# 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.

**Data Set:** bike\_rental\_dataset.csv

**Data Description**

|  |  |
| --- | --- |
| **Variable** | **Description** |
| instant | Record index |
| dteday | Date |
| season | Season (1: springer, 2: summer, 3: fall, 4: winter) |
| yr | Year (0: 2011, 1:2012) |
| mnth | Month (1 to 12) |
| holiday | Weather day is a holiday or not |
| weekday | Day of the week |
| workingday | Working day (1: neither weekend nor holiday, 0: other days) |
| weathersit | 1: Clear, few clouds, partly cloudy, partly cloudy  2: Mist + cloudy, mist + broken clouds, mist + few clouds, mist  3: Light snow, light rain + thunderstorm + scattered clouds, light rain + scattered clouds  4: Heavy rain + ice pallets |
| temp | Normalized temperature in Celsius; The values are divided into 41 (max) |
| atemp | Normalized feeling temperature in Celsius; The values are divided into 50 (max) |
| hum | Normalized humidity; The values are divided into 100 (max) |
| windspeed | Normalized wind speed; The values are divided into 67 (max) |
| casual | Count of casual users |
| registered | Count of registered users |
| cnt | Count of total rental bikes including both casual and registered |

**Steps to Perform:**

1.    Exploratory data analysis  
•    Load dataset and libraries  
•    Perform data type conversion of the attributes  
•    Carry out the missing value analysis  
2. Attributes distributions 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  
3. Split the dataset into train and test dataset  
4. Create a model using the random forest algorithm  
5. Predict the performance of the model on the test dataset

# Task

# Load necessary libraries

library(ggplot2)

library(readxl)

library(dplyr)

library(randomForest)

library(caret)

# Load Dataset

file\_path <- "C:/Users/shint/Desktop/Fortray/5. Data Analytics with R/Project 2\_Bike Rental Prediction/bike\_rental\_dataset.xlsx"

bike\_df <- read\_excel(file\_path)

# Data type conversion

bike\_df <- bike\_df %>%

mutate(

dteday = as.Date(dteday),

yr = as.factor(yr),

mnth = as.factor(mnth),

season = as.factor(season),

holiday = as.factor(holiday),

weekday = as.factor(weekday),

workingday = as.factor(workingday),

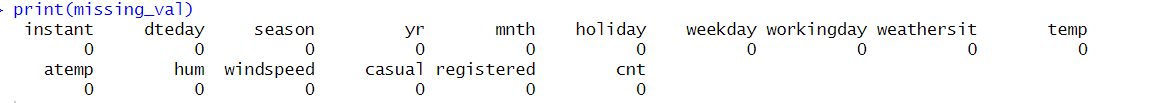
weathersit = as.factor(weathersit)

)

# Missing value analysis

missing\_val <- colSums(is.na(bike\_df))

print(missing\_val)



# Plot monthly distribution of total bike rentals

monthly\_plot <- bike\_df %>%

group\_by(mnth) %>%

summarise(total\_rentals = sum(cnt)) %>%

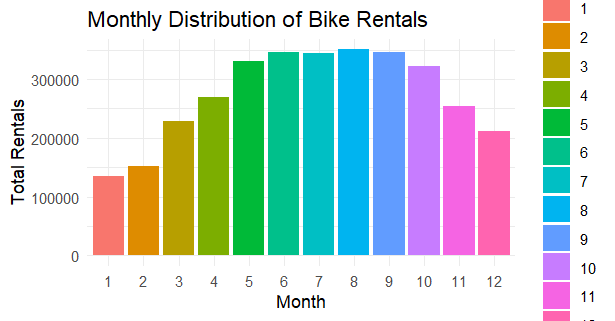
ggplot(aes(x = mnth, y = total\_rentals, fill = mnth)) +

geom\_bar(stat = "identity") +

labs(x = "Month", y = "Total Rentals", title = "Monthly Distribution of Bike Rentals") +

theme\_minimal()

print(monthly\_plot)



# Plot yearly distribution of total bike rentals

yearly\_plot <- bike\_df %>%

group\_by(yr) %>%

summarise(total\_rentals = sum(cnt)) %>%

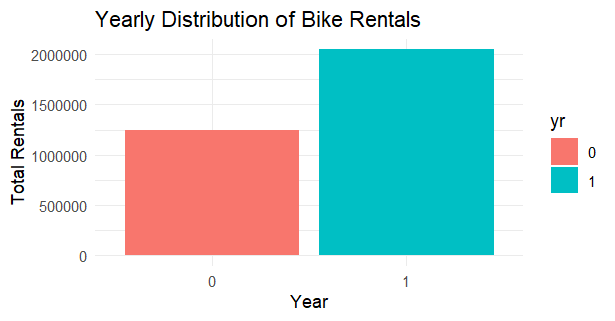
ggplot(aes(x = yr, y = total\_rentals, fill = yr)) +

geom\_bar(stat = "identity") +

labs(x = "Year", y = "Total Rentals", title = "Yearly Distribution of Bike Rentals") +

theme\_minimal()

print(yearly\_plot)



# Outlier Analysis

# Section: Boxplot for Bike Rental Count with Outliers

# Boxplot for bike rental count with outliers

boxplot(bike\_df$cnt, main = 'Bike Rental Count', sub = ifelse(length(boxplot.stats(bike\_df$cnt)$out) == 0, "No Outliers", paste("Outliers: ", boxplot.stats(bike\_df$cnt)$out)),

ylab = 'Count', col = "cyan", border = "blue")

# Add statistical values

text(1, boxplot.stats(bike\_df$cnt)$stats[1], paste("Min:", round(boxplot.stats(bike\_df$cnt)$stats[1], 2)), pos = 4, cex = 1)

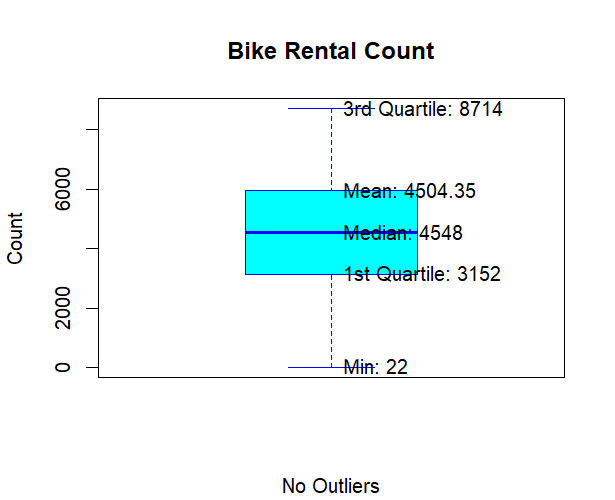
text(1, boxplot.stats(bike\_df$cnt)$stats[2], paste("1st Quartile:", round(boxplot.stats(bike\_df$cnt)$stats[2], 2)), pos = 4, cex = 1)

text(1, boxplot.stats(bike\_df$cnt)$stats[3], paste("Median:", round(boxplot.stats(bike\_df$cnt)$stats[3], 2)), pos = 4, cex = 1)

text(1, boxplot.stats(bike\_df$cnt)$stats[4], paste("Mean:", round(mean(bike\_df$cnt), 2)), pos = 4, cex = 1)

text(1, boxplot.stats(bike\_df$cnt)$stats[5], paste("3rd Quartile:", round(boxplot.stats(bike\_df$cnt)$stats[5], 2)), pos = 4, cex = 1)

text(1, boxplot.stats(bike\_df$cnt)$stats[6], paste("Max:", round(boxplot.stats(bike\_df$cnt)$stats[6], 2)), pos = 4, cex = 1)



# Section: Boxplots for Outliers in Temperature, Feel-like Temperature, Humidity, and Windspeed

# Set up the layout for multiple boxplots

par(mfrow = c(2, 2))

# Box plot for temperature outliers

boxplot(bike\_df$temp, main = "Temperature", sub = ifelse(length(boxplot.stats(bike\_df$temp)$out) == 0, "No Outliers", paste("Outliers: ", boxplot.stats(bike\_df$temp)$out)),

col = "#FF6347", border = "#8B0000", notch = TRUE, outline = FALSE)

# Box plot for feel-like temperature outliers

boxplot(bike\_df$atemp, main = "Feel-like Temperature", sub = ifelse(length(boxplot.stats(bike\_df$atemp)$out) == 0, "No Outliers", paste("Outliers: ", boxplot.stats(bike\_df$atemp)$out)),

col = "pink", border = "red", notch = TRUE, outline = FALSE)

# Box plot for humidity outliers

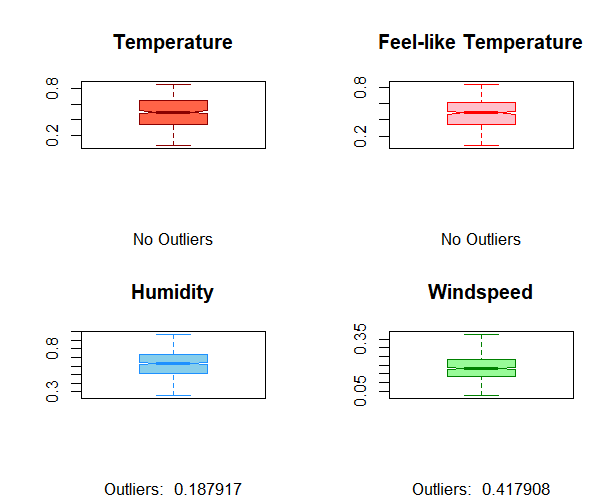
boxplot(bike\_df$hum, main = "Humidity", sub = ifelse(length(boxplot.stats(bike\_df$hum)$out) == 0, "No Outliers", paste("Outliers: ", boxplot.stats(bike\_df$hum)$out)),

col = "#87CEEB", border = "#1E90FF", notch = TRUE, outline = FALSE)

# Box plot for windspeed outliers

boxplot(bike\_df$windspeed, main = "Windspeed", sub = ifelse(length(boxplot.stats(bike\_df$windspeed)$out) == 0, "No Outliers", paste("Outliers: ", boxplot.stats(bike\_df$windspeed)$out)),

col = "#98FB98", border = "#008000", notch = TRUE, outline = FALSE)



# Outlier Replacement and Imputation

# Section: Replacing and Imputing Outliers in Humidity and Windspeed

# Function to replace outliers with NA

replace\_outliers <- function(x) {

q <- quantile(x, c(0.25, 0.75))

iqr <- q[2] - q[1]

lower\_bound <- q[1] - 1.5 \* iqr

upper\_bound <- q[2] + 1.5 \* iqr

x[x < lower\_bound | x > upper\_bound] <- NA

return(x)

}

# Apply the function to windspeed and humidity

bike\_df$windspeed <- replace\_outliers(bike\_df$windspeed)

bike\_df$hum <- replace\_outliers(bike\_df$hum)

# Impute missing values using mean imputation method

bike\_df$windspeed[is.na(bike\_df$windspeed)] <- mean(bike\_df$windspeed, na.rm = TRUE)

bike\_df$hum[is.na(bike\_df$hum)] <- mean(bike\_df$hum, na.rm = TRUE)

# Plot for numerical variables in combined dataset

# Select numerical columns for histogram and normal probability plot

numerical\_columns <- sapply(bike\_df[, 8:15], is.numeric)

# Histograms for numerical variables

for (column in names(bike\_df[, 8:15][, numerical\_columns])) {

hist(bike\_df[, column], main = paste("Histogram for", column),

xlab = column, col = "skyblue", border = "black")

}

# Normal probability plots for numerical variables

for (column in names(bike\_df[, 8:15][, numerical\_columns])) {

qqnorm(bike\_df[, column], main = paste("Normal Probability Plot for", column))

qqline(bike\_df[, column], col = 2)

# Add insight annotation

annotation <- "Some data points are deviating from normality in a good way."

text(quantile(bike\_df[, column], 1.0), quantile(bike\_df[, column], 0.1), annotation, adj = c(0, 1), cex = 0.8, col = "darkgreen")

}

# Section: Correlation Analysis of Numerical Variables in Combined Dataset

# Identify numeric columns for correlation analysis

numeric\_columns <- sapply(bike\_df[, 8:15], is.numeric)

# Create a correlation plot

corrgram(bike\_df[, 8:15][, numeric\_columns], order = FALSE, upper.panel = panel.pie, text.panel = panel.txt, main = 'Correlation Plot')

# Add insight on positive and negative correlations

cat("Positive Correlations: temp, atemp, and yr have positive correlations with the target variable.\n")

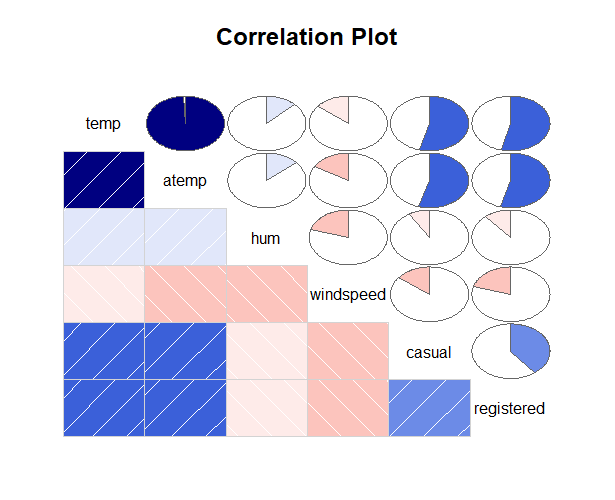
cat("Negative Correlations: weathersit, hum, and windspeed have negative correlations with the target variable.\n")

# Identify variables that may not be needed for further analysis based on correlation

cat("\nVariables with weak correlation (abs(correlation) <= 0.1) with the target variable:\n")

weak\_corr\_vars <- names(bike\_df[, 8:15][, numeric\_columns])[sapply(bike\_df[, 8:15][, numeric\_columns], function(x) abs(cor(x, bike\_df$cnt)) <= 0.1)]

print(weak\_corr\_vars)



# Split the dataset into training and testing sets

set.seed(123) # for reproducibility

train\_index <- sample(1:nrow(bike\_df), 0.7 \* nrow(bike\_df))

train\_data <- bike\_df[train\_index, ]

test\_data <- bike\_df[-train\_index, ]

# Confirm dimensions of the training and testing datasets

cat("Dimensions of Training Data:", dim(train\_data), "\n")

cat("Dimensions of Testing Data:", dim(test\_data), "\n")

A close up of a number

Description automatically generated

# Train the Random Forest model

set.seed(456) # for reproducibility

rf\_model <- randomForest(cnt ~ ., data = train\_data, ntree = 200)

# Display the trained model

print(rf\_model)

A computer code with black text

Description automatically generated

# Predict using the trained model

rf\_predictions <- predict(rf\_model, newdata = test\_data)

# Evaluate model performance

rmse <- sqrt(mean((rf\_predictions - test\_data$cnt)^2))

mae <- mean(abs(rf\_predictions - test\_data$cnt))

cat("Random Forest Model Performance on Test Data:\n")

cat("RMSE:", rmse, "\n")

cat("MAE:", mae, "\n")

A close-up of a text

Description automatically generated

# Reset the plotting area

dev.off()

# Set up a new plotting window with larger dimensions

windows(width = 12, height = 8)

# Increase plot size and adjust margins

par(mar = c(5, 5, 4, 2) + 0.1, oma = c(1, 1, 1, 1), cex = 1.5)

# Plot actual vs predicted values with increased line width and point size

plot(test\_data$cnt, col = 'blue', type = 'l', ylim = c(0, max(test\_data$cnt, rf\_predictions)),

xlab = 'Index', ylab = 'Count', main = 'Actual vs Predicted Values', lwd = 3, cex.lab = 1.5, cex.axis = 1.5, cex.main = 2)

lines(rf\_predictions, col = 'red', lwd = 3)

# Add points for better visibility

points(test\_data$cnt, col = 'blue', pch = 16, cex = 1.5)

points(rf\_predictions, col = 'red', pch = 16, cex = 1.5)

# Add legend with increased text size and position it outside the plot

legend("topright", inset = c(-0.2, 0), legend = c("Actual", "Predicted"), col = c("blue", "red"),

lty = 1, lwd = 3, cex = 1.5, pch = 16, xpd = TRUE)

A graph of a graph showing value

Description automatically generated with medium confidence

A graph showing the difference between value and price

Description automatically generated with medium confidence