Bike-Sharing Demand

Shril Patel

March 3, 2025

Outline

- Executive Summary
- Introduction
- Methodology
- Results
- Conclusion
- Appendix

Executive Summary

- Rising Importance of Bike Sharing:
  - Over the past few decades, bike sharing has become increasingly significant as more people seek healthier and more livable cities where such activities are readily accessible.
- Prediction of bike rental:
  - We discovered that a polynomial model with more terms and interactions, achieved the best performance.
  - Key factors include temperature, rainfall, humidity, peak hour, and weekdays, indicating weathers significant impact on bike rentals.
- Dataset Overview:
  - The Dataset encompasses weather information (including temperature, humidity, windspeed, etc.). Hourly bike rentals counts, and date details for the capital bike share system from 2017 to 2018.

Introduction

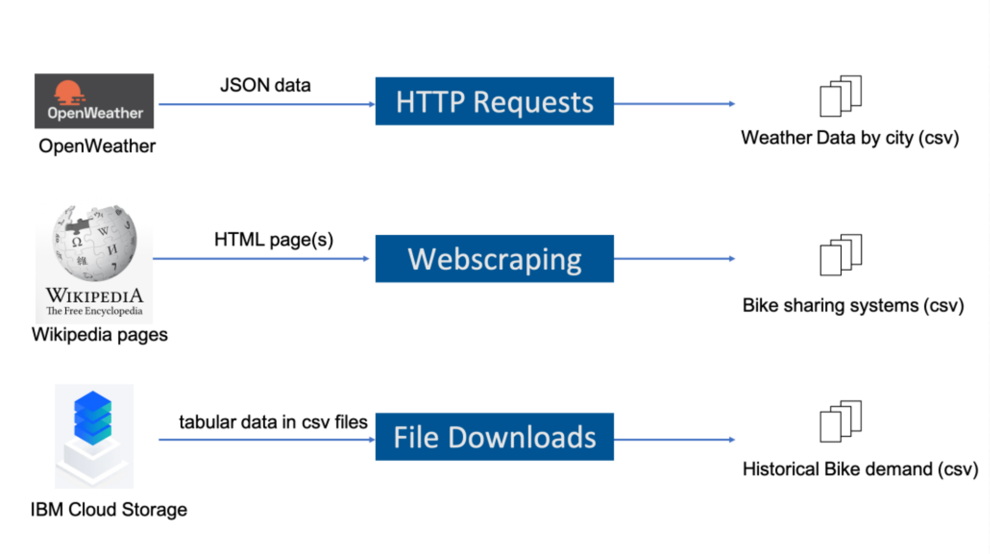
- Problem Defining:
  - It is important for each of these cities to provide a reliable supply of rental bikes to optimize accessibility at all times.
- Solid Background:
  - The global bike sharing cities dataset is an HTML table on the Wikipedia page list of bicycle-sharing systems.
- Supportive Tool:
  - The open Weather API allows users to access current and forecasted weather data for any location including over 200,00 cities.
- The Goal:
  - Minimizing program costs, including bike supply to meet demand, is important. Predicting hourly bike needs based on weather helps optimize supply.

Methodology

- Perform Data Collection
- Perform Data Wrangling
  - With Regular Expressions
  - With dplyr
- Perform exploratory data analysis (EDA) using SQL and visualization
- Perform predictive analysis using regression models
  - How to build baseline model
  - How to improve the baseline model
- Build a R Shiny dashboard

Methodology

Data Collection



- 5-day weather forecast for a list of cities from OpenWeather API.
- Global Bike Sharing Systems Dataset which is an HTML table on the Wikipedia page list of bike-sharing systems: URL. It lists active bicycle-sharing systems around the world.
- World Cities Data which contains information such as name, latitude, and longitude, about major cities around the world.
- The Seoul Sharing Demand Data set which contains weather information (Temperature, Humidity, Visibility, Dewpoint, Solar radiation, Snowfall, Rainfall), and the number of bikes rented per hour and date

Data Wrangling

- This stage involved cleaning the data by checking for missing values, mis-formatted values and/or unexpected noises.
- The first step involved the use of the R package ‘string’ and regular expression language to standardize column names, remove unwanted reference links from tables and extract numeric values from rows.
- The next step involved the use of the ‘dplyr’ package to detect and handle missing value in the data tables, create indicator (dummy) variable for categorical variables, and normalize the data using min-max normalization.

Raw Data → Regular Expressions → ~Standardization of column names ~Removal of undesired reference links ~Extraction of numeric values

Raw Data → DPLYR → ~Detection and handling of missing values ~Creation of indicator variables for categorical variables ~Data Normalization

EDA with SQL

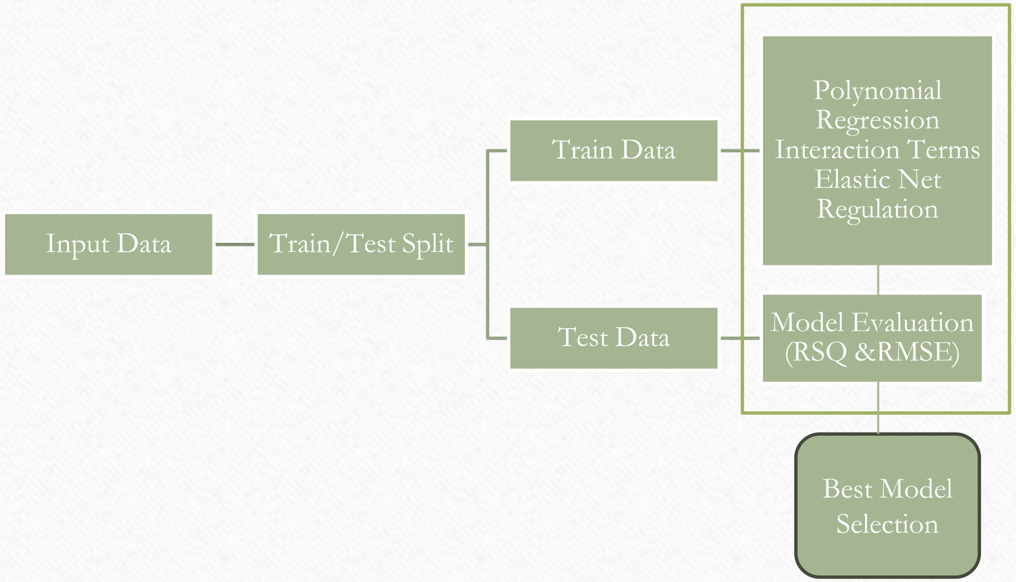
- Exploratory Data Analysis was performed on the datasets using Structured Query Language (SQL). The queries performed to;
  - Determine how many records are in the seoul_bike_sharing dataset.
  - Determine how many hours had non-zero rented bike count.
  - Query the weather forecast for Seoul over the next 3 hours.
  - Find which seasons are included in the Seoul bike sharing dataset.
  - Find the first and last dates in the Seoul bike sharing dataset.
  - Determine the date and hour had the most bike rentals.
  - Determine the average hourly temperature and the average number of bike rentals per hour over each season. List the top ten results by average bike count.
  - Find the average hourly bike count during each season.
  - Determine the average TEMPERATURE, HUMIDITY, WIND_SPEED, VISIBILITY, DEW_POINT_TEMPERATURE, SOLAR_RADIATION, RAINFALL, and SNOWFALL per season.
  - Determine the Total Bike Count and City Info for Seoul.
  - Find all city names and coordinates with comparable bike scale to Seoul's sharing systems.

EDA with Data Visualization

- Exploratory Data Analysis was also performed to visually inspect dataset using the ggplot2 library. The ggplot2 library was used;
  - Create a scatter plot of 'RENTED_BIKE_COUNT' vs 'DATE'
  - Create the same plot of the 'RENTED_BIKE_COUNT' time series, but now add 'HOURS' as the color.
  - Create a histogram overlaid with a kernel density curve.
  - Create a scatter plot to visualize the correlation between 'RENTED_BIKE_COUNT' by 'SEASONS'
  - Create a display of four boxplots of 'RENTED_BIKE_COUNT' vs. 'HOUR' grouped by 'SEASONS'.

Predictive Analysis

- This stage involved building and refining a regression model to predict the number of bikes that would be rented at each hour of the day by factoring in weather and non-weather conditions.
- The cleaned and processed data is divided into two sets: Train set (for model creation and refinement via polynomial regression, interaction, and regularization) and Test set (for Model Evaluation). The constructed models were evaluated using RMSE and RSQ as our assessment metrics, and the regression algorithm was chosen based on the lowest RMSE and highest RSQ values.



Build a R Shiny Dashboard

- The results of the predictive linear regression model were combined with an interactive dashboard created using the shiny package in R. This dashboard contained;
  - A basic max bike prediction overview map.
  - A static temperature trend line.
  - An interactive bike-sharing demand prediction trend line.
  - A static humidity and bike-sharing demand prediction correlation plot.

Results

- Exploratory Data Analysis Results
- Predictive Analysis Results
- A dashboard demo in screenshots

EDA with SQL

Busiest Bike Rental Times

- On June 19, 2018, at 6:00pm, the highest number of bike rentals in Seoul was recorded.

Task 6 - Subquery - 'all-time high'

determine which date and hour had the most bike rentals.

Solution 6

[10]:

```
# provide your solution here
dbGetQuery(conn, "SELECT DATE, HOUR, RENTED_BIKE_COUNT as Maximum_COUNT FROM seoul_bike_sharing
WHERE RENTED_BIKE_COUNT = (SELECT MAX(RENTED_BIKE_COUNT) FROM seoul_bike_sharing)")
```

A data.frame: 1 × 3

DATE	HOUR	Maximum_COUNT
<chr>	<dbl>	<dbl>
19/06/2018	18	3556

Hourly Popularity and Temperature by Seasons

- The result of the query showed that more bikes were rented during Summer, Autumn or Spring, between the hours of 5pm and 10pm and at temperatures between 15 degrees Celsius and 30 degrees Celsius which are typical Autumn, Summer and Spring temperatures.

Task 7 - Hourly popularity and temperature by season ¶

Determine the average hourly temperature and the average number of bike rentals per hour over each season. List the top ten results by average bike count.

Solution 7

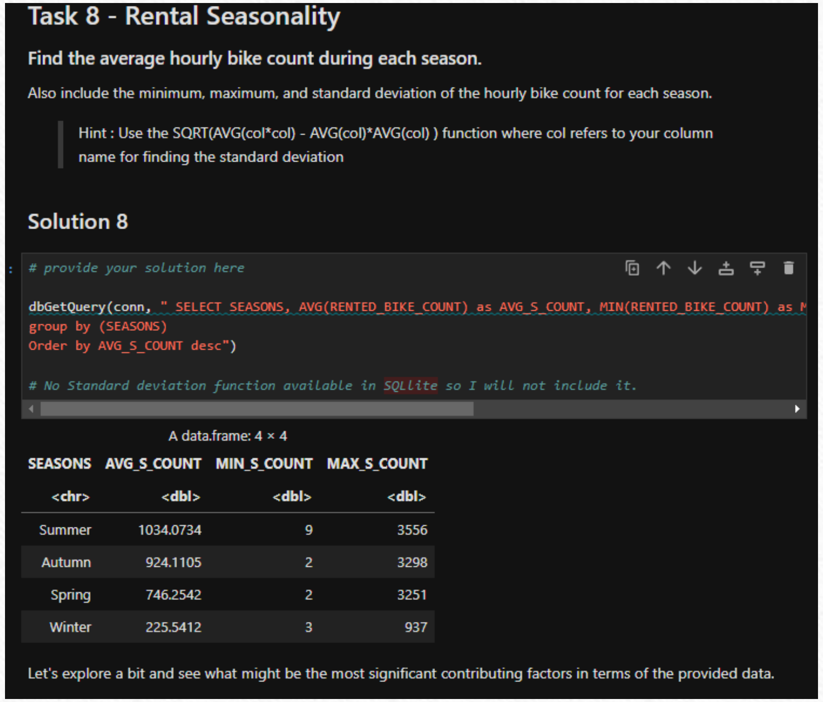
```
# provide your solution here
dbGetQuery(conn, "SELECT SEASONS, HOUR, AVG(RENTED_BIKE_COUNT), AVG(TEMPERATURE) FROM seoul_bike_sharing GROUP
```

A data.frame: 10 × 4

SEASONS	HOUR	AVG(RENTED_BIKE_COUNT)	AVG(TEMPERATURE)
<chr>	<dbl>	<dbl>	<dbl>
Summer	18	2135.141	29.38791
Autumn	18	1983.333	16.03185
Summer	19	1889.250	28.27378
Summer	20	1801.924	27.06630
Summer	21	1754.065	26.27826
Spring	18	1689.311	15.97222
Summer	22	1567.870	25.69891
Autumn	17	1562.877	17.27778
Summer	17	1526.293	30.07691
Autumn	19	1515.568	15.06346

Rental Seasonality

- The result of the query showed that on average more bikes were rented during Summer and fewer bikes were rented during Winter



Task 8 - Rental Seasonality

Find the average hourly bike count during each season.

Also include the minimum, maximum, and standard deviation of the hourly bike count for each season.

> Hint : Use the SQRT(AVG(col*col) - AVG(col)*AVG(col)) function where col refers to your column name for finding the standard deviation

Solution 8

```
# provide your solution here

dbGetQuery(conn, " SELECT SEASONS, AVG(RENTED_BIKE_COUNT) as AVG_S_COUNT, MIN(RENTED_BIKE_COUNT) as M
group by (SEASONS)
Order by AVG_S_COUNT desc")

# No Standard deviation function available in SQLlite so I will not include it.
```

A data.frame: 4 × 4

SEASONS	AVG_S_COUNT	MIN_S_COUNT	MAX_S_COUNT
<chr>	<dbl>	<dbl>	<dbl>
Summer	1034.0734	9	3556
Autumn	924.1105	2	3298
Spring	746.2542	2	3251
Winter	225.5412	3	937

Let's explore a bit and see what might be the most significant contributing factors in terms of the provided data.

Weather Seasonality

- The results showed that the temperature, rainfall, humidity, snowfall, etc. readings were very correlated with the season of the year. For example, there was a higher average temperature and rainfall in the summer, while snowfall was confined to the winter season. It also showed a correlation between the number of rented bikes and the season of the year.

Task 9 - Weather Seasonality

Consider the weather over each season. On average, what were the TEMPERATURE, HUMIDITY, WIND_SPEED, VISIBILITY, DEW_POINT_TEMPERATURE, SOLAR_RADIATION, RAINFALL, and SNOWFALL per season?

Include the average bike count as well , and rank the results by average bike count so you can see if it is correlated with the weather at all.

Solution 9

```
# provide your solution here
dbGetQuery(conn," SELECT SEASONS, AVG(RENTED_BIKE_COUNT) as AVG_S_COUNT, AVG(TEMPERATURE) as AVG_S_TEMP, AVG(HUMIDITY) as AVG_S_HUMIDITY, AVG(WIND_SPEED) as AV
group by (SEASONS)
Order by AVG_S_COUNT desc")
```

A data.frame: 4 × 10

SEASONS	AVG_S_COUNT	AVG_S_TEMP	AVG_S_HUMIDITY	AVG_WIND_SPEED	AVG_VISIBILITY	AVG_DEW_POINT	AVG_SOLAR_RADIATION	AVG_RAINFALL	AVG_SNOWFALL
<chr>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
Summer	1034.0734	26.587711	64.98143	1.609420	1501.745	18.750136	0.7612545	0.25348732	0.00000000
Autumn	924.1105	13.821580	59.04491	1.492101	1558.174	5.150594	0.5227827	0.11765617	0.06350026
Spring	746.2542	13.021685	58.75833	1.857778	1240.912	4.091389	0.6803009	0.18694444	0.00000000
Winter	225.5412	-2.540463	49.74491	1.922685	1445.987	-12.416667	0.2981806	0.03282407	0.24750000

Bike-Sharing Info in Seoul

- The result of the query showed that there were 20,000 bikes available for rent in the city of Seoul.

Task 10 - Total Bike Count and City Info for Seoul

Use an implicit join across the WORLD_CITIES and the BIKE_SHARING_SYSTEMS tables to determine the total number of bikes avaialble in Seoul, plus the following city information about Seoul: CITY, COUNTRY, LAT, LON, POPULATION, in a single view.

Notice that in this case, the CITY column will work for the WORLD_CITIES table, but in general you would have to use the CITY_ASCII column.

Solution 10

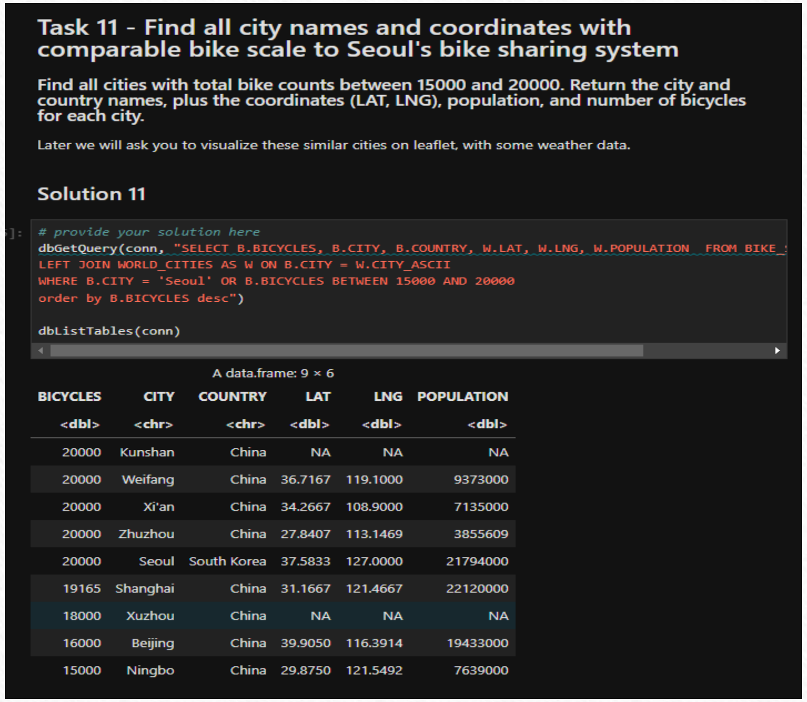
```
# provide your solution here
dbGetQuery(conn, "SELECT B.BICYCLES, B.CITY, B.COUNTRY, W.LAT, W.LNG, W.POPULATION  FROM BIKE_
LEFT JOIN WORLD_CITIES AS W ON B.CITY = W.CITY_ASCII
WHERE B.CITY = 'Seoul'")
```

A data.frame: 1 × 6

BICYCLES	CITY	COUNTRY	LAT	LNG	POPULATION
<dbl>	<chr>	<chr>	<dbl>	<dbl>	<dbl>
20000	Seoul	South Korea	37.5833	127	21794000

Cities Similar to Seoul

- The result of the query showed that six cities in China namely; Zhuzhou, Xi’an, Weifang, Shanghai, Beijing and Ningbo had similar number of rental bikes like Seoul.



Task 11 - Find all city names and coordinates with comparable bike scale to Seoul's bike sharing system

Find all cities with total bike counts between 15000 and 20000. Return the city and country names, plus the coordinates (LAT, LNG), population, and number of bicycles for each city.

Later we will ask you to visualize these similar cities on leaflet, with some weather data.

Solution 11

```
# provide your solution here
dbGetQuery(conn, "SELECT B.BICYCLES, B.CITY, B.COUNTRY, W.LAT, W.LNG, W.POPULATION  FROM BIKE_
LEFT JOIN WORLD_CITIES AS W ON B.CITY = W.CITY_ASCII
WHERE B.CITY = 'Seoul' OR B.BICYCLES BETWEEN 15000 AND 20000
order by B.BICYCLES desc")

dbListTables(conn)
```

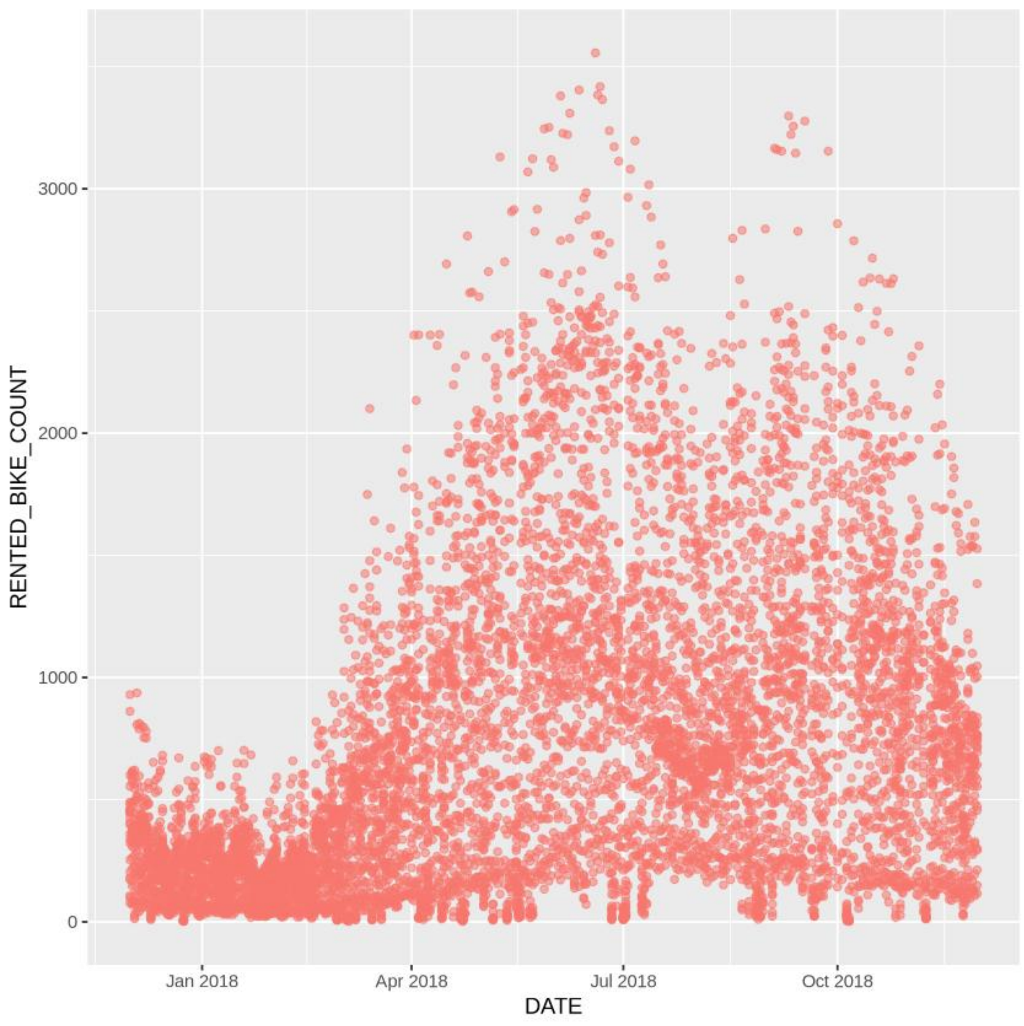
A data.frame: 9 × 6

BICYCLES	CITY	COUNTRY	LAT	LNG	POPULATION
<dbl>	<chr>	<chr>	<dbl>	<dbl>	<dbl>
20000	Kunshan	China	NA	NA	NA
20000	Weifang	China	36.7167	119.1000	9373000
20000	Xi'an	China	34.2667	108.9000	7135000
20000	Zhuzhou	China	27.8407	113.1469	3855609
20000	Seoul	South Korea	37.5833	127.0000	21794000
19165	Shanghai	China	31.1667	121.4667	22120000
18000	Xuzhou	China	NA	NA	NA
16000	Beijing	China	39.9050	116.3914	19433000
15000	Ningbo	China	29.8750	121.5492	7639000

EDA with Visualization

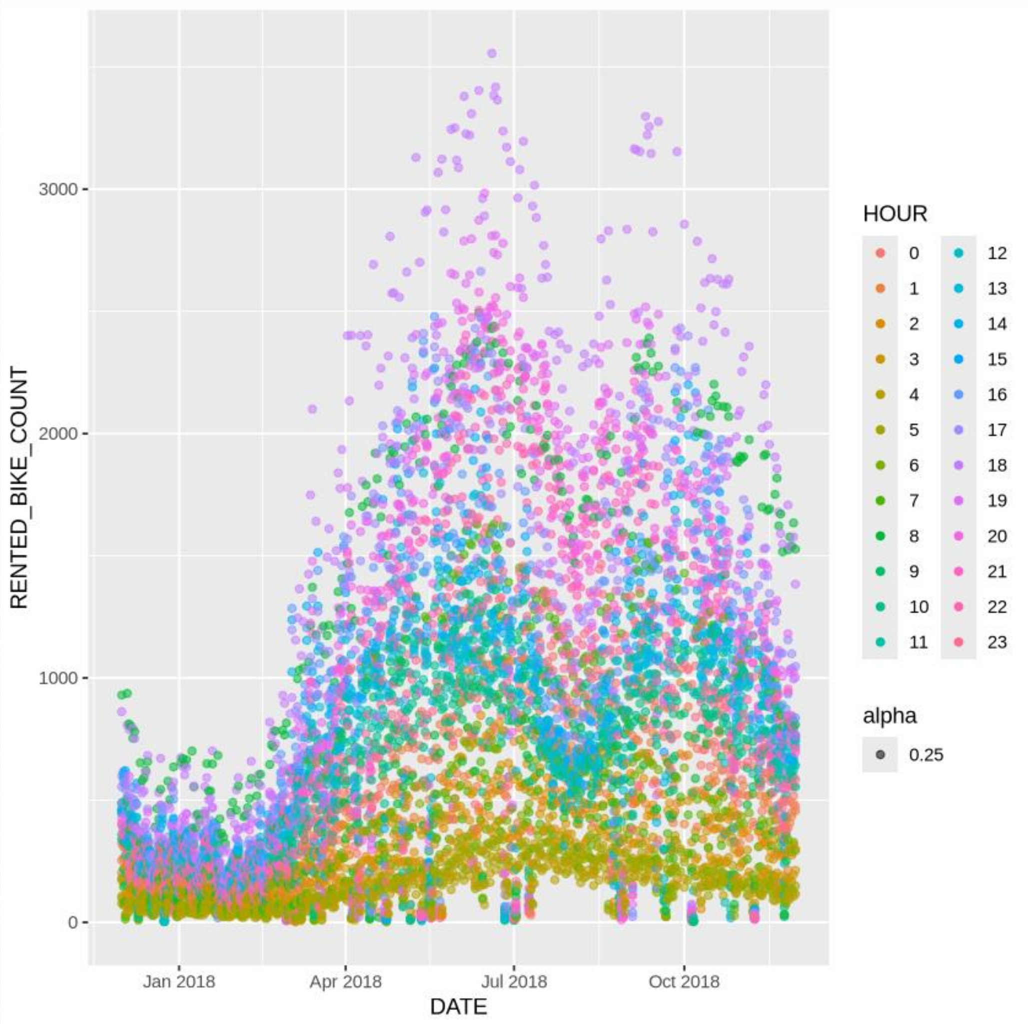
Bike Rental Vs. Date

- The number of bicycle rentals starts to rise in the spring and peaks in the summer months of June and July. We also observe a slight decline in August as the season draws to a close, followed by another high in September. The number of hired bikes also gradually declines as the winter months approach as we approach the conclusion of the year



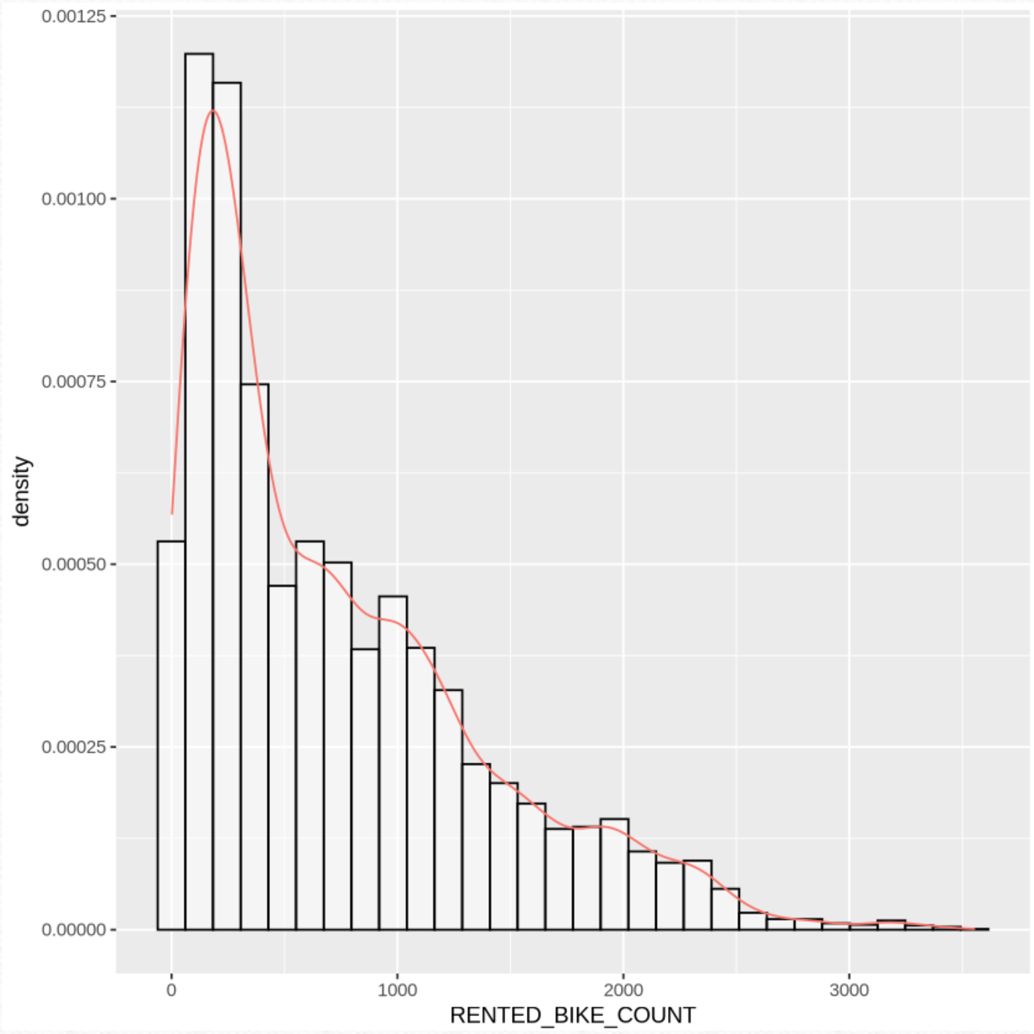
Bike Rental Vs. Datetime

- The largest hourly bike rentals are observed between the hours of 6pm and 7pm. In the mornings around 7 a.m., we also observe a moderate to high volume of bike rentals. Generally, more bike rentals occur during the day, and this pattern continues throughout the year.



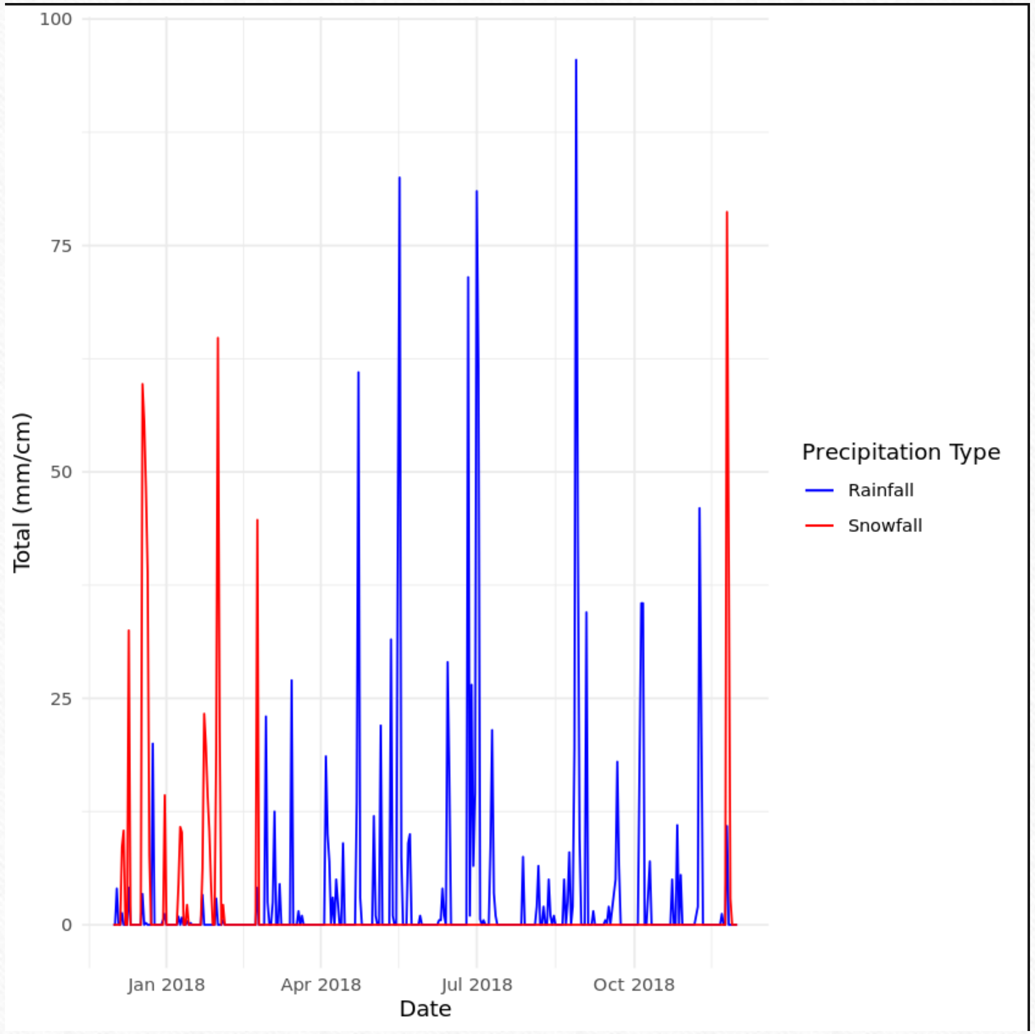
Bike Rental Histogram

- It also demonstrates that rental bike counts ranging from 0 to 500 occur most frequently in the sample. The 'mode,' or most often rented number of bikes, is about 250. Because the density curve is tilted to the left, the mean is smaller than the median.



Daily Total Rainfall & Snowfall

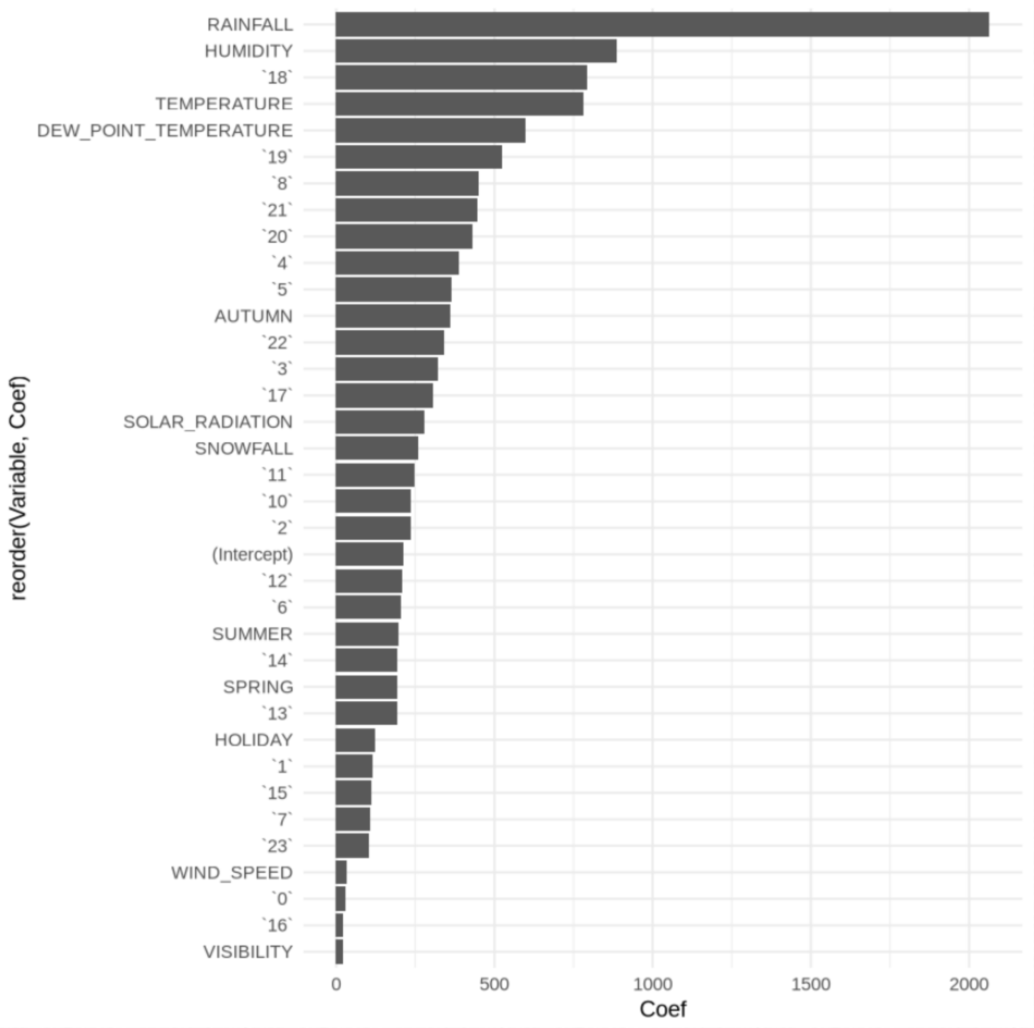
- Snowfall primarily occurs during winter days. Rainfall occurs almost year-round, with its main peaks in June, July, and September



Predictive Analysis

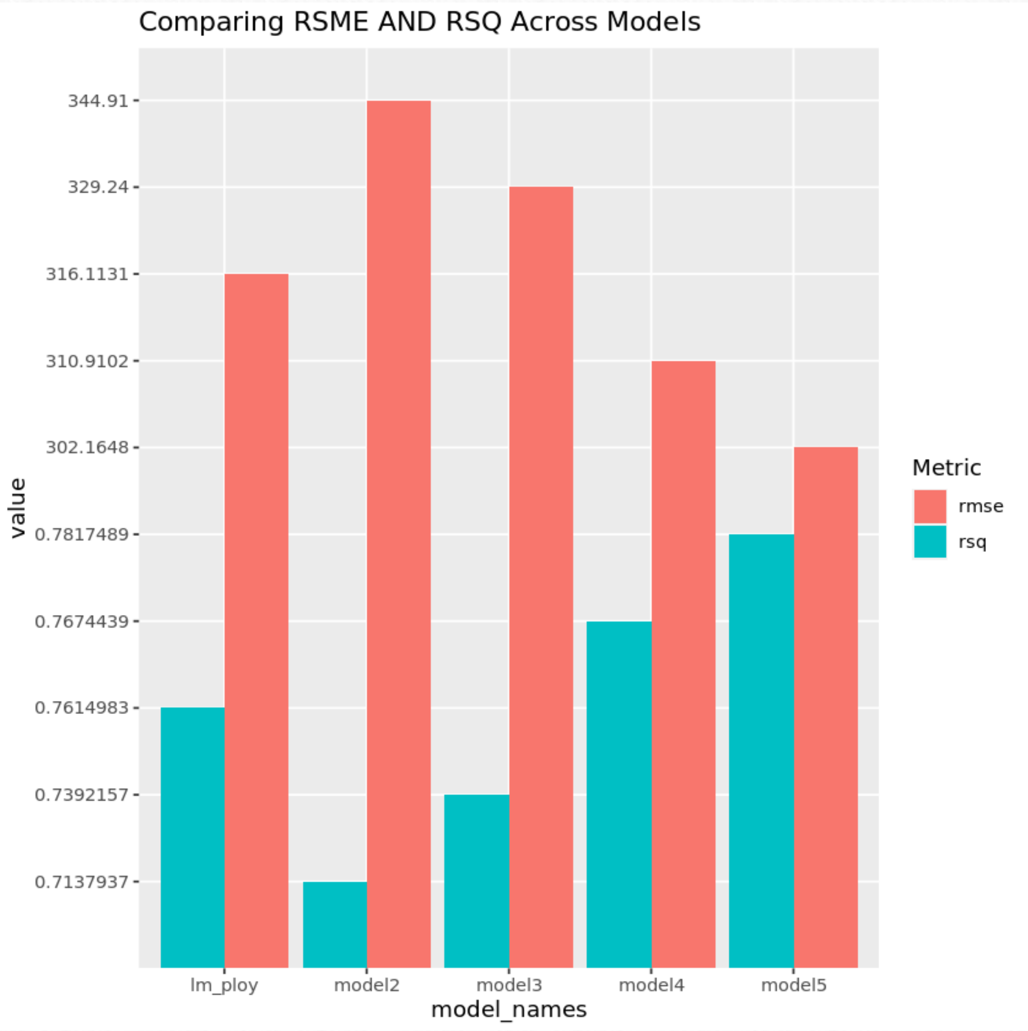
Ranked Coefficients

- Rainfall, humidity, the 6 PM temperature, and overall temperature are the strongest predictors of bike rentals.
- Rainfall and humidity make biking less comfortable and practical.
- Temperature, especially at 6 PM—a key time for commuting and leisure—encourages biking when conditions are favorable. Together, these factors highlight the influence of weather on outdoor activity preferences.



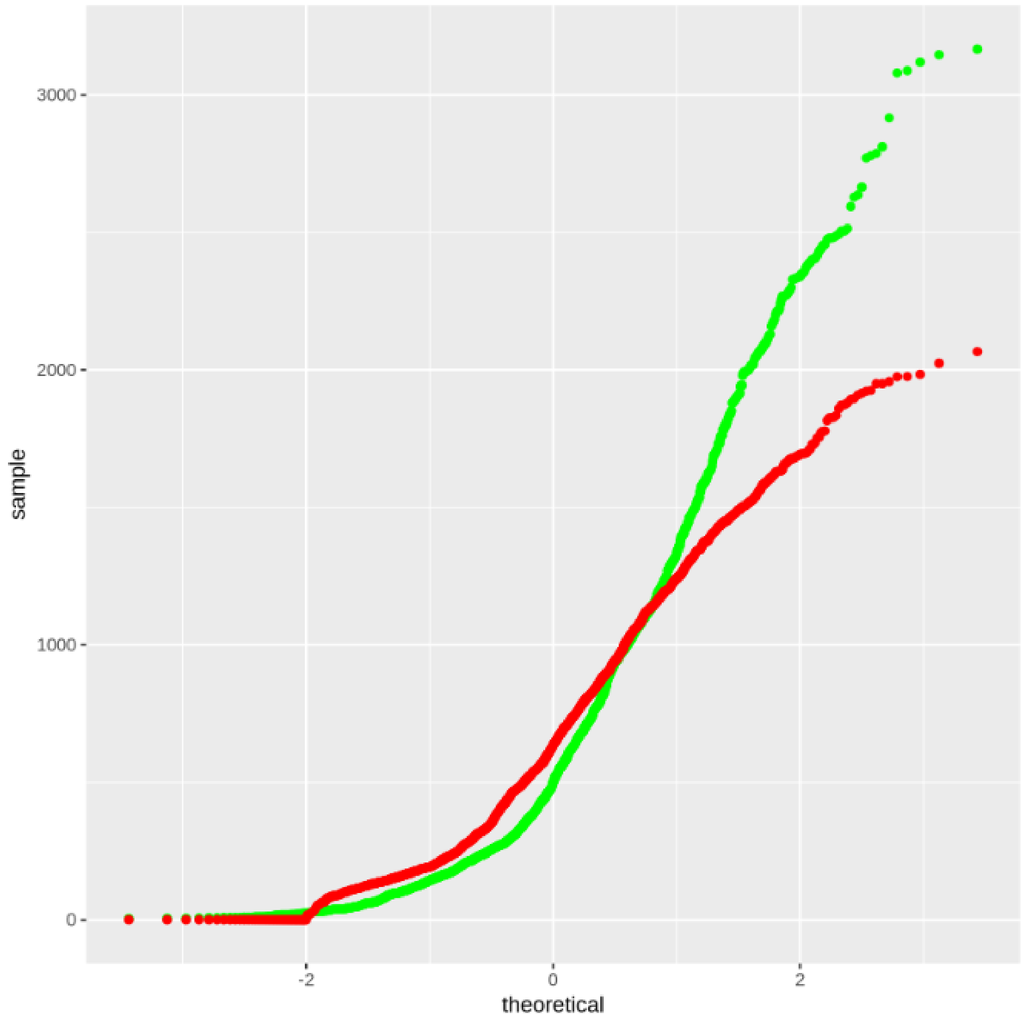
Model Evaluation

- The Root Mean Squared Error (RMSE) and R Squared (RSQ) metrics were used to assess the performance of each model. The bar charts on the right show the RMSE and RSQ performance of all models.



Q-Q Plot of the Best Model

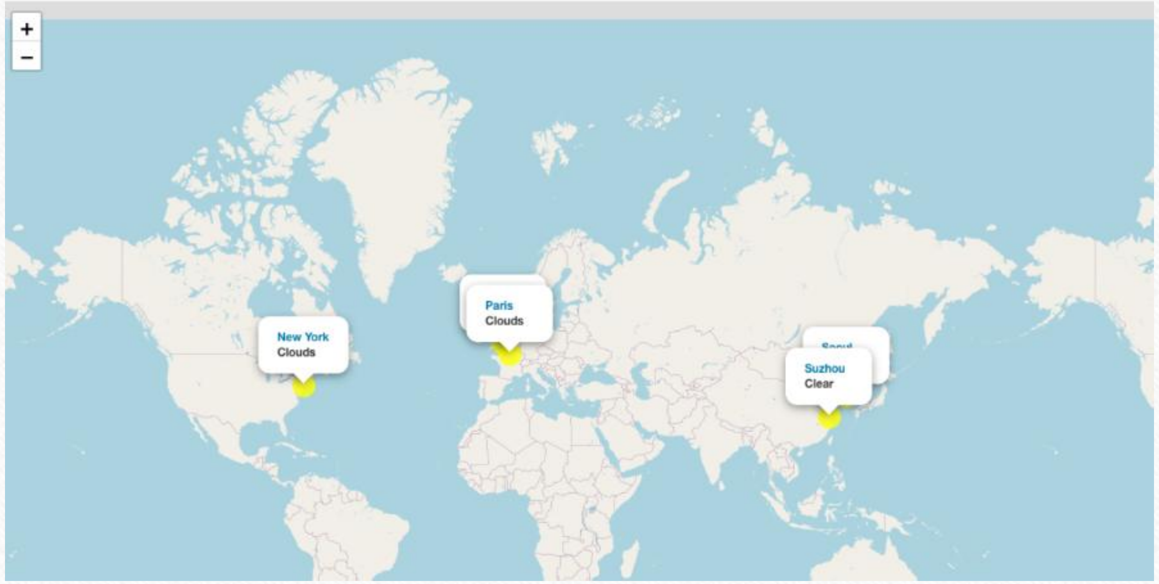
- Q-Q plot of the best model's test results vs the truths



DashBoard

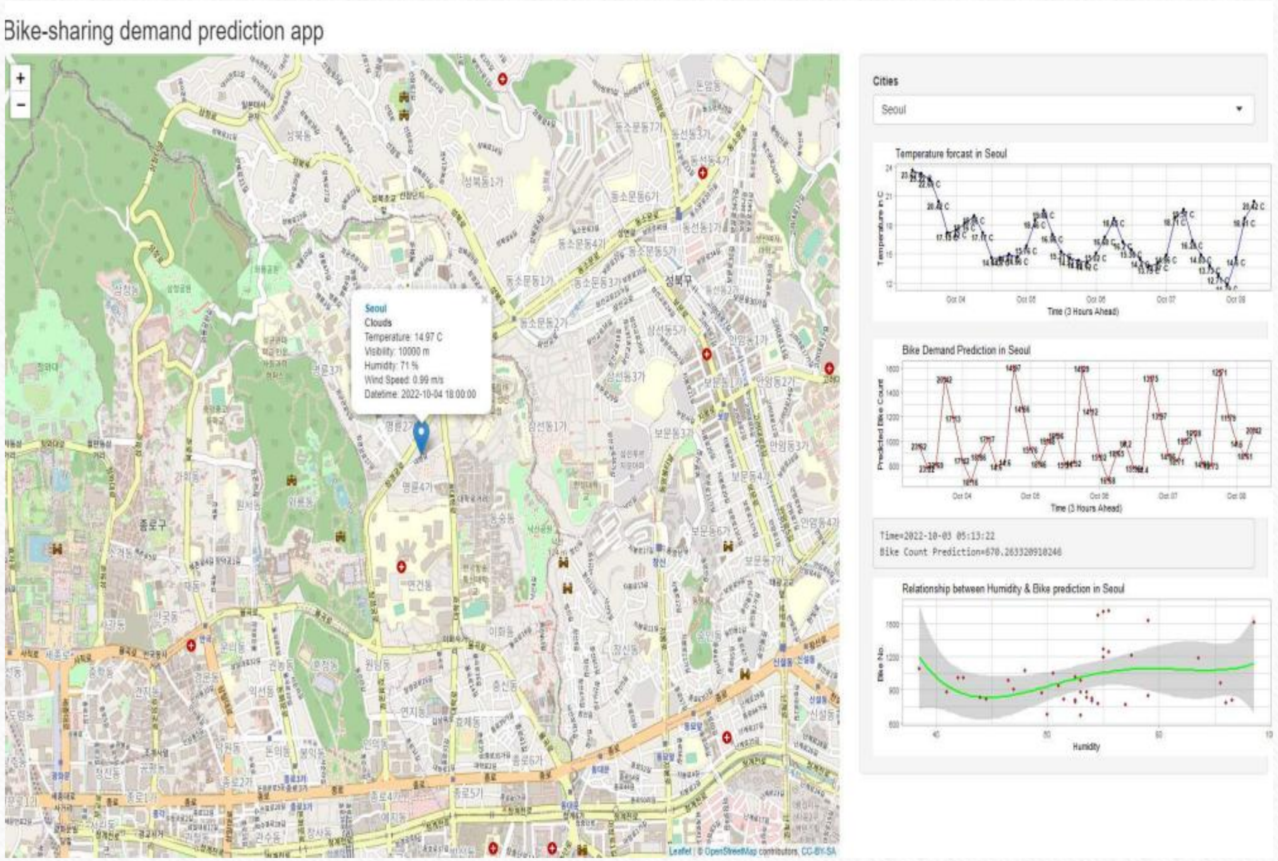
Global Bike Demand Prediction

Map of predicted bike demand for the cities of Seoul, Suzhou, London, New York and Paris.



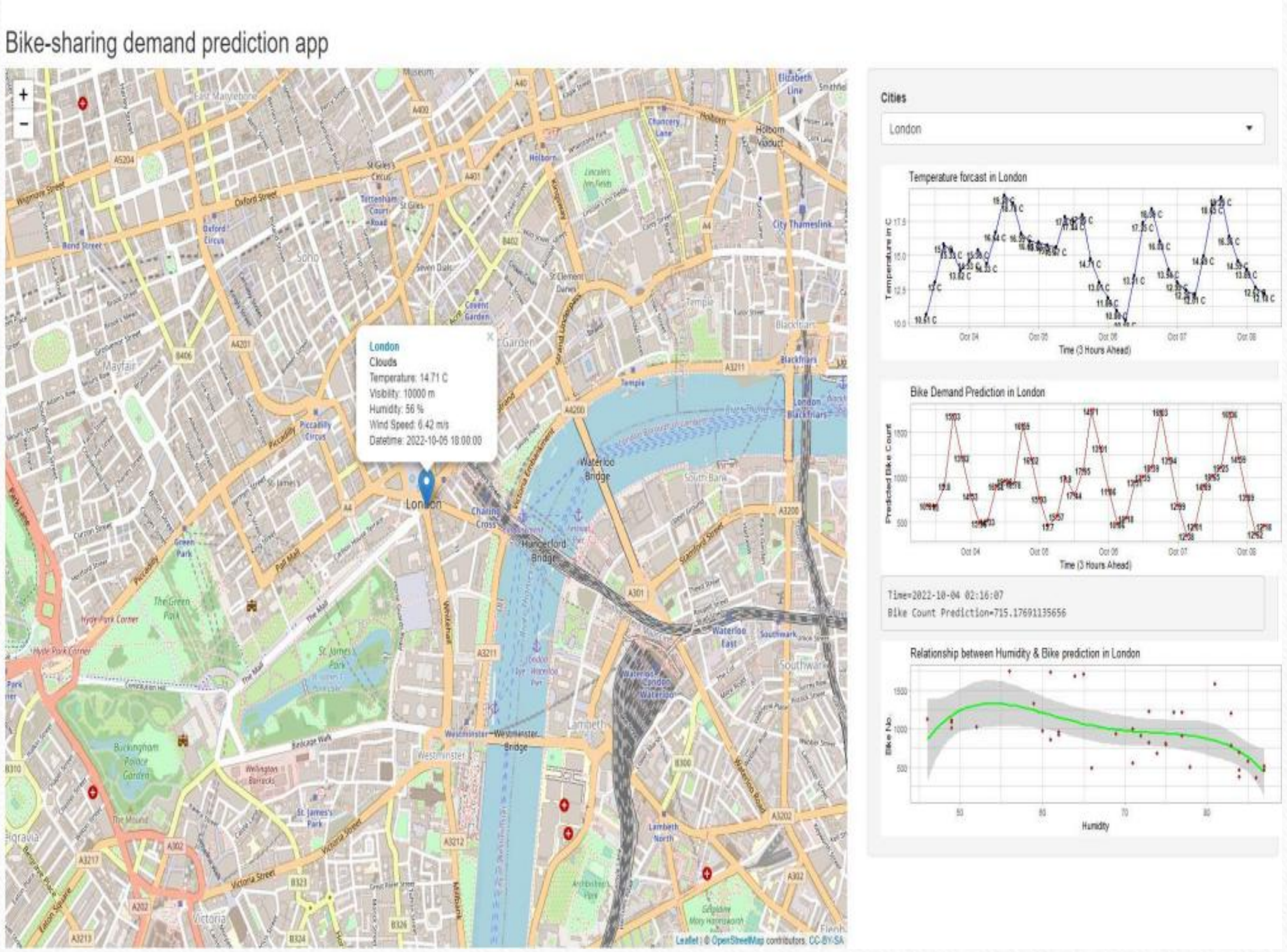
Dashboard City View: Seoul

The following screenshot displays Seoul's temperature forecast and bike demand forecast. However, the impact of humidity on predicted bike demand is missing due to technical reasons.



Dashboard City View: London

The following screenshot displays London's temperature forecast, bike demand forecast, and the impact of humidity on predicted bike demand.



Conclusion

- The summer months of June and July saw the biggest demand for bike sharing.
- During the day, the peak demand for rental bikes is between 6 and 7 p.m.
- During the winter, there is virtually little demand for bicycles.
- The quantity of bikes hired throughout each hour of any given day is heavily influenced by temperature and humidity.

Appendix

- Web Scrapping
- OpenWeather API
- Data Wrangling – Regular Expression
- EDA – SQL
- EDA with Visualiztion

Appendix: Web Scrapping

```
[1]: # Check if need to install rvest` library
require("rvest")

library(rvest)
```

```
Loading required package: rvest
```

TASK: Extract bike sharing systems HTML table from a Wiki page and convert it into a data frame

TODO: Get the root HTML node

```
[2]: url <- "https://en.wikipedia.org/wiki/List_of_bicycle-sharing_systems"
# Get the root HTML node by calling the `read_html()` method with URL
```

Note that this HTML page at least contains three child <table> nodes under the root HTML node. So, you will need to use html_nodes(root_node, "table") function to get all its child <table> nodes:

```
<html>
  <table>(table1)</table>
  <table>(table2)</table>
  <table>(table3)</table>
  ...
</html>
```

table_nodes <- html_nodes(root_node, "table")

You can use a for loop to print each table, and then you will see that the actual the bike sharing table is the first element table_nodes[[1]] .

Next, you need to convert this HTML table into a data frame using the html_table() function. You may choose to include fill = TRUE argument to fill any empty table rows/columns.

```
# Convert the bike-sharing system table into a dataframe

# Load required libraries
library(rvest) # For web scraping
library(dplyr) # For data manipulation

# Define the URL
url <- "https://en.wikipedia.org/wiki/List_of_bicycle-sharing_systems"

# Read the HTML content of the webpage
webpage <- read_html(url)

# Extract all the tables on the page
tables <- html_table(webpage, fill = TRUE)

# Identify the table of interest (inspect the tables list to find the correct one)
# For example, assume the desired table is the first table in the list
bike_sharing_table <- tables[[1]]

# Convert the table to a dataframe
bike_sharing_df <- as.data.frame(bike_sharing_table)

# View the dataframe
head(bike_sharing_df)
```

```
Attaching package: 'dplyr'


The following objects are masked from 'package:stats':

    filter, lag


The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union
```

A data.frame: 6 × 8

	Country	Country	City / Region	Name	System	Operator	Launched	Discontinued
	<chr>	<chr>	<chr>	<chr>	<chr>	<chr>	<chr>	<chr>
1	Albania	Albania	Tirana[5]	Ecovolis			March 2011	Discontinued
2	Argentina	Argentina	Buenos Aires[6][7]	Ecobici	Serttel Brasil[8]	Bike In Baires Consortium[9]	2010	
3	Argentina	Argentina	Mendoza[10]	Metrobici			2014	
4	Argentina	Argentina	Rosario	Mi Bici Tu Bici[11]			2 December 2015	
5	Argentina	Argentina	San Lorenzo, Santa Fe	Biciudad	Biciudad		27 November 2016	
6	Australia	Australia	Melbourne[12]	Melbourne Bike Share	PBSC & 8D	Motivate	June 2010	30 November 2019[13]

Summarize the bike sharing system data frame

```
# Summarize the dataframe
summary(bike_sharing_df)
```

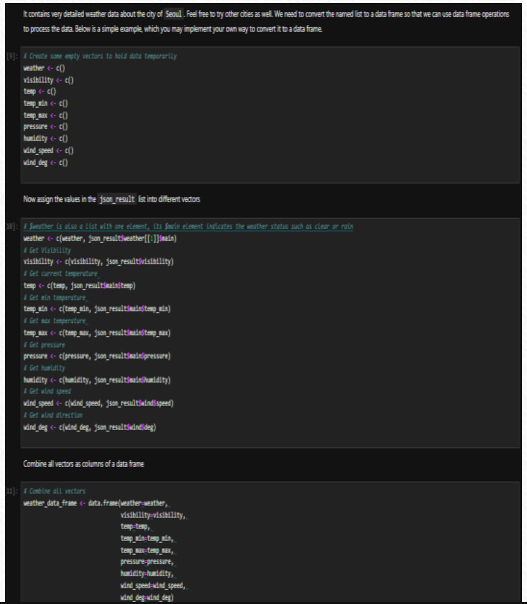
```
   Country             Country            City / Region          Name          
 Length:889          Length:889         Length:889          Length:889        
 Class :character    Class :character   Class :character    Class :character  
 Mode  :character    Mode  :character   Mode  :character    Mode  :character  
    System             Operator             Launched          Discontinued     
 Length:889          Length:889         Length:889          Length:889        
 Class :character    Class :character   Class :character    Class :character  
 Mode  :character    Mode  :character   Mode  :character    Mode  :character  
```

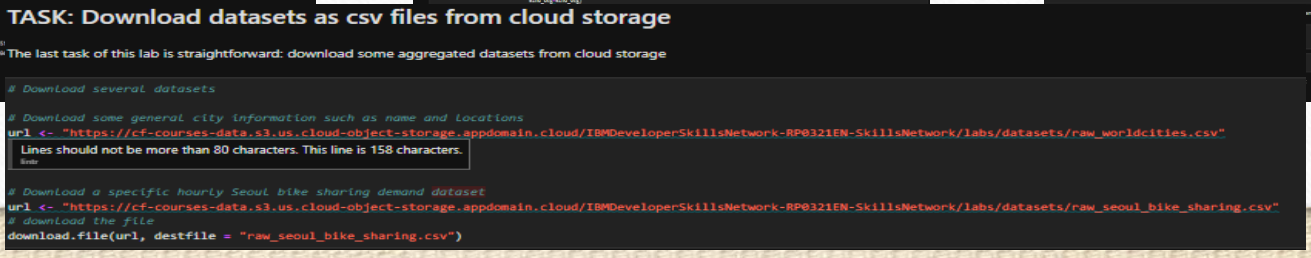
Export the data frame as a csv file called raw_bike_sharing_systems.csv

```
# Export the dataframe into a csv file
write.csv(bike_sharing_df, "raw_bike_sharing_systems.csv", row.names = FALSE)
```

For more details about webscraping with rvest , please refer to the previous webscraping notebook here:

Appendix: OpenWeather API





Appendix: Data Wrangling – Regular Expressions

```r
library(tidyverse)
library(stringr)  # For string manipulation
library(readr)    # For reading CSV files (read_csv())
library(dplyr)
#### TASK: Standardize column names for all collected datasets
setwd("C:/Users/[illegible]/Desktop/IBM CERTIFICATE/9 Capstone Project")

dataset_list <- c('raw_bike_sharing_systems.csv', 'raw_seoul_bike_sharing.csv', 'raw_cities_weather_forecast.csv', 'raw_worldcities.csv')

for (dataset_name in dataset_list) {

  # Read dataset
  dataset <- read_csv(dataset_name)

# Standardized its columns:

  # Convert all column names to uppercase

  names(dataset) <- toupper(names(dataset))

  # Replace any white space separators by underscores, using the str_replace_all function

  names(dataset) <- str_replace_all(names(dataset), " ", "_")

  # Save the dataset
  write.csv(dataset, dataset_name, row.names = FALSE)
}


# Print a summary for each data set to check whether the column names were correctly converted

for (dataset_name in dataset_list) {
  dataset <- read_csv(dataset_name)
  print(colnames(dataset))
}

####TASK: Remove undesired reference links from the scraped bike-sharing systems dataset

# First load the dataset
bike_sharing_df <- read_csv("raw_bike_sharing_systems.csv")
head(bike_sharing_df)

# Select the four columns
sub_bike_sharing_df <- bike_sharing_df %>% select(COUNTRY, CITY, SYSTEM, BICYCLES)

#Types of the selected columns
sub_bike_sharing_df %>%
  summarize_all(class) %>%
  gather(variable, class)

#ABOUT COLUMN BICLYCLE:
#Let's see why it wasn't loaded as a numeric column:possibly some entries contain characters.
# grepl searches a string for non-digital characters, and returns TRUE or FALSE
# if it finds any non-digital characters, then the bicyle column is not purely numeric
find_character <- function(strings) grepl("[^0-9]", strings)

#To find any elements in the Bicycles column containing non-numeric characters
sub_bike_sharing_df %>%
  select(BICYCLES) %>%
  filter(find_character(BICYCLES)) %>%
  slice(0:10)
#RESULT: many rows have non-numeric characters

# Define a 'reference link' character class,
# `[A-z0-9]` means at least one character
# `\\[` and `\\]` means the character is wrapped by [], such as for [12] or [abc]
ref_pattern <- "\\[[A-z0-9]+\\]"
find_reference_pattern <- function(strings) grepl(ref_pattern, strings)

# Check whether the COUNTRY column has any reference links
sub_bike_sharing_df %>%
  select(COUNTRY) %>%
  filter(find_reference_pattern(COUNTRY)) %>%
  slice(0:10)
#RESULT: COUNTRY IS CLEAN

# Check whether the CITY column has any reference links
sub_bike_sharing_df %>%
  select(CITY) %>%
  filter(find_reference_pattern(CITY)) %>%
  slice(0:10)
#RESULT: CITY HAS REFERENCE LINKS TO BE REMOVED
```

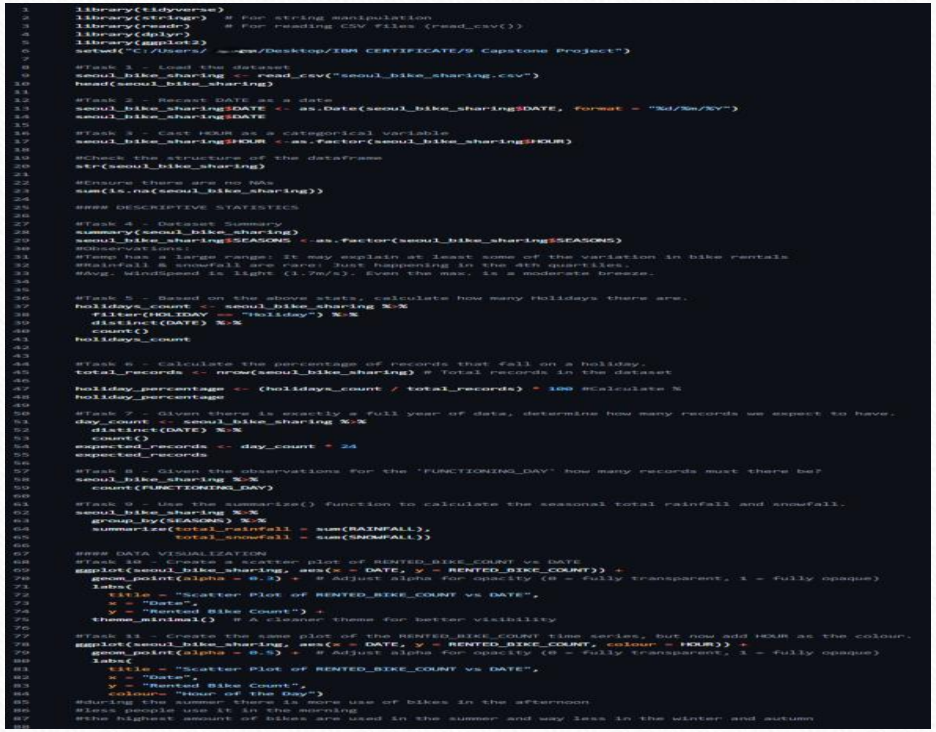
```r
# Check whether the System column has any reference links
sub_bike_sharing_df %>%
  select(SYSTEM) %>%
  filter(find_reference_pattern(SYSTEM)) %>%
  slice(0:10)
#RESULT: SYSTEM HAS REFERENCE LINKS TO BE REMOVED

##CONCLUSION:
#CITY and SYSTEM columns have some undesired reference links
#BICYCLES column has both reference links and some textual annotations.


####TASK: Remove undesired reference links using regular expressions

# To replace all reference links with an empty character for columns CITY and SYSTEM
# Define the remove_ref function
remove_ref <- function(strings) {
  # Pattern to match reference links, e.g., [1], [2], etc.
  ref_pattern <- "\\[\\d+\\]"  # Matches text like [1], [23], etc.

  # Replace all matched substrings with a white space using str_replace_all
  result <- str_replace_all(strings, ref_pattern, "")

  # Trim the result to remove any unnecessary spaces
  result <- str_trim(result)

  # Return the cleaned string
  return(result)
}

# Use the function to remove the reference links
# Apply remove_ref to CITY and SYSTEM columns
sub_bike_sharing_df <- sub_bike_sharing_df %>%
  mutate(SYSTEM = remove_ref(SYSTEM),
         CITY = remove_ref(CITY))

# Check whether all reference links are removed
# Select specific columns and filter rows with references
result <- sub_bike_sharing_df %>%
  select(CITY, SYSTEM, BICYCLES) %>%
  filter(find_reference_pattern(CITY) |
           find_reference_pattern(SYSTEM) |
           find_reference_pattern(BICYCLES))

# Print the result to check if any references remain
print(result)
head(result)


####TASK: Extract the numeric value using regular expressions

# Extract the first number
extract_num <- function(columns){

  # Define a digital pattern
  digitals_pattern <- "\\d+"  # Matches any sequence of digits

  # Find the first match using str_extract
  result <- str_extract(columns, digitals_pattern)

  # Convert the result to numeric using the as.numeric() function
  result <- as.numeric(result)

  # Return the numeric result
  return(result)
}

# Use the mutate() function on the BICYCLES column
sub_bike_sharing_df <- sub_bike_sharing_df %>%
  mutate(BICYCLES = extract_num(BICYCLES))

summary(sub_bike_sharing_df$BICYCLES)

# Write the dataset to a CSV file
write.csv(sub_bike_sharing_df, "C:/Users/[illegible]/Desktop/IBM CERTIFICATE/9 Capstone Project/bike_sharing_systems.csv", row.names = FALSE)
```

Appendix: EDA – SQL

```r
#Install RSQLite package
install.packages("https://cran.r-project.org/src/contrib/Archive/RSQLite/RSQLite_0.10.0.tar.gz", repos = NULL, type = "source", dependencies = TRUE)
library("RSQLite")
setwd("C:/Users/[redacted]/Desktop/IBM CERTIFICATE/9 Capstone Project")

library(tidyverse)
library(stringr)  # For string manipulation
library(readr)    # For reading CSV files (read_csv())
library(dplyr)
library(ggplot2)

#Establish Connection
conn <- dbConnect(RSQLite::SQLite(),"RDB.sqlite")

#Read Datasets
SEOUL_BIKE_SHARING <- read_csv("seoul_bike_sharing.csv")
CITIES_WEATHER_FORECAST <- read_csv("cities_weather_forecast.csv")
BIKE_SHARING_SYSTEMS <- read_csv("bike_sharing_systems.csv")
WORLD_CITIES <- read_csv("world_cities.csv")

#Load Tables
dbWriteTable(conn, "SEOUL_BIKE_SHARING", SEOUL_BIKE_SHARING, overwrite=TRUE, header = TRUE)
dbWriteTable(conn, "CITIES_WEATHER_FORECAST", CITIES_WEATHER_FORECAST, overwrite=TRUE, header = TRUE)
dbWriteTable(conn, "BIKE_SHARING_SYSTEMS", BIKE_SHARING_SYSTEMS, overwrite=TRUE, header = TRUE)
dbWriteTable(conn, "WORLD_CITIES", WORLD_CITIES, overwrite=TRUE, header = TRUE)
dbListTables(conn)


#### TASK 1: Determine how many records are in the seoul_bike_sharing dataset.
dbGetQuery(conn, 'SELECT COUNT(*) AS Records FROM SEOUL_BIKE_SHARING')

#### TASK 2: Determine how many hours had non-zero rented bike count.
dbGetQuery(conn, "SELECT count(HOUR) as Numer_of_hours FROM SEOUL_BIKE_SHARING
WHERE RENTED_BIKE_COUNT > 0")

#### TASK 3: Query the the weather forecast for Seoul over the next 3 hours.
#Recall that the records in the CITIES_WEATHER_FORECAST dataset are 3 hours apart, so we just need the first record from the query

dbGetQuery(conn, "SELECT * FROM CITIES_WEATHER_FORECAST
WHERE CITY = 'Seoul'
Limit 1")

#### TASK 4: Find which seasons are included in the seoul bike sharing dataset.
dbGetQuery(conn, "SELECT distinct SEASONS as Seasons
FROM SEOUL_BIKE_SHARING")

#### TASK 5: Find the first and last dates in the Seoul Bike Sharing dataset.
dbGetQuery(conn, "SELECT MIN(DATE) as Start_Date, MAX(DATE) as End_Date
FROM SEOUL_BIKE_SHARING")

#### TASK 6: Determine which date and hour had the most bike rentals.
dbGetQuery(conn, "
SELECT DATE, HOUR, RENTED_BIKE_COUNT AS Maximum_COUNT
FROM SEOUL_BIKE_SHARING
WHERE RENTED_BIKE_COUNT = (SELECT MAX(RENTED_BIKE_COUNT) FROM SEOUL_BIKE_SHARING)")

#### TASK 7: Determine the average hourly temperature and the average number of bike rentals per hour over each season. List the top ten results by average bike count.
dbGetQuery(conn, "SELECT SEASONS, HOUR, AVG(RENTED_BIKE_COUNT), AVG(TEMPERATURE)
           FROM SEOUL_BIKE_SHARING
           GROUP BY SEASONS, HOUR
           ORDER BY AVG(RENTED_BIKE_COUNT) DESC
           LIMIT 10")
```

```r
#### TASK 8: Find the average hourly bike count during each season.
#Include:  min, max, and sd of the hourly bike count for each season
dbGetQuery(conn, "
SELECT
    SEASONS,
    AVG(RENTED_BIKE_COUNT) AS Avg_Bike_Count,
    MIN(RENTED_BIKE_COUNT) AS Min_Bike_Count,
    MAX(RENTED_BIKE_COUNT) AS Max_Bike_Count,
    SQRT(AVG(RENTED_BIKE_COUNT * RENTED_BIKE_COUNT) - AVG(RENTED_BIKE_COUNT) * AVG(RENTED_BIKE_COUNT)) AS Std_Dev_Bike_Count
FROM SEOUL_BIKE_SHARING
GROUP BY SEASONS")

#### TASK 9: Consider the weather over each season.
#On avg, what were the TEMPERATURE, HUMIDITY, WIND_SPEED, VISIBILITY, DEW_POINT_TEMPERATURE, SOLAR_RADIATION, RAINFALL, and SNOWFALL per season?
#Include the average bike count as well , and rank the results by average bike count
dbGetQuery(conn, "
SELECT
    SEASONS,
    AVG(TEMPERATURE) AS Avg_Temperature,
    AVG(HUMIDITY) AS Avg_Humidity,
    AVG(WIND_SPEED) AS Avg_WindSpeed,
    AVG(VISIBILITY) AS Avg_Visibility,
    AVG(DEW_POINT_TEMPERATURE) AS Avg_DewPointTemp,
    AVG(SOLAR_RADIATION) AS Avg_SolarRad,
    AVG(RAINFALL) AS Avg_Rainfall,
    AVG(SNOWFALL) AS Avg_Snowfall,
    AVG(RENTED_BIKE_COUNT) AS Avg_RentedBikes
FROM SEOUL_BIKE_SHARING
GROUP BY SEASONS
ORDER BY AVG(RENTED_BIKE_COUNT) DESC")

#### TASK 10: Use an implicit join across the WORLD_CITIES and the BIKE_SHARING_SYSTEMS tables to determine the total number of bikes avaialble in Seoul, plus the following c
dbGetQuery(conn, "SELECT B.BICYCLES, B.CITY, B.COUNTRY, W.LAT, W.LNG, W.POPULATION
FROM BIKE_SHARING_SYSTEMS AS B
LEFT JOIN WORLD_CITIES AS W
ON B.CITY = W.CITY_ASCII
WHERE B.CITY = 'Seoul'")

#TASK 11: Find all cities with total bike counts between 15000 and 20000.
#Return the city and country names, plus the coordinates (LAT, LNG), population, and number of bicycles for each city.

dbGetQuery(conn, "
SELECT
    B.BICYCLES,
    B.CITY,
    B.COUNTRY,
    W.LAT,
    W.LNG,
    W.POPULATION
FROM BIKE_SHARING_SYSTEMS AS B
LEFT JOIN WORLD_CITIES AS W
    ON B.CITY = W.CITY_ASCII
WHERE B.CITY = 'Seoul'
   OR (B.BICYCLES BETWEEN 15000 AND 20000)
ORDER BY B.BICYCLES DESC
")

dbListTables(conn)

# Once you're done with the connection, close it
dbDisconnect(conn)
```

EDA with Visualizations



```r
library(tidyverse)
library(stringr)   # For string manipulation
library(readr)     # For reading CSV files (read_csv())
library(dplyr)
library(ggplot2)
setwd("C:/Users/[redacted]/Desktop/IBM CERTIFICATE/9 Capstone Project")

#Task 1 - Load the dataset
seoul_bike_sharing <- read_csv("seoul_bike_sharing.csv")
head(seoul_bike_sharing)

#Task 2 - Recast DATE as a date
seoul_bike_sharing$DATE <- as.Date(seoul_bike_sharing$DATE, format = "%d/%m/%Y")
seoul_bike_sharing$DATE

#Task 3 - Cast HOUR as a categorical variable
seoul_bike_sharing$HOUR <-as.factor(seoul_bike_sharing$HOUR)

#Check the structure of the dataframe
str(seoul_bike_sharing)

#Ensure there are no NAs
sum(is.na(seoul_bike_sharing))

#### DESCRIPTIVE STATISTICS

#Task 4 - Dataset Summary
summary(seoul_bike_sharing)
seoul_bike_sharing$SEASONS <-as.factor(seoul_bike_sharing$SEASONS)
#Observations:
#Temp has a large range: It may explain at least some of the variation in bike rentals
#Rainfall & snowfall are rare: Just happening in the 4th quartiles.
#Avg. WindSpeed is light (1.7m/s). Even the max. is a moderate breeze.


#Task 5 - Based on the above stats, calculate how many Holidays there are.
holidays_count <- seoul_bike_sharing %>%
  filter(HOLIDAY == "Holiday") %>%
  distinct(DATE) %>%
  count()
holidays_count


#Task 6 - Calculate the percentage of records that fall on a holiday.
total_records <- nrow(seoul_bike_sharing) # Total records in the dataset

holiday_percentage <- (holidays_count / total_records) * 100 #Calculate %
holiday_percentage

#Task 7 - Given there is exactly a full year of data, determine how many records we expect to have.
day_count <- seoul_bike_sharing %>%
  distinct(DATE) %>%
  count()
expected_records <- day_count * 24
expected_records

#Task 8 - Given the observations for the 'FUNCTIONING_DAY' how many records must there be?
seoul_bike_sharing %>%
  count(FUNCTIONING_DAY)

#Task 9 - Use the summarize() function to calculate the seasonal total rainfall and snowfall.
seoul_bike_sharing %>%
  group_by(SEASONS) %>%
  summarize(total_rainfall = sum(RAINFALL),
            total_snowfall = sum(SNOWFALL))

#### DATA VISUALIZATION
#Task 10 - Create a scatter plot of RENTED_BIKE_COUNT vs DATE
ggplot(seoul_bike_sharing, aes(x = DATE, y = RENTED_BIKE_COUNT)) +
  geom_point(alpha = 0.3) +  # Adjust alpha for opacity (0 = fully transparent, 1 = fully opaque)
  labs(
    title = "Scatter Plot of RENTED_BIKE_COUNT vs DATE",
    x = "Date",
    y = "Rented Bike Count") +
  theme_minimal()  # A cleaner theme for better visibility

#Task 11 - Create the same plot of the RENTED_BIKE_COUNT time series, but now add HOUR as the colour.
ggplot(seoul_bike_sharing, aes(x = DATE, y = RENTED_BIKE_COUNT, colour = HOUR)) +
  geom_point(alpha = 0.5) +  # Adjust alpha for opacity (0 = fully transparent, 1 = fully opaque)
  labs(
    title = "Scatter Plot of RENTED_BIKE_COUNT vs DATE",
    x = "Date",
    y = "Rented Bike Count",
    colour= "Hour of the Day")
#during the summer there is more use of bikes in the afternoon
#less people use it in the morning
#the highest amount of bikes are used in the summer and way less in the winter and autumn
```

```r
#Task 12 - Create a histogram overlaid with a kernel density curve
ggplot(seoul_bike_sharing, aes(x = RENTED_BIKE_COUNT)) +
  geom_histogram(aes(y = ..density..), bins = 30, fill = "lightblue", color = "black", alpha = 0.6) +  # Histogram
  geom_density(alpha = 0.3, fill = "red") +  # Kernel Density Curve
  labs(
    title = "Histogram and Kernel Density Curve of RENTED_BIKE_COUNT",
    x = "Rented Bike Count",
    y = "Density")
#happens that less than 1000 bikes are rented in general
#around 3K are less likely to be rented

### Correlation between two variables (scatter plot)

#Task 13 - Use a scatter plot to visualize the correlation between RENTED_BIKE_COUNT and TEMPERATURE by SEASONS
#WITH SEASONALITY
ggplot(seoul_bike_sharing, aes(x = TEMPERATURE, y = RENTED_BIKE_COUNT, colour = factor(HOUR))) +
  geom_point(alpha = 0.5) +  # Scatter plot with opacity
  facet_wrap(~ SEASONS) +  # Separate by SEASON
  labs(title = "Bike Rentals vs Temperature by Season") +
  theme_minimal()  # Minimal theme
#WITHOUT SEASONALITY
ggplot(seoul_bike_sharing) +
  geom_point(aes(x=TEMPERATURE,y=RENTED_BIKE_COUNT,colour=HOUR),alpha=1/5)

##Describe the patterns you see
#Visually, we can see some strong correlations as approximately linear patterns.


### Outliers (Boxplot)
#Task 14 - Create a display of four boxplots of RENTED_BIKE_COUNT vs. HOUR grouped by SEASONS.
ggplot(seoul_bike_sharing, aes(x = factor(HOUR), y = RENTED_BIKE_COUNT)) +
  geom_boxplot() +  # Create boxplot
  facet_wrap(~ SEASONS) +  # Separate by SEASON
  labs(title = "Boxplot of RENTED_BIKE_COUNT vs HOUR by SEASON",
       x = "Hour of Day",
       y = "Rented Bike Count") +
  theme_minimal()


##Compare and contrast the key features of these boxplots between seasons.
#Although the overall scale of bike rental counts changes with the seasons, key features remain very similar.
#For example, peak demand times are the same across all seasons, at 8 am and 6 pm.


#Task 15 - Group the data by DATE, and use the summarize() function to calculate the daily total rainfall and snowfall.
daily_weather <- seoul_bike_sharing %>%
  group_by(DATE) %>%
  summarise(
    total_rainfall = sum(RAINFALL, na.rm = TRUE),
    total_snowfall = sum(SNOWFALL, na.rm = TRUE)
  )
daily_weather

ggplot(seoul_bike_sharing, aes(x = DATE, y = RAINFALL)) +
  geom_point(alpha = 0.3) +  # Adjust alpha for opacity (0 = fully transparent, 1 = fully opaque)
  labs(
    title = "Scatter Plot of RAINFALL vs DATE",
    x = "Date",
    y = "Rainfall") +
  theme_minimal()  # A cleaner theme for better visibility

#Task 16 - Determine how many days had snowfall
days_with_snowfall <- seoul_bike_sharing %>%
  group_by(DATE) %>%
  summarise(total_snowfall = sum(SNOWFALL, na.rm = TRUE)) %>%
  filter(total_snowfall > 0) %>%
  count()
days_with_snowfall

#Visualization to show bike count depending on the day of the week.
seoul_bike_sharing %>%
  mutate(DAY_OF_WEEK = as.factor(wday(DATE, abbr = TRUE, label = TRUE))) %>%
  ggplot(aes(x = DAY_OF_WEEK, y = RENTED_BIKE_COUNT, color = HOUR, alpha = 0.25)) +
  geom_point() +
  facet_wrap(~SEASONS)

####part of PPT presentation
# Task 17 - Group the data by DATE and calculate the daily total rainfall and snowfall
daily_weather <- seoul_bike_sharing %>%
  group_by(DATE) %>%
  summarise(
    total_rainfall = sum(RAINFALL, na.rm = TRUE),
    total_snowfall = sum(SNOWFALL, na.rm = TRUE)
  )

# Reshaping data to long format for easy plotting
daily_weather_long <- daily_weather %>%
  gather(key = "weather_type", value = "total", total_rainfall, total_snowfall)

# Bar chart for daily total rainfall and snowfall
ggplot(daily_weather_long, aes(x = DATE, y = total, fill = weather_type)) +
  geom_bar(stat = "identity", position = "dodge") +  # 'dodge' separates the bars for rainfall and snowfall
  labs(
    title = "Daily Total Rainfall and Snowfall",
    x = "Date",
    y = "Total (mm)",
    fill = "Weather Type"
  ) +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))  # Rotate x-axis labels for better readability
```