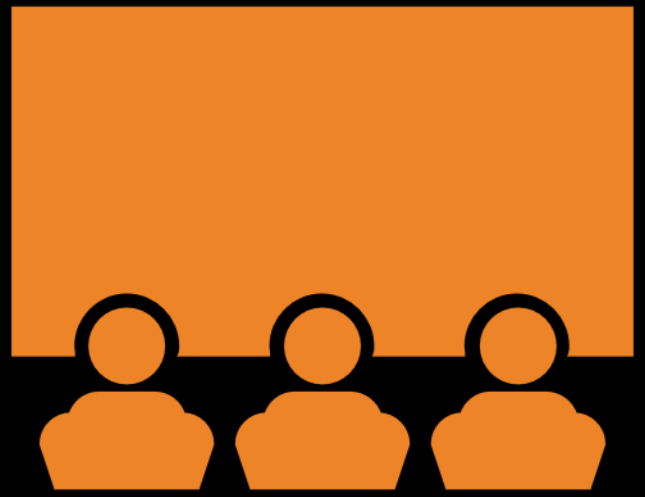
IBM DATA SCIENCE WITH R CAPSTONE PROJECT

ADJEI ANTHONY JUNIOR

MARCH 2023

OUTLINE



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

EXECUTIVE SUMMARY

- collect data from multiple sources (Download from cloud storage, Web scrapping, and API request GET calls
- Data wrangling with dplyr and stringr
  - Standardize column names for all collected datasets
  - Detect and handle missing values
  - Create indicator (dummy) variables for categorical variables
  - Remove undesired references and numeric values
- Exploratory Data Analysis with SQL(RSQLITE), tidyverse and ggplot2
- Build baseline linear regression models using the tidymodels
- Build a dashboard with R shiny and leaflet

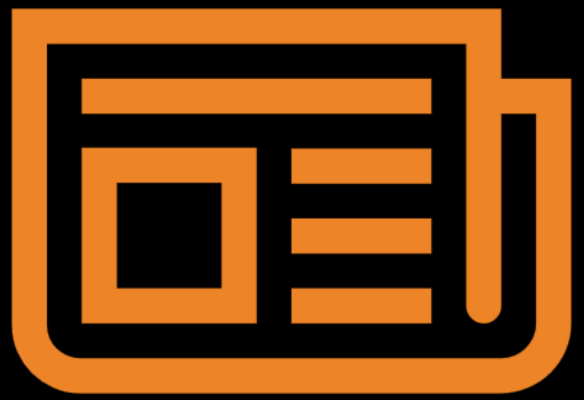
INTRODUCTION

Analyzing how the weather would affect bike-sharing demand in urban areas. To complete this project the following steps were taken :

- collect and process related weather and bike-sharing demand data from various sources,
- perform exploratory data analysis on the data,
- build predictive models to predict bike-sharing demand.
- combine results and connect to a live dashboard

to display an interactive map and associated visualization of the current weather estimated bike demand. Then create an insightful and informative slide and present it

METHODOLOGY



- Perform data collection
- Perform data wrangling
- Perform exploratory data analysis (EDA) using SQL and visualization
- Perform predictive analysis using regression models
  - To build the baseline model
  - To improve the baseline model
- Build an R Shiny dashboard app

DATA COLLECTION

web scraping

- Rvest library is required
- bike sharing systems HTML table extract from a wiki page
- bike-sharing system table converted into a data frame

API request GET calls

- Httr library is required
- make an HTTP request to the current weather API and create empty vector to hold data temporarily
- Get 5-day weather forecasts for a list of cities using the Open Weather API

Download data from cloud storage

- Use the URL address provide
- And download the data frame

WEB SCRAPING CODES

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

```
# Convert the bike-sharing system table into a dataframe
df_bike<-html_table(table)
```

Summarize the bike sharing system data frame

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

```
   Country              City                Name              System
 Length:549         Length:549         Length:549         Length:549
 Class :character   Class :character   Class :character   Class :character
 Mode  :character   Mode  :character   Mode  :character   Mode  :character
   Operator            Launched         Discontinued         Stations
 Length:549         Length:549         Length:549         Length:549
 Class :character   Class :character   Class :character   Class :character
 Mode  :character   Mode  :character   Mode  :character   Mode  :character
   Bicycles         Daily ridership
 Length:549         Length:549
 Class :character   Class :character
 Mode  :character   Mode  :character
```

API REQUEST GET CALLS CODES

```
library(httr)
```

```
Loading required package: httr
```

The API base URL to get current weather is https://api.openweathermap.org/data/2.5/weather

```
# URL for Current Weather API
current_weather_url <- 'https://api.openweathermap.org/data/2.5/weather'
```

Next, let's create a list to hold URL parameters for current weather API

```
# need to be replaced by your real API key
your_api_key <- "your_api_key"
# Input `q` is the city name
# Input `appid` is your API KEY,
# Input `units` are preferred units such as Metric or Imperial
current_query <- list(q = "Seoul", appid = "2ac1e71c6dde1ef14952e2ef0d1c331d", units="metric")
```

```
]: # $weather is also a list with one element, its $main elem
weather <- c(weather, json_result$weather[[1]]$main)
# Get Visibility
visibility <- c(visibility, json_result$visibility)
# Get current temperature
temp <- c(temp, json_result$main$temp)
# Get min temperature
temp_min <- c(temp_min, json_result$main$temp_min)
# Get max temperature
temp_max <- c(temp_max, json_result$main$temp_max)
# Get pressure
pressure <- c(pressure, json_result$main$pressure)
# Get humidity
humidity <- c(humidity, json_result$main$humidity)
# Get wind speed
wind_speed <- c(wind_speed, json_result$wind$speed)
# Get wind direction
```

```
81]: # Get forecast data for a given city list

get_weather_forecaset_by_cities <- function(city_names){
    df <- data.frame()
    for (city_name in city_names){
        # Forecast API URL
        forecast_url <- 'https://api.openweathermap.org/data/2.5/forecast'
        # Create query parameters
        forecast_query <- list(q = city_name, appid = "2ac1e71c6dde1ef14952e2ef0d1c331d", units="metric")
        # Make HTTP GET call for the given city
         response<-GET(forecast_url,query= forecast_query )
        # Note that the 5-day forecast JSON result is a list of lists. You can print the reponse to check the results
        json_list<-content(response,as="parsed" )
        results <- json_list$list

        # Loop the json result
        for(result in results) {
            city <- c(city,city_name)
             weather<-  c(weather, result$weather[[1]]$main)
               visibility<-  c(visibility,result$visibility)
               temp1<-c(temp,result$main$temp)
               tem_min1 <- c(temp_min,result$main$temp_min)
               tem_max1 <- c(temp_max,result$main$temp_max)
               pressure1<- c(pressure,result$main$pressure)
               humidity1<- c(humidity,result$main$humidity)
               wind_speed1<- c(wind_speed,result$main$wind$speed)
               wind_deg1<- c(wind_deg,result$main$wind$deg)
               forecast_datetime1<-c(forecast_datetime,result$dt_txt)
               season1<-c(season,result$main$season)

        }
             # Add the R Lists into a data frame
                                weather_data_frame<-data.frame(city=city,
                                            weather=weather,
                                            visibility=visibility,
                                            temp=temp,
                                            temp_min=temp_min,
                                            temp_max=temp_max,
                                            pressure=pressure,
                                            humidity=humidity,
                                            wind_speed=wind_speed,
                                            wind_deg=wind_deg,
```

DOWNLOADING DATA FROM CLOUD STORAGE CODES

```
# Download several datasets

# Download some general city information such as name and locations
url <- "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-RP0321EN-SkillsNetwor
# download the file
download.file(url, destfile = "raw_worldcities.csv")

# Download a specific hourly Seoul bike sharing demand dataset
url <- "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-RP0321EN-SkillsNetwor
# download the file
download.file(url, destfile = "raw_seoul_bike_sharing.csv")
```

DATA WRANGLING

- Standardize column names for all collected datasets

Change columns names to upper case and replace space with underscore

- Extract the numeric value using regular expressions
- Remove undesired reference links using regular expressions
- Detect and handle missing values
- Create indicator (dummy) variables for categorical variables
- Normalize data

```
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)
}
```

```
# remove reference link
remove_ref <- function(strings) {
 ref_pattern <- "\\[[A-z0-9]+\\]"
    # Replace all matched substrings with a white space using str_replace_all()
   strings<-str_replace_all(strings,ref_pattern," ")
    # Trim the reslt if you want
     strings <- str_squish(strings)
    # return(result)
    return(strings)

}
```

EXPLORATORY DATA ANALYSIS WITH SQL

Exploratory data analysis done using SQL gave insights into the Soul city data set to find:

- The total records of the dataset
- The number of records with non-zero rent bike counts
- The types of seasons in the city
- The start and end date
- The average temperature of the city
- The type of season with the highest and the lowest rent bike count

Exploratory Data Analysis with Data Visualization

Scatter plot of Rent Bike Number vs Date

Number of Bikes Rented vs Date:: scatter plot
Number of Bikes Rented vs Temperature:: scatter plot
Number of Bikes Rented vs Hour:: boxplot
Total daily Rainfall over the year

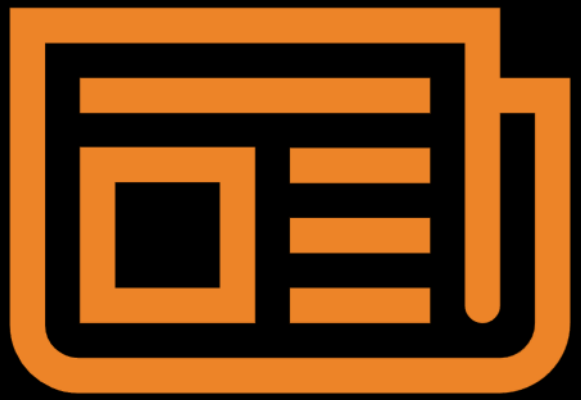
PREDICTIVE ANALYSIS

- Data set was split into training and testing data with the training taking 75% of the total dataset.
- A linear regression model was built with weather variables in the dataset with the number of bike rent variables as the target
- Another model linear regression model was built using all the variable numbers of bike rent
- The two models were used to predict data using the test data
- R-squared root mean squared error were determine to compare both models

BUILD A R SHINY DASHBOARD

- Leaflet was used to create maps of selected cities
- A dropdown menu for city selection was provided for interaction

RESULTS



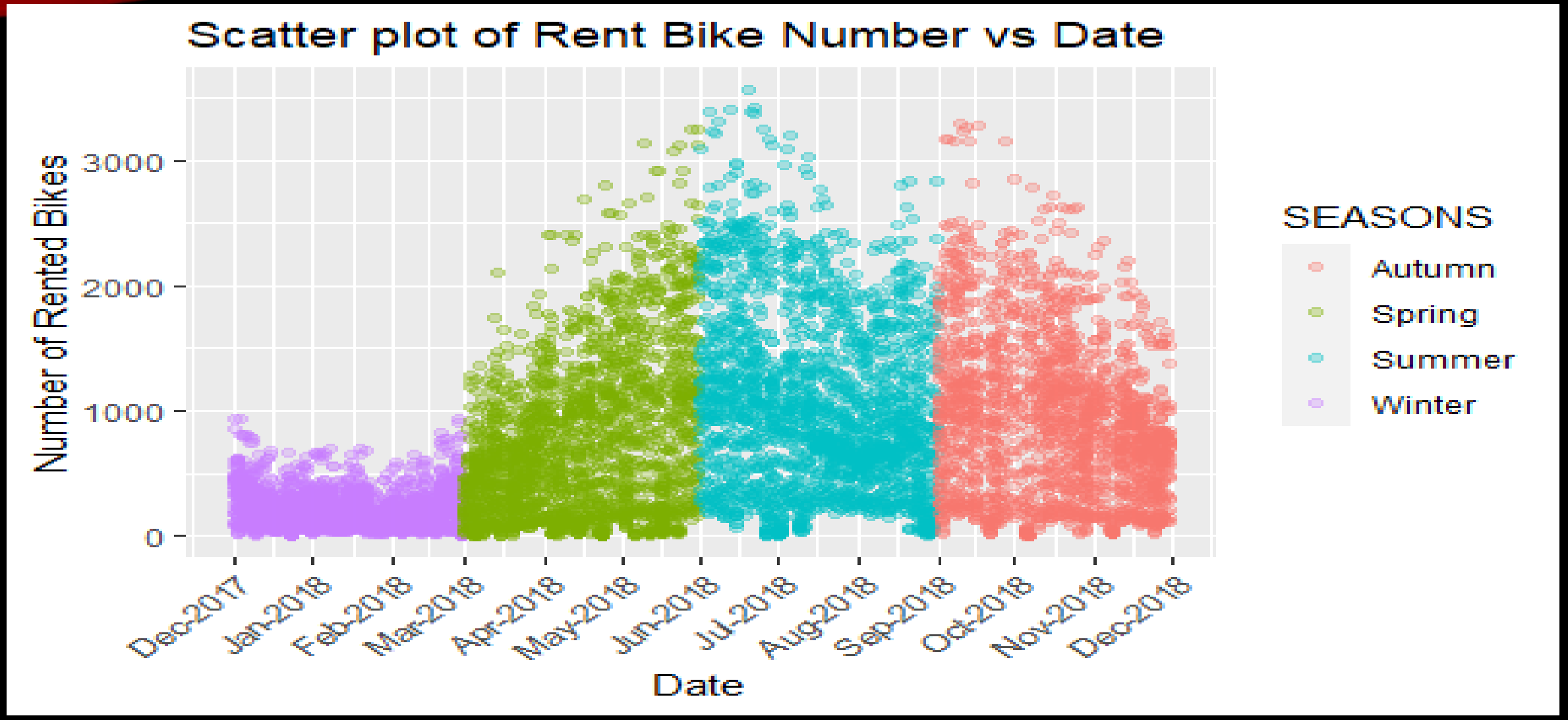
- Exploratory data analysis results
- Predictive analysis results
- A dashboard demo in screenshots

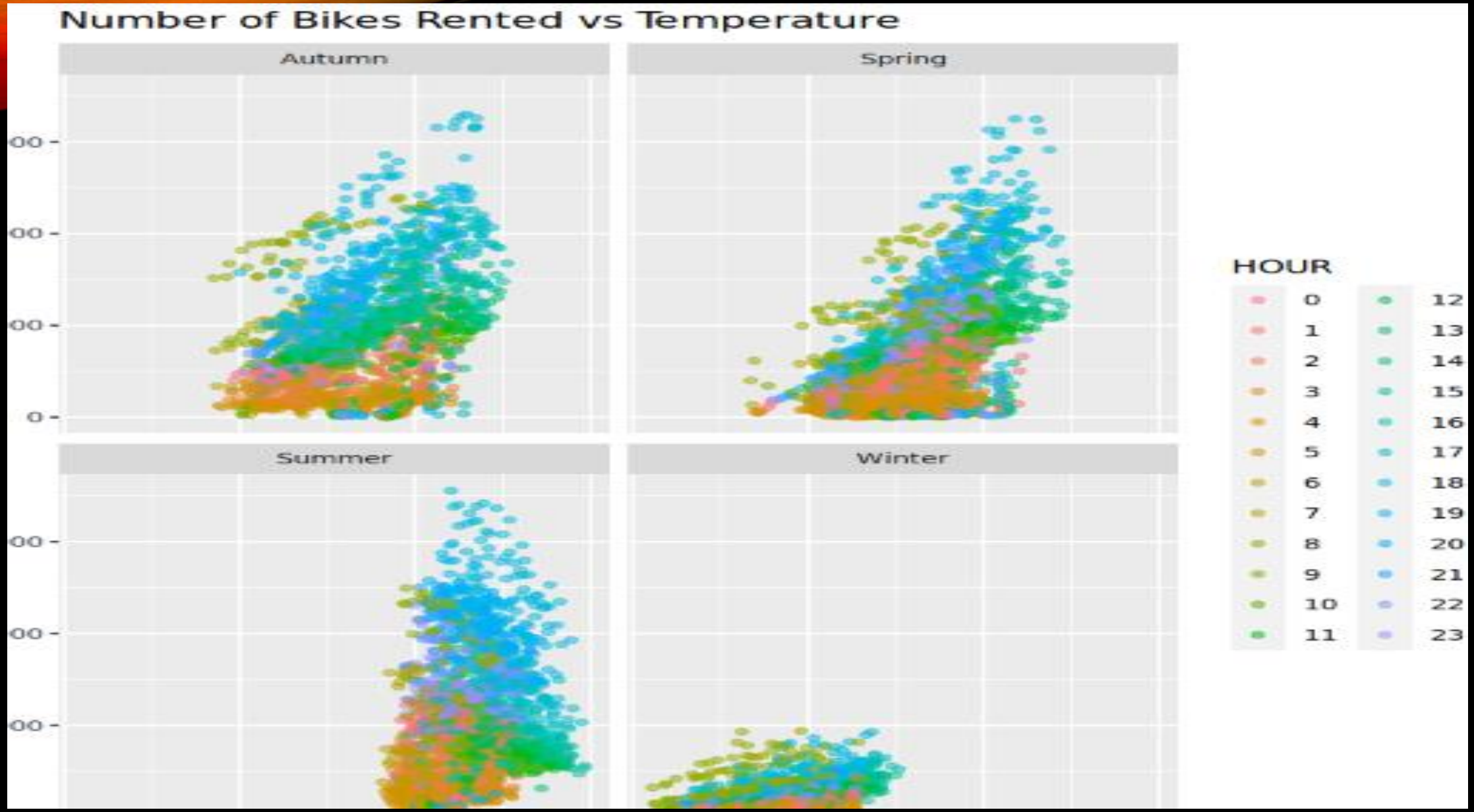
EXPLORATORY DATA ANALYSIS WITH SQL

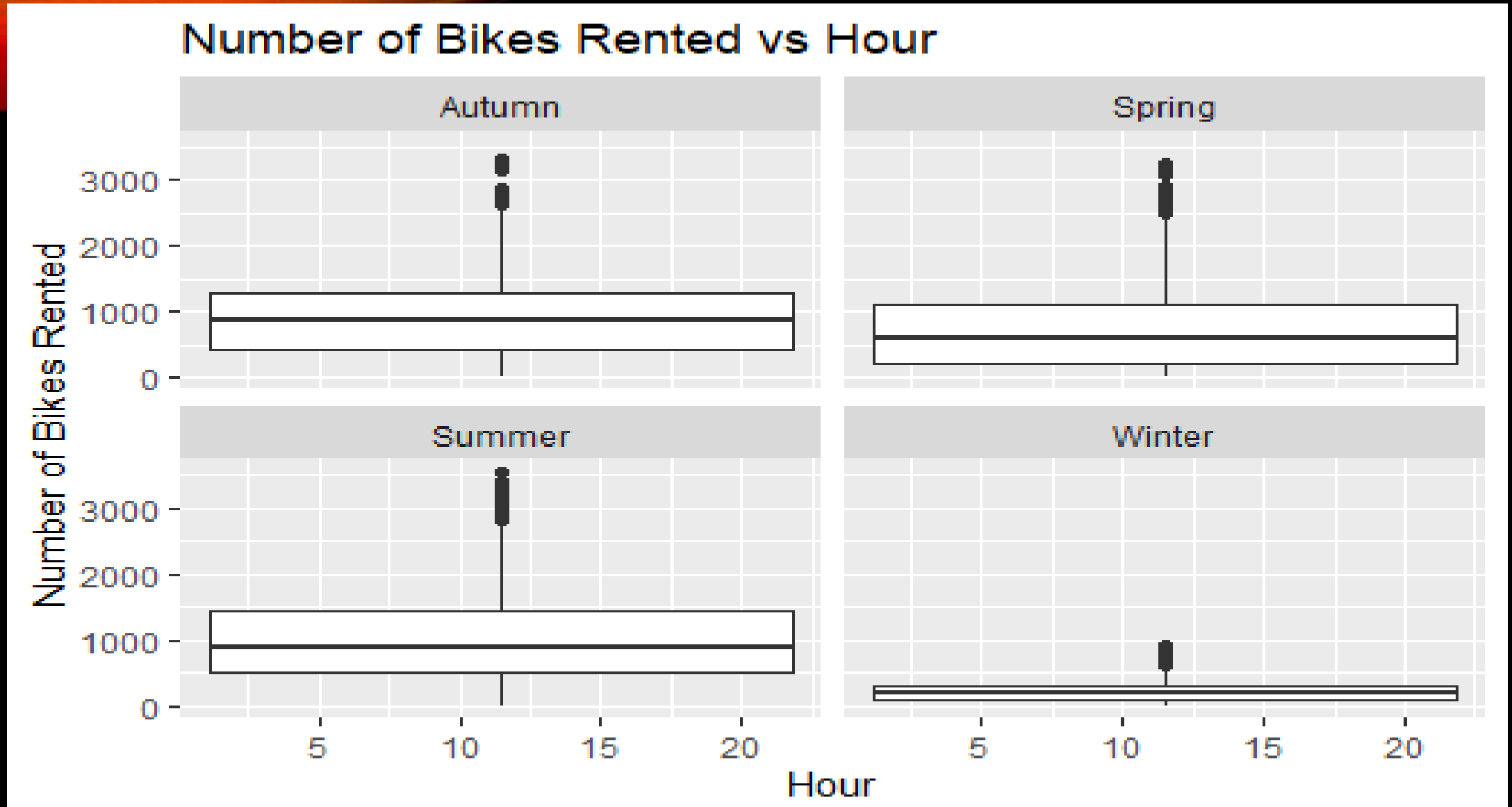
Exploratory data analysis done using SQL gave insights into the Soul city data set

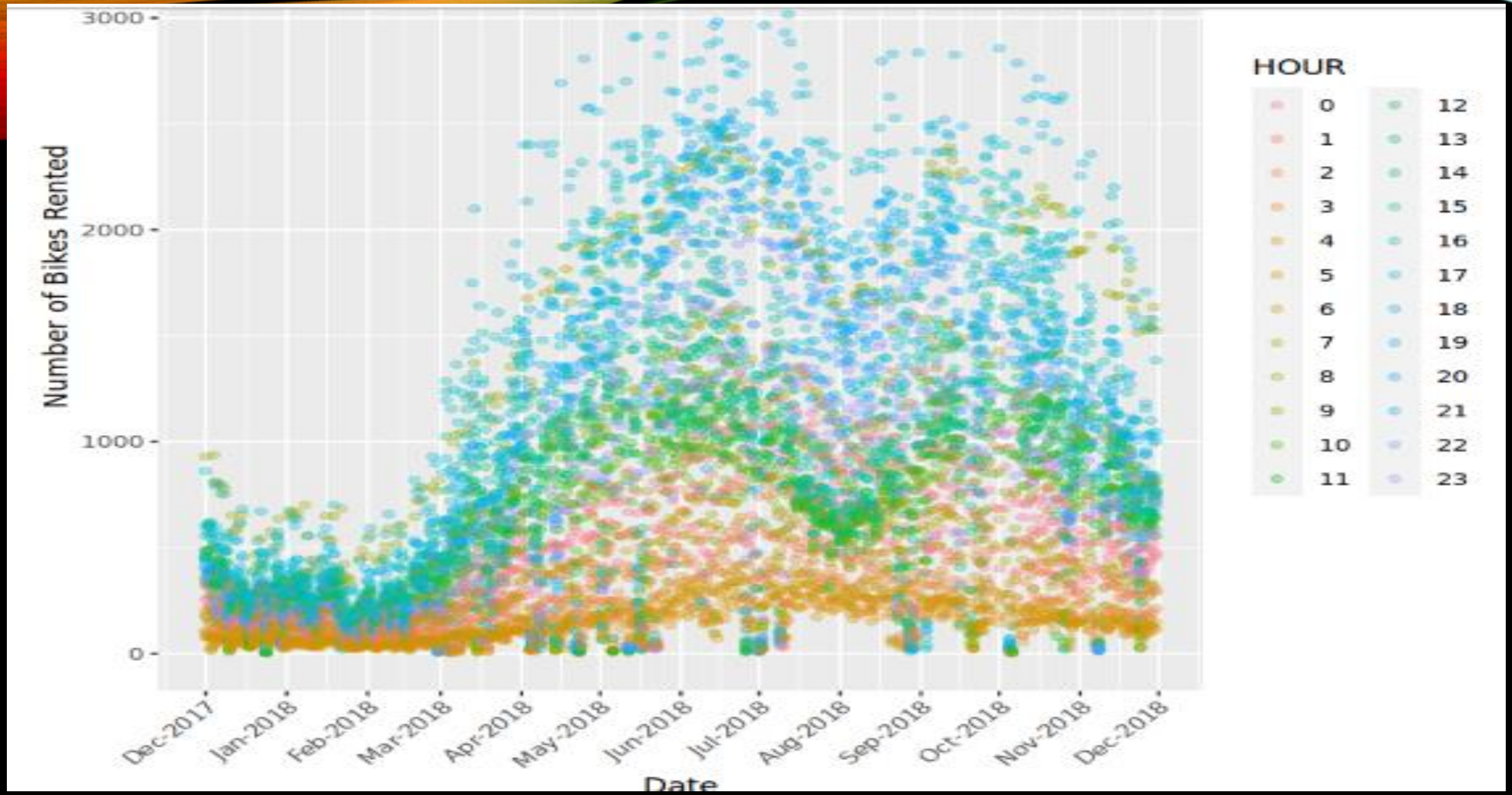
- There were 8465 in the Seoul bike-sharing dataset
- Out of the 8465 there were 8113 records with non-zero rented bike count
- The soul city has winter, spring, summer, and Autumn seasons
- The first and last data of the soul city record is from 31st Dec 2017 and 1st Jan 2018 respectively
- 19th may 2018 had more rent bike count than any other day
- Average temperature of the soul city is 12.7723
- On average the summer season had more rent bike count and winter had the lowest count

EDA with Visualization




Number of Bikes Rented vs Temperature
Autumn
Spring
Summer
Winter
HOUR
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23


Number of Bikes Rented vs Hour
Autumn
Spring
Summer
Winter
Number of Bikes Rented
3000
2000
1000
0
5
10
15
20
Hour


3000
2000
1000
0
Number of Bikes Rented
Dec-2017
Jan-2018
Feb-2018
Mar-2018
Apr-2018
May-2018
Jun-2018
Jul-2018
Aug-2018
Sep-2018
Oct-2018
Nov-2018
Dec-2018
Date
HOUR
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23

PREDICTIVE ANALYSIS

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Go to file/function | Addins | Project: (None)

Source

R 4.2.1 · C:/Users/hp/OneDrive/Desktop/Data/IBM PROJECT/sql/

```
Linear Regression Model Specification (regression)

Computational engine: lm

> lm_model_weather <- lm(RENTED_BIKE_COUNT ~ TEMPERATURE + HUMIDITY + WIND_SPEED + VISIBILITY + DEW
_POINT_TEMPERATURE + SOLAR_RADIATION + RAINFALL + SNOWFALL, train_data)
> summary(lm_model_weather)

Call:
lm(formula = RENTED_BIKE_COUNT ~ TEMPERATURE + HUMIDITY + WIND_SPEED +
    VISIBILITY + DEW_POINT_TEMPERATURE + SOLAR_RADIATION + RAINFALL +
    SNOWFALL, data = train_data)

Residuals:
     Min       1Q   Median       3Q      Max
-1348.46  -294.03   -57.28   208.59  2329.78

Coefficients:
                      Estimate Std. Error t value Pr(>|t|)
(Intercept)             156.71      58.07   2.699  0.00698 **
TEMPERATURE            2399.74     261.66   9.171  < 2e-16 ***
HUMIDITY               -918.38     126.79  -7.243  4.9e-13 ***
WIND_SPEED              404.47      48.16   8.399  < 2e-16 ***
VISIBILITY               12.56      24.86   0.505  0.61351
DEW_POINT_TEMPERATURE  -316.92     278.83  -1.137  0.25575
SOLAR_RADIATION        -444.85      34.69 -12.824  < 2e-16 ***
RAINFALL              -1764.01     182.65  -9.658  < 2e-16 ***
SNOWFALL                317.78     131.58   2.415  0.01576 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 487.3 on 6339 degrees of freedom
Multiple R-squared:  0.4303,    Adjusted R-squared:  0.4296
F-statistic: 598.5 on 8 and 6339 DF,  p-value: < 2.2e-16

>
```

Environment | History | Connections | Tutorial

Import Dataset | 437 MiB | List

R | Global Environment

Data	
bike_sharing_df	8465 obs. of 39 variables
bike_split	Large initial_split (4 elements, 2.7 M…
lm_model_weath…	Large lm (12 elements, 2.4 MB)
test_data	2117 obs. of 39 variables
train_data	6348 obs. of 39 variables
Values	
dataset_url	"https://cf-courses-data.s3.us.cloud-obje…

Files | Plots | Packages | Help | Viewer | Presentation

Zoom | Export

Type here to search

11:56 AM
30/03/2023

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Go to file/function | Addins | Project: (None)

Source

R 4.2.1 · C:/Users/hp/OneDrive/Desktop/Data/IBM PROJECT/sql/

```
Call:
lm(formula = RENTED_BIKE_COUNT ~ ., data = train_data)

Residuals:
     Min       1Q   Median       3Q      Max
-1401.45  -218.96    -7.31   199.53  1780.67

Coefficients: (3 not defined because of singularities)
                    Estimate Std. Error t value Pr(>|t|)
(Intercept)           212.20      51.04   4.158 3.26e-05 ***
TEMPERATURE           782.66     212.13   3.690 0.000227 ***
HUMIDITY             -886.73      99.49  -8.913  < 2e-16 ***
WIND_SPEED             31.91      40.27   0.792 0.428169
VISIBILITY             21.87      20.26   1.079 0.280439
DEW_POINT_TEMPERATURE  598.39     221.37   2.703 0.006888 **
SOLAR_RADIATION        276.88      41.47   6.677 2.64e-11 ***
RAINFALL             -2064.64     143.28 -14.410  < 2e-16 ***
SNOWFALL               260.97     103.50   2.522 0.011709 *
`0`                    -29.30      34.26  -0.855 0.392515
`1`                   -116.85      33.72  -3.465 0.000533 ***
`10`                  -237.52      32.74  -7.255 4.48e-13 ***
`11`                  -247.38      33.85  -7.309 3.02e-13 ***
`12`                  -208.34      34.39  -6.059 1.45e-09 ***
`13`                  -191.35      35.04  -5.461 4.90e-08 ***
`14`                  -192.44      34.44  -5.588 2.39e-08 ***
`15`                  -109.73      34.40  -3.190 0.001429 **
`16`                    23.13      34.00   0.680 0.496431
`17`                   305.55      34.15   8.946  < 2e-16 ***
`18`                   794.80      34.02  23.364  < 2e-16 ***
`19`                   522.99      34.25  15.268  < 2e-16 ***
`2`                   -237.21      33.74  -7.030 2.28e-12 ***
`20`                   432.00      34.13  12.657  < 2e-16 ***
`21`                   446.58      34.09  13.100  < 2e-16 ***
`22`                   342.64      33.85  10.123  < 2e-16 ***
`23`                   103.81      33.85   3.066 0.002175 **
`3`                   -319.87      34.23  -9.346  < 2e-16 ***
```

Environment | History | Connections | Tutorial

Import Dataset | 440 MiB | List

R | Global Environment

Data

bike_sharing_df	8465 obs. of 39 variables
bike_split	Large initial_split (4 elements, 2.7 M…
lm_model_all	List of 12
lm_model_weath…	Large lm (12 elements, 2.4 MB)
test_data	2117 obs. of 39 variables
train_data	6348 obs. of 39 variables

values

Files | Plots | Packages | Help | Viewer | Presentation

Zoom | Export

Type here to search

11:57 AM
30/03/2023

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Go to file/function | Addins | Project: (None)

lbm project.R | IBM PROJECT BULDING MODELS.R*

Source on Save | Run | Source

```
63
64
65  ## Calculating the R-squared and RMSE for the two test results:
66
67    rsq_weather <- rsq(test_results_weather, TRUTH, PREDICTED)
68  rsq_weather
69  rsq_all <- rsq(test_results_all, TRUTH, PREDICTED)
70  rsq_all
71
72  rmse_weather <- rmse(test_results_weather, TRUTH, PREDICTED)
73  rmse_weather
74  rmse_all <- rmse(test_results_all, TRUTH, PREDICTED)
75  rmse_all
76
77
78  lm_model_all
79
80
81  # Sort coefficient list
82
```

66:1 (Top Level) R Script

Environment | History | Connections | Tutorial

Import Dataset | 441 MiB | List

R | Global Environment

lm_model_weath…	Large lm (12 elements, 2.4 MB)
rmse_all	1 obs. of 3 variables
rmse_weather	1 obs. of 3 variables
rsq_all	1 obs. of 3 variables
rsq_weather	1 obs. of 3 variables
test_data	2117 obs. of 39 variables
test_results_a…	2117 obs. of 2 variables

Files | Plots | Packages | Help | Viewer | Presentation

Zoom | Export

R 4.2.1 · C:/Users/hp/OneDrive/Desktop/Data/IBM PROJECT/sql/

```
lm(formula = RENTED_BIKE_COUNT ~ ., data = train_data)

Coefficients:
          (Intercept)           TEMPERATURE              HUMIDITY            WIND_SPEED
               212.20                782.66               -886.73                 31.91
           VISIBILITY  DEW_POINT_TEMPERATURE      SOLAR_RADIATION              RAINFALL
                21.87                598.39                276.88              -2064.64
             SNOWFALL                   `0`                   `1`                   `10`
               260.97                -29.30               -116.85               -237.52
                  `11`                  `12`                  `13`                  `14`
              -247.38               -208.34               -191.35               -192.44
                  `15`                  `16`                  `17`                  `18`
              -109.73                 23.13                305.55                794.80
                  `19`                   `2`                   `20`                  `21`
```

Type here to search

11:58 AM
30/03/2023

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Go to file/function | Addins | Project: (None)

Ibm project.R | IBM PROJECT BULDING MODELS.R*

Source on Save | Run | Source

```
79
80
81  # Sort coefficient list
82  row_name = rownames(data.frame(lm_model_all$coefficients))
83  lm_all_abs_coefficient = data.frame(coefficient = c(abs(lm_model_all$coefficients)))
84  coefficient_data = cbind(row_name,lm_all_abs_coefficient)
85  row.names(coefficient_data)
86
87  coefficient_data = coefficient_data %>%
88    arrange(desc(coefficient))
89
90  head(coefficient_data)
91
92
93
94  # Visualize the list using ggplot and geom_bar
95  ggplot(coefficient_data) +
96    geom_col(aes(x = coefficient, y = reorder(row_name, coefficient, sum))) +
97    labs(title = "Absolute Coefficient of Linear Regression Model
98         (Using all variables)", x = "Coefficient Value", y = "Independent Variables")
```

94:1 (Top Level) R Script

R 4.2.1 · C:/Users/hp/OneDrive/Desktop/Data/IBM PROJECT/sql/

```
                              row_name coefficient
RAINFALL                      RAINFALL   2064.6375
HUMIDITY                      HUMIDITY    886.7303
`18`                             `18`    794.8032
TEMPERATURE                TEMPERATURE    782.6579
DEW_POINT_TEMPERATURE DEW_POINT_TEMPERATURE    598.3873
`19`                             `19`    522.9882
> # Visualize the list using ggplot and geom_bar
> ggplot(coefficient_data) +
+   geom_col(aes(x = coefficient, y = reorder(row_name, coefficient, sum))) +
+   labs(title = "Absolute Coefficient of Linear Regression Model
+        (Using all variables)", x = "Coefficient Value", y = "Independent Variables")
>
```

Environment | History | Connections | Tutorial

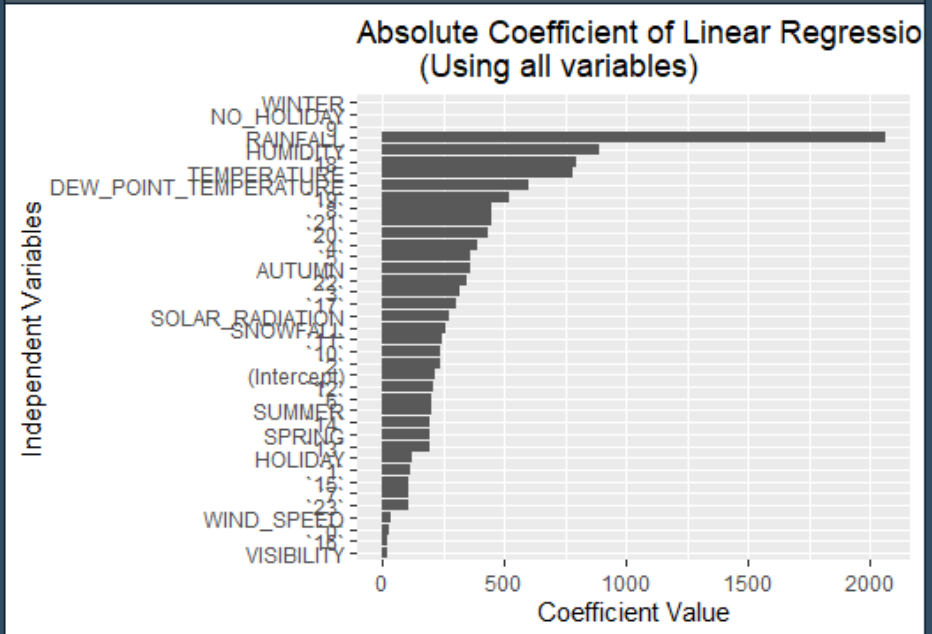
Import Dataset | 438 MiB | List

R | Global Environment

rsq_weather	1 obs. of 3 variables
test_data	2117 obs. of 39 variables
test_results_a...	2117 obs. of 2 variables
test_results_w...	2117 obs. of 2 variables
train_data	6348 obs. of 39 variables
Values	
dataset_url	"https://cf-courses-data.s3.us.cloud-obje...
row_name	chr [1:39] "(Intercept)" "TEMPERATURE" "H...

Files | Plots | Packages | Help | Viewer | Presentation

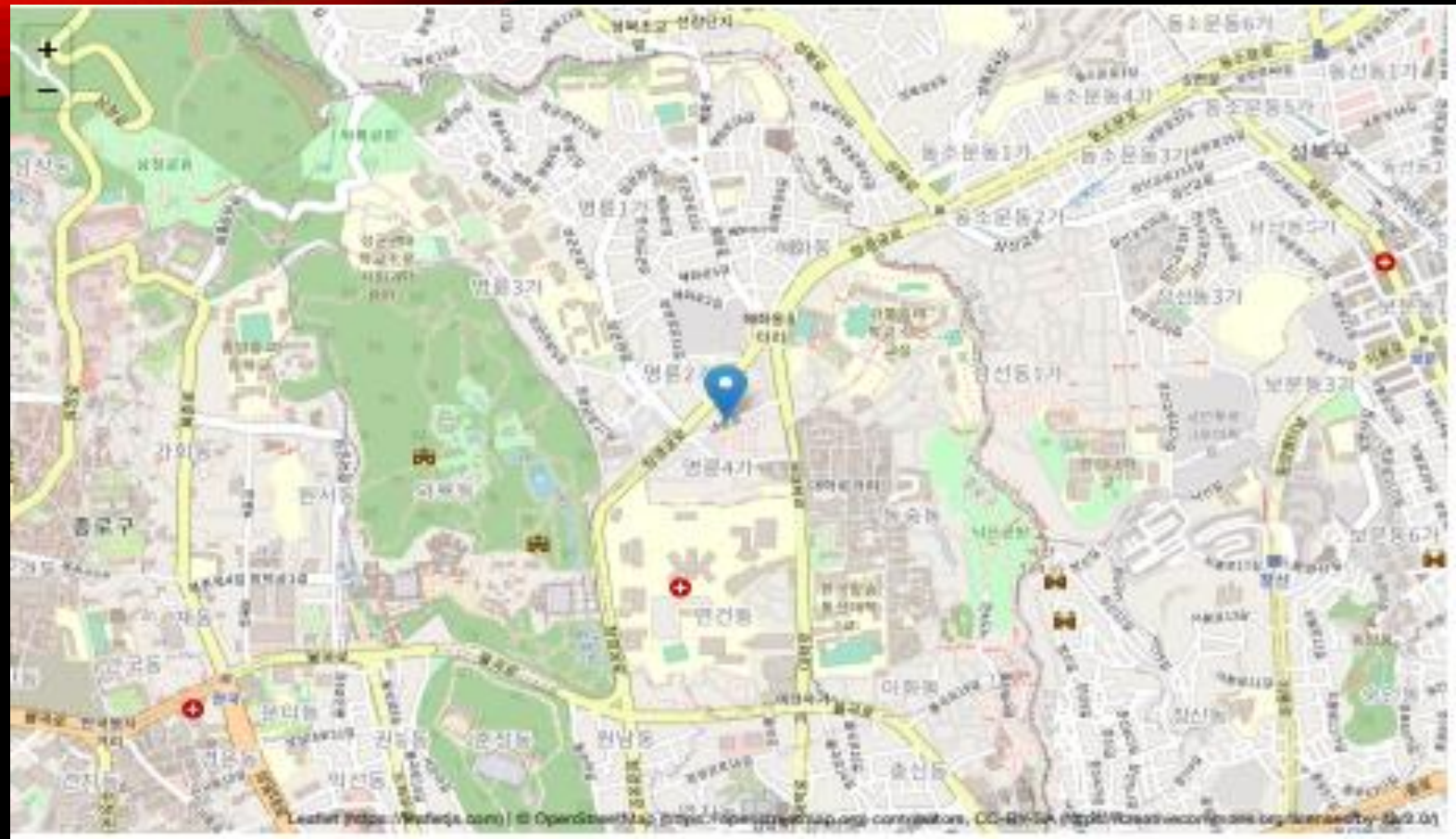
Zoom | Export | Publish

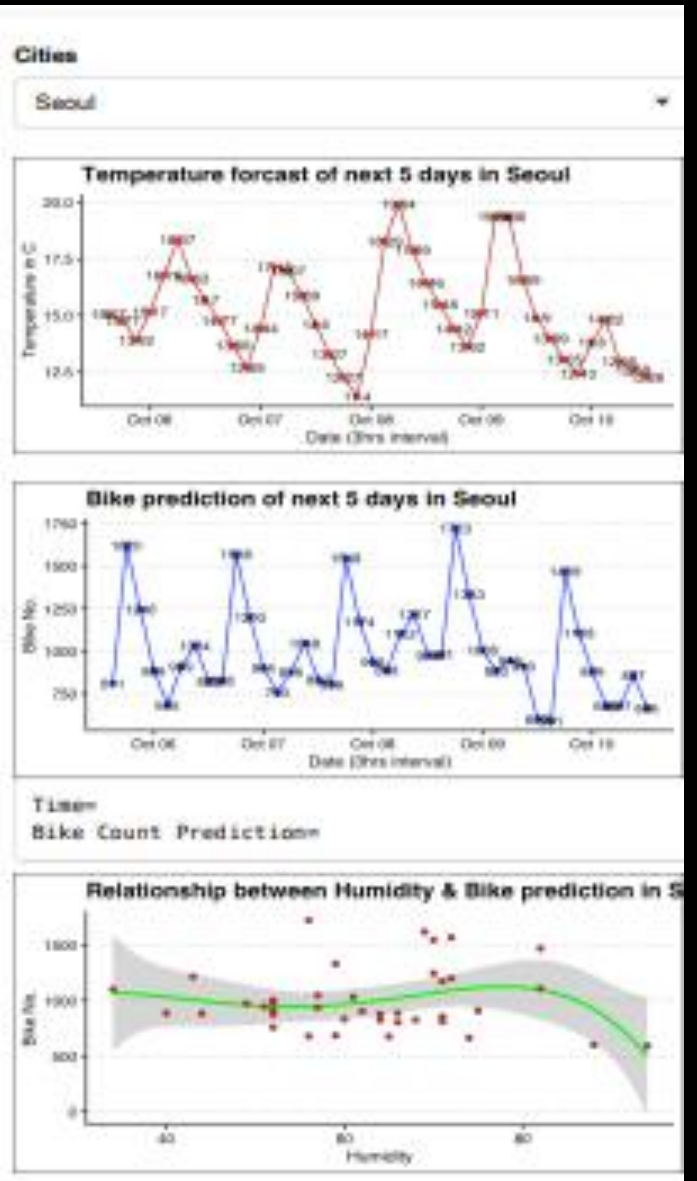


Type here to search

11:58 AM
30/03/2023

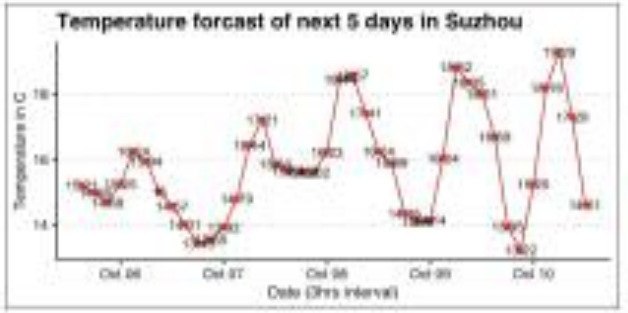
DASHBOARD




Cities
Seoul
Temperature forcast of next 5 days in Seoul
Bike prediction of next 5 days in Seoul
Time=
Bike Count Prediction=
Relationship between Humidity & Bike prediction in S

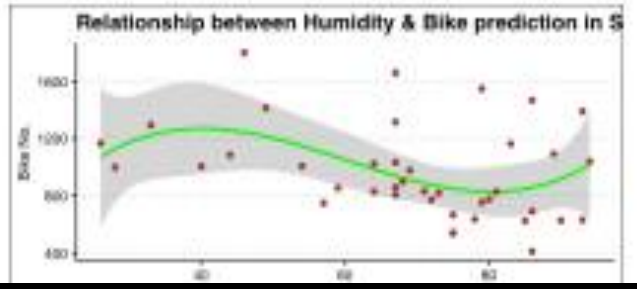
Cities

Suzhou



Time=

Bike Count Prediction=



CONCLUSION

- factors like humidity, season, and temperature have more influence on bike rentals than others.
- bike rentals took place the most in summer and autumn with winter following up behind.

WEB SCRAPING CODES

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

```
# Convert the bike-sharing system table into a dataframe
df_bike<-html_table(table)
```

Summarize the bike sharing system data frame

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

```
   Country              City                Name              System
 Length:549         Length:549         Length:549         Length:549
 Class :character   Class :character   Class :character   Class :character
 Mode  :character   Mode  :character   Mode  :character   Mode  :character
   Operator            Launched         Discontinued         Stations
 Length:549         Length:549         Length:549         Length:549
 Class :character   Class :character   Class :character   Class :character
 Mode  :character   Mode  :character   Mode  :character   Mode  :character
   Bicycles         Daily ridership
 Length:549         Length:549
 Class :character   Class :character
 Mode  :character   Mode  :character
```

API REQUEST GET CALLS CODES

```
library(httr)
```

```
Loading required package: httr
```

The API base URL to get current weather is https://api.openweathermap.org/data/2.5/weather

```
# URL for Current Weather API
current_weather_url <- 'https://api.openweathermap.org/data/2.5/weather'
```

Next, let's create a list to hold URL parameters for current weather API

```
# need to be replaced by your real API key
your_api_key <- "your_api_key"
# Input `q` is the city name
# Input `appid` is your API KEY,
# Input `units` are preferred units such as Metric or Imperial
current_query <- list(q = "Seoul", appid = "2ac1e71c6dde1ef14952e2ef0d1c331d", units="metric")
```

```
]: # $weather is also a list with one element, its $main elem
weather <- c(weather, json_result$weather[[1]]$main)
# Get Visibility
visibility <- c(visibility, json_result$visibility)
# Get current temperature
temp <- c(temp, json_result$main$temp)
# Get min temperature
temp_min <- c(temp_min, json_result$main$temp_min)
# Get max temperature
temp_max <- c(temp_max, json_result$main$temp_max)
# Get pressure
pressure <- c(pressure, json_result$main$pressure)
# Get humidity
humidity <- c(humidity, json_result$main$humidity)
# Get wind speed
wind_speed <- c(wind_speed, json_result$wind$speed)
# Get wind direction
```

```
81]: # Get forecast data for a given city list

get_weather_forecaset_by_cities <- function(city_names){
    df <- data.frame()
    for (city_name in city_names){
        # Forecast API URL
        forecast_url <- 'https://api.openweathermap.org/data/2.5/forecast'
        # Create query parameters
        forecast_query <- list(q = city_name, appid = "2ac1e71c6dde1ef14952e2ef0d1c331d", units="metric")
        # Make HTTP GET call for the given city
         response<-GET(forecast_url,query= forecast_query )
        # Note that the 5-day forecast JSON result is a list of lists. You can print the reponse to check the results
        json_list<-content(response,as="parsed" )
        results <- json_list$list

        # Loop the json result
        for(result in results) {
            city <- c(city,city_name)
             weather<-  c(weather, result$weather[[1]]$main)
               visibility<-  c(visibility,result$visibility)
               temp1<-c(temp,result$main$temp)
               tem_min1 <- c(temp_min,result$main$temp_min)
               tem_max1 <- c(temp_max,result$main$temp_max)
               pressure1<- c(pressure,result$main$pressure)
               humidity1<- c(humidity,result$main$humidity)
               wind_speed1<- c(wind_speed,result$main$wind$speed)
               wind_deg1<- c(wind_deg,result$main$wind$deg)
               forecast_datetime1<-c(forecast_datetime,result$dt_txt)
               season1<-c(season,result$main$season)

        }
             # Add the R Lists into a data frame
                                weather_data_frame<-data.frame(city=city,
                                            weather=weather,
                                            visibility=visibility,
                                            temp=temp,
                                            temp_min=temp_min,
                                            temp_max=temp_max,
                                            pressure=pressure,
                                            humidity=humidity,
                                            wind_speed=wind_speed,
                                            wind_deg=wind_deg,
```

DOWNLOADING DATA FROM CLOUD STORAGE CODES

```
# Download several datasets

# Download some general city information such as name and locations
url <- "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-RP0321EN-SkillsNetwor
# download the file
download.file(url, destfile = "raw_worldcities.csv")

# Download a specific hourly Seoul bike sharing demand dataset
url <- "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-RP0321EN-SkillsNetwor
# download the file
download.file(url, destfile = "raw_seoul_bike_sharing.csv")
```