Applied
Data Science
with
R Capstone project

RAMISETTY SAISRINIVAS

August 28th 2024

GITHUB URL

https://github.com/SR000777/R-Data-Science-Capstone-Project.git

Outline

Executive Summary

- Collect relevant data from global bike sharing systems on public web pages, weather data via Open Weather APIs, and aggregated tabular data from cloud storage
- Data wrangling with string r and regular expressions and dply r to remove the noise from data and convert the undesired data forma
- We seek to examine bike-sharing data, joined with daily Seoul, Suzhou, London, New York, and Paris weather data, to study the impact of weather on shared bike usage and generate a predictive model which can estimate the number of trips that would be taken on each day.
- Exploratory Data Analysis with SQL and using an R notebook to perform exploratory data analysis using tidy verse and the ggplot2 R pack
- Predicting Hourly Rented Bike Count using Basic Linear Regression Models and refining the Baseline Regression Models.
- Building a bike-sharing demand prediction app with R Shiny and Leaflet which is enhancing the Bike-Sharing Demand Prediction App with City Details Plots.

Introduction

- The purpose of this project analysis on bike-sharing demand and weather data collection is to help better understand the process of building a bike- sharing demand prediction app with R Shiny and Leaflet.

- In the initial task of collecting data, used web scrape a Global Bike-Sharing Systems Wiki Page and Open Weather APIs Calls to aggregate tabular data from cloud storage.

- In the face of the data sorting and classification task of the original dataset with missing data, Regular Expressions and dply r are used to help with data wrangling.

- Performing Exploratory Data Analysis with SQL, Tidyverse & ggplot2 to help solve exploratory problems with datasets

- In order to solve the demand forecasting problem of shared bicycles, building a baseline Regression Model and improvements for forecasting

- For interactive R Shiny dashboard to be able to visualize and predict, build a bike-sharing demand prediction app with R Shiny and Leaflet to show the max predicted bike-sharing demand in the next 5 days

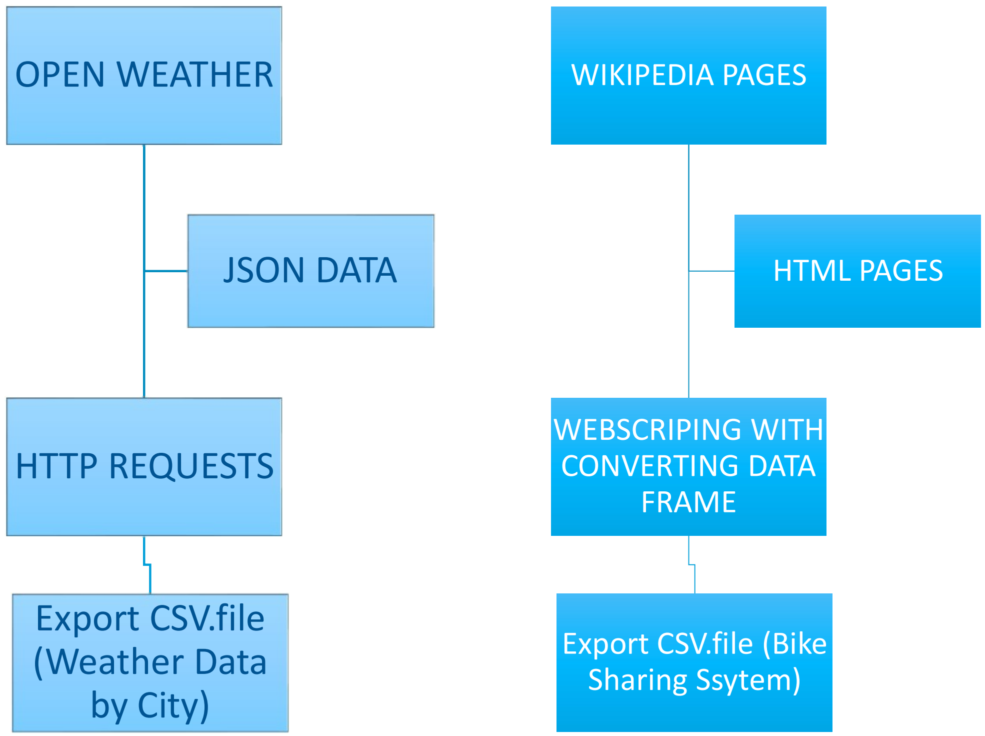
Methodology

- Data collection
- Data wrangling
- Performing Exploratory Data Analysis with SQL, Tidyverse & ggplot2
- Predictive analysis by using Regression Models and improvements
- Build a bike-sharing demand prediction app with R Shiny and Leaflet

Methodology

Data collection

- Two ways of collecting relevant data from various sources:
- The use of HTTP requests to collect the JSON data of Open Weather
- Using web scraping to collect the HTML pages from Wikipedia web



GitHub URL:
https://github.com/SR000777/R-Data-Science-Capstone-Project.git

Data wrangling

- We have collected required bike-sharing data for further exploratory, visual, and predictive analysis tasks. However, some data may contain missing, mis formatted and/or unexpected noises. Such sources of noise may downgrade the analysis performance significantly. Thus, we need to perform data wrangling before further analyzing the data.
- Data wrangling aims to remove the noise from data and convert the undesired data format to a format that is likely to be better for analysis.
  - Data wrangling with stringr and regular expressions
    - Standardize column names for all collected datasets
    - Remove undesired reference links using regular expressions
    - Extract numeric values using regular expressions
  - Lab: Data wrangling with dplyr
    - Detect and handle missing values
    - Create indicator (dummy) variables for categorical variables
    - Normalize data

GitHub URL:

https://github.com/SR000777/R-Data-Science-Capstone-Project.git

EDA with SQL

Using SQL queries with the RODBC R package by establishing Db2 connection firstly:

- Determine how many records are in the seoul_bike_sharing dataset
- For how many operational hours had non-zero rented bike count
- Query the 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 which date and hour had the most bike rentals
- The top ten average bike counts can be used to calculate the average hourly temperature and the average number of bike rentals per hour throughout each season
- Find the average hourly bike count during each season
- Consider the weather over each season
- Determine the total number of bikes available

GitHub URL:

https://github.com/SR000777/R-Data-Science-Capstone-Project/blob/2db285b785c7bb470252c6d3b95927aa6be54944/Lab-SQL-EDA.ipynb

EDA with SQL

Task 1 - Record Count

Determine how many records are in the seoul_bike_sharing dataset.

Solution 1

In [2]:
```
# provide your solution here
# Load the dataset
seoul_bike_sharing <- read.csv("seoul_bike_sharing.csv")

# Determine the number of records
num_records <- nrow(seoul_bike_sharing)

# Print the number of records
print(num_records)
```

```
[1] 8465
```

Task 3 - Weather Outlook

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.

Solution 3

In [4]:
```
# provide your solution here
# Load the dataset
# Read the CSV file
cities_weather_forecast <- read.csv("cities_weather_forecast.csv")
cities_weather_forecast [1, ]
```

A data.frame: 1 × 3

	City	Temperature	Condition
	<fct>	<dbl>	<fct>
1	Seoul	30.10879	Cloudy

Task 4 - Seasons

Find which seasons are included in the seoul bike sharing dataset.

Solution 4

In [5]:
```
# Read the CSV file
seoul_bike_sharing <- read.csv("seoul_bike_sharing.csv")

# Identify duplicated values in the 'SEASONS' column
duplicated_seasons <- seoul_bike_sharingSEASONS[duplicated(seoulbike,sharingSEASONS)]

# Remove duplicates to get unique values
unique_seasons <- unique(seoul_bike_sharing$SEASONS)

# Print the unique values in a table format
print(unique_seasons)
```

```
[1] Winter Spring Summer Autumn
Levels: Autumn Spring Summer Winter
```

Task 5 - Date Range

Find the first and last dates in the Seoul Bike Sharing dataset.

Solution 5

In [6]:
```
# Read the CSV file
seoul_bike_sharing <- read.csv("seoul_bike_sharing.csv")

# Convert the DATE column to Date type with the appropriate format
seoul_bike_sharing$DATE <- as.Date(seoul_bike_sharing$DATE, format="%d/%m/%Y")

# Find the first and last date
first_date <- min(seoul_bike_sharing$DATE, na.rm=TRUE)
last_date <- max(seoul_bike_sharing$DATE, na.rm=TRUE)

# Print the results
print(paste("First date:", first_date))
print(paste("Last date:", last_date))
```

```
[1] "First date: 2017-12-01"
[1] "Last date: 2018-11-30"
```

EDA with data visualization

Using SQL with tidyverse and the ggplot2 R packages:

• Load the dataset; Recast DATE; Cast HOURS as a categorical variable;

For Descriptive Statistics:

• Use dataset summary to describe the seoul_bike_sharing dataset; calculate how many holidays there are; calculate the percentage of records that fall on a holiday; Given there is exactly a full year of data, determine how many records we expect to have; Given the observations of how many records must there be;

For drilling down:

• Calculate the seasonal total rainfall and snowfall

For data visluzation:

• 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 colour; Create a histogram overlaid with a kernel density curve; Use a scatter plot to visualize the correlation between RENTED_BIKE_COUNT and TEMPERATURE; Create a display of four boxplots of RENTED_BIKE_COUNT vs. HOUR grouped by SEASONS; Group the data by DATE and calculate the daily total rainfall and snowfall; Determine how many days had snowfall

GitHub URL:

https://github.com/SR000777/R-Data-Science-Capstone-Project/blob/2db285b785c7bb470252c6d3b95927aa6be54944/Lab-ggplot2-EDA.ipynb

EDA with data visualization

Task 5 - Based on the above stats, calculate how many Holidays there are.

Solution 5:

In [8]:
```r
# provide your solution here
holiday_count <- sum(seoul_bike_sharing$HOLIDAY == "Holiday")
holidays <- (holiday_count/24)
holidays
```

17

Task 6 - Calculate the percentage of records that fall on a holiday.

Solution 6

In [9]:
```r
# provide your solution here
total_count <- nrow(seoul_bike_sharing)
holiday_percentage <- (holiday_count / total_count) * 100
holiday_percentage
```

4.8198464264619

Task 8 - Given the observations for the 'FUNCTIONING_DAY' how many records must there be?

Solution 8

In [11]:
```r
# provide your solution here
functioning_day_count <- sum(seoul_bike_sharing$FUNCTIONING_DAY == "Yes")
functioning_day_count
```

8465

Task 9 - Load the dplyr package, group the data by `SEASONS`, and use the `summarize()` function to calculate the seasonal total rainfall and snowfall.

Solution 9

In [12]:
```r
total_rainfall_snowfall <- aggregate(cbind(RAINFALL, SNOWFALL) ~ SEASONS, data = seoul_bike_sharing, FUN = sum)

# Rename the columns for clarity
names(total_rainfall_snowfall)[names(total_rainfall_snowfall) == "RAINFALL"] <- "total_rainfall"
names(total_rainfall_snowfall)[names(total_rainfall_snowfall) == "SNOWFALL"] <- "total_snowfall"

total_rainfall_snowfall
```

A data.frame: 4 × 3

SEASONS	total_rainfall	total_snowfall
<fct>	<dbl>	<dbl>
Autumn	227.9	123.0
Spring	403.8	0.0
Summer	559.7	0.0
Winter	70.9	534.6

Task 15 - Group the data by `DATE`, and use the summarize() function to calculate the daily total rainfall and snowfall.

Also, go ahead and plot the results if you wish.

Solution 15

In [23]:
```r
# Read the CSV file
seoul_bike_sharing <- read.csv("seoul_bike_sharing.csv")

# Summarize the data
daily_rain_snow <- summarize(
  group_by(seoul_bike_sharing, DATE),
  TOTAL_RAINFALL = sum(RAINFALL, na.rm = TRUE),
  TOTAL_SNOWFALL = sum(SNOWFALL, na.rm = TRUE)
)

# View the first few rows of the summarized data
head(daily_rain_snow)
```

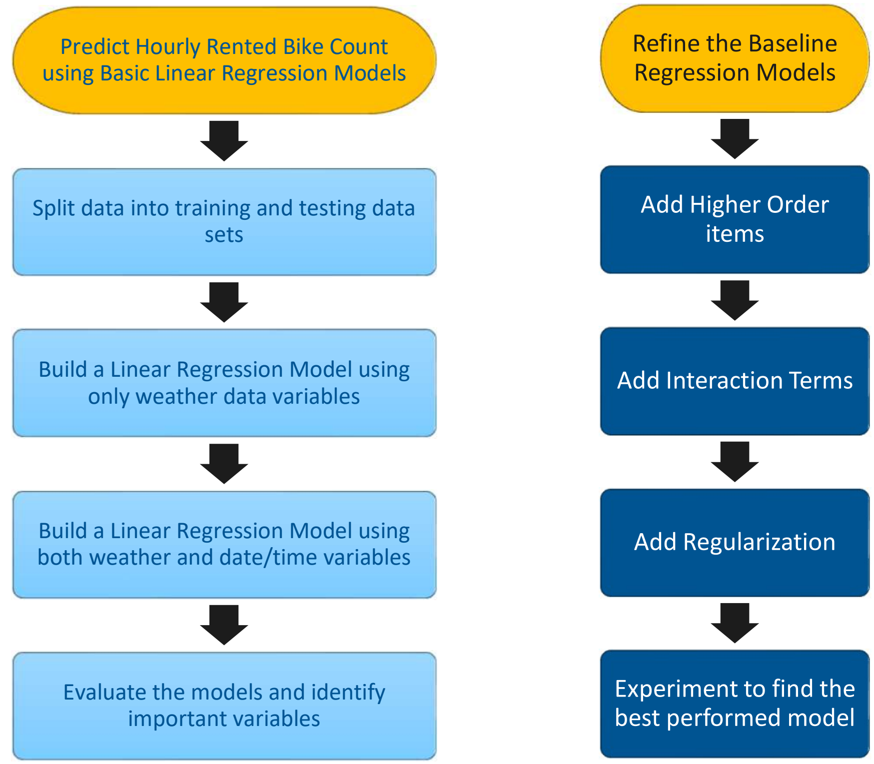
A tibble: 6 × 3

DATE	TOTAL_RAINFALL	TOTAL_SNOWFALL
<fct>	<dbl>	<dbl>
01/01/2018	0.0	0.0
01/02/2018	0.0	21.7
01/03/2018	2.5	0.0
01/04/2018	0.0	0.0
01/05/2018	0.0	0.0
01/06/2018	0.0	0.0

Predictive analysis

The ways of predicting bike-sharing demand using regression models and improvements:

- Build basic linear regression models to predict the hourly rented bike count using related weather and date information
- Use methods like adding polynomial and interaction terms to refine the baseline regression models



GitHub URL:
https://github.com/SR000777/R-Data-Science-Capstone-Project/blob/2db285b785c7bb470252c6d3b95927aa6be54944/lab-jupyter-linear-models-baselinse.ipynb

Build a R Shiny dashboard

- Build an interactive R Shiny dashboard to be able to visualize weather forecast data and predicted hourly bike-sharing demand for the following cities, New York, USA, Paris, France, Suzhou, China, and London, UK.
- Leaflet-based interactive map that shows the max predicted bike-sharing demand in the next 5 days
- Built a R Shiny app with leaflet to show the max bike-sharing demand predictions for each city:
- Use ggplot to render some more detailed plots such as bike-sharing prediction trend, temperature trend, humidity and bike-sharing demand prediction correlation, when users zoom-in to a city

GitHub URL:

https://github.com/SR000777/R-Data-Science-Capstone-Project.git

Results

- Results indicate all variable related model performs better than weather-based model.
- A non-linear relationship between bikes rented and temperature.
- Temperature and humidity are the strongest indicators.
- Analysis shows Autumn: 357.978, Spring: 194.42, and Summer: 172.901 as the strongest date predictors in the all-variable model.
- All weather-based model has a higher R squared and RMSE compared to all variables-model.
- The Shiny app visualizes bike-sharing demand prediction for three cities: Seoul, Suzhou, London, New York, and Paris. The app contains three types of plots for each city namely Temperature Trend, Bike-sharing Demand Prediction Trend, Humidity vs. Bike-sharing Demand Prediction

EDA with SQL

Busiest bike rental times

Task 6 - Subquery - 'all-time high'

determine which date and hour had the most bike rentals.

Solution 6

In [7]:

```
# Read the CSV file into a data frame
seoul_bike_sharing <- read.csv("seoul_bike_sharing.csv")

# Find the maximum value in the RENTED_BIKE_COUNT column
max_rented_bike_count <- max(seoul_bike_sharing$RENTED_BIKE_COUNT, na.rm = TRUE)

# Print the result (if needed)
print(max_rented_bike_count)
```

```
[1] 3556
```

- The all time high most bike rentals happened is “3556”

GitHub URL:
https://github.com/SR000777/R-Data-Science-Capstone-Project/blob/2db285b785c7bb470252c6d3b95927aa6be54944/Lab-SQL-EDA.ipynb

Hourly popularity and temperature by seasons

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

In [8]:

```
# Read the CSV file
seoul_bike_sharing <- read.csv("seoul_bike_sharing.csv")

# Compute the mean of TEMPERATURE and RENTED_BIKE_COUNT by SEASONS
result <- aggregate(cbind(TEMPERATURE, RENTED_BIKE_COUNT) ~ SEASONS,
                    data = seoul_bike_sharing,
                    FUN = mean)

# Print the result
print(result)
```

```
  SEASONS TEMPERATURE RENTED_BIKE_COUNT
1  Autumn   13.821580          924.1105
2  Spring   13.021685          746.2542
3  Summer   26.587711         1034.0734
4  Winter   -2.540463          225.5412
```

GitHub URL:
https://github.com/SR000777/R-Data-Science-Capstone-Project/blob/2db285b785c7bb470252c6d3b95927aa6be54944/Lab-SQL-EDA.ipynb

Rental Seasonality

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.

Solution 8

In [9]:

```r
# Read the CSV file
seoul_bike_sharing <- read.csv("seoul_bike_sharing.csv")

# Average hourly bike count
avg_bike_count <- aggregate(RENTED_BIKE_COUNT ~ SEASONS,
                            data = seoul_bike_sharing,
                            FUN = mean)

# Minimum hourly bike count
min_bike_count <- aggregate(RENTED_BIKE_COUNT ~ SEASONS,
                            data = seoul_bike_sharing,
                            FUN = min)

# Maximum hourly bike count
max_bike_count <- aggregate(RENTED_BIKE_COUNT ~ SEASONS,
                            data = seoul_bike_sharing,
                            FUN = max)

# Standard deviation of hourly bike count
sd_bike_count <- aggregate(RENTED_BIKE_COUNT ~ SEASONS,
                           data = seoul_bike_sharing,
                           FUN = sd)

# Merge all results into one data frame
result <- merge(avg_bike_count, min_bike_count, by = "SEASONS", suffixes = c("_mean", "_min"))
result <- merge(result, max_bike_count, by = "SEASONS")
result <- merge(result, sd_bike_count, by = "SEASONS")

# Rename the columns for clarity
colnames(result) <- c("SEASONS", "Average_Bike_Count", "Min_Bike_Count", "Max_Bike_Count", "SD_Bike_Count")

# Print the result
print(result)
```

	SEASONS	Average_Bike_Count	Min_Bike_Count	Max_Bike_Count	SD_Bike_Count
1	Autumn	924.1105	2	3298	617.5479
2	Spring	746.2542	2	3251	618.6680
3	Summer	1034.0734	9	3556	690.2448
4	Winter	225.5412	3	937	150.3722

- The left screenshot shows Rental Seasonality for the four seasons namely
  - Autumn
  - Spring
  - Summer
  - Winter

Weather Seasonality

- The right side screenshot shows the Weather Seasonality

- This provides the result season wise with average temperature, humidity, windspeed, visibility, solar radiation etc.

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

In [10]:

```r
# Read the CSV file
seoul_bike_sharing <- read.csv("seoul_bike_sharing.csv")

# Compute the average of various columns by SEASONS
result <- aggregate(cbind(TEMPERATURE, HUMIDITY, WIND_SPEED, VISIBILITY,
                          DEW_POINT_TEMPERATURE, SOLAR_RADIATION, RAINFALL,
                          SNOWFALL, RENTED_BIKE_COUNT) ~ SEASONS,
                    data = seoul_bike_sharing,
                    FUN = mean)

# Rename columns for clarity
colnames(result) <- c("SEASONS", "Average_Temperature", "Average_Humidity",
                      "Average_Wind_Speed", "Average_Visibility",
                      "Average_Dew_Point_Temperature", "Average_Solar_Radiation",
                      "Average_Rainfall", "Average_Snowfall", "Average_Bike_Count")

# Rank the results by Average_Bike_Count
result <- result[order(-result$Average_Bike_Count), ]

# Print the result
print(result)
```

```
  SEASONS Average_Temperature Average_Humidity Average_Wind_Speed
3  Summer           26.587711         64.98143           1.609420
1  Autumn           13.821580         59.04491           1.492101
2  Spring           13.021685         58.75833           1.857778
4  Winter           -2.540463         49.74491           1.922685
  Average_Visibility Average_Dew_Point_Temperature Average_Solar_Radiation
3           1501.745                     18.750136               0.7612545
1           1558.174                      5.150594               0.5227827
2           1240.912                      4.091389               0.6803009
4           1445.987                    -12.416667               0.2981806
  Average_Rainfall Average_Snowfall Average_Bike_Count
3       0.25348732       0.00000000          1034.0734
1       0.11765617       0.06350026           924.1105
2       0.18694444       0.00000000           746.2542
4       0.03282407       0.24750000           225.5412
```

Bike-sharing info in 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

In [11]:

```
# Load necessary libraries
# No libraries required for base R approach

# Read the CSV files
world_cities <- read.csv("raw_worldcities.csv")
bike_sharing_systems <- read.csv("bike_sharing_systems.csv")

# Merge the two datasets on the CITY column
merged_data <- merge(world_cities, bike_sharing_systems, all.x = TRUE)

# Filter the merged data for Seoul
seoul_data <- merged_data[merged_data$CITY == "Seoul", ]

# Calculate the total number of bikes available in Seoul
total_bikes <- sum(seoul_data$TOTAL_BIKES, na.rm = TRUE)

# Select relevant columns and add total bikes to the data
result <- seoul_data[, c("CITY", "COUNTRY", "LAT", "LNG", "POPULATION")]

# Print the result
print(result)
```

```
       CITY      COUNTRY     LAT LNG POPULATION
21003 Seoul South Korea 37.5833 127   21794000
```

- This shows the city bike count, country name, latitude, longitude and population info relating to seoul city

Cities similar to Seoul

- The cities which are similar to seoul bike sharing system are
  - Beijing
  - Ningbo
  - Shanghai
  - Weifang
  - Zhuzhou

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

In [12]:

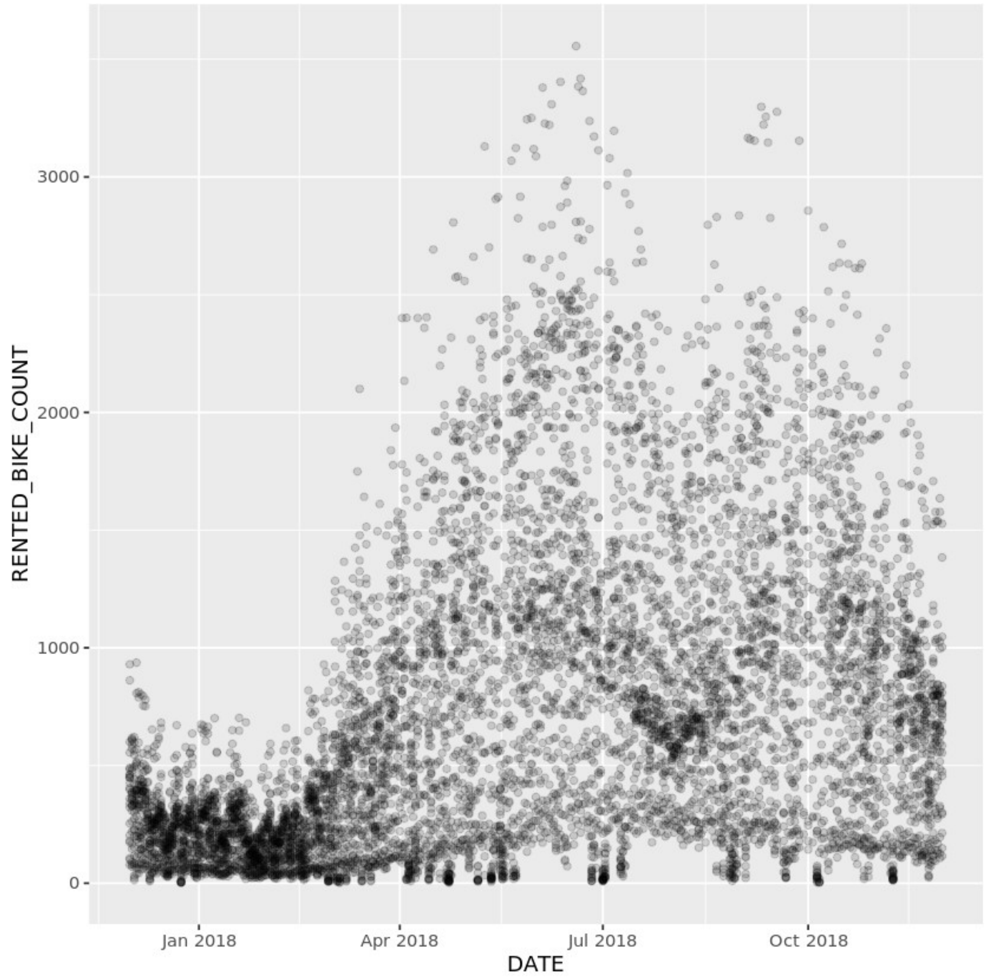
```r
# Read the CSV files
world_cities <- read.csv("raw_worldcities.csv")
bike_sharing_systems <- read.csv("bike_sharing_systems.csv")

# Merge the dataframes
df_joined <- merge(bike_sharing_systems, world_cities)

# Select relevant columns
result <- df_joined[, c("CITY", "COUNTRY", "LAT", "LNG", "POPULATION", "BICYCLES")]


# Filter rows based on 'BICYCLES' values
data <- result[!is.na(result["BICYCLES"]) & (df_joined["BICYCLES"] >= 15000) & (result["BICYCLES"] <= 20000), ]


# Print the result
print(data)
```

```
        CITY     COUNTRY     LAT      LNG POPULATION BICYCLES
32   Beijing       China 39.9050 116.3914   19433000    16000
60    Ningbo       China 29.8750 121.5492    7639000    15000
62  Shanghai       China 31.1667 121.4667   22120000    19165
67   Weifang       China 36.7167 119.1000    9373000    20000
76   Zhuzhou       China 27.8407 113.1469    3855609    20000
271    Seoul South Korea 37.5833 127.0000   21794000    20000
```

EDA with Visualization

Bike rental vs. Date

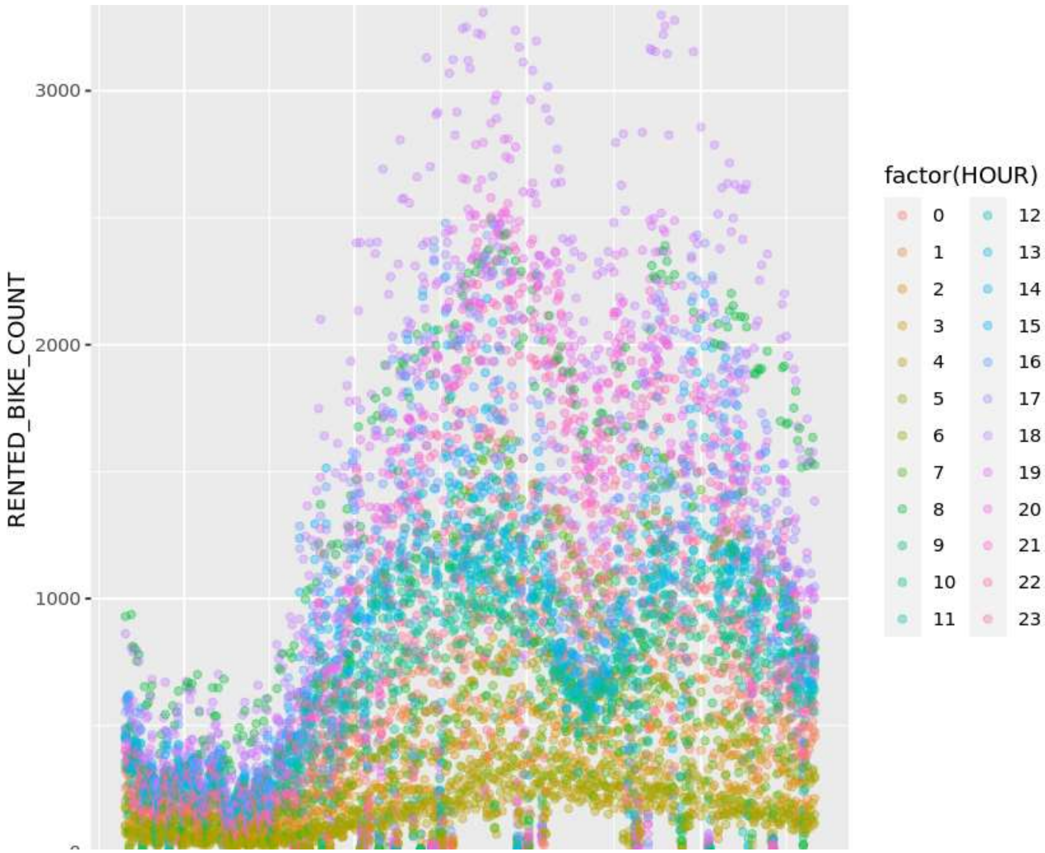
- The right side image show a scatter plot of RENTED_BIKE_COUNT vs. DATE.
- It shows the relationship between the rented bike count and the date



GitHub URL:
https://github.com/SR000777/R-Data-Science-Capstone-Project/blob/2db285b785c7bb470252c6d3b95927aa6be54944/Lab-ggplot2-EDA.ipynb

Bike rental vs. Datetime

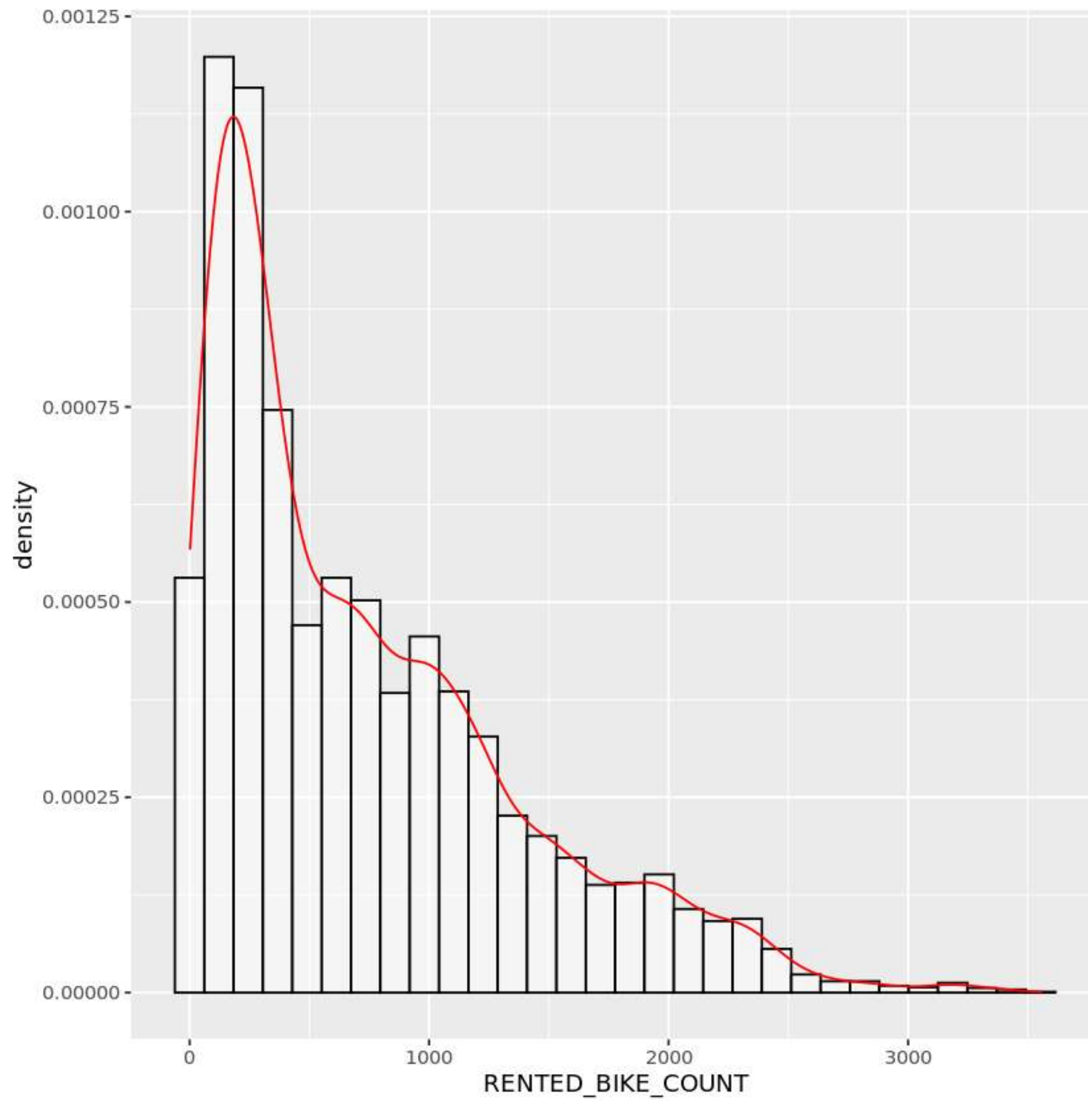
- The image shows the same plot of the RENTED_BIKE_COUNT time series, but now add HOURS as the colour



GitHub URL:
https://github.com/SR000777/R-Data-Science-Capstone-Project/blob/2db285b785c7bb470252c6d3b95927aa6be54944/Lab-ggplot2-EDA.ipynb

Bike rental histogram

- The image shows a histogram overlaid with a kernel density curve.
- It shows the relationship between rented bike count and the density



Daily total rainfall and snowfall

Task 15 - Group the data by `DATE`, and use the summarize() function to calculate the daily total rainfall and snowfall.

Also, go ahead and plot the results if you wish.

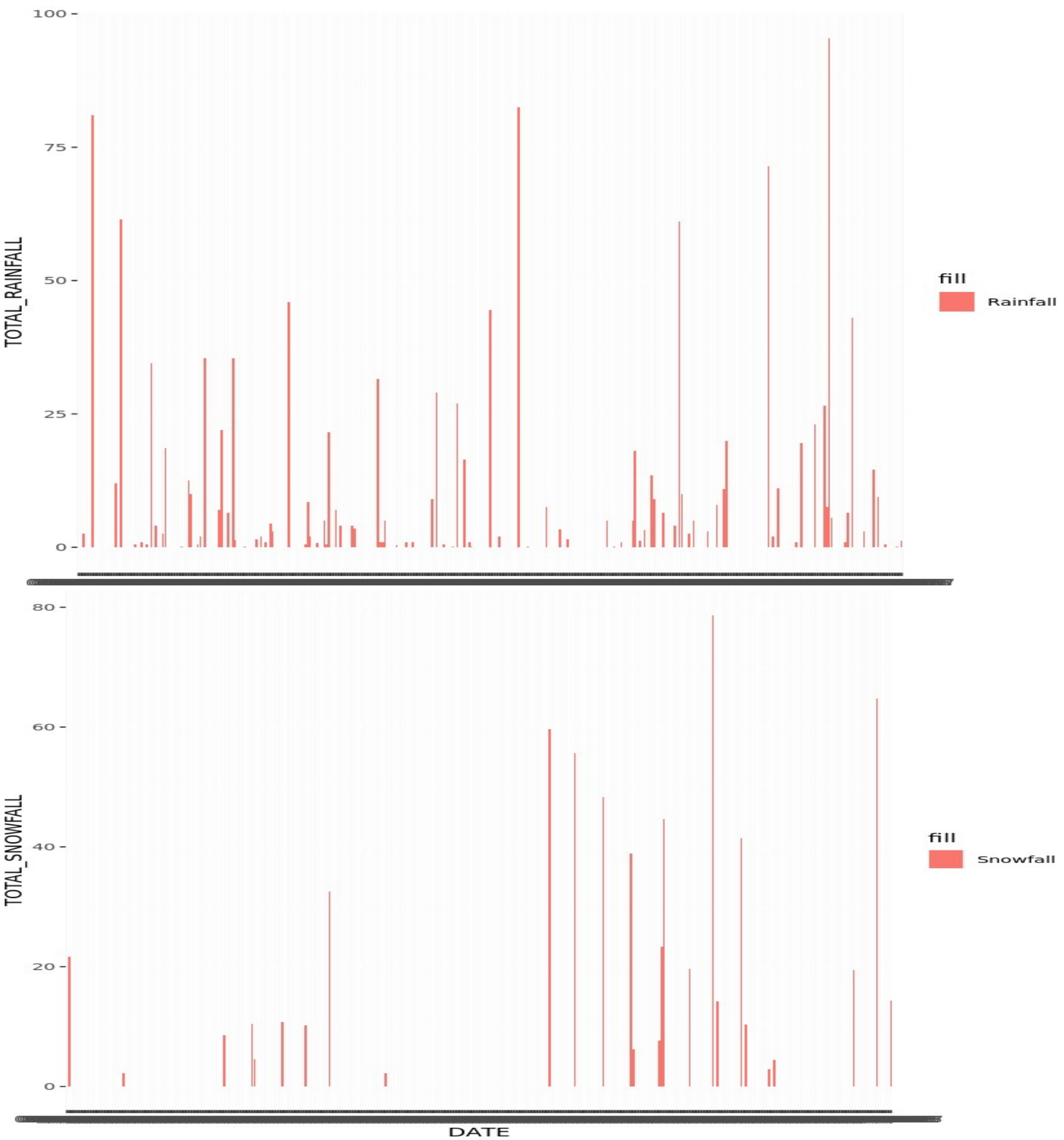
Solution 15

In [23]:

```
# Read the CSV file
seoul_bike_sharing <- read.csv("seoul_bike_sharing.csv")

# Summarize the data
daily_rain_snow <- summarize(
  group_by(seoul_bike_sharing, DATE),
  TOTAL_RAINFALL = sum(RAINFALL, na.rm = TRUE),
  TOTAL_SNOWFALL = sum(SNOWFALL, na.rm = TRUE)
)

# View the first few rows of the summarized data
head(daily_rain_snow)
```

A tibble: 6 × 3

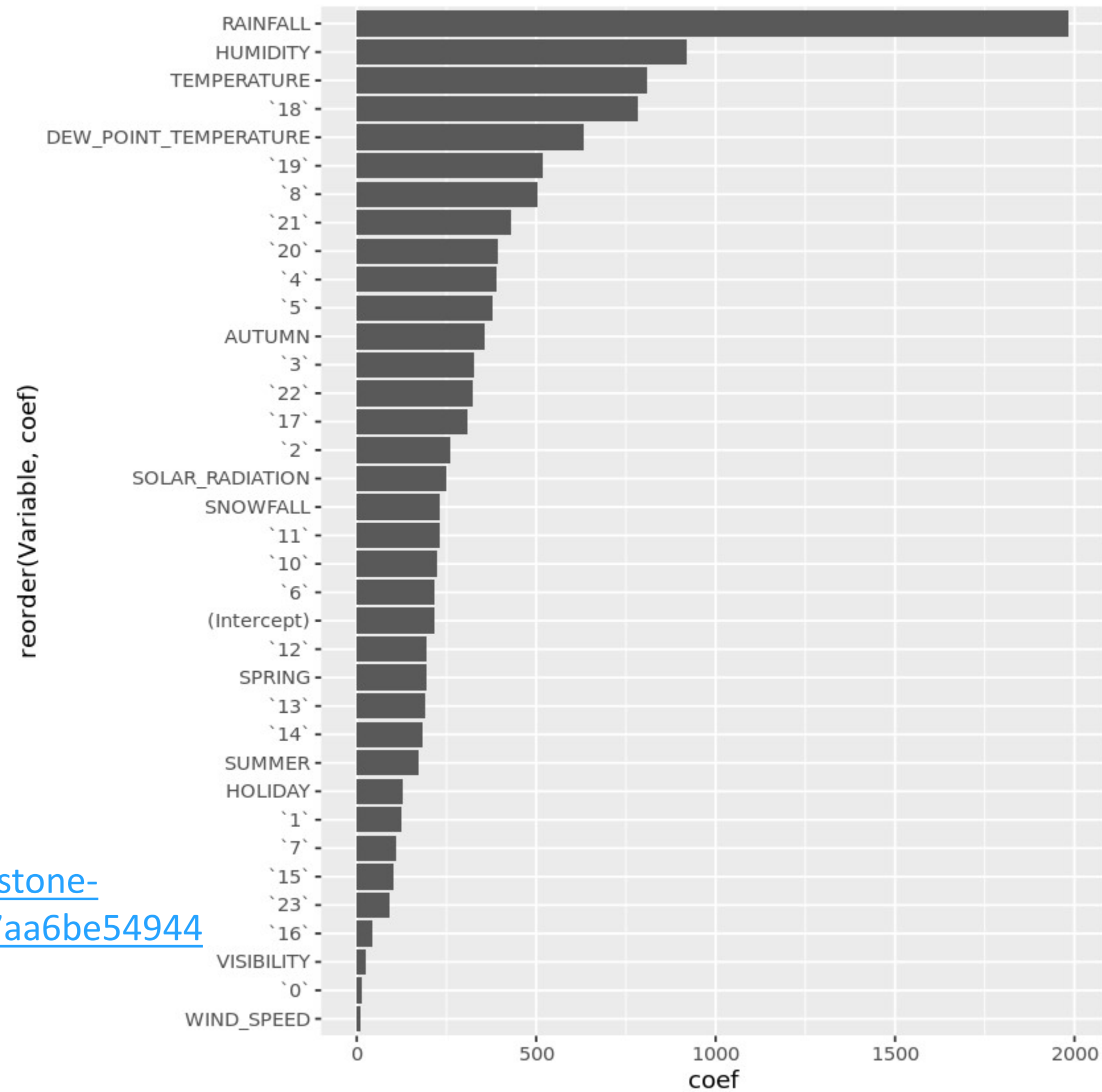
DATE	TOTAL_RAINFALL	TOTAL_SNOWFALL
<fct>	<dbl>	<dbl>
01/01/2018	0.0	0.0
01/02/2018	0.0	21.7
01/03/2018	2.5	0.0
01/04/2018	0.0	0.0
01/05/2018	0.0	0.0
01/06/2018	0.0	0.0



Predictive analysis

Ranked coefficients

- The image shows the bar chart which ranked coefficients in the descending order.

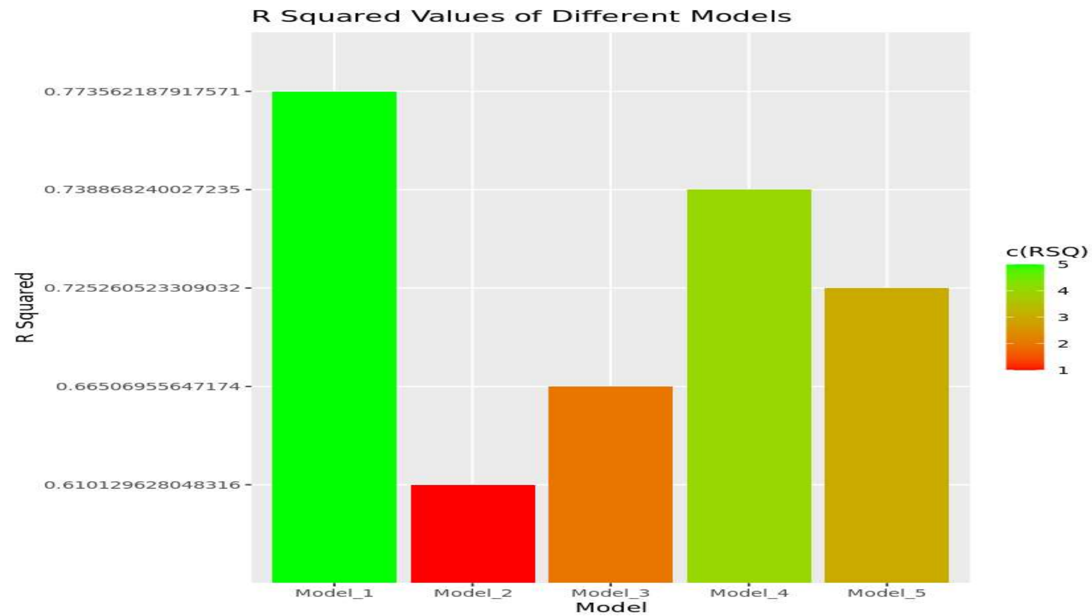


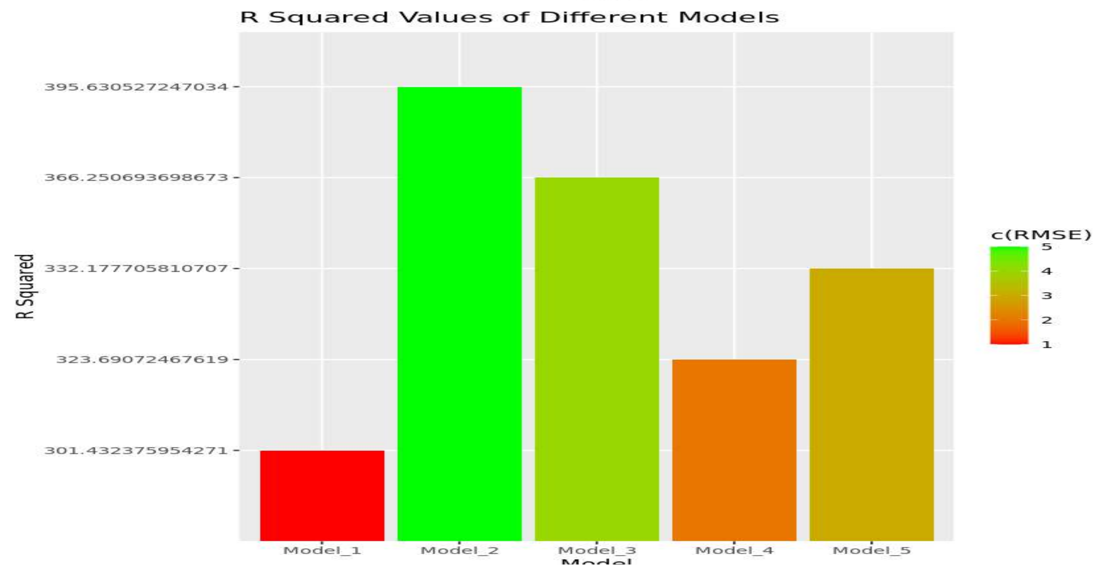
GitHub URL:
https://github.com/SR000777/R-Data-Science-Capstone-Project/blob/2db285b785c7bb470252c6d3b95927aa6be54944/lab-jupyter-linear-models-baselinse.ipynb

Model evaluation

- The bar chart shows 5 different models RMSE and R-squared using grouped bar chart
- Model 1 has high RSQ where as Model 2 has high RMSE

GitHub URL:
https://github.com/SR000777/R-Data-Science-Capstone-Project/blob/2db285b785c7bb470252c6d3b95927aa6be54944/lab-jupyter-linear-models-refinments.ipynb





Find the best performing model

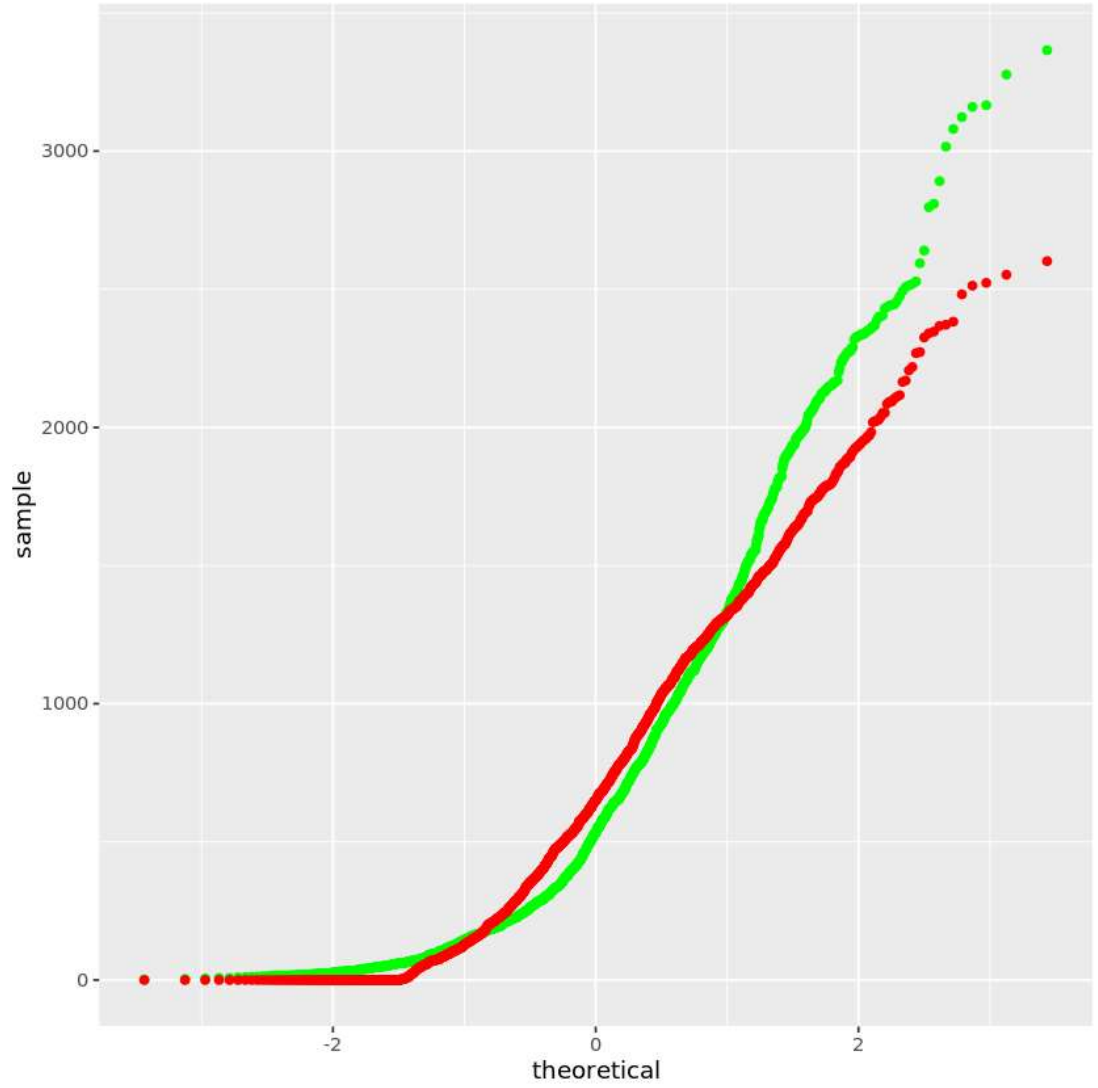
```
In [26]: #model 4

model_4_recipe <- glmnet_spec %>% fit(RENTED_BIKE_COUNT ~ RAINFALL*HUMIDITY*TEMPERATURE  +
                                    `18`*`19`*`8`*`21`*`20`*`4`*`5`*SPRING*SUMMER*AUTUMN*HOLIDAY +
                                      poly(RAINFALL, 8) + poly(HUMIDITY, 5) + poly(TEMPERATURE, 5) +
                                      poly(DEW_POINT_TEMPERATURE, 5) + poly(SOLAR_RADIATION, 5) +
                                      poly(SNOWFALL, 5) + SPRING  + AUTUMN + SUMMER + HOLIDAY +
                                      `18` + `19` + `8` + `21` + `20`+ `0` + `23` + `4` + `5` + WIND_SPEED +
                                      VISIBILITY, data = train_data)

model_4 <- model_prediction(model_4_recipe, test_data)
model_evaluation(model_4)
model_4_rmse <- rmse(model_4, truth = truth, estimate = .pred)
model_4_rsq <- rsq(model_4, truth = truth, estimate = .pred)
```

```
# A tibble: 1 x 3
  .metric .estimator .estimate
  <chr>   <chr>          <dbl>
1 rsq     standard       0.739
# A tibble: 1 x 3
  .metric .estimator .estimate
  <chr>   <chr>          <dbl>
1 rmse    standard        324.
```

- The criteria to select the best performing model is model with:
  - RMSE must be less than 330
  - R-squared must be larger than 0.72
  - Shown a screenshot of the model performance
- The best performing model in terms of above criteria is model 4.

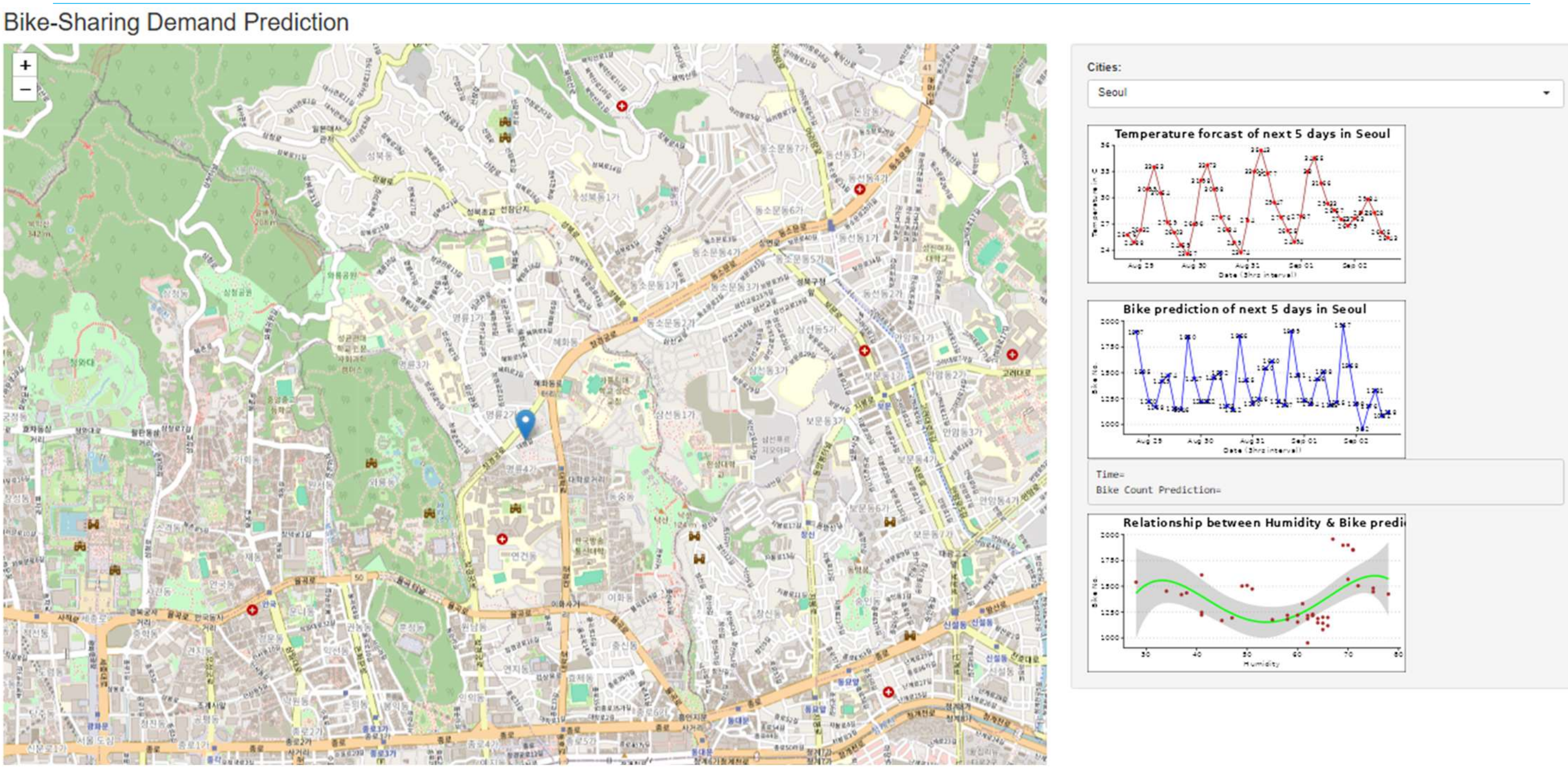
Q-Q plot of the best model

- The image shows the Q-Q plot of the best model i.e. model 4

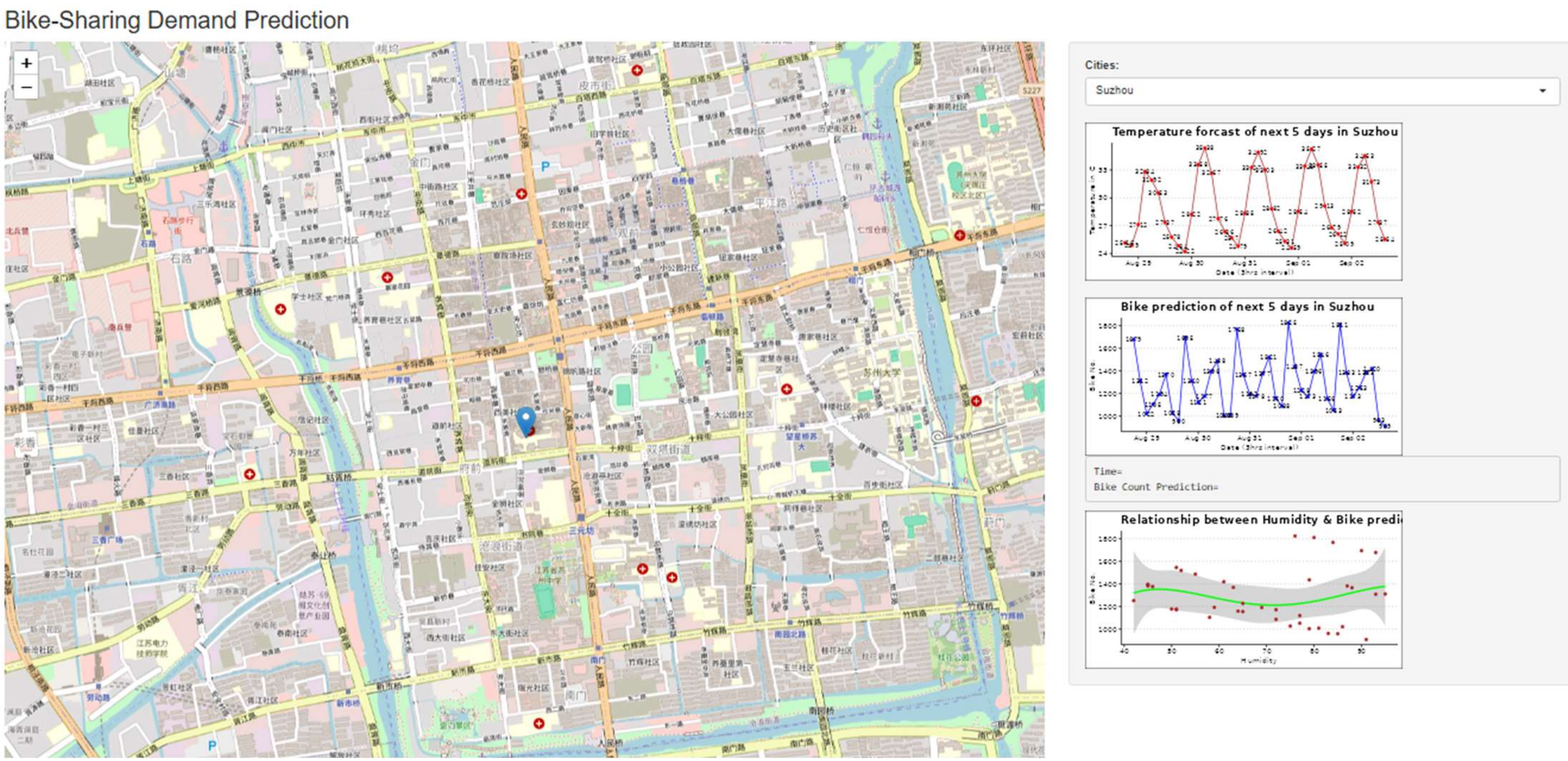


Dashboard

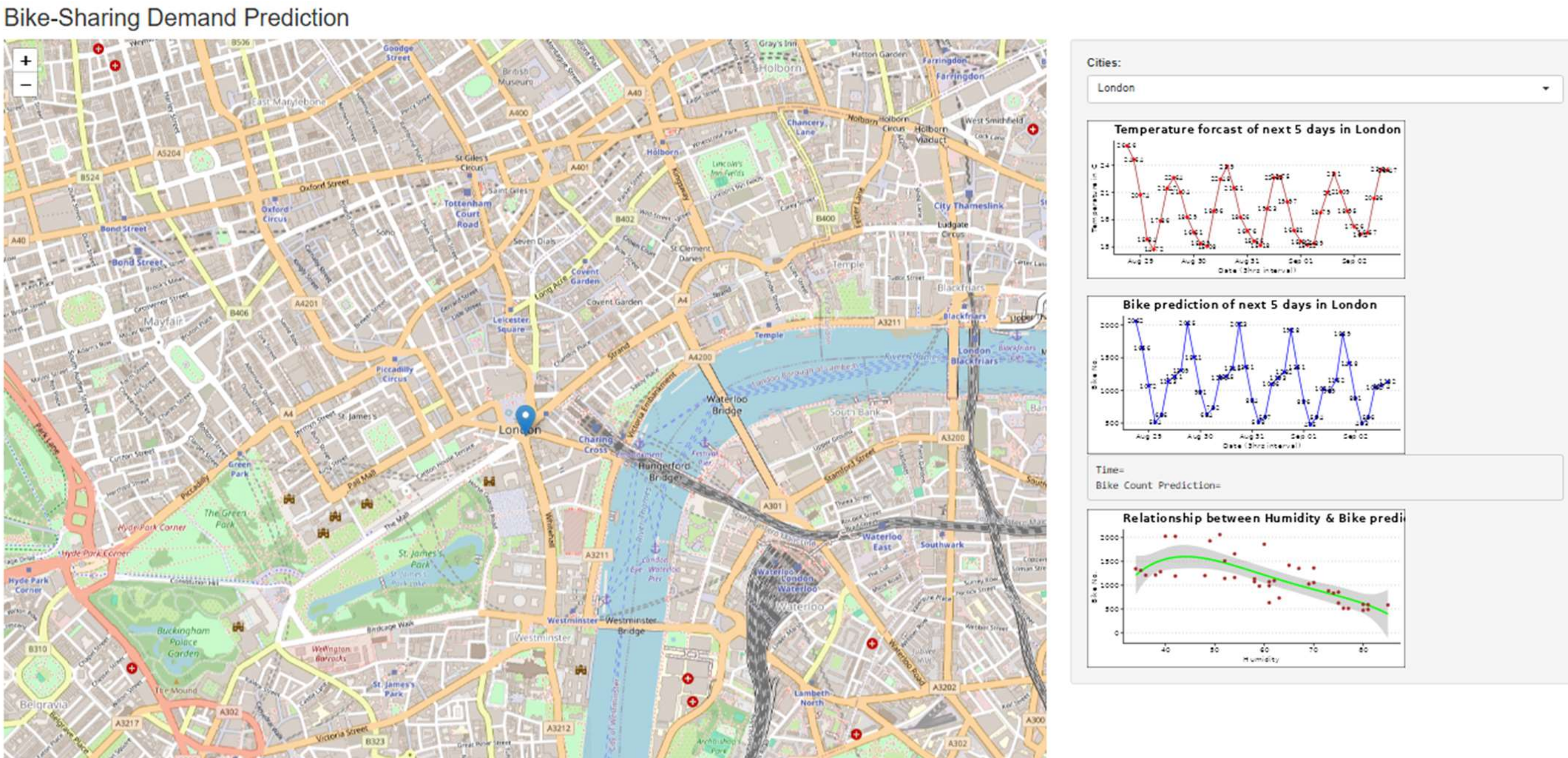
Seoul Bike-sharing prediction on a map



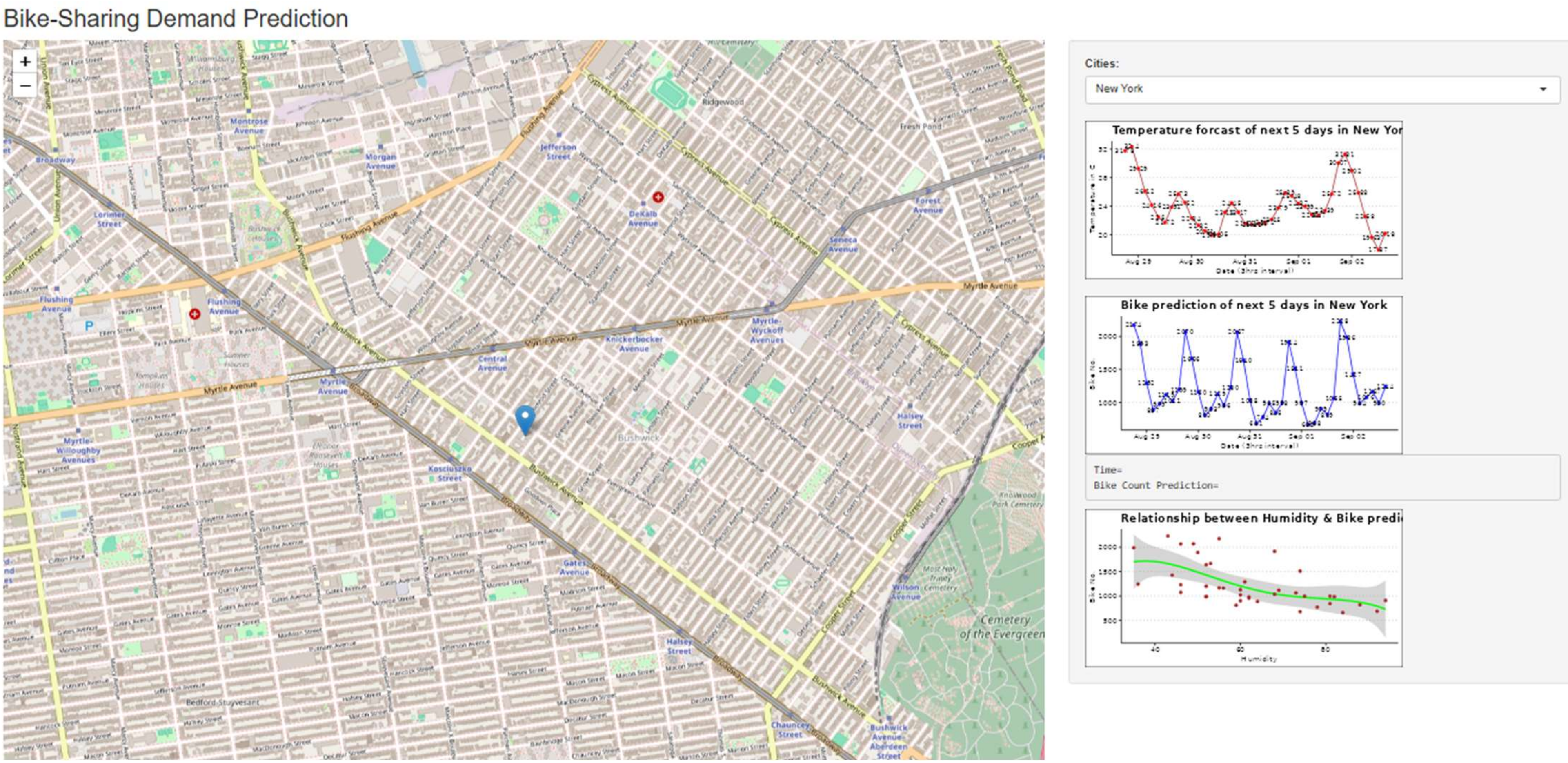
Suzhou Bike-sharing prediction on a map



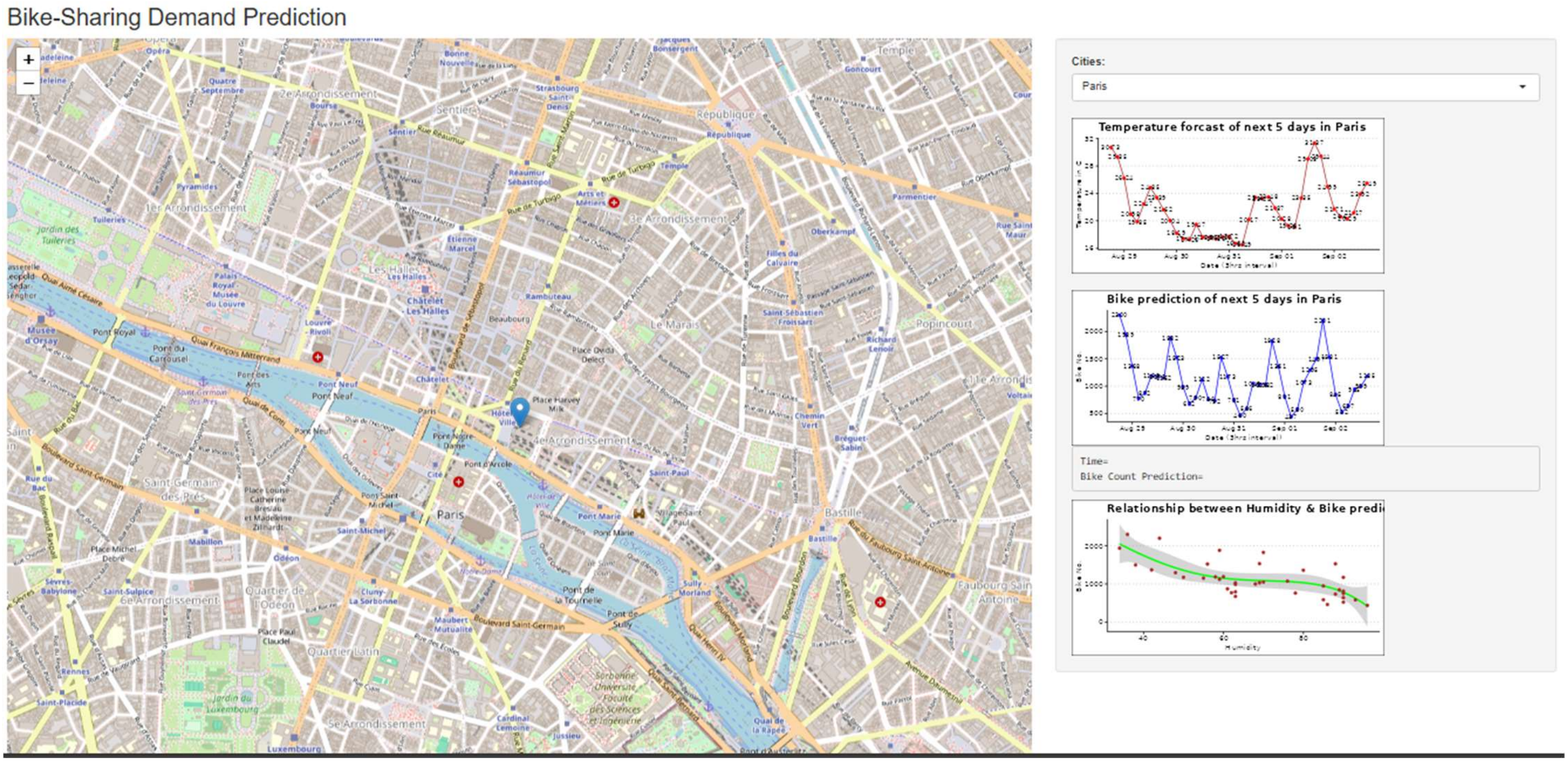
London Bike-sharing prediction on a map



New York Bike-sharing prediction on a map



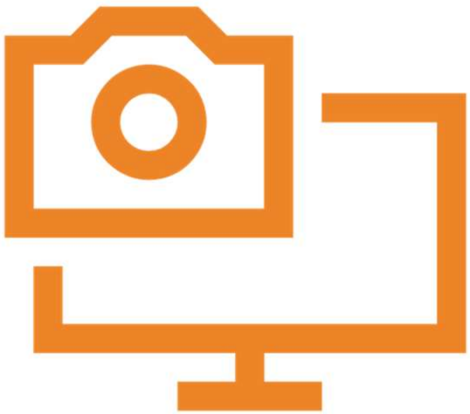
Paris Bike-sharing prediction on a map



CONCLUSION

- Implement weather-adaptive strategies such as dynamic pricing based on weather conditions to encourage bike-sharing.
- Results indicate all variable related model performs better than weather-based model.
- The larger the R-squared and smaller the RMSE, the better the model. The best performing model in terms of given criteria is model 4.
- Improve infrastructure to make bike-sharing more appealing during less favorable weather conditions, such as covered bike paths and better lighting.

APPENDIX



- [GitHub URL]

  https://github.com/SR000777/R-Data-Science-Capstone-Project.git

- Thanks to all the instructors of the course
- Thanks to IBM Skills Network Team
- Thanks to Edx Team.