Project 2: Bike Rental Prediction

Perform the following tasks on the dataset provided using R:

1. Exploratory data analysis:

- Load the dataset and the relevant libraries
- Perform data type conversion of the attributes
- Carry out the missing value analysis

Code:

```
# 1. Exploratory data analysis :

# Load dataset and relevant libraries =>
# Load relevant libraries
library(readxl) # for data reading
library(dplyr) # for data manipulation
library(ggplot2)
library(randomForest)
library(caret) # Add caret library for train/test splitting
library(reshape2) # Add reshape2 for melt function

# Load dataset
bike_data <- readxl::read_excel("C:/Users/garvit/Desktop/Intern/data.xlsx")
print(bike_data)

# Perform data type conversion on the attributes
bike_data$season <- as.factor(bike_data$season) # Used for classification
purposes
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)

# Carry out Missing value analysis
missing_values <- colSums(is.na(bike_data))
print(missing_values)
```

Output :

```
> print(bike_data)
# A tibble: 731 × 16
   instant dteday              season    yr  mnth holiday weekday workingday weathersit
     <dbl> <dttm>                <dbl> <dbl> <dbl>   <dbl>   <dbl>      <dbl>      <dbl>
 1       1 2011-01-01 00:00:00      1     0     1       0       6          0          2
 2       2 2011-01-02 00:00:00      1     0     1       0       0          0          2
 3       3 2011-01-03 00:00:00      1     0     1       0       1          1          1
 4       4 2011-01-04 00:00:00      1     0     1       0       2          1          1
 5       5 2011-01-05 00:00:00      1     0     1       0       3          1          1
 6       6 2011-01-06 00:00:00      1     0     1       0       4          1          1
 7       7 2011-01-07 00:00:00      1     0     1       0       5          1          2
 8       8 2011-01-08 00:00:00      1     0     1       0       6          0          2
 9       9 2011-01-09 00:00:00      1     0     1       0       0          0          1
10      10 2011-01-10 00:00:00      1     0     1       0       1          1          1
# i 721 more rows
# i 7 more variables: temp <dbl>, atemp <dbl>, hum <dbl>, windspeed <dbl>,
#   casual <dbl>, registered <dbl>, cnt <dbl>
# i Use `print(n = ...)` to see more rows
```

```
> print(missing_values)
   instant     dteday     season         yr       mnth    holiday    weekday
         0          0          0          0          0          0          0
workingday weathersit       temp      atemp        hum  windspeed     casual
         0          0          0          0          0          0          0
registered        cnt
         0          0
>
```

2. Attributes distribution and trends :

- Plot monthly distribution of the total number of bikes rented
- Plot yearly distribution of the total number of bikes rented
- Plot boxplot for outliers' analysis

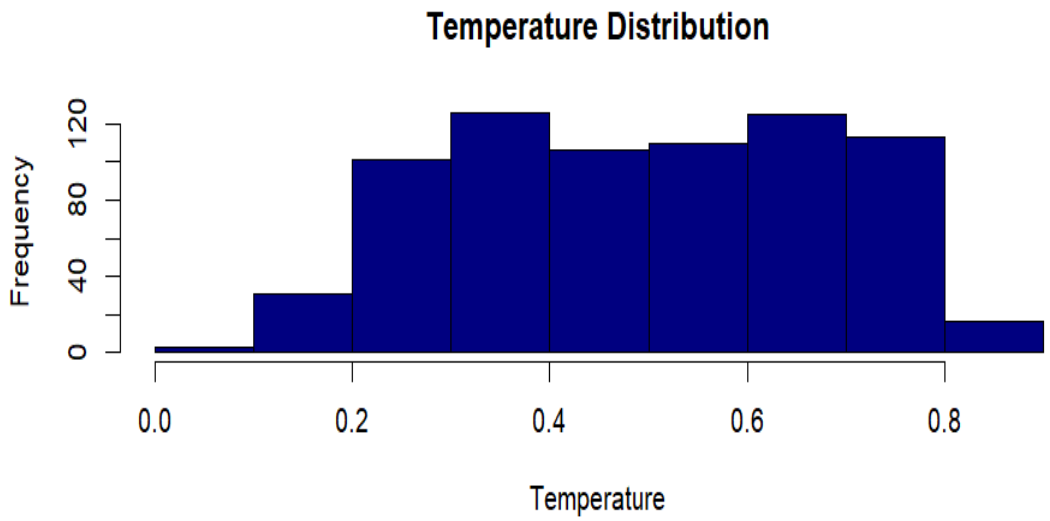
Code :

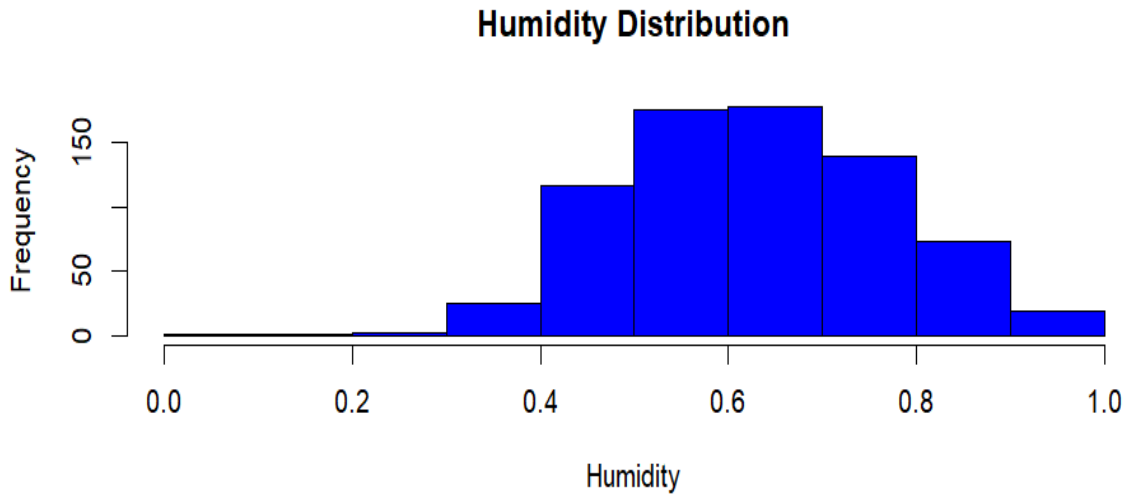
```
# 2. Attributes distribution and trends :

# Histograms :

# Histogram for Temperature Distribution
hist(bike_data$temp, col = "navyblue", xlab = "Temperature", ylab =
"Frequency", main = "Temperature Distribution")

# Histogram for Humidity Distribution
```

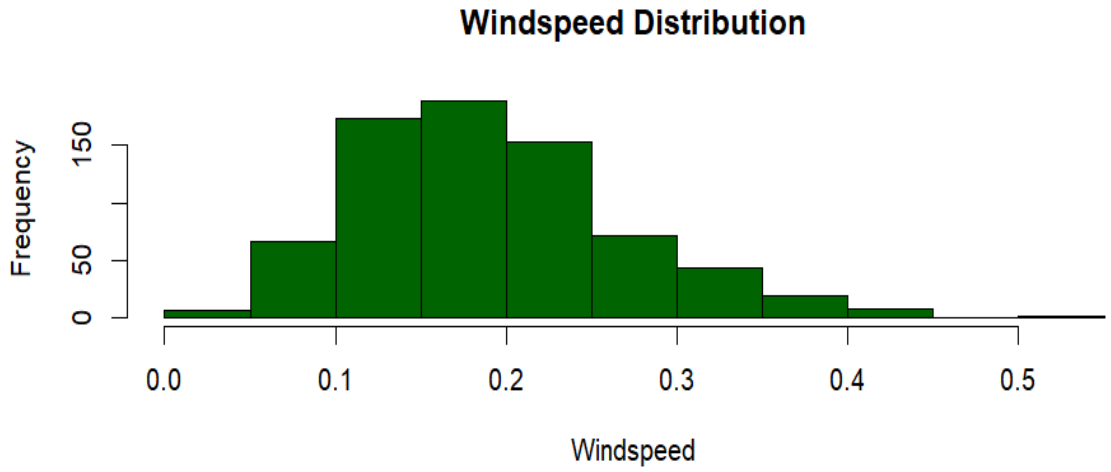
```
hist(bike_data$hum, col = "blue", xlab = "Humidity", ylab = "Frequency", main
= "Humidity Distribution")

# Histogram for Windspeed Distribution
hist(bike_data$windspeed, col = "dark green", xlab = "Windspeed", ylab =
"Frequency", main = "Windspeed Distribution")




# Bar graphs :

# Plot monthly distribution of total number of bikes rented
ggplot(bike_data, aes(x = mnth, y = cnt)) +
  geom_bar(stat = "summary", fun = "sum", fill = "skyblue", color = "black") +
  labs(title = "Monthly Distribution of Total Bikes Rented",
       x = "Month",
       y = "Total Bikes Rented")

# Plot yearly distribution of total number of bikes rented
ggplot(bike_data, aes(x = yr, y = cnt)) +
  geom_bar(stat = "summary", fun = "sum", fill = "lightgreen", color =
"black") +
  labs(title = "Yearly Distribution of Total Bikes Rented",
       x = "Year",
       y = "Total Bikes Rented")




# Scatter plots :

# Count with respect to temperature and humidity together
ggplot(bike_data, aes(temp, cnt)) +
  geom_point(aes(color = hum), alpha = 0.5) +
  labs(title = "Bikes count vs temperature and humidity", x = "Normalized
temperature", y = "Count") +
  scale_color_gradientn(colors = c('blue', 'light blue', 'dark blue', 'light
green', 'yellow', 'dark orange', 'black')) +
  theme_bw()

# Count with respect to windspeed and weather together
ggplot(bike_data, aes(x = windspeed, y = cnt)) +
  geom_point(aes(color = factor(weathersit)), alpha = 0.5) +  # Convert to
factor to ensure it's treated as discrete
  labs(title = "Bikes count vs windspeed and weather", x = "Windspeed", y =
"Count") +
```

```
  scale_color_manual(values = c('blue', 'light blue', 'dark blue', 'light
green', 'yellow', 'dark orange', 'black')) +  # Use scale_color_manual for
discrete scale
  theme_bw()

# Count with respect to temperature and season together
ggplot(bike_data, aes(x = temp, y = cnt)) +
  geom_point(aes(color = season), alpha = 0.5) +
  labs(title = "Bikes count vs temperature and season", x = "Normalized
temperature", y = "Count") +
  scale_color_manual(values = c('blue', 'light blue', 'dark blue', 'light
green', 'yellow', 'dark orange', 'black')) +
  theme_bw()


# Plot boxplot for outliers' analysis =>
# We will consider only numeric variables for boxplot
numeric_vars <- bike_data[, sapply(bike_data, is.numeric)]
# Remove 'instant' column as it is just an index
numeric_vars <- numeric_vars[, -1]

# Melt the data, specifying 'instant' as the id variable
melted_data <- melt(numeric_vars, id.vars = NULL)

ggplot(melted_data, aes(x = variable, y = value)) +
  geom_boxplot(fill = "orange", color = "black") +
  labs(title = "Boxplot for Outliers' Analysis",
       x = "Variable",
       y = "Value")
```

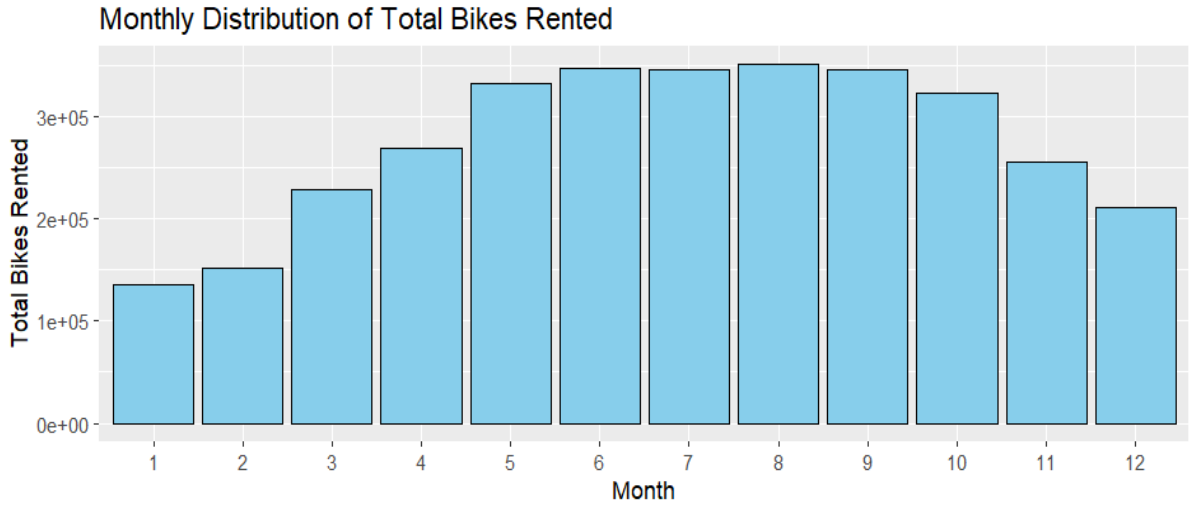
Output : a) Histograms :



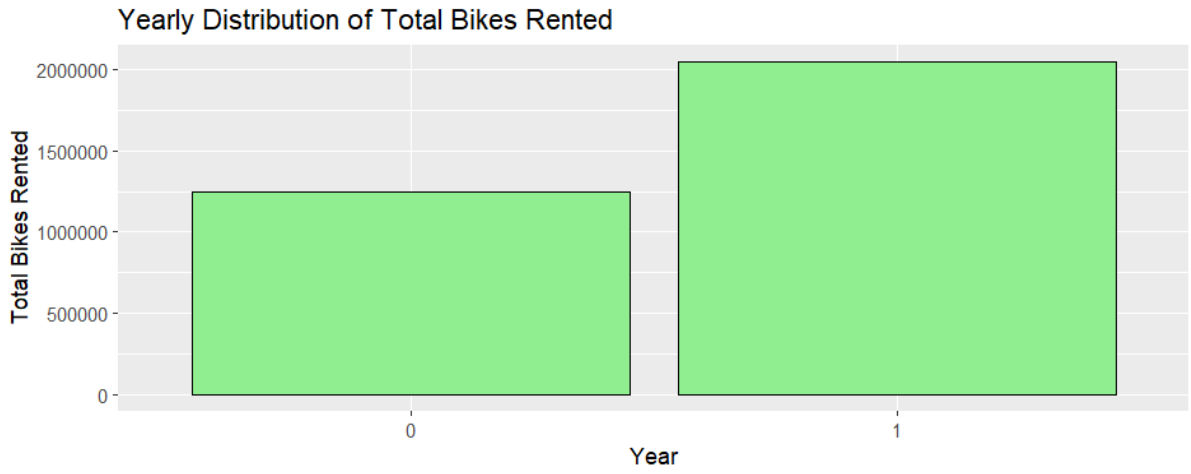




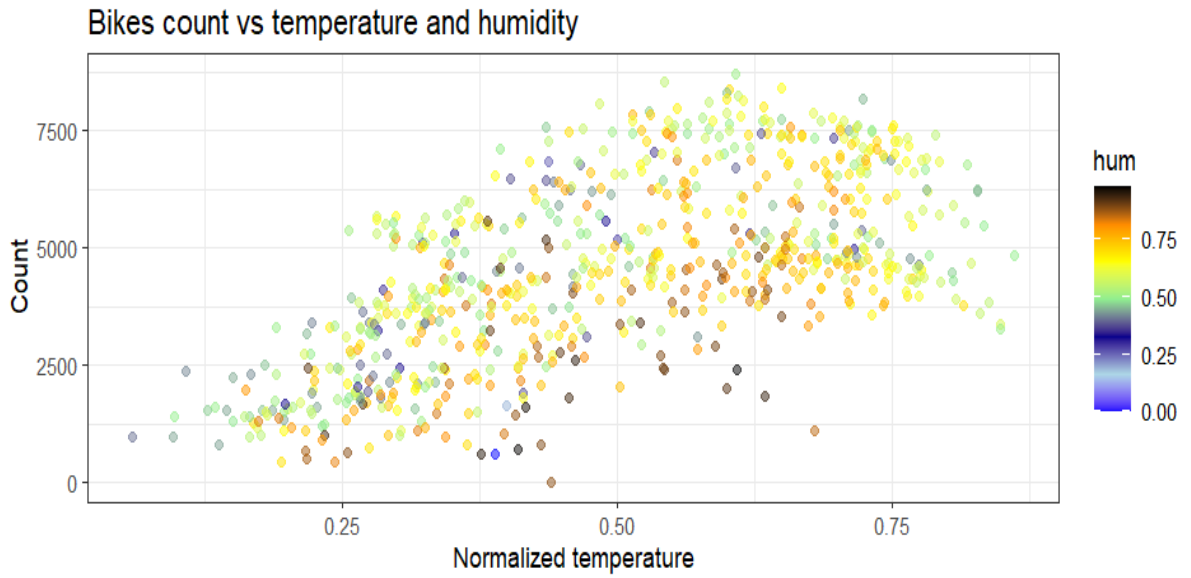
b) Bar Graphs : i) Conclusion : Most bikes are rented in 8th month (Aug.)

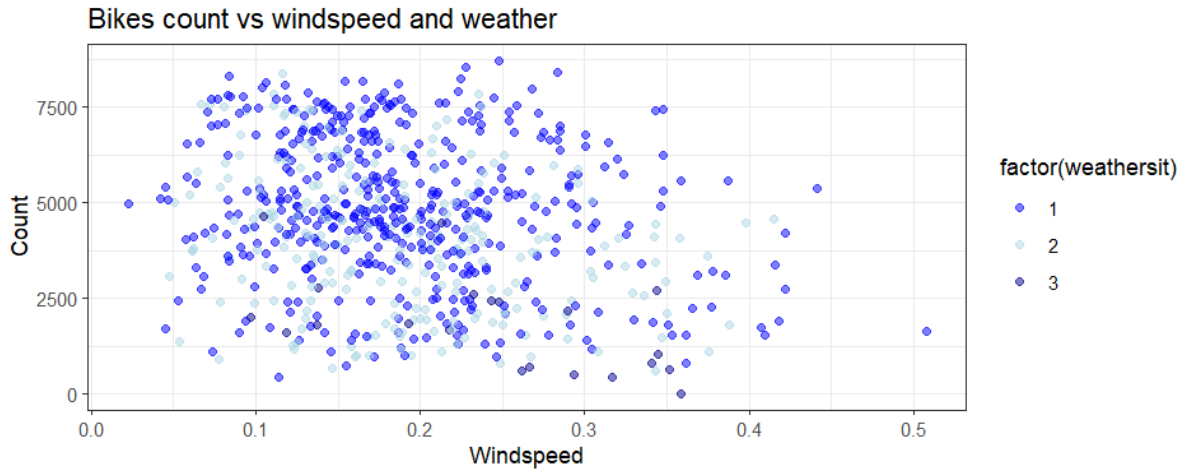


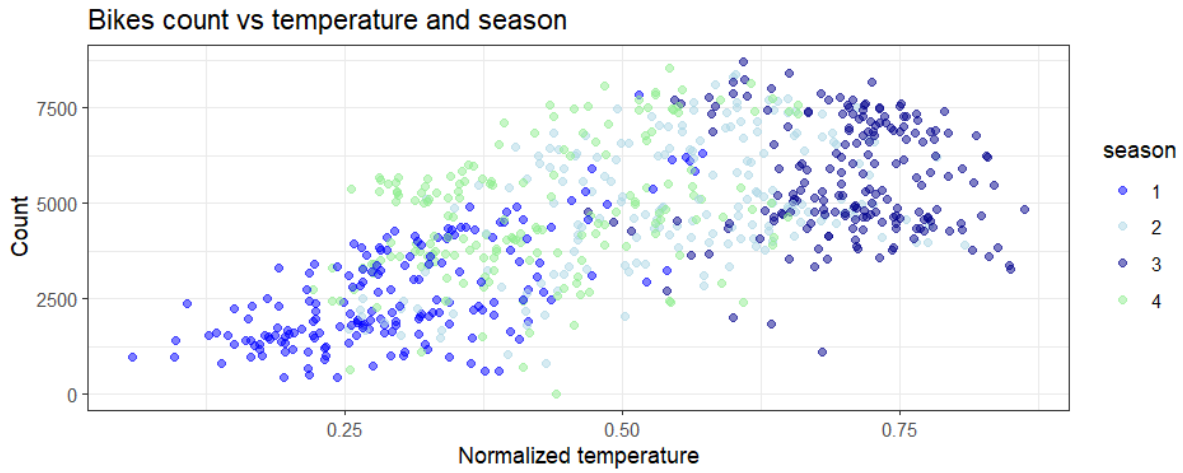
ii) Conclusion : Most bikes are rented in year 2012 (year : 1)



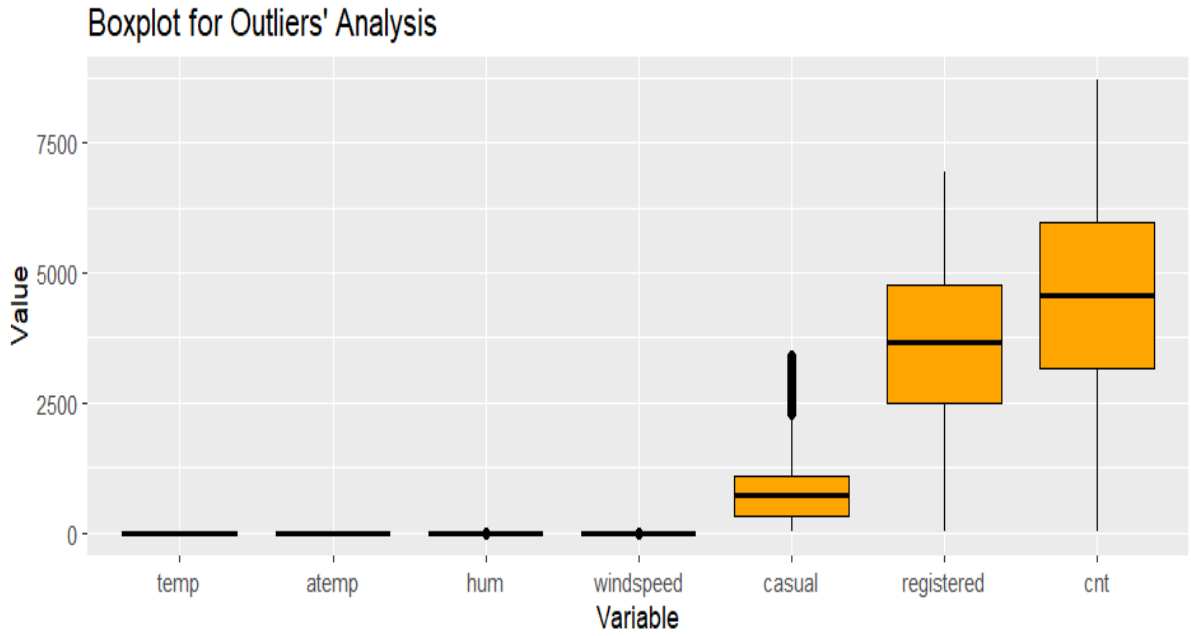
c) Scatter Plots :







d) Boxplots :



3. Split the dataset into training and testing sets :

Code :

```
train_index <- createDataPartition(y = bike_data$cnt, p = 0.8, list = FALSE)
train_data <- bike_data[train_index, ]
test_data <- bike_data[-train_index, ]
print(train_data)
print(test_data)
```

Output :

```
> print(train_data)
# A tibble: 587 × 16
   instant dteday              season yr    mnth  holiday weekday workingday weathersit
     <dbl> <dttm>              <fct>  <fct> <fct> <fct>   <fct>   <fct>      <fct>
 1       1 2011-01-01 00:00:00 1      0     1     0       6       0          2
 2       4 2011-01-04 00:00:00 1      0     1     0       2       1          1
 3       5 2011-01-05 00:00:00 1      0     1     0       3       1          1
 4       6 2011-01-06 00:00:00 1      0     1     0       4       1          1
 5       7 2011-01-07 00:00:00 1      0     1     0       5       1          2
 6       8 2011-01-08 00:00:00 1      0     1     0       6       0          2
 7       9 2011-01-09 00:00:00 1      0     1     0       0       0          1
 8      11 2011-01-11 00:00:00 1      0     1     0       2       1          2
 9      12 2011-01-12 00:00:00 1      0     1     0       3       1          1
10      13 2011-01-13 00:00:00 1      0     1     0       4       1          1
# i 577 more rows
# i 7 more variables: temp <dbl>, atemp <dbl>, hum <dbl>, windspeed <dbl>,
#   casual <dbl>, registered <dbl>, cnt <dbl>
# i Use `print(n = ...)` to see more rows
> print(test_data)
# A tibble: 144 × 16
   instant dteday              season yr    mnth  holiday weekday workingday weathersit
     <dbl> <dttm>              <fct>  <fct> <fct> <fct>   <fct>   <fct>      <fct>
 1       2 2011-01-02 00:00:00 1      0     1     0       0       0          2
 2       3 2011-01-03 00:00:00 1      0     1     0       1       1          1
 3      10 2011-01-10 00:00:00 1      0     1     0       1       1          1
 4      15 2011-01-15 00:00:00 1      0     1     0       6       0          2
 5      18 2011-01-18 00:00:00 1      0     1     0       2       1          2
 6      28 2011-01-28 00:00:00 1      0     1     0       5       1          2
 7      29 2011-01-29 00:00:00 1      0     1     0       6       0          1
 8      33 2011-02-02 00:00:00 1      0     2     0       3       1          2
 9      45 2011-02-14 00:00:00 1      0     2     0       1       1          1
10      49 2011-02-18 00:00:00 1      0     2     0       5       1          1
# i 134 more rows
# i 7 more variables: temp <dbl>, atemp <dbl>, hum <dbl>, windspeed <dbl>,
#   casual <dbl>, registered <dbl>, cnt <dbl>
# i Use `print(n = ...)` to see more rows
```

4. Create a model using Random Forest Algorithm :

Code :

```
# Specify the formula for the model
# Here, we predict 'cnt' based on other variables
formula <- cnt ~ season + yr + mnth + holiday + weekday +
  workingday + weathersit + temp + atemp + hum + windspeed +
  casual + registered
```

```r
# Random Forest

# Train the Random Forest model
rf_model <- randomForest(formula, data = train_data)

# Print the summary of the model
print(rf_model)
```

Output :

```
> # Print the summary of the model
> print(rf_model)

Call:
 randomForest(formula = formula, data = train_data)
               Type of random forest: regression
                     Number of trees: 500
No. of variables tried at each split: 4

          Mean of squared residuals: 68273.27
                    % Var explained: 98.15
```

5.Predict the performance of the model on the test dataset

Code :

```r
# Predict on the test dataset
predictions <- predict(rf_model, newdata = test_data)
print(predictions)

# Evaluate the model (e.g., using RMSE)
rmse <- sqrt(mean((predictions - test_data$cnt)^2))
print(paste("Root Mean Squared Error (RMSE):", rmse))
```

Output :

```
> print(predictions)
       1        2        3        4        5        6        7        8
1286.4785 1491.8877 1398.2837 1292.0774  849.8678 1264.9499 1148.7517 1616.2935
       9       10       11       12       13       14       15       16
2084.7246 3271.5790 1672.1720 2131.3340 1275.3038 1907.2148 2081.6853 2185.0095
      17       18       19       20       21       22       23       24
3529.3863 1902.1931 1791.0088 2252.3085 1434.1489 4130.8911 4057.6685 4567.6863
      25       26       27       28       29       30       31       32
2777.9658 4301.9685 4395.5447 4310.7515 4662.9693 4714.1086 4047.0600 4978.5086
      33       34       35       36       37       38       39       40
4927.6217 4614.8753 4472.8471 5558.3172 3872.4859 4735.9741 4015.8986 4663.3547
      41       42       43       44       45       46       47       48
4380.2989 5155.9626 5548.5135 2383.3081 4750.4083 4596.6852 4796.5953 4662.4852
      49       50       51       52       53       54       55       56
3822.7738 4019.7044 4696.0900 5224.4836 3848.3306 2533.8497 3918.7228 3698.4690
      57       58       59       60       61       62       63       64
4038.7545 4099.4502 3792.7469 2964.0822 2967.1190 3256.3206 3290.1357 3642.9796
      65       66       67       68       69       70       71       72
2713.4542 1361.2542 1521.1717 2223.3251 2895.0117 4134.3820 4418.4172 2351.3642
      73       74       75       76       77       78       79       80
2216.4349 3220.5409 3969.4608 3950.6284 3768.7001 4342.6516 2517.4735 5019.5744
      81       82       83       84       85       86       87       88
4321.4654 3239.2524 4237.3013 6258.5540 4520.6473 5672.7768 5873.7332 5509.0130
      89       90       91       92       93       94       95       96
5969.4023 6688.1504 6253.2768 4314.5167 6576.9253 4875.7059 6296.9574 7494.4691
      97       98       99      100      101      102      103      104
6925.7564 6169.9648 6111.0927 6776.6826 7270.7636 7007.7521 7538.9158 7291.3320
     105      106      107      108      109      110      111      112
6547.6620 6436.0951 6788.2267 5617.7931 6403.1954 4978.1180 7233.1681 7263.6974
     113      114      115      116      117      118      119      120
6170.9211 7384.4974 6792.7802 7025.5330 7128.9501 7088.2395 6882.4973 7420.0475
     121      122      123      124      125      126      127      128
6478.7906 5770.9169 7494.7501 7642.8232 7640.2024 7410.7932 7487.3720 7735.1321
     129      130      131      132      133      134      135      136
3949.5762 7287.0846 7383.6333 5413.3345 6131.7085 5327.3792 5315.0901 5489.9425
     137      138      139      140      141      142      143      144
3716.5110 5206.8163 5404.4254 5107.9898 4669.5634 1923.5443 2921.0731 2461.3042
>
```

```
> # Evaluate the model (e.g., using RMSE)
> rmse <- sqrt(mean((predictions - test_data$cnt)^2))
> print(paste("Root Mean Squared Error (RMSE):", rmse))
[1] "Root Mean Squared Error (RMSE): 258.108900555969"
>
```