Project Script

```r
# Description

# Problem Statement:

# In bike-sharing systems, the entire process from membership to rental and return has been
automated.
# Using these systems, users can easily rent a bike from one location and return it to
another.
# Hence, a bike rental company wants to understand and predict the number of bikes rented
daily based on the environment and seasons.

# Objective: The objective of this case is to predict bike rental counts
# based on environmental and seasonal settings with the help of a machine learning
algorithm.


# Steps to Perform:

# 1.    Exploratory data analysis
# •   Load dataset and libraries

# Load necessary libraries
library(ggplot2)
library(dplyr)

# Load the dataset

bike_data <- read.csv(file.choose())

# Display the first few rows of the dataset
head(bike_data)



# •   Perform data type conversion of the attributes

# Exploratory Data Analysis (EDA)
## Summary statistics
summary(bike_data)

## Visualize distributions of numeric variables
hist(bike_data$temp, main="Temperature Distribution", xlab="Normalized Temperature")
hist(bike_data$atemp, main="Feeling Temperature Distribution", xlab="Normalized Feeling
Temperature")
```

```r
hist(bike_data$hum, main="Humidity Distribution", xlab="Normalized Humidity")
hist(bike_data$windspeed, main="Wind Speed Distribution", xlab="Normalized Wind Speed")

## Visualize bike rentals over time
bike_data$dteday <- as.Date(bike_data$dteday) # Convert dteday to Date object
ggplot(bike_data, aes(x=dteday, y=cnt)) + geom_line() + labs(title="Daily Bike Rentals Over Time", x="Date", y="Bike Rental Count")

# Data Type Conversion
## Convert 'season', 'yr', 'mnth', 'holiday', 'weekday', 'workingday', 'weathersit' to factor as they are categorical variables
bike_data$season <- as.factor(bike_data$season)
bike_data$yr <- as.factor(bike_data$yr)
bike_data$mnth <- as.factor(bike_data$mnth)
bike_data$holiday <- as.factor(bike_data$holiday)
bike_data$weekday <- as.factor(bike_data$weekday)
bike_data$workingday <- as.factor(bike_data$workingday)
bike_data$weathersit <- as.factor(bike_data$weathersit)

# Check the structure of the data after conversions
str(bike_data)



# •    Carry out the missing value analysis

## Missing Value Analysis
# Check for missing values in the dataset
missing_values <- sum(is.na(bike_data))
cat("Total missing values in the dataset:", missing_values, "\n")

# If there are missing values, to see where they are.
if(missing_values > 0) {
  missing_values_by_column <- sapply(bike_data, function(x) sum(is.na(x)))
  cat("Missing values by column:\n")
  print(missing_values_by_column)

  # To impute missing values or drop them
  bike_data_clean <- na.omit(bike_data)
  cat("Rows after removing missing values:", nrow(bike_data_clean), "\n")
}

# Visualizing the distribution of total bike rentals
library(ggplot2)
ggplot(bike_data, aes(x = cnt)) +
  geom_histogram(binwidth = 100, fill = "blue", color = "black") +
```

```r
  theme_minimal() +
  labs(title = "Distribution of Total Bike Rentals", x = "Total Rentals", y = "Frequency")



# 2. Attributes distributions and trends
# •	Plot monthly distribution of the total number of bikes rented

# Plot monthly distribution of the total number of bikes rented
monthly_rental <- bike_data %>%
  group_by(mnth) %>%
  summarise(total_rental = sum(cnt))

# Plot the monthly distribution
ggplot(monthly_rental, aes(x = factor(mnth), y = total_rental)) +
  geom_bar(stat = "identity", fill = "skyblue") +
  labs(title = "Monthly Distribution of Total Bike Rentals",
       x = "Month",
       y = "Total Rental Count") +
  theme_minimal()



# •	Plot yearly distribution of the total number of bikes rented

# Convert the 'dteday' column to Date type
bike_data$dteday <- as.Date(bike_data$dteday, format="%Y-%m-%d")

# Extract year from the 'dteday' column
bike_data$year <- format(bike_data$dteday, "%Y")

# Summarize total bikes rented by year
yearly_data <- bike_data %>%
  group_by(year) %>%
  summarise(total_rentals = sum(cnt))

# Plotting the yearly distribution of total bike rentals
ggplot(yearly_data, aes(x=year, y=total_rentals)) +
  geom_bar(stat="identity", fill="steelblue") +
  theme_minimal() +
  labs(title="Yearly Distribution of Total Bike Rentals",
       x="Year",
       y="Total Rentals")
```

```
# •	Plot boxplot for outliers analysis

# Function to plot boxplots for outlier analysis
plot_outliers <- function(data, feature_name) {
  # Using ggplot2 to create a boxplot to visualize outliers
  ggplot(data, aes_string(x = feature_name)) +
    geom_boxplot(fill = "skyblue", color = "darkblue") +
    labs(title = paste("Boxplot for Outliers Analysis -", feature_name),
         x = feature_name,
         y = "Value") +
    theme_minimal()
}

# Attributes to be analyzed for outliers
attributes <- c("temp", "atemp", "hum", "windspeed", "casual", "registered", "cnt")

# Loop through attributes and plot boxplots for each
for(attribute in attributes) {
  print(plot_outliers(bike_data, attribute))
}




# 3. Split the dataset into train and test dataset

# Load necessary libraries
install.packages("readr")
library(readr) # For reading CSV files

install.packages("caret")
library(caret) # For data splitting and modeling


# Split the dataset into training and test sets
# First, set a seed for reproducibility
set.seed(123)

# Splitting the data - 70% for training, 30% for testing
splitIndex <- createDataPartition(bike_data$cnt, p = .7, list = FALSE, times = 1)
trainData <- bike_data[ splitIndex,]
testData  <- bike_data[-splitIndex,]

# Just to confirm the size of the split
cat("Training set rows:", nrow(trainData), "\n")
cat("Test set rows:", nrow(testData), "\n")
```

```
# Fitting the model on the training data
model <- lm(cnt ~ temp, data = trainData)

# Summary of the model to check its performance metrics
summary(model)

# Predicting on the test data
predictions <- predict(model, newdata = testData)

# Comparing actual vs predicted values (for evaluation, consider using metrics like RMSE,
MAE)
comparison <- data.frame(Actual = testData$cnt, Predicted = predictions)
head(comparison)




# 4. Create a model using the random forest algorithm

# Load necessary libraries

install.packages("randomForest")
library(randomForest)

# Create a Random Forest model
# We will predict the 'cnt' variable using all other variables except 'instant', 'dteday', 'casual',
and 'registered'
model <- randomForest(cnt ~ . -instant -dteday -casual -registered, data=trainData,
ntree=500, importance=TRUE)

# Step 4: Evaluate the model
# Predict on test data
predictions <- predict(model, testData)

# Calculate RMSE (Root Mean Squared Error)
rmse <- sqrt(mean((predictions - testData$cnt)^2))
cat("RMSE on test data: ", rmse, "\n")

# Calculate R-squared value
r2 <- cor(predictions, testData$cnt)^2
cat("R-squared: ", r2, "\n")

# Print the importance of variables
importance(model)
```

```r
# 5. Predict the performance of the model on the test dataset

# Split the data into training and testing sets
set.seed(123)
trainIndex <- createDataPartition(bike_data$cnt, p = 0.7, list = FALSE)
training <- bike_data[trainIndex,]
testing <- bike_data[-trainIndex,]

# Train a machine learning model
model <- train(cnt ~ season + yr + mnth + holiday + weekday + workingday + weathersit +
temp + atemp + hum + windspeed, data = training, method = "lm")

# Make predictions on the testing set
predictions <- predict(model, newdata = testing)

# Evaluate the model performance
performance <- RMSE(predictions, testing$cnt)
print(paste("Root Mean Squared Error:", performance))
```