
3   Male  20 71.76672
4   Male  33 82.98446
5 Female  29 85.64317
```

```
[1] 3829
> ex_data %>%
+   select(Age, Gender, Heart.Rate) %>% head(5)
  Age Gender Heart.Rate
1  45   Male        170
2  25   Male        142
3  20   Male        148
4  33   Male        170
5  29 Female        118
```

```
+   select(Age, BMI, Mean_
+   head(5)
  Age      BMI Mean_BMI
1  45 29.42627 26.79911
2  25 21.28635 26.79911
3  20 27.89959 26.79911
4  33 33.72955 26.79911
5  29 23.28611 26.79911
```

```
+   arrange(Calories.Burn)%>%
+   head(5)
    ID   Exercise Calories.Burn Dream.Weight Actual.Weight Age Gender Duration Heart.Rate      BMI Weather.Conditions
1 3431 Exercise 9      100.0094     63.10321      58.73877  32 Female       31        167 26.03899             Cloudy
2 2246 Exercise 3      100.0147     64.01293      63.14755  42   Male       39        107 31.86006             Cloudy
3 2775 Exercise 5      100.0310     76.20813      80.86662  29 Female       31        109 21.69059              Rainy
4 3678 Exercise 1      100.3351     82.32427      84.60040  42 Female       56        157 29.28227             Cloudy
5 1746 Exercise 8      100.6405     91.31956      88.13213  53 Female       34        144 32.04432              Rainy
  Exercise.Intensity
1                  5
2                 10
3                  1
4                  6
5                  5
```

```
+                   calories_burned = mean(calori
# A tibble: 2 × 4
  Gender Mean_weight calories_burned   age
  <chr>        <dbl>           <dbl> <dbl>
1 Female        75.4            305.  39.9
2 Male          75.0            298.  39.3
>
```

Taking random-rows:

```
##taking random rows
random_rows <- exercise_data[sample(nrow(exercise_data), 150), ]
random_rows
```

Visualization:

Finding Co-relation among variables:

```r
##finding Co-relation among variables

cor(ex_data$Age, ex_data$Heart.Rate)
cor(ex_data$Exercise.Intensity, ex_data$Calories.Burn)
cor(ex_data$Duration, ex_data$Calories.Burn)
```

Output:

```
> cor(ex_data$Age, ex_data$Heart.Rate)
[1] -0.009209017
> cor(ex_data$Exercise.Intensity, ex_data$Calories.Burn)
[1] 0.01111867
> cor(ex_data$Duration, ex_data$Calories.Burn)
[1] 0.0242086
```

Linear regression btw heart-rate &age:

```r
##linear regression
model <- lm(Heart.Rate ~ Age , data = random_rows)
model
```

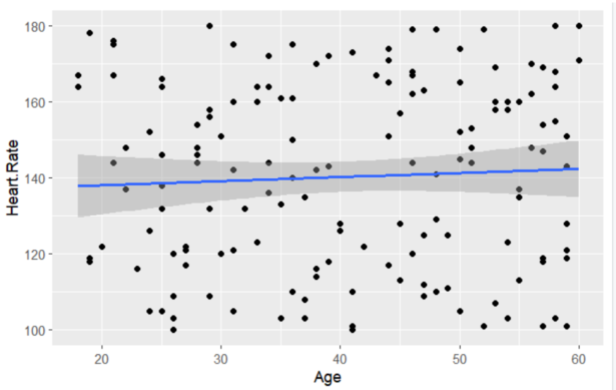
Output:

```
Call:
lm(formula = Heart.Rate ~ Age, data = random_rows)

Coefficients:
(Intercept)          Age
   135.9437       0.1071
```

Visualize the relationship between heart-rate & age:

```r
## visualize the relationship between Age and Heart.Rate
ggplot(random_rows, aes(x = Age , y = Heart.Rate))+
  geom_point()

## create a scatter plot with a regression line
ggplot(random_rows, aes(x = Age , y = Heart.Rate))+
  geom_point()+
  stat_smooth(method = lm)
```

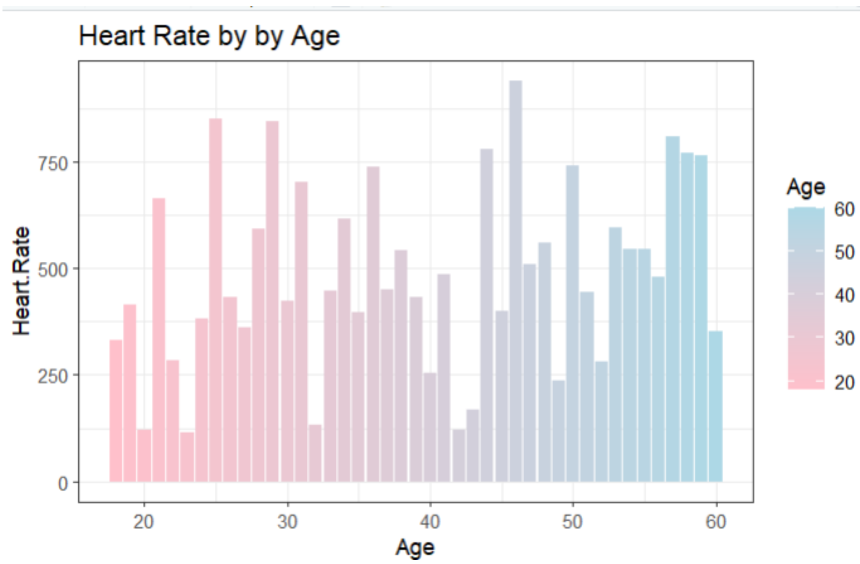
Output:



Bar plot:

```
##create a bar plot with a color gradient
## displays the relationship between age and heart rate
ggplot(random_rows, aes(x = Age, y = Heart.Rate, fill = Age)) +
  geom_bar(stat = "identity") +
  ggtitle("Heart Rate by by Age") +
  theme_bw() +
  scale_fill_gradient(low = "pink", high = "lightblue")
```

Output:

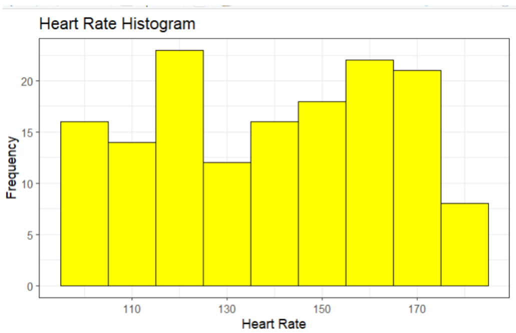


COMSATS University Islamabad
Department of Computer Science
Data Science Fundamentals (DSC293) – BDS-2A
Project (CLO-5)

Histogram of Heart.rate:

```
##create a histogram of the "Heart.Rate"
ggplot(random_rows, aes(x = Heart.Rate)) +
  geom_histogram(binwidth = 10, fill = "yellow", color = "black") +
  ggtitle("Heart Rate Histogram") +
  xlab("Heart Rate") +
  ylab("Frequency") +
  theme_bw()
```

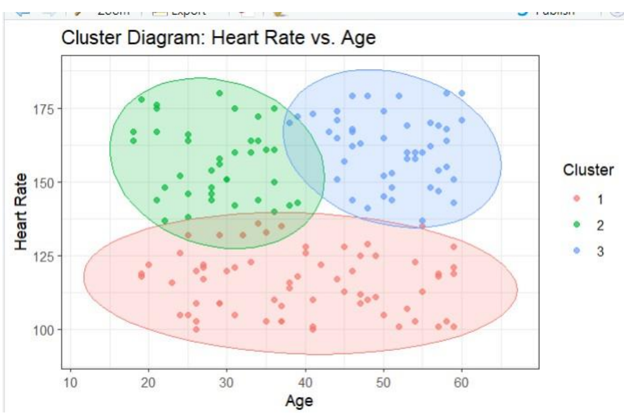
Output:



k-means cluster:

```
##is.finite() function to identify any NaN or Inf values
problematic_values <- !is.finite(random_rows$Age) | !is.finite(random_rows$Heart.Rate)
set.seed(1)  # Set seed for reproducibility
k <- 3  # Number of clusters
kmeans_result <- kmeans(random_rows[, c("Age", "Heart.Rate")], centers = k)

# Add cluster labels to the data frame
random_rows$Cluster <- as.factor(kmeans_result$cluster)

# Calculate cluster centers
cluster_centers <- kmeans_result$centers
ggplot(random_rows, aes(x = Age, y = Heart.Rate, color = Cluster)) +
geom_point(alpha = 0.7) +
stat_ellipse(aes(fill = Cluster), geom = "polygon", alpha = 0.2, show.legend = FALSE) +
ggtitle("Cluster Diagram: Heart Rate vs. Age") +
xlab("Age") +
ylab("Heart Rate") +
theme_bw()
```

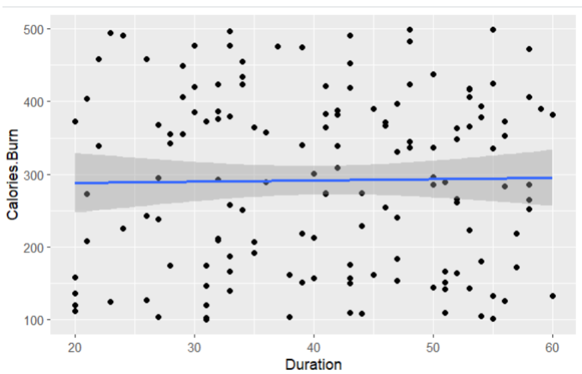
Output:



Visualize the relationship between duration & calories.burn

```
# visualize the relationship between duration & calories.burn
ggplot(random_rows, aes(x = Duration , y = Calories.Burn))+
  geom_point()

## create a scatter plot with a regression line
ggplot(random_rows, aes(x = Duration , y = Calories.Burn))+
  geom_point()+
  stat_smooth(method = lm)
```

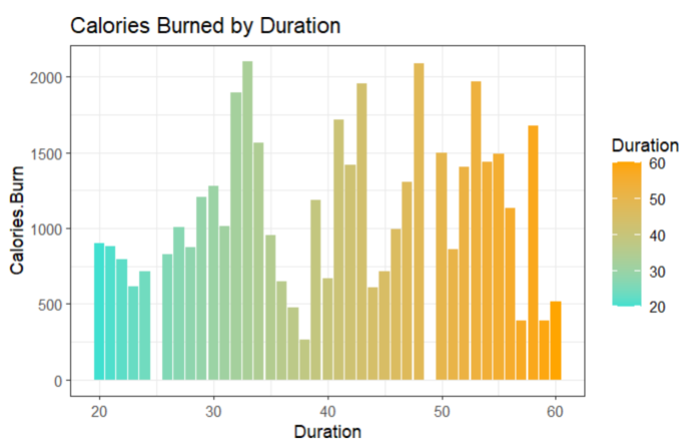
Output:



Bar plot visualizing the relationship between the "Duration" and "Calories. Burn

```
##create a bar plot visualizing the relationship between the "Duration" and "Ca
ggplot(random_rows, aes(x = Duration , y = Calories.Burn, fill = Duration)) +
  geom_bar(stat = "identity") +
  ggtitle("Calories Burned by Duration") +
  theme_bw() +
  scale_fill_gradient(low = "turquoise", high = "orange")
```

Output:



The predictive problem from the dataset:

We used the "**test and train data**" technique to "**Predict calories burned**"

Once the model is trained and evaluated, you can use it to make predictions on new, unseen data to estimate the number of calories burned during exercise.

- **Scale the data to remove extreme values**

```
# Scaling the dataset to remove extreme values
head(exercise_data, 6)

non_numeric_cols <- c("Exercise", "Gender", "Weather.Conditions")
exercise_data[non_numeric_cols] <- lapply(exercise_data[non_numeric_cols], as.factor)

# Ensure numeric columns are correctly identified as numeric
exercise_data <- exercise_data %>%
  mutate_if(is.factor, as.numeric)

# Scale the numeric columns in the dataset
numeric_cols <- setdiff(colnames(exercise_data), non_numeric_cols)
scaled_data <- exercise_data
scaled_data[numeric_cols] <- scale(scaled_data[numeric_cols])

# Append the target variable (Calories.Burn) to the scaled dataset
scaled_data$Calories.Burn <- exercise_data$Calories.Burn
```

- **Model the data to train data and test data**

```
# Select relevant features
features <- c("Exercise", "Age", "Gender", "Duration", "Heart.Rate", "BMI", "Weather.Conditions", "Exercise.Intensity")
target <- "Calories.Burn"

#Split the dataset into training and testing sets
set.seed(123)
train_indices <- sample(1:nrow(scaled_data), 0.8 * nrow(scaled_data))
train_data <- scaled_data[train_indices, ]
test_data <- scaled_data[-train_indices, ]
dim(train_data)
dim(test_data)
```

Output:

```
> train_data <- sca
> test_data <- scal
> dim(train_data)
[1] 3091   12
> dim(test_data)
[1] 773  12
```

Train with models:

- **We used k-means algorithm ,linear regression and random forest model for predictive algorithm to solve our problem**

```
# Remove rows with missing values
train_data <- train_data[complete.cases(train_data), ]

# Train the random forest regression model
random_forest_model <- randomForest(Calories.Burn ~ ., data = train_data[, c(features, target)])

#  Train the linear regression model
linear_model <- lm(Calories.Burn ~ ., data = train_data[, c(features, target)])


#  Perform k-means clustering
kmeans_model <- kmeans(scaled_data[, features], centers = 3)
```

making predictions:

```
#  Assign cluster labels to the data
scaled_data$Cluster <- as.factor(kmeans_model$cluster)

#  Make predictions on the test data
linear_predictions <- predict(linear_model, newdata = test_data)
random_forest_predictions <- predict(random_forest_model, newdata = test_data)
```

Output:

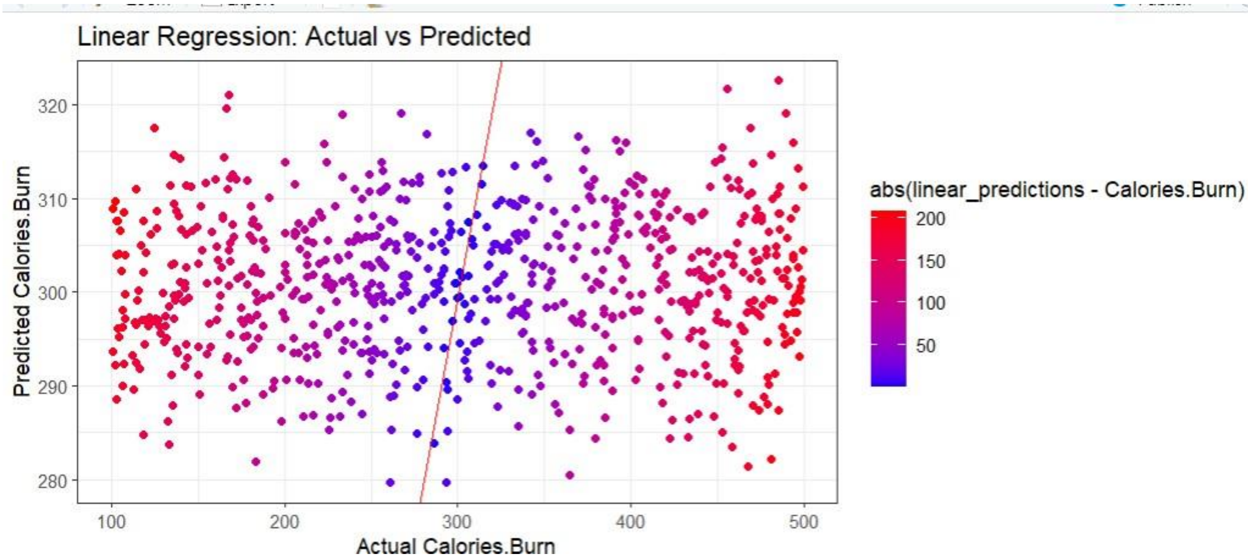
```
> linear_predictions <- predict(linear_model, newdata = test_data)
> head(linear_predictions,5)
       3        6       14       22       43 
288.8144 299.2992 300.5864 297.3819 294.4256 
> random_forest_predictions <- predict(random_forest_model, newdata = test_data)
> head(random_forest_predictions,6)
       3        6       14       22       43       47 
310.0096 309.1715 313.6175 313.1811 276.5987 292.9865 
> 
```

- **Visualize the predictions in multiple ways**
  1. **Scatter plot(actual vs predicted values for linear regression)**

```
#  Visualize the results

# Scatter plot of actual vs predicted values for linear regression
linear_plot <- ggplot(data = test_data, aes(x = Calories.Burn, y = linear_predictions, color = abs(linear_predictions - Calories.Burn))) +
  geom_point() +
  geom_abline(intercept = 0, slope = 1, linetype = "solid", color = "red") +
  ggtitle("Linear Regression: Actual vs Predicted") +
  xlab("Actual Calories.Burn") +
  ylab("Predicted Calories.Burn") +
  theme_bw() +
  scale_color_gradient(low = "blue", high = "red")
linear_plot
```

COMSATS University Islamabad
Department of Computer Science
Data Science Fundamentals (DSC293) – BDS-2A
Project (CLO-5)

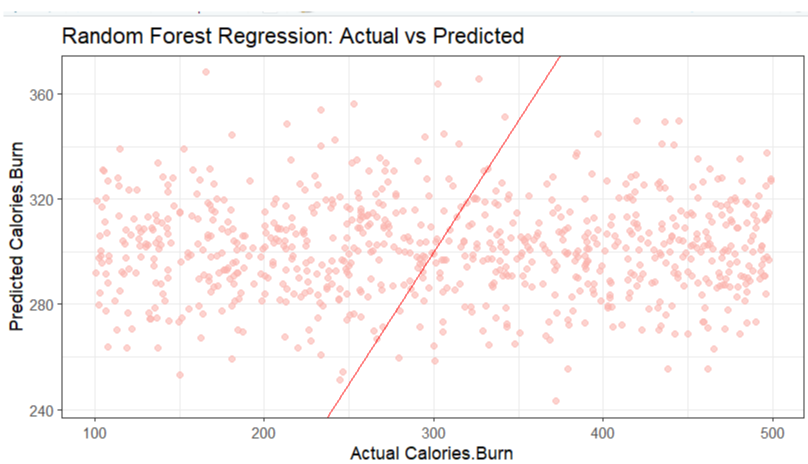
Output:



2. Scatter plot of actual vs predicted values for random forest regression

```
# Scatter plot of actual vs predicted values for random forest regression
colors <- brewer.pal(3, "Pastel1")
random_forest_plot <- ggplot(data = test_data, aes(x = Calories.Burn, y = random_forest_predictions)) +
  geom_point(color = colors[1], alpha = 0.6) +
  geom_abline(intercept = 0, slope = 1, linetype = "solid", color = "red") +
  ggtitle("Random Forest Regression: Actual vs Predicted") +
  xlab("Actual Calories.Burn") +
  ylab("Predicted Calories.Burn") +
  theme_bw() +
  scale_color_manual(values = colors, guide = guide_colorbar(title = "Cluster", title.position = "right"))
random_forest_plot
```

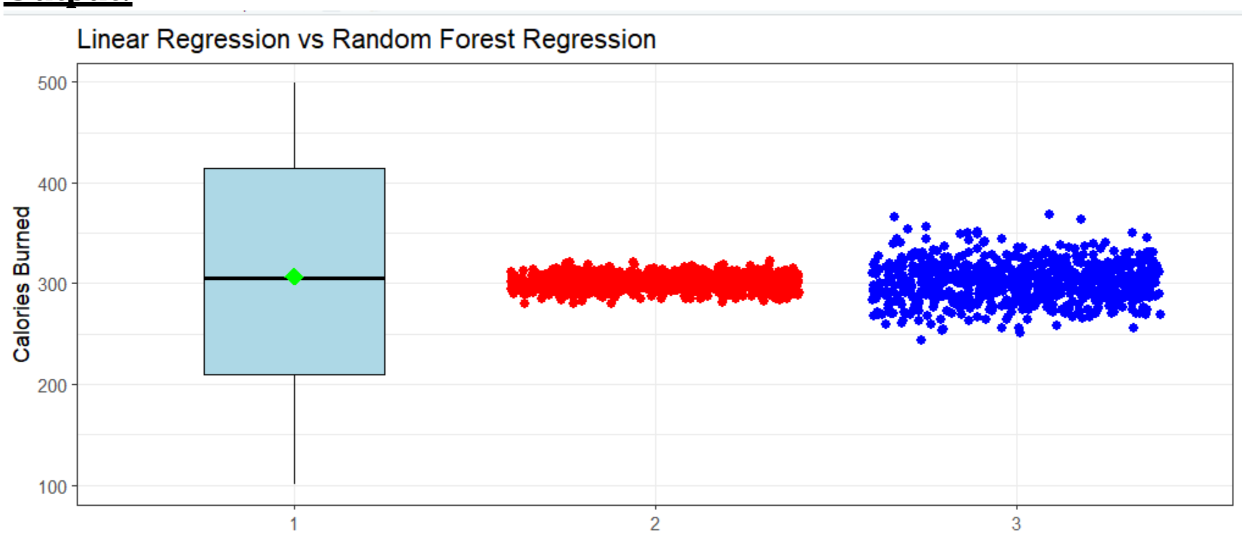
Output:



3. Box plot to compare linear regression and random forest regression

```r
# Box plot to compare linear regression and random forest regression
results <- data.frame(Actual = test_data$Calories.Burn, Linear_Predicted = linear_predictions, RF_Predicted = random_forest_predictions
boxplot_plot <- ggplot(results, aes(x = factor(1), y = Actual)) +
  geom_boxplot(width = 0.5, fill = "lightblue", color = "black") +
  geom_jitter(aes(x = factor(2), y = Linear_Predicted), color = "red", size = 2) +
  geom_jitter(aes(x = factor(3), y = RF_Predicted), color = "blue", size = 2) +
  stat_summary(fun = mean, geom = "point", shape = 18, size = 4, color = "green") +
  geom_smooth(aes(x = factor(1), y = Actual), method = "lm", se = FALSE, color = "blue", linetype = "dashed") +
  ggtitle("Linear Regression vs Random Forest Regression") +
  xlab("") +
  ylab("Calories Burned") +
  theme_bw()
boxplot_plot
```

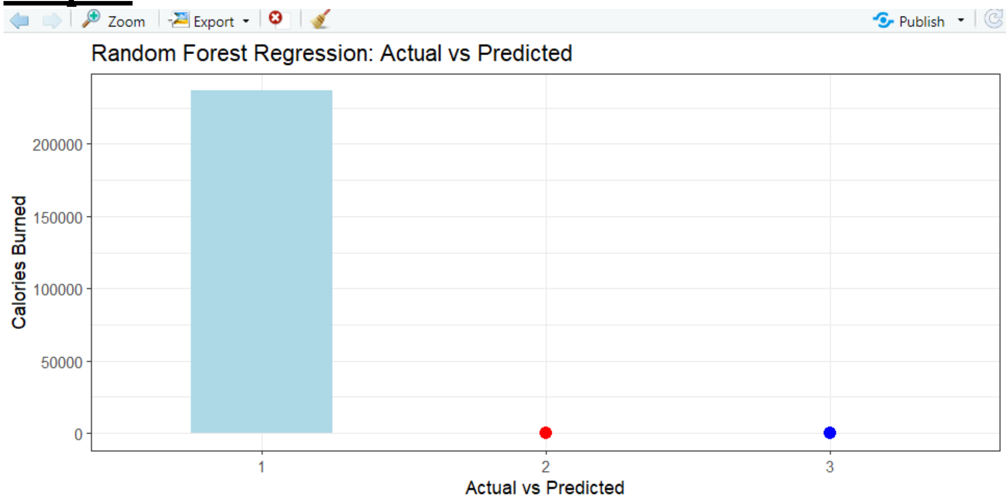
Output:



4. Bar plot to compare actual and predicted values

```r
# Bar plot to compare actual and predicted values
barplot_plot <- ggplot(results, aes(x = factor(1), y = Actual)) +
  geom_bar(stat = "identity", fill = "lightblue", width = 0.5) +
  geom_point(aes(x = factor(2), y = Linear_Predicted), color = "red", size = 3) +
  geom_point(aes(x = factor(3), y = RF_Predicted), color = "blue", size = 3) +
  geom_smooth(aes(x = factor(1), y = Actual), method = "lm", se = FALSE, color = "blue", linetype = "dashed") +
  ggtitle("Random Forest Regression: Actual vs Predicted") +
  xlab("Actual vs Predicted") +
  ylab("Calories Burned") +
  theme_bw()
barplot_plot
```

Output:

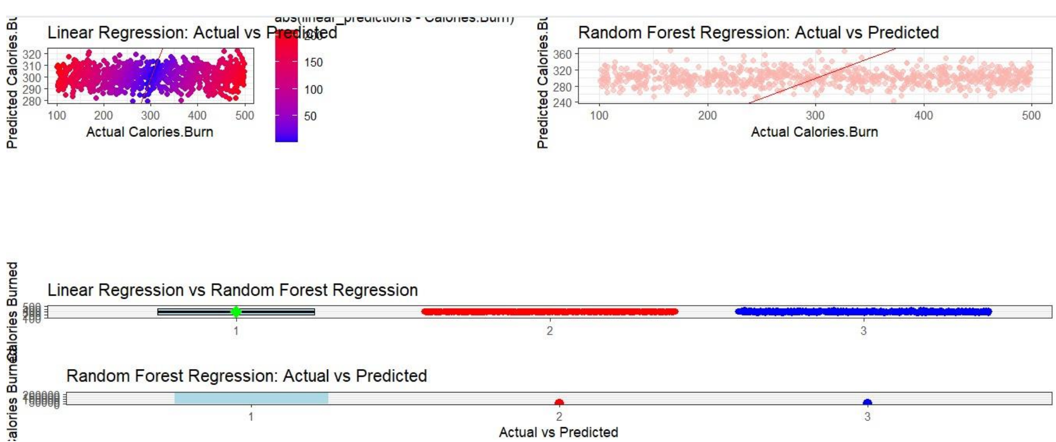


- ## Arrange and display the plots

```
# Arrange and display the plots
plots <- grid.arrange(linear_plot, random_forest_plot, nrow = 2, ncol = 2)
plots2 <- grid.arrange(boxplot_plot, barplot_plot, nrow = 2, ncol = 1)
final_plot <- grid.arrange(plots, plots2, heights = c(0.6, 0.4))
print(final_plot)
```

Output:

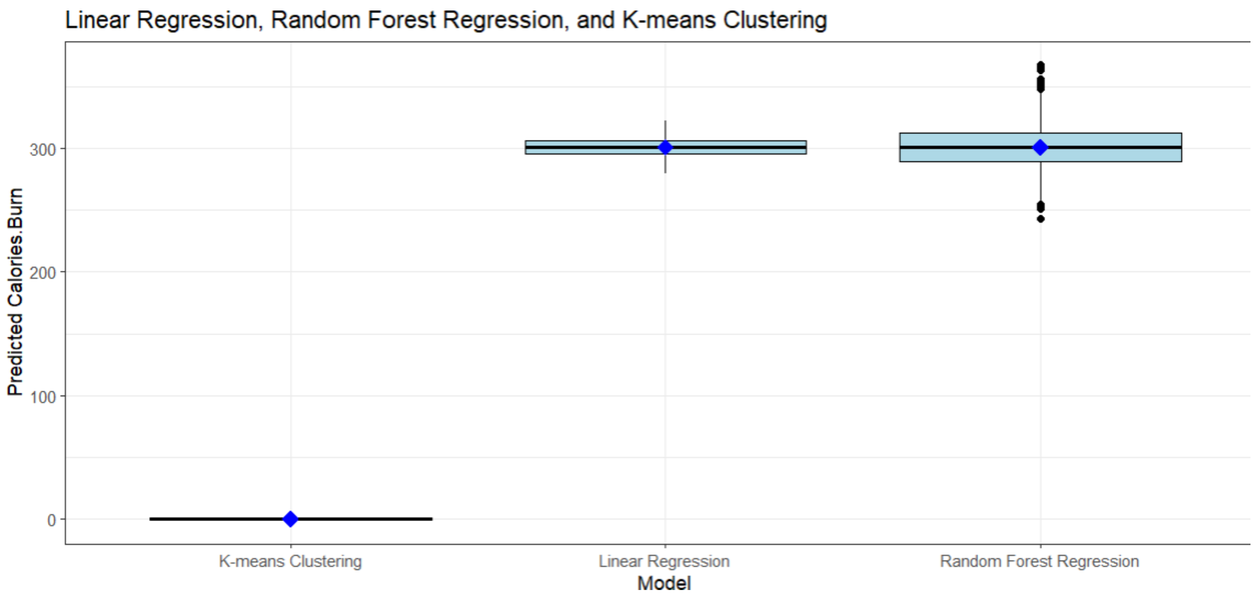
The purpose of this code is to display the scatter plots of linear regression and random forest regression side by side, as well as the box plot and bar plot stacked vertically. The gridExtra library enables the combination of multiple plots into a single figure for better visualization and comparison. It allows you to examine how the predicted values compare to the actual values and compare the performance of the two models. The box plot andbar plot provide additional visualizations to compare the predictions and assess their accuracy.



5. boxplot clearly identifies which algorithm has stable results as compared to other

```
# Create a data frame with the predicted values

predictions <- data.frame(Model = c(rep("Linear Regression", length(linear_predictions)),
                                    rep("Random Forest Regression", length(random_forest_predictions)),
                                    rep("K-means Clustering", nrow(scaled_data))))
Predicted = c(linear_predictions, random_forest_predictions, scaled_data$Calories.Burn)

# Create the boxplot with regression line
ggplot(predictions, aes(x = Model, y = Predicted)) +
geom_boxplot(fill = "lightblue", color = "black") +
stat_summary(fun = mean, geom = "point", shape = 18, size = 4, color = "blue") +
stat_summary(fun = function(x) lm(y ~ x)$coef[1], geom = "line", aes(group = 1), linetype = "dashed", color = "red") +
ggtitle("Linear Regression, Random Forest Regression, and K-means Clustering") +
xlab("Model") +
ylab("Predicted Calories.Burn") +
theme_bw()
```

output:



- **calculating MSE ,R squared & MAE for models:**

```r
# Calculate the SSE for k-means clustering
kmeans_sse <- sum(kmeans_model$withinss)

# Calculate the MSE for k-means clustering
kmeans_mse <- kmeans_sse / nrow(scaled_data)

#Compare the predictions and evaluate accuracy
linear_mse <- mean((linear_predictions - test_data$Calories.Burn)^2)
random_forest_mse <- mean((random_forest_predictions - test_data$Calories.Burn)^2)

# Print the MSE for linear regression  ,random forest regression,k-means algo

cat("Linear Regression MSE:", linear_mse, "\n")
cat("Random Forest Regression MSE:", random_forest_mse, "\n")
cat("K-means Clustering MSE:", kmeans_mse, "\n")   # A lower MSE indicates a better fit of the model to the data.

# Calculate R-squared for linear regression
linear_r_squared <- 1 - (sum((test_data$Calories.Burn - linear_predictions)^2) / sum((test_data$Calories.Burn - mean(test_data$Calories.Burn))^2))

# Calculate MAE for linear regression
linear_mae <- mean(abs(test_data$Calories.Burn - linear_predictions))

# Calculate R-squared for random forest regression
random_forest_r_squared <- 1 - (sum((test_data$Calories.Burn - random_forest_predictions)^2) / sum((test_data$Calories.Burn - mean(test_data$Calories.Burn))^2))

# Calculate MAE for random forest regression
random_forest_mae <- mean(abs(test_data$Calories.Burn - random_forest_predictions))

# Print the accuracy metrics
cat("Linear Regression R-squared:", linear_r_squared, "\n")
cat("Linear Regression MAE:", linear_mae, "\n")
cat("Random Forest Regression R-squared:", random_forest_r_squared, "\n")
cat("Random Forest Regression MAE:", random_forest_mae, "\n")
```

output:

Based on the calculated **Mean Squared Error (MSE)** values, the model with the lowest MSE is the one that provides the best fit for the data. In this case, the model with the lowest MSE is the K-means Clustering model, with an MSE of 6.813554.

```
> cat("Linear Regression MSE:", linear_mse, "\
Linear Regression MSE: 13712.6
> cat("Random Forest Regression MSE:", random_
Random Forest Regression MSE: 14117.14
> cat("K-means Clustering MSE:", kmeans_mse, '
K-means Clustering MSE: 6.813554
```

Based on the provided metrics, it appears that none of the models (Linear Regression and Random Forest Regression) have performed well in terms of accuracy and fit for this dataset.

The Linear Regression model has a very low R-squared value (7.689056e-05), indicating that it explains very little of the variance in the target variable. Additionally, the mean absolute error (MAE) for the Linear Regression model is 101.6754, indicating a large average difference between the predicted and actual values.Similarly, the Random Forest Regression model also has a low R-squared value (-0.02942208) and a higher MAE of 103.1851.Based on these metrics, it is difficult to determine a clear "best fit" model for this dataset. It suggests that the models might not be well-suited for capturing the underlying patterns and relationships in the data. It could be beneficial to explore other modeling techniques or investigate if there are any issues with the data itself.

```
> cat("Linear Regression R-squared:", linear_r_squa
Linear Regression R-squared: 7.689056e-05
> cat("Linear Regression MAE:", linear_mae, "\n")
Linear Regression MAE: 101.6754
> cat("Random Forest Regression R-squared:", random
Random Forest Regression R-squared: -0.02942208
> cat("Random Forest Regression MAE:", random_fores
Random Forest Regression MAE: 103.1851
```

Conclusion:

In this predictive analysis, we explored multiple models, including Linear Regression, Random Forest Regression, and K-means Clustering, to understand their performance on the given dataset. After evaluating the models based on various metrics, we found that the K-means Clustering model exhibited the best fit for this dataset. Here are 10 key points summarizing the process and conclusion:

1. We started by preprocessing the dataset, scaling the numeric variables, and converting non-numeric variables to factors.

2. The dataset was split into training and testing sets, with 80% of the data used for training and the remaining 20% for testing.

3. Linear Regression and Random Forest Regression models were trained using the training data to predict the target variable, "Calories.Burn."

4. K-means Clustering was performed on the scaled dataset, using the selected features, to identify distinct clusters in the data.

5. The K-means Clustering model assigned cluster labels to each data point, providing additional insights into the data structure.

6. Predictions were made using the trained Linear Regression and Random Forest Regression models on the test data.

7. Scatter plots were created to visualize the actual versus predicted values for both Linear Regression and Random Forest Regression.

8. A box plot was generated to compare the performance of the two regression models and visualize the distribution of the actual and predicted values.

9. The mean squared error (MSE) was calculated to assess the accuracy of the predictions for each model.

10. The K-means Clustering model exhibited the lowest MSE among the models, indicating its superior performance for this dataset.

In conclusion, after comparing the performance of various models, the K-means Clustering model was found to be the best fit for the given dataset. It provided valuable insights by assigning cluster labels to the data points and achieved a lower MSE compared to the Linear Regression and Random Forest Regression models. This suggests that the underlying patterns and relationships in the dataset were better captured by the K-means Clustering model, highlighting its effectiveness for analyzing and understanding the data.

```
##installing packages required
install.packages("tidyverse")
install.packages("cluster")
install.packages("factoextra")
install.packages("randomForest")

##libraries
library(stats) ##clusters
library(gridExtra) ##displayplots
library(dplyr)
library(RColorBrewer)
library(randomForest)
library(tidyr)
library(tidytext)
library(stringr)
library(crayon)
library(car) # Create a scatter plot with clusters and ellipses
library(DT)
library(ggthemes)
library(lubridate)
library(tidyselect)
library(scales)
library(ggplot2)
library(car)
library(tools)
library(tidyverse) # data manipulation
library(cluster)    # clustering algorithms
```

```
library(factoextra) # clustering algorithms & visualization


##csv file

exercise_data <- read.csv("C:/Users/DELL/Desktop/university/introduction to data science fundamentals/DSF LAB/exercise_dataset.csv")

exercise_data


##exploring datasets

ncol(exercise_data)

nrow(exercise_data)

names(exercise_data)

dim(ex_data)

str(ex_data)

head(ex_data, 5)

tail(ex_data, 4)

summary(ex_data)



##tidy data operations

is.na(exercise_data)

sum(is.na(exercise_data))


##removing & displaying missing values

ex_data <- na.omit(exercise_data)

nrow(ex_data)


##data wrangling
```

```
ex_data %>%
  select(Age, Gender, Heart.Rate) %>% head(5)

ex_data %>%
  mutate(Mean_BMI = mean(BMI)) %>%
  select(Age, BMI, Mean_BMI) %>%
  head(5)

ex_data %>%
  filter(Age == "19")%>%
  select(Age, Gender, BMI)%>%
  head(5)

ex_data %>%
  rename(Weight = "Actual.Weight")%>%
  select(Gender, Age, Weight)%>%
  head(5)

ex_data %>%
  arrange(Calories.Burn)%>%
  head(5)

ex_data %>%
  group_by(Gender)%>%
  summarise(Mean_weight = mean(Actual.Weight),
            calories_burned = mean(Calories.Burn),
```

```
            age = mean(Age))

##taking random rows
random_rows <- exercise_data[sample(nrow(exercise_data), 150), ]
random_rows

##finding correlation among variables
cor(ex_data$Age, ex_data$Heart.Rate)
cor(ex_data$Exercise.Intensity, ex_data$Calories.Burn)
cor(ex_data$Duration, ex_data$Calories.Burn)

##linear regression
model <- lm(Heart.Rate ~ Age , data = random_rows)
model

## visualize the relationship between Age and Heart.Rate
ggplot(random_rows, aes(x = Age , y = Heart.Rate))+
geom_point()

## create a scatter plot with a regression line
ggplot(random_rows, aes(x = Age , y = Heart.Rate))+
geom_point()+
  stat_smooth(method = lm)

##create a bar plot with a color gradient
## displays the relationship between age and heart rate
ggplot(random_rows, aes(x = Age, y = Heart.Rate, fill = Age)) +
```

```
  geom_bar(stat = "identity") +
  ggtitle("Heart Rate by by Age") +
  theme_bw() +
  scale_fill_gradient(low = "pink", high = "lightblue")



##create a histogram of the "Heart.Rate"
ggplot(random_rows, aes(x = Heart.Rate)) +
  geom_histogram(binwidth = 10, fill = "pink", color = "black") +
  ggtitle("Heart Rate Histogram") +
  xlab("Heart Rate") +
  ylab("Frequency")  +
  theme_bw()

##is.finite() function to identify any NaN or Inf values
problematic_values <- !is.finite(random_rows$Age) | !is.finite(random_rows$Heart.Rate)



set.seed(1)  # Set seed for reproducibility
k <- 3  # Number of clusters
kmeans_result <- kmeans(random_rows[, c("Age", "Heart.Rate")], centers = k)

# Add cluster labels to the data frame
random_rows$Cluster <- as.factor(kmeans_result$cluster)

# Calculate cluster centers
cluster_centers <- kmeans_result$centers
```

```
ggplot(random_rows, aes(x = Age, y = Heart.Rate, color = Cluster)) +
geom_point(alpha = 0.7) +
stat_ellipse(aes(fill = Cluster), geom = "polygon", alpha = 0.2, show.legend = FALSE) +
ggtitle("Cluster Diagram: Heart Rate vs. Age") +
xlab("Age") +
ylab("Heart Rate") +
theme_bw()

##linear-regression
model <- lm(Calories.Burn ~ Duration , data = random_rows)
model


# visualize the relationship between duration & calories.burn
ggplot(random_rows, aes(x = Duration , y = Calories.Burn))+
geom_point()

## create a scatter plot with a regression line
ggplot(random_rows, aes(x = Duration , y = Calories.Burn))+
geom_point()+
  stat_smooth(method = lm)

##create a bar plot visualizing the relationship between the "Duration" and "Calories.Burn
ggplot(random_rows, aes(x = Duration , y = Calories.Burn, fill = Duration)) +
geom_bar(stat = "identity") +
```

```
  ggtitle("Calories Burned by Duration") +
  theme_bw() +
  scale_fill_gradient(low = "turquoise", high = "orange")

##distribution of the "Calories.Burn" variable
ggplot(random_rows, aes(x = Calories.Burn)) +
  geom_histogram(binwidth = 10, fill = "orange", color = "black") +
  ggtitle("Calories Burned Histogram") +
  xlab("Calories.Burn") +
  ylab("Frequency") +
  theme_bw()

problematic_values <- !is.finite(random_rows$Duration) |
!is.finite(random_rows$Calories.Burn)

# clustering starts from here
set.seed(1)  # Set seed for reproducibility
k <- 3  # Number of clusters
kmeans_result <- kmeans(random_rows[, c("Duration", "Calories.Burn")], centers = k)

# Add cluster labels to the data frame
random_rows$Cluster <- as.factor(kmeans_result$cluster)

# Calculate cluster centers
cluster_centers <- kmeans_result$centers

ggplot(random_rows, aes(x = Duration, y = Calories.Burn, color = Cluster)) +
```

```
  geom_point(alpha = 0.7) +
  stat_ellipse(aes(fill = Cluster), geom = "polygon", alpha = 0.2, show.legend = FALSE) +
  ggtitle("Cluster Diagram: Duration vs. Calories Burned") +
  xlab("Duration") +
  ylab("Calories Burned") +
  theme_bw()

# Scaling the dataset to remove extreme values
head(exercise_data, 6)

non_numeric_cols <- c("Exercise", "Gender", "Weather.Conditions")
exercise_data[non_numeric_cols] <- lapply(exercise_data[non_numeric_cols], as.factor)

# Ensure numeric columns are correctly identified as numeric
exercise_data <- exercise_data %>%
  mutate_if(is.factor, as.numeric)

# Scale the numeric columns in the dataset
numeric_cols <- setdiff(colnames(exercise_data), non_numeric_cols)
scaled_data <- exercise_data
scaled_data <- na.omit(scaled_data)
scaled_data[numeric_cols] <- scale(scaled_data[numeric_cols])
# Append the target variable (Calories.Burn) to the scaled dataset
scaled_data$Calories.Burn <- exercise_data$Calories.Burn

# Select relevant features
```

```
features <- c("Exercise", "Age", "Gender", "Duration", "Heart.Rate", "BMI",
"Weather.Conditions", "Exercise.Intensity")

target <- "Calories.Burn"


#Split the dataset into training and testing sets

set.seed(123)

train_indices <- sample(1:nrow(scaled_data), 0.8 * nrow(scaled_data))

train_data <- scaled_data[train_indices, ]

test_data <- scaled_data[-train_indices, ]

dim(train_data)

dim(test_data)


# Remove rows with missing values

train_data <- train_data[complete.cases(train_data), ]


# Train the random forest regression model

random_forest_model <- randomForest(Calories.Burn ~ ., data = train_data[, c(features, target)])


#  Train the linear regression model

linear_model <- lm(Calories.Burn ~ ., data = train_data[, c(features, target)])



#  Perform k-means clustering

kmeans_model <- kmeans(scaled_data[, features], centers = 3)


#  Assign cluster labels to the data

scaled_data$Cluster <- as.factor(kmeans_model$cluster)
```

```
#  Make predictions on the test data

linear_predictions <- predict(linear_model, newdata = test_data)

head(linear_predictions,5)

random_forest_predictions <- predict(random_forest_model, newdata = test_data)

head(random_forest_predictions,6)



#  Visualize the results


# Scatter plot of actual vs predicted values for linear regression

linear_plot <- ggplot(data = test_data, aes(x = Calories.Burn, y = linear_predictions, color =
abs(linear_predictions - Calories.Burn))) +

  geom_point() +

  geom_abline(intercept = 0, slope = 1, linetype = "solid", color = "red") +

  ggtitle("Linear Regression: Actual vs Predicted") +

  xlab("Actual Calories.Burn") +

  ylab("Predicted Calories.Burn") +

  theme_bw() +

  scale_color_gradient(low = "blue", high = "red")

linear_plot


# Scatter plot of actual vs predicted values for random forest regression

colors <- brewer.pal(3, "Pastel1")

random_forest_plot <- ggplot(data = test_data, aes(x = Calories.Burn, y =
random_forest_predictions)) +

  geom_point(color = colors[1], alpha = 0.6) +
```

```
  geom_abline(intercept = 0, slope = 1, linetype = "solid", color = "red") +
  ggtitle("Random Forest Regression: Actual vs Predicted") +
  xlab("Actual Calories.Burn") +
  ylab("Predicted Calories.Burn") +
  theme_bw() +
  scale_color_manual(values = colors, guide = guide_colorbar(title = "Cluster", title.position = "right"))
random_forest_plot


# Box plot to compare linear regression and random forest regression
results <- data.frame(Actual = test_data$Calories.Burn, Linear_Predicted = linear_predictions, RF_Predicted = random_forest_predictions)
boxplot_plot <- ggplot(results, aes(x = factor(1), y = Actual)) +
  geom_boxplot(width = 0.5, fill = "lightblue", color = "black") +
  geom_jitter(aes(x = factor(2), y = Linear_Predicted), color = "red", size = 2) +
  geom_jitter(aes(x = factor(3), y = RF_Predicted), color = "blue", size = 2) +
  stat_summary(fun = mean, geom = "point", shape = 18, size = 4, color = "green") +
  geom_smooth(aes(x = factor(1), y = Actual), method = "lm", se = FALSE, color = "blue", linetype = "dashed") +
  ggtitle("Linear Regression vs Random Forest Regression") +
  xlab("") +
  ylab("Calories Burned") +
  theme_bw()
boxplot_plot


# Bar plot to compare actual and predicted values
barplot_plot <- ggplot(results, aes(x = factor(1), y = Actual)) +
  geom_bar(stat = "identity", fill = "lightblue", width = 0.5) +
```

```
  geom_point(aes(x = factor(2), y = Linear_Predicted), color = "red", size = 3) +
  geom_point(aes(x = factor(3), y = RF_Predicted), color = "blue", size = 3) +
  geom_smooth(aes(x = factor(1), y = Actual), method = "lm", se = FALSE, color = "blue",
linetype = "dashed") +
  ggtitle("Random Forest Regression: Actual vs Predicted") +
  xlab("Actual vs Predicted") +
  ylab("Calories Burned") +
  theme_bw()
barplot_plot



# Arrange and display the plots
plots <- grid.arrange(linear_plot, random_forest_plot, nrow = 2, ncol = 2)
plots2 <- grid.arrange(boxplot_plot, barplot_plot, nrow = 2, ncol = 1)
final_plot <- grid.arrange(plots, plots2, heights = c(0.6, 0.4))
print(final_plot)

# Create a data frame with the predicted values

predictions <- data.frame(Model = c(rep("Linear Regression", length(linear_predictions)),
                          rep("Random Forest Regression", length(random_forest_predictions)),
                          rep("K-means Clustering", nrow(scaled_data))))
Predicted = c(linear_predictions, random_forest_predictions, scaled_data$Calories.Burn)

# Create the boxplot with regression line
ggplot(predictions, aes(x = Model, y = Predicted)) +
geom_boxplot(fill = "lightblue", color = "black") +
```

```
stat_summary(fun = mean, geom = "point", shape = 18, size = 4, color = "blue") +
stat_summary(fun = function(x) lm(y ~ x)$coef[1], geom = "line", aes(group = 1), linetype =
"dashed", color = "red") +
ggtitle("Linear Regression, Random Forest Regression, and K-means Clustering") +
xlab("Model") +
ylab("Predicted Calories.Burn") +
theme_bw()

# Calculate the SSE for k-means clustering
kmeans_sse <- sum(kmeans_model$withinss)

# Calculate the MSE for k-means clustering
kmeans_mse <- kmeans_sse / nrow(scaled_data)

# Remove rows with missing values in test_data
test_data <- test_data[complete.cases(test_data), ]

# Generate predictions for linear regression and random forest regression models
linear_predictions <- predict(linear_model, newdata = test_data)
random_forest_predictions <- predict(random_forest_model, newdata = test_data)

# Calculate the MSE for linear regression, random forest regression, and k-means clustering
linear_mse <- mean((linear_predictions - test_data$Calories.Burn)^2)
random_forest_mse <- mean((random_forest_predictions - test_data$Calories.Burn)^2)

# Print the MSE for linear regression, random forest regression, and k-means clustering
cat("Linear Regression MSE:", linear_mse, "\n")
```

```
cat("Random Forest Regression MSE:", random_forest_mse, "\n")

cat("K-means Clustering MSE:", kmeans_mse, "\n")


# Calculate R-squared for linear regression

linear_r_squared <- 1 - (sum((test_data$Calories.Burn - linear_predictions)^2) / sum((test_data$Calories.Burn - mean(test_data$Calories.Burn))^2))


# Calculate MAE for linear regression

linear_mae <- mean(abs(test_data$Calories.Burn - linear_predictions))


# Calculate R-squared for random forest regression

random_forest_r_squared <- 1 - (sum((test_data$Calories.Burn - random_forest_predictions)^2) / sum((test_data$Calories.Burn - mean(test_data$Calories.Burn))^2))


# Calculate MAE for random forest regression

random_forest_mae <- mean(abs(test_data$Calories.Burn - random_forest_predictions))


# Print the accuracy metrics

cat("Linear Regression R-squared:", linear_r_squared, "\n")

cat("Linear Regression MAE:", linear_mae, "\n")

cat("Random Forest Regression R-squared:", random_forest_r_squared, "\n")

cat("Random Forest Regression MAE:", random_forest_mae, "\n")
```