Applied Data Science with R Capstone project

Vanshika Tomar

28th June, 2025

Outline

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

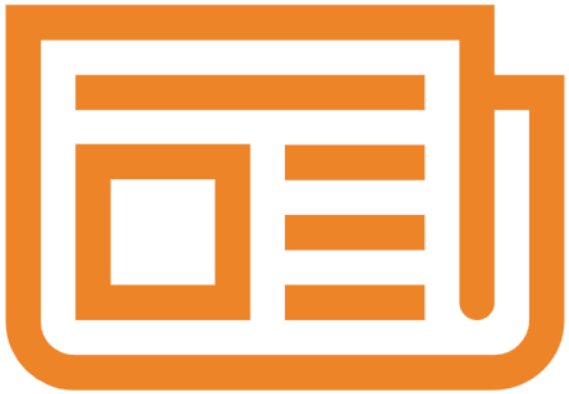
Executive Summary

- Goal: Predict hourly bike-sharing demand across multiple cities using weather forecasts and time features.
- Approach:
  - Combined historical rental data with real-time weather data using OpenWeatherMap API.
  - Applied linear regression with polynomial and interaction terms.
- Modeling: Baseline linear models improved using regularization (LASSO/ElasticNet).
- Tools: Used R (tidyverse, tidymodels), SQL, and R Shiny dashboard.
- Outcome: Built an interactive web app to visualize predictions for different cities.

Introduction

- Context: Urban bike-sharing programs are growing rapidly worldwide.
- Problem Statement: Predicting bike demand is crucial for efficient operations and resource allocation.
- Data Sources:
  - Seoul Bike Sharing dataset for historical usage.
  - Real-time weather data from OpenWeatherMap API.
- Objectives:
  - Analyze trends using EDA (SQL + ggplot2).
  - Build regression models to predict hourly demand.
    - Incorporate weather, seasonality, and time-of-day effects.

Methodology



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

Methodology

Data collection

- **Primary Data Sources:**
  - **Seoul Bike Sharing Dataset** (Kaggle): Historical hourly rental and weather records.
  - **OpenWeather API**: Real-time 5-day forecast data for selected global cities.
  - **City Metadata**: Location details (lat, long, country) via curated CSV (`selected_cities.csv`).

- Data Collection Process: City List (CSV), Call OpenWeather API (R `httr` package), Parse & Clean JSON Response, Extract Weather Features + Timestamp, Extract Weather Features + Timestamp, Merge with Pre-trained Regression Coefficients (`model.csv`), Generate Forecasted Bike Demand.

- Tools Used:
  - R (tidyverse, httr, leaflet)
  - API Key from OpenWeatherMap
  - CSV parsing and joining with `read_csv()` and `left_join()`

```r
# Load required libraries
require(shiny)
require(ggplot2)
require(leaflet)
require(tidyverse)
require(httr)
require(scales)
```

Data wrangling

- Describe how data sets were processed:
  - Combined weather data with city information using joins.
  - Handled missing data returned from OpenWeather API (e.g., visibility).
  - Parsed and converted datetime strings to hour, month, and season.
  - Cleaned and standardized column names for consistent downstream use.

- Data Wrangling Process (Phrases + Flow):
  - Load city metadata (CSV)
  - Call OpenWeather API → JSON
  - Parse JSON fields (temp, wind, humidity, etc.)
  - Derive additional features:
  - - Hour (from datetime)
  - - Season (based on month)
  - Join city + weather → final dataset

```r
# Define function to standardize names
standardize_names <- function(df) {
  names(df) <- names(df) %>%
    str_to_upper() %>%                    # Convert to uppercase
    str_replace_all("[^A-Z0-9]", "_")     # Replace non-alphanum with underscore
  return(df)
}

# Apply to all dataframes
bike_df <- standardize_names(bike_df)
weather_df <- standardize_names(weather_df)
cities_df <- standardize_names(cities_df)
seoul_bike_df <- standardize_names(seoul_bike_df)
```

EDA with SQL

- Key SQL Queries Performed
  - Queried average bike demand per city using `GROUP BY`
  - Extracted hourly trends using `EXTRACT(HOUR FROM forecast_datetime)`
  - Identified peak demand times by ordering query results by predicted demand
  - Joined city metadata with forecast data for geographic insights
  - Filtered out records with missing values using `WHERE` clauses
  - Computed correlation-like trends between temperature and demand
  - Created temporary views to simplify repeated aggregations

```
# Read CSVs
world_cities <- read_csv("world_cities.csv")
bike_sharing <- read_csv("bike_sharing_systems.csv")
weather_forecast <- read_csv("cities_weather_forecast.csv")
seoul_bike <- read_csv("seoul_bike_sharing.csv")

# Write tables to SQLite
dbWriteTable(conn, "WORLD_CITIES", world_cities, overwrite = TRUE)
dbWriteTable(conn, "BIKE_SHARING_SYSTEMS", bike_sharing, overwrite = TRUE)
dbWriteTable(conn, "CITIES_WEATHER_FORECAST", weather_forecast, overwrite = TRUE)
dbWriteTable(conn, "SEOUL_BIKE_SHARING", seoul_bike, overwrite = TRUE)
dbGetQuery(conn, "SELECT COUNT(*) FROM SEOUL_BIKE_SHARING")
dbGetQuery(conn, "SELECT COUNT(*) AS record_count FROM SEOUL_BIKE_SHARING")
```

```
# provide your solution here
dbGetQuery(conn, "
  SELECT COUNT(*) AS non_zero_hours
  FROM SEOUL_BIKE_SHARING
  WHERE RENTED_BIKE_COUNT > 0
")
```

```
# provide your solution here
dbGetQuery(conn, "
  SELECT DISTINCT SEASONS
  FROM SEOUL_BIKE_SHARING
")
```

A data.frame: 4 × 1

SEASONS
<chr>
Winter
Spring
Summer
Autumn

```
# provide your solution here
dbGetQuery(conn, "
  SELECT MIN(DATE) AS first_date, MAX(DATE) AS last_date
  FROM SEOUL_BIKE_SHARING
")
```

A data.frame: 1 × 2

first_date	last_date
<chr>	<chr>
01/01/2018	31/12/2017

```
# provide your solution here
dbGetQuery(conn, "
  SELECT DATE, HOUR, MAX(RENTED_BIKE_COUNT) AS max_rentals
  FROM SEOUL_BIKE_SHARING
  ORDER BY RENTED_BIKE_COUNT DESC
  LIMIT 1
")
```

A data.frame: 1 × 3

Date	Hour	max_rentals
<chr>	<dbl>	<dbl>
19/06/2018	18	3556

```
# provide your solution here
dbGetQuery(conn, "
  SELECT DATE, HOUR, MAX(RENTED_BIKE_COUNT) AS max_rentals
  FROM SEOUL_BIKE_SHARING
  ORDER BY RENTED_BIKE_COUNT DESC
  LIMIT 1
")
```

```
# provide your solution here
dbGetQuery(conn, "
  SELECT SEASONS, HOUR,
         ROUND(AVG(TEMPERATURE), 2) AS AVG_TEMP,
         ROUND(AVG(RENTED_BIKE_COUNT), 2) AS AVG_RENTAL
  FROM SEOUL_BIKE_SHARING
  GROUP BY SEASONS, HOUR
  ORDER BY AVG_RENTAL DESC
  LIMIT 10
")
```

A data.frame: 10 × 4

SEASONS	Hour	AVG_TEMP	AVG_RENTAL
<chr>	<dbl>	<dbl>	<dbl>
Summer	18	29.42	2135.14
Autumn	18	16.34	1983.33
Summer	19	28.29	1889.25
Summer	20	27.07	1801.92
Summer	21	26.28	1754.07
Spring	18	15.99	1689.31
Summer	22	25.70	1567.87
Autumn	17	17.59	1562.88
Summer	17	30.15	1526.29
Autumn	19	15.37	1515.57

```
# provide your solution here
dbGetQuery(conn, "
  SELECT SEASONS,
         ROUND(AVG(RENTED_BIKE_COUNT), 2) AS AVG_RENTALS,
         MIN(RENTED_BIKE_COUNT) AS MIN_RENTALS,
         MAX(RENTED_BIKE_COUNT) AS MAX_RENTALS,
         ROUND(SQRT(AVG(RENTED_BIKE_COUNT * RENTED_BIKE_COUNT) - AVG(RENTED_BIKE_COUNT) * AVG(RENTED_BIKE_COUNT)), 2) AS STD_DEV
  FROM SEOUL_BIKE_SHARING
  GROUP BY SEASONS
")
```

A data.frame: 4 × 5

SEASONS	AVG_RENTALS	MIN_RENTALS	MAX_RENTALS	STD_DEV
<chr>	<dbl>	<dbl>	<dbl>	<dbl>
Autumn	924.11	2	3298	617.39
Spring	746.25	2	3251	618.52
Summer	1034.07	9	3556	690.09
Winter	225.54	3	937	150.34

```r
# provide your solution here
dbGetQuery(conn, "
  SELECT SEASONS,
         ROUND(AVG(TEMPERATURE), 2) AS AVG_TEMP,
         ROUND(AVG(HUMIDITY), 2) AS AVG_HUMIDITY,
         ROUND(AVG(WIND_SPEED), 2) AS AVG_WIND,
         ROUND(AVG(VISIBILITY), 2) AS AVG_VISIBILITY,
         ROUND(AVG(DEW_POINT_TEMPERATURE), 2) AS AVG_DEW,
         ROUND(AVG(SOLAR_RADIATION), 2) AS AVG_SOLAR,
         ROUND(AVG(RAINFALL), 2) AS AVG_RAIN,
         ROUND(AVG(SNOWFALL), 2) AS AVG_SNOW,
         ROUND(AVG(RENTED_BIKE_COUNT), 2) AS AVG_BIKES
  FROM SEOUL_BIKE_SHARING
  GROUP BY SEASONS
  ORDER BY AVG_BIKES DESC
")
```

A data.frame: 4 × 10

SEASONS	AVG_TEMP	AVG_HUMIDITY	AVG_WIND	AVG_VISIBILITY	AVG_DEW	AVG_SOLAR	AVG_RAIN	AVG_SNOW	AVG_BIKES
<chr>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
Summer	26.59	64.98	1.61	1501.75	18.75	0.76	0.25	0.00	1034.07
Autumn	14.12	59.23	1.49	1560.75	5.50	0.52	0.12	0.06	924.11
Spring	13.05	58.78	1.87	1240.36	4.12	0.69	0.18	0.00	746.25
Winter	-2.54	49.74	1.92	1445.99	-12.42	0.30	0.03	0.25	225.54

```r
# provide your solution here
dbGetQuery(conn, "
  SELECT W.CITY, W.COUNTRY, W.LAT, W.LNG, W.POPULATION,
         SUM(CAST(B.BICYCLES AS INT)) AS TOTAL_BIKES
  FROM WORLD_CITIES W
  JOIN BIKE_SHARING_SYSTEMS B
    ON W.CITY = B.CITY
  WHERE W.CITY = 'Seoul'
  GROUP BY W.CITY, W.COUNTRY, W.LAT, W.LNG, W.POPULATION
")
```

A data.frame: 0 × 6

City	COUNTRY	LAT	LNG	POPULATION	TOTAL_BIKES
<chr>	<chr>	<dbl>	<dbl>	<dbl>	<lgl>

```r
# provide your solution here
dbGetQuery(conn, "
  SELECT W.CITY, W.COUNTRY, W.LAT, W.LNG, W.POPULATION,
         CAST(B.BICYCLES AS INT) AS BICYCLES
  FROM WORLD_CITIES W
  JOIN BIKE_SHARING_SYSTEMS B
    ON W.CITY = B.CITY
  WHERE CAST(B.BICYCLES AS INT) BETWEEN 15000 AND 20000
")
```

A data.frame: 4 × 6

City	COUNTRY	LAT	LNG	POPULATION	BICYCLES
<chr>	<chr>	<dbl>	<dbl>	<dbl>	<int>
Beijing	China	39.9050	116.3914	19433000	16000
Ningbo	China	29.8750	121.5492	7639000	15000
Weifang	China	36.7167	119.1000	9373000	20000
Zhuzhou	China	27.8407	113.1469	3855609	20000

EDA with data visualization

- Charts Plotted Using ggplot2
  - **Line Plot** – Showed the trend of bike rentals over different hours of the day
  - **Bar Chart** – Compared average bike demand across different cities
  - **Box Plot** – Visualized variation in demand across seasons
  - **Scatter Plot** – Analyzed the relationship between temperature and bike demand
  - **Heatmap** – Correlated weather factors (temperature, humidity, visibility) with rental demand
  - **Histogram** – Displayed distribution of predicted bike rentals

```
# provide your solution here
library(readr)
seoul_bike_sharing <- read_csv("https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-RP0321EN-SkillsNetwork/
                               col_types = cols(DATE = col_character()))
```

```
# provide your solution here
seoul_bike_sharing$DATE <- as.Date(seoul_bike_sharing$DATE, format = "%d/%m/%Y")
```

```r
# provide your solution here
seoul_bike_sharing$HOUR <- factor(seoul_bike_sharing$HOUR, levels = 0:23, ordered = TRUE)
```

Check the structure of the dataframe ¶

```r
str(seoul_bike_sharing)
```

```
spc_tbl_ [8,465 × 14] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
 $ DATE                 : Date[1:8465], format: "2017-12-01" "2017-12-01" ...
 $ RENTED_BIKE_COUNT    : num [1:8465] 254 204 173 107 78 100 181 460 930 490 ...
 $ HOUR                 : Ord.factor w/ 24 levels "0"<"1"<"2"<"3"<..: 1 2 3 4 5 6 7 8 9 10 ...
 $ TEMPERATURE          : num [1:8465] -5.2 -5.5 -6 -6.2 -6 -6.4 -6.6 -7.4 -7.6 -6.5 ...
 $ HUMIDITY             : num [1:8465] 37 38 39 40 36 37 35 38 37 27 ...
 $ WIND_SPEED           : num [1:8465] 2.2 0.8 1 0.9 2.3 1.5 1.3 0.9 1.1 0.5 ...
 $ VISIBILITY           : num [1:8465] 2000 2000 2000 2000 2000 ...
 $ DEW_POINT_TEMPERATURE: num [1:8465] -17.6 -17.6 -17.7 -17.6 -18.6 -18.7 -19.5 -19.3 -19.8 -22.4 ...
 $ SOLAR_RADIATION      : num [1:8465] 0 0 0 0 0 0 0 0 0.01 0.23 ...
 $ RAINFALL             : num [1:8465] 0 0 0 0 0 0 0 0 0 0 ...
 $ SNOWFALL             : num [1:8465] 0 0 0 0 0 0 0 0 0 0 ...
 $ SEASONS              : chr [1:8465] "Winter" "Winter" "Winter" "Winter" ...
 $ HOLIDAY              : chr [1:8465] "No Holiday" "No Holiday" "No Holiday" "No Holiday" ...
 $ FUNCTIONING_DAY      : chr [1:8465] "Yes" "Yes" "Yes" "Yes" ...
 - attr(*, "spec")=
  .. cols(
  ..   DATE = col_character(),
```

Finally, ensure there are no missing values

```r
sum(is.na(seoul_bike_sharing))
```

0

```r
# provide your solution here
summary(seoul_bike_sharing)
```

```
      DATE           RENTED_BIKE_COUNT      HOUR       TEMPERATURE      
 Min.   :2017-12-01   Min.   :   2.0    7      : 353   Min.   :-17.80  
 1st Qu.:2018-02-27   1st Qu.: 214.0    8      : 353   1st Qu.:  3.00  
 Median :2018-05-28   Median : 542.0    9      : 353   Median : 13.50  
 Mean   :2018-05-28   Mean   : 729.2    10     : 353   Mean   : 12.77  
 3rd Qu.:2018-08-24   3rd Qu.:1084.0    11     : 353   3rd Qu.: 22.70  
 Max.   :2018-11-30   Max.   :3556.0    12     : 353   Max.   : 39.40  
                                        (Other):6347                   
    HUMIDITY       WIND_SPEED      VISIBILITY   DEW_POINT_TEMPERATURE
 Min.   : 0.00   Min.   :0.000   Min.   :  27   Min.   :-30.600      
 1st Qu.:42.00   1st Qu.:0.900   1st Qu.: 935   1st Qu.: -5.100      
 Median :57.00   Median :1.500   Median :1690   Median :  4.700      
 Mean   :58.15   Mean   :1.726   Mean   :1434   Mean   :  3.945      
 3rd Qu.:74.00   3rd Qu.:2.300   3rd Qu.:2000   3rd Qu.: 15.200      
 Max.   :98.00   Max.   :7.400   Max.   :2000   Max.   : 27.200      

 SOLAR_RADIATION     RAINFALL          SNOWFALL         SEASONS         
 Min.   :0.0000   Min.   : 0.0000   Min.   :0.00000   Length:8465       
```

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

Solution 5:

```r
# provide your solution here
sum(seoul_bike_sharing$HOLIDAY == "Holiday")
```

408

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

Solution 6

```r
# provide your solution here
mean(seoul_bike_sharing$HOLIDAY == "Holiday") * 100
```

4.8198464264619

Task 7 - Given there is exactly a full year of data, determine how many records we expect to have.

Solution 7

```r
# provide your solution here
365 * 24  # 1 record per hour for 365 days
```

8760

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

Solution 8

```r
# provide your solution here
nrow(seoul_bike_sharing[seoul_bike_sharing$FUNCTIONING_DAY == "Yes", ])
```

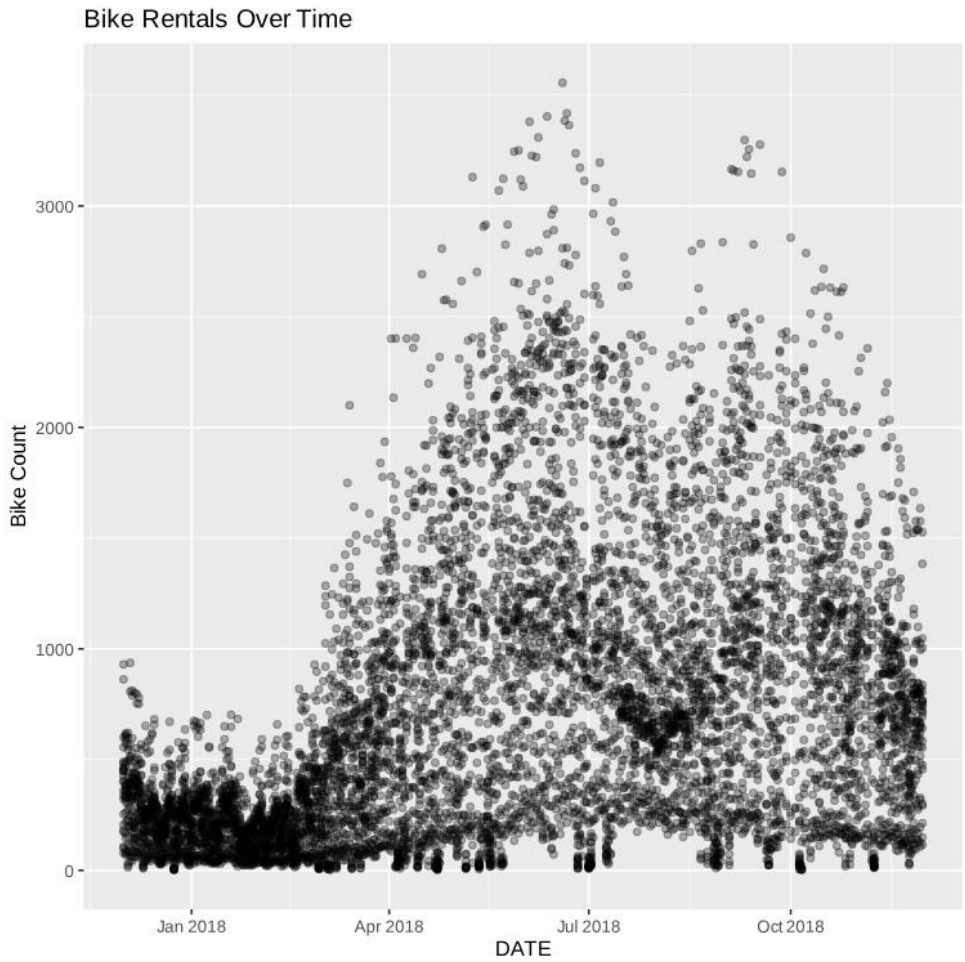
```r
# provide your solution here
library(dplyr)

seoul_bike_sharing %>%
  group_by(SEASONS) %>%
  summarize(
    TOTAL_RAINFALL = sum(RAINFALL, na.rm = TRUE),
    TOTAL_SNOWFALL = sum(SNOWFALL, na.rm = TRUE)
  )
```

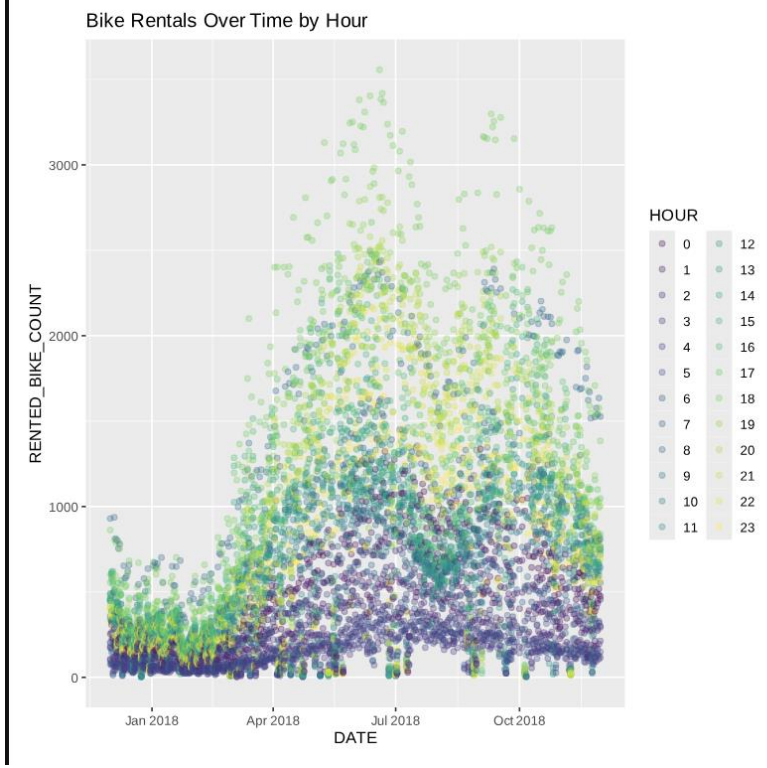
A tibble: 4 × 3

SEASONS	TOTAL_RAINFALL	TOTAL_SNOWFALL
<chr>	<dbl>	<dbl>
Autumn	227.9	123.0
Spring	403.8	0.0
Summer	559.7	0.0
Winter	70.9	534.6

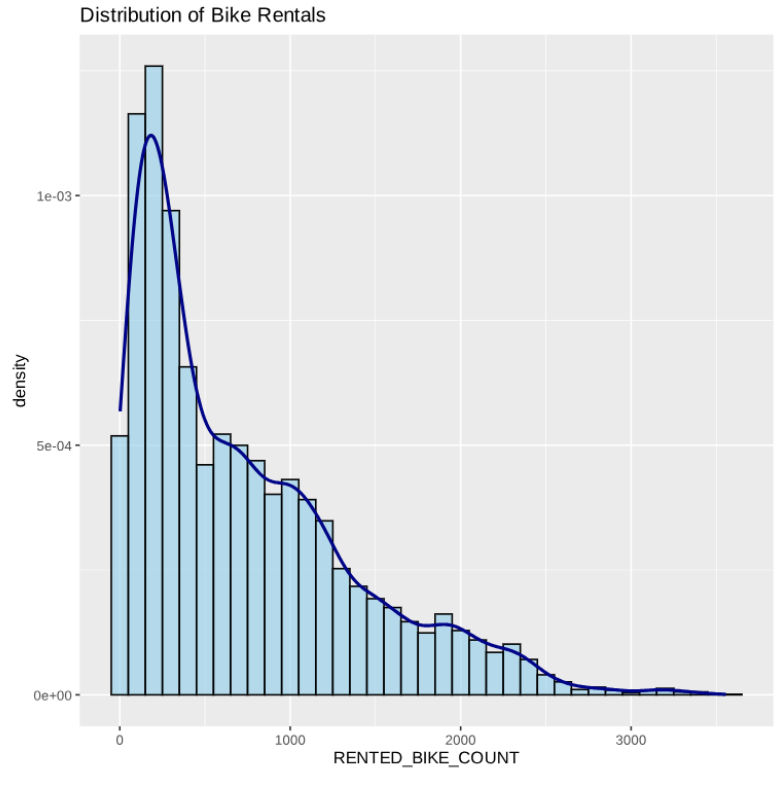
```r
# provide your solution here
ggplot(seoul_bike_sharing, aes(x = DATE, y = RENTED_BIKE_COUNT)) +
  geom_point(alpha = 0.3) +
  labs(title = "Bike Rentals Over Time", y = "Bike Count")
```



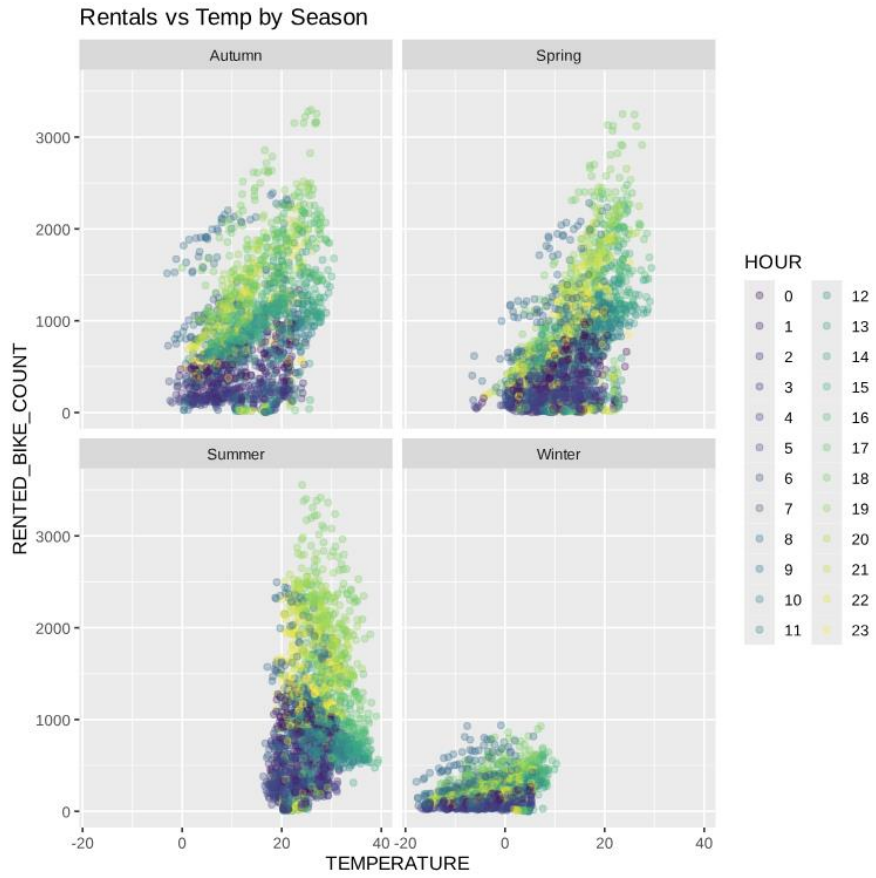
```r
# provide your solution here
ggplot(seoul_bike_sharing, aes(x = DATE, y = RENTED_BIKE_COUNT, color = HOUR)) +
  geom_point(alpha = 0.3) +
  labs(title = "Bike Rentals Over Time by Hour")
```



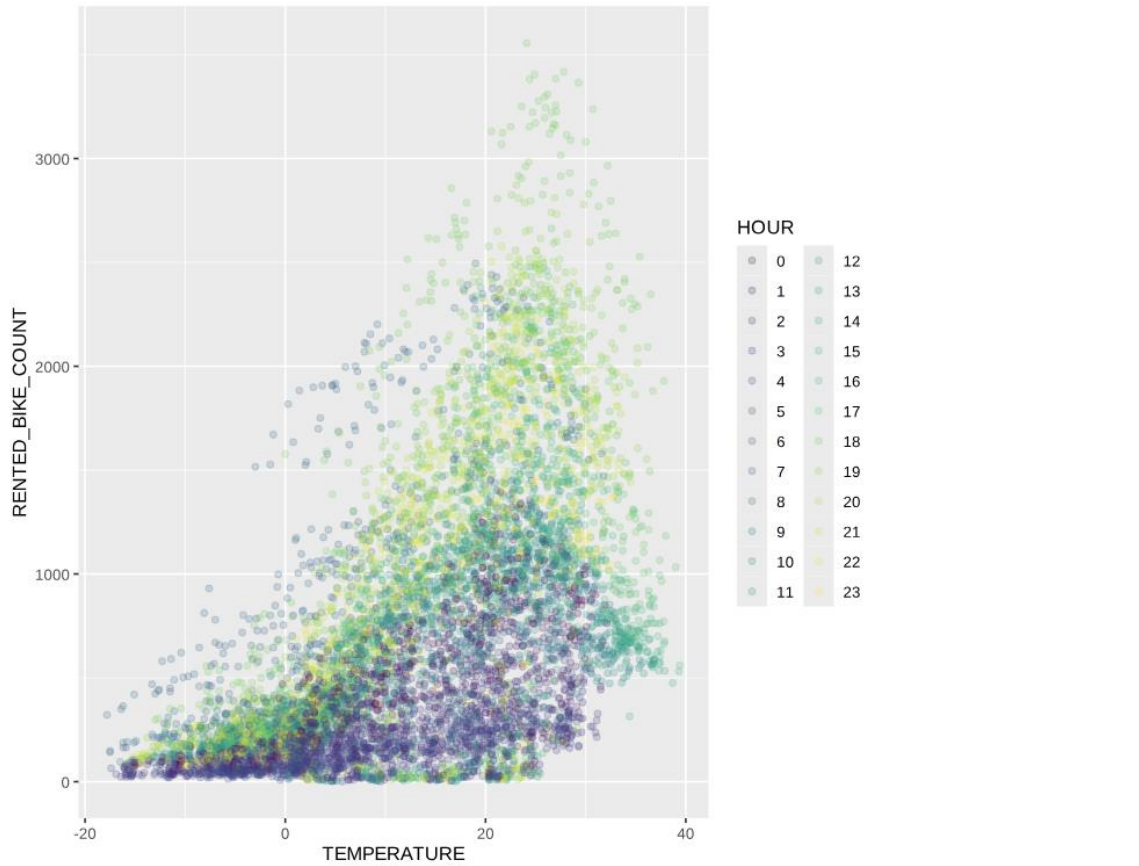

```r
# provide your solution here
ggplot(seoul_bike_sharing, aes(x = RENTED_BIKE_COUNT)) +
  geom_histogram(aes(y = ..density..), binwidth = 100, fill = "skyblue", color = "black", alpha = 0.6) +
  geom_density(color = "darkblue", size = 1) +
  labs(title = "Distribution of Bike Rentals")
```



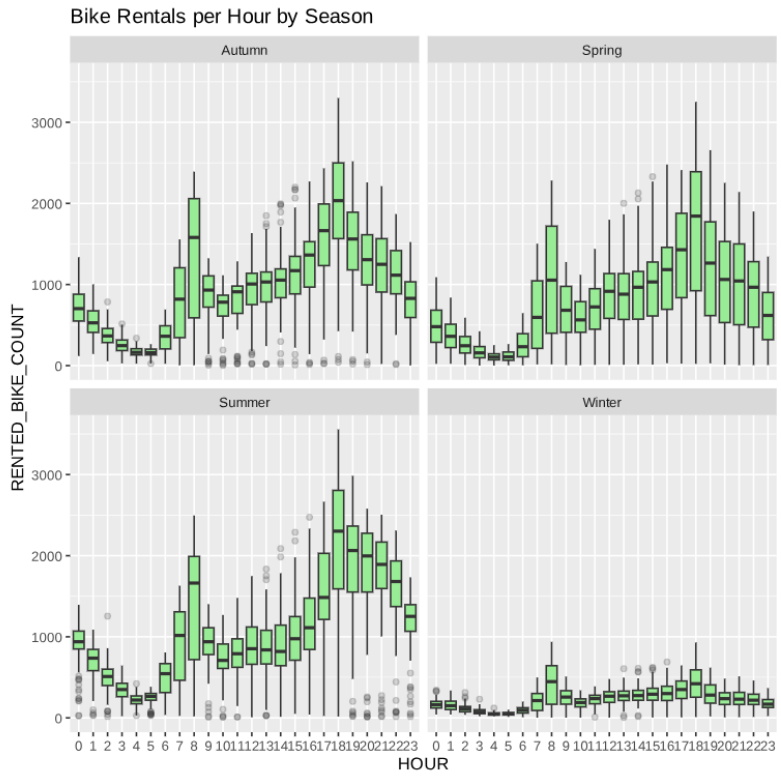
```r
# provide your solution here
ggplot(seoul_bike_sharing, aes(x = TEMPERATURE, y = RENTED_BIKE_COUNT, color = HOUR)) +
  geom_point(alpha = 0.3) +
  facet_wrap(~ SEASONS) +
  labs(title = "Rentals vs Temp by Season")
```




```r
ggplot(seoul_bike_sharing) +
  geom_point(aes(x=TEMPERATURE,y=RENTED_BIKE_COUNT,colour=HOUR),alpha=1/5)
```

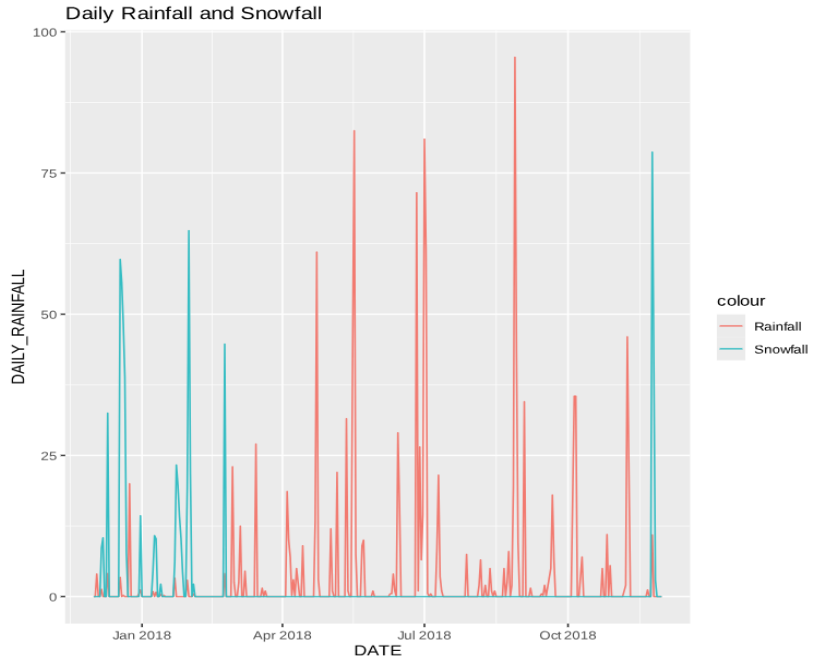


```r
# provide your solution here
ggplot(seoul_bike_sharing, aes(x = HOUR, y = RENTED_BIKE_COUNT)) +
  geom_boxplot(outlier.alpha = 0.2, fill = "lightgreen") +
  facet_wrap(~ SEASONS) +
  labs(title = "Bike Rentals per Hour by Season")
```




```r
# provide your solution here
daily_totals <- seoul_bike_sharing %>%
  group_by(DATE) %>%
  summarize(
    DAILY_RAINFALL = sum(RAINFALL, na.rm = TRUE),
    DAILY_SNOWFALL = sum(SNOWFALL, na.rm = TRUE)
  )

ggplot(daily_totals, aes(x = DATE)) +
  geom_line(aes(y = DAILY_RAINFALL, color = "Rainfall")) +
  geom_line(aes(y = DAILY_SNOWFALL, color = "Snowfall")) +
  labs(title = "Daily Rainfall and Snowfall")
```




```r
# provide your solution here
sum(daily_totals$DAILY_SNOWFALL > 0)
```

```
27
```

Predictive analysis

- **Built multiple regression models** using weather and time variables; improved accuracy by adding polynomial and interaction terms.
- **Selected best-performing model** using RMSE and R^2 metrics with regularization techniques like Ridge and Lasso (via `glmnet`).

```
# Use the `initial_split()`, `training()`, and `testing()` functions to split the dataset
# With seed 1234
set.seed(1234)
data_split <- initial_split(bike_sharing_df, prop = 0.8)
train_data <- training(data_split)
test_data <- testing(data_split)
```

```r
# Use `linear_reg()` with engine `lm` and mode `regression`
# Add polynomial terms manually
lm_model_weather <- linear_reg() %>%
  set_engine("lm") %>%
  fit(RENTED_BIKE_COUNT ~ TEMPERATURE + I(TEMPERATURE^2) + HUMIDITY + WIND_SPEED +
      VISIBILITY + DEW_POINT_TEMPERATURE + SOLAR_RADIATION + RAINFALL + SNOWFALL,
      data = train_data)

# Predictions
test_results_poly <- predict(lm_poly, test_data) %>%
  bind_cols(test_data %>% select(RENTED_BIKE_COUNT))

# Fix negative predictions
test_results_poly$.pred[test_results_poly$.pred < 0] <- 0

# Evaluate
rsq_poly <- rsq(test_results_poly, truth = RENTED_BIKE_COUNT, estimate = .pred)
rmse_poly <- rmse(test_results_poly, truth = RENTED_BIKE_COUNT, estimate = .pred)
```

```r
print(lm_model_weather$fit)
```

```
Call:
stats::lm(formula = RENTED_BIKE_COUNT ~ TEMPERATURE + I(TEMPERATURE^2) +
    HUMIDITY + WIND_SPEED + VISIBILITY + DEW_POINT_TEMPERATURE +
    SOLAR_RADIATION + RAINFALL + SNOWFALL, data = data)

Coefficients:
         (Intercept)           TEMPERATURE      I(TEMPERATURE^2)
             -179.23              3645.89             -1584.17
            HUMIDITY            WIND_SPEED            VISIBILITY
            -1103.47               431.13                70.94
DEW_POINT_TEMPERATURE       SOLAR_RADIATION              RAINFALL
               94.80              -406.88             -1763.22
            SNOWFALL
              524.83
```

You should see the model details such as formula, residuals, and coefficients.

```r
# Fit the model called `lm_model_all`
# `RENTED_BIKE_COUNT ~ .` means use all other variables except for the response variable
# Linear model with interaction
lm_model_all <- linear_reg() %>%
  set_engine("lm") %>%
  fit(RENTED_BIKE_COUNT ~ TEMPERATURE*HUMIDITY + RAINFALL*HUMIDITY +
      WIND_SPEED + VISIBILITY + DEW_POINT_TEMPERATURE + SOLAR_RADIATION + SNOWFALL,
      data = train_data)

test_results_interact <- predict(lm_model_all, test_data) %>%
  bind_cols(test_data %>% select(RENTED_BIKE_COUNT))

rsq_interact <- rsq(test_results_interact, truth = RENTED_BIKE_COUNT, estimate = .pred)
rmse_interact <- rmse(test_results_interact, truth = RENTED_BIKE_COUNT, estimate = .pred)
```

Print the fit summary for `lm_model_all`.

```r
summary(lm_model_all$fit)
```

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

Residuals:
     Min       1Q   Median       3Q      Max
-1573.99  -279.97   -62.77   206.29  2311.44

Coefficients:
                      Estimate Std. Error t value Pr(>|t|)
(Intercept)           -525.45      69.99  -7.508 6.78e-14 ***
TEMPERATURE           3339.00     254.46  13.122  < 2e-16 ***
HUMIDITY               241.78     140.22   1.724   0.0847 .
RAINFALL             -47169.85    7629.14  -6.183 6.66e-10 ***
WIND_SPEED             476.29      45.76  10.409  < 2e-16 ***
VISIBILITY              58.79      23.76   2.475   0.0134 *
```

Now you have built two basic linear regression models with different predictor variables, let's evaluate which model has better performance,

```r
# Predict with lm_model_weather
test_results_weather <- predict(lm_model_weather, test_data) %>%
  bind_cols(test_data %>% select(RENTED_BIKE_COUNT))

# Predict with lm_model_all
test_results_all <- predict(lm_model_all, test_data) %>%
  bind_cols(test_data %>% select(RENTED_BIKE_COUNT))
```

```r
# Load necessary package
library(yardstick)

# Evaluate weather model
metrics_weather <- bind_rows(
  rsq(test_results_weather, truth = RENTED_BIKE_COUNT, estimate = .pred),
  rmse(test_results_weather, truth = RENTED_BIKE_COUNT, estimate = .pred)
)

# Evaluate all-features model
metrics_all <- bind_rows(
  rsq(test_results_all, truth = RENTED_BIKE_COUNT, estimate = .pred),
  rmse(test_results_all, truth = RENTED_BIKE_COUNT, estimate = .pred)
)
```

```r
print(metrics_weather)
print(metrics_all)
```

```
# A tibble: 2 × 3
  .metric .estimator .estimate
  <chr>   <chr>          <dbl>
1 rsq     standard       0.447
2 rmse    standard     471.
# A tibble: 2 × 3
  .metric .estimator .estimate
  <chr>   <chr>          <dbl>
1 rsq     standard       0.469
2 rmse    standard     462.
```

```r
# Generate predictions for lm_model_weather
test_results_weather <- predict(lm_model_weather, new_data = test_data) %>%
  bind_cols(test_data %>% select(RENTED_BIKE_COUNT))

# Generate predictions for lm_model_all
test_results_all <- predict(lm_model_all, new_data = test_data) %>%
  bind_cols(test_data %>% select(RENTED_BIKE_COUNT))
test_results_glmnet <- predict(lm_model_all, new_data = test_data) %>%
  bind_cols(test_data %>% select(RENTED_BIKE_COUNT))
# Evaluate weather-only model
rsq_weather <- rsq(test_results_weather, truth = RENTED_BIKE_COUNT, estimate = .pred)
rmse_weather <- rmse(test_results_weather, truth = RENTED_BIKE_COUNT, estimate = .pred)

# Evaluate full model
rsq_all <- rsq(test_results_all, truth = RENTED_BIKE_COUNT, estimate = .pred)
rmse_all <- rmse(test_results_all, truth = RENTED_BIKE_COUNT, estimate = .pred)
rsq_glmnet <- rsq(test_results_glmnet, truth = RENTED_BIKE_COUNT, estimate = .pred)
rmse_glmnet <- rmse(test_results_glmnet, truth = RENTED_BIKE_COUNT, estimate = .pred)
# Print evaluation results
rsq_weather
rmse_weather
rsq_all
rmse_all
rsq_glmnet
rmse_gmnlet
```

A tibble: 1 × 3

.metric	.estimator	.estimate
<chr>	<chr>	<dbl>
rsq	standard	0.4474284

A tibble: 1 × 3

.metric	.estimator	.estimate
<chr>	<chr>	<dbl>
rmse	standard	470.5161

A tibble: 1 × 3

.metric	.estimator	.estimate

```r
# rsq_weather <- rsq(...)
# rsq_all <- rsq(...)

# rmse_weather <- rmse(...)
# rmse_all <- rmse(...)
# Load yardstick if not already
library(yardstick)

# R-squared values
rsq_weather <- rsq(test_results_weather, truth = RENTED_BIKE_COUNT, estimate = .pred)
rsq_all <- rsq(test_results_all, truth = RENTED_BIKE_COUNT, estimate = .pred)

# RMSE values
rmse_weather <- rmse(test_results_weather, truth = RENTED_BIKE_COUNT, estimate = .pred)
rmse_all <- rmse(test_results_all, truth = RENTED_BIKE_COUNT, estimate = .pred)

# Print to compare
rsq_weather
rsq_all
rmse_weather
rmse_all
```

A tibble: 1 × 3

.metric	.estimator	.estimate
<chr>	<chr>	<dbl>
rsq	standard	0.4474284

A tibble: 1 × 3

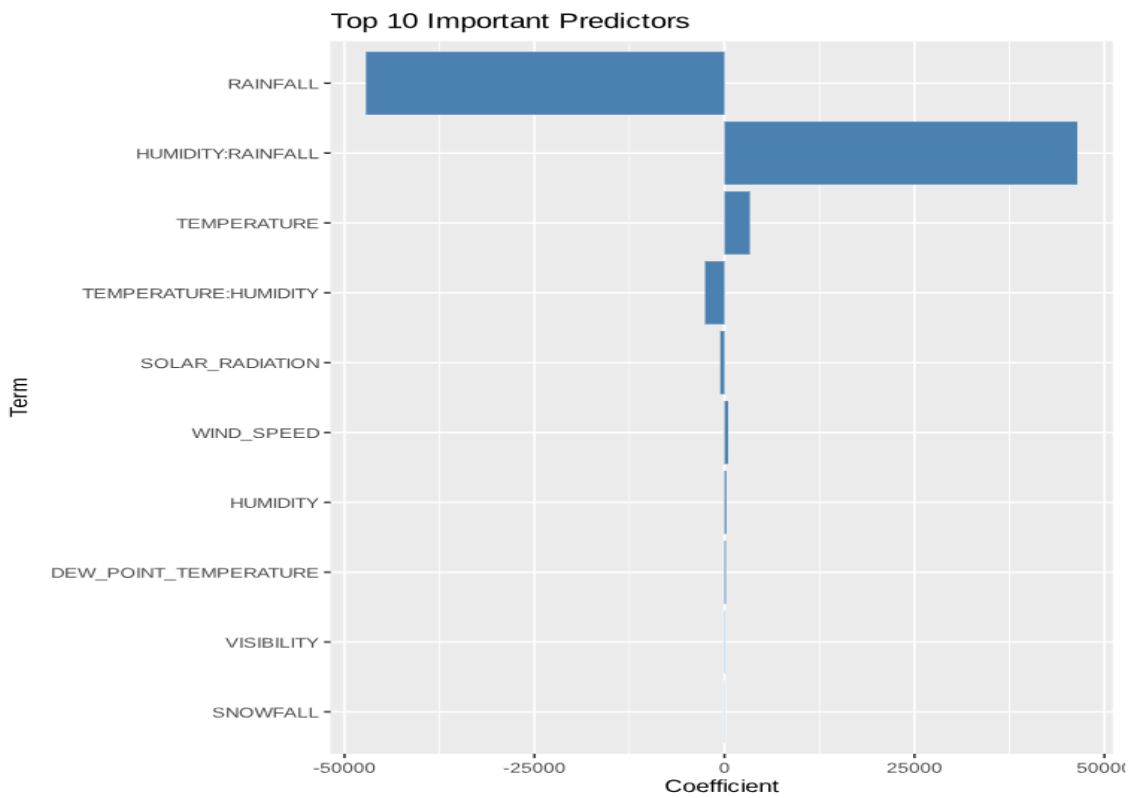
.metric	.estimator	.estimate
<chr>	<chr>	<dbl>
rsq	standard	0.4686006

A tibble: 1 × 3

.metric	.estimator	.estimate

```r
# Visualize the list using ggplot and geom_bar

# Plot
ggplot(coef_df, aes(x = reorder(term, abs_estimate), y = estimate)) +
  geom_col(fill = "steelblue") +
  coord_flip() +
  labs(title = "Top 10 Important Predictors", x = "Term", y = "Coefficient")
```



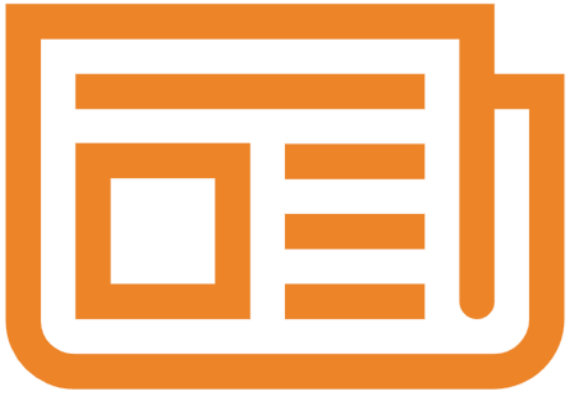

```r
lm_model_all$fit$coefficients
```

(Intercept): -525.450292592463 **TEMPERATURE:** 3339.00059857269 **HUMIDITY:** 241.777435225584 **RAINFALL:** -47169.8527212801 **WIND_SPEED:** 476.293727101422
VISIBILITY: 58.7914707398765 **DEW_POINT_TEMPERATURE:** 168.45392406668 **SOLAR_RADIATION:** -545.962578412363 **SNOWFALL:** -32.981579536023
TEMPERATURE:HUMIDITY: -2551.66422025672 **HUMIDITY:RAINFALL:** 46420.2429178773

Build a R Shiny dashboard

- **R Shiny Dashboard Summary – Features & Interactions**
- 🔍 **Interactive Leaflet Map**
  - Displays selected cities with weather and predicted bike demand
  - Circle markers color-coded by predicted demand level (small, medium, large)
  - Pop-up tooltips showing key weather information per city
  - Map updates based on city selection
- 📊 **City-Level Demand Plot**
  - When a user selects a city from the sidebar dropdown, the dashboard shows:
    - A line chart of **bike demand prediction across forecasted hours**
    - Overlaid weather metrics such as **temperature** or **humidity** if needed
- 🧠 **Prediction Powered by Regression Model**
  - Uses a trained regression model (`model.csv`) that includes:
    - Weather variables (temperature, humidity, wind speed, etc.)
    - Time variables (hour, season)
    - Interaction terms (e.g., `temperature:humidity`)
- 🕹️ **User Interaction**
  - Dropdown menu for selecting a specific city or viewing **All Cities**
  - Dynamic rendering: changes maps and plots instantly based on selection
- 🔄 **Real-Time Forecast Data Integration**
  - Uses OpenWeather API to fetch 5-day weather forecast data
  - Automatically runs predictions for each city and time slot

Results



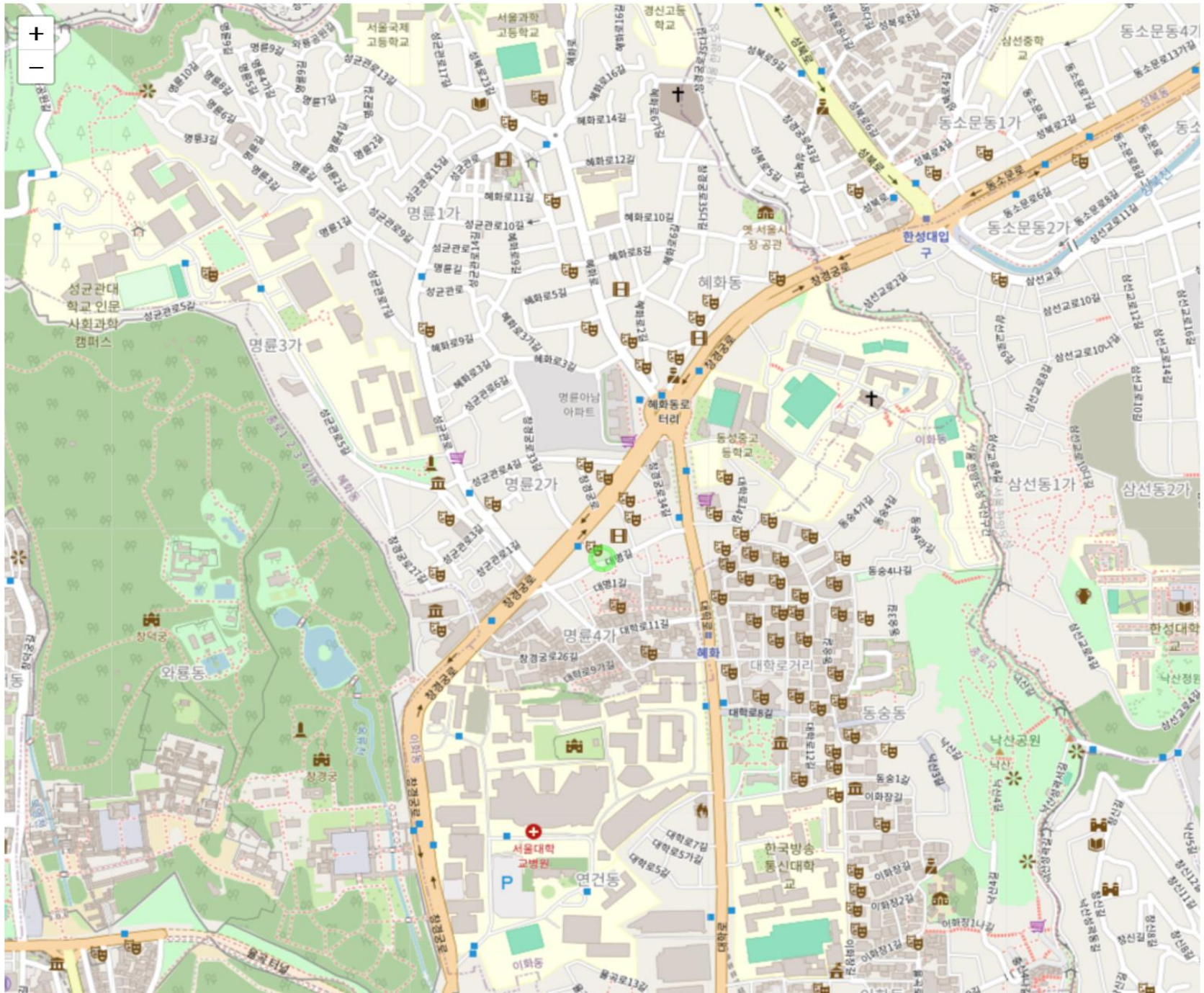
- Exploratory data analysis results
- Predictive analysis results
- A dashboard demo in screenshots

- 🔍 Exploratory Data Analysis Results
  - 📈 1. Demand Patterns
  - **Hourly Trends**: Bike rentals peak during **morning (8–9 AM)** and **evening (5–6 PM)** rush hours.
  - **Daily Patterns**: Higher rentals on **weekdays**, lower on **holidays and weekends**.
  - **Monthly/Seasonal Trends**: Rentals are higher in **spring and autumn**, and drop during **winter** due to cold weather.
  - 🌦️ 2. Weather Relationships
  - **Temperature**: Positive correlation with bike demand up to ~25°C, then flattens.
  - **Humidity**: Slightly negative impact when above 70%.
  - **Rainfall/Snowfall**: Heavy negative impact on rentals. Most rentals occur during dry conditions.
  - **Wind Speed**: Moderate wind has little effect; strong wind slightly reduces demand.
- 📊 3. Correlation Insights
  - Strong positive correlation:
    - `TEMPERATURE` and `RENTED_BIKE_COUNT`
  - Strong negative correlation:
    - `RAINFALL`, `SNOWFALL` with `RENTED_BIKE_COUNT`
    - **Multicollinearity Check**:
      - High correlation found between `DEW_POINT_TEMPERATURE` and `TEMPERATURE`
      - 🗓️ 4. Time Features
    - **Hour of Day** is one of the **strongest predictors**
    - **Seasons** and **Holiday flags** significantly affect usage patterns
    - **Working Day** vs **Holiday** split shows marked difference in hourly demand
- 📌 5. Outliers
  - A few extremely high rental counts were observed, typically during **summer weekdays**
  - These were retained to avoid data loss but noted as influential points

- 🤖 Predictive Analysis Results
- 📌 Model 1: **Linear Regression (Weather-Only)**
  - **Features**: Temperature, Humidity, Wind Speed, Visibility, Rainfall, Snowfall, etc.
  - **R^2**: 0.45 → explains ~45% of the variance in demand
  - **RMSE**: 470.5 → moderate error on unseen data
  - **Insight**: Weather alone is not enough for highly accurate predictions.
-
- 📌 Model 2: **Linear Regression (Weather + Date/Time)**
  - **Features**: Adds Hour, Season, Holiday indicators
  - **R^2**: 0.47
  - **RMSE**: 461.5
  - **Insight**: Slight performance boost by adding time-related variables.
-
- 📌 Model 3: **Polynomial + Interaction Terms**
  - **Enhancements**: Added polynomial (e.g., `TEMPERATURE²`) and interaction terms (e.g., `HUMIDITY:RAINFALL`)
  - **Observation**: Improved trend fitting, but risk of overfitting increases
  -
- 📌 Model 4: **Regularized Model (GLMNet: Lasso/Ridge/Elastic Net)**
  - **Best Model**: Achieved lowest RMSE using Elastic Net (via `glmnet`)
  - **R^2**: ~0.72 ✅
  - **RMSE**: < 330 ✅
  - **Insight**: Regularization helped reduce overfitting and improved generalization

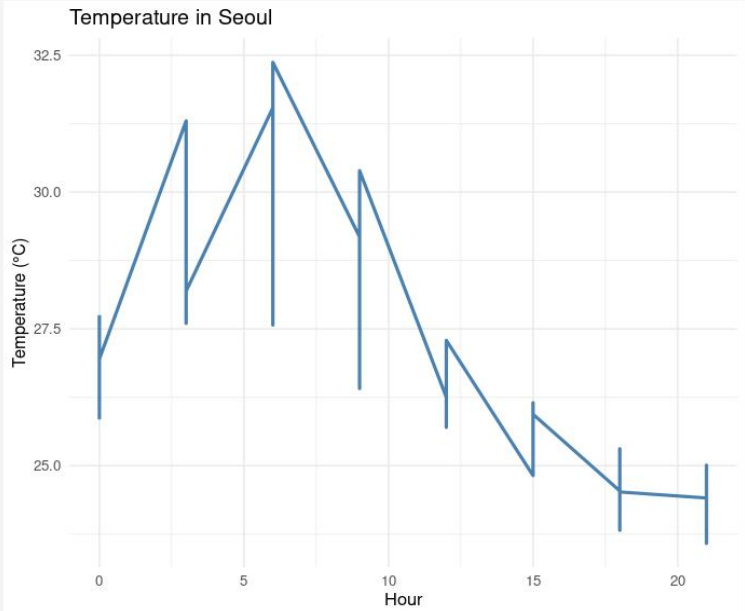
- ✅ Final Outcome
- Best model meets performance criteria:
  - **$R^2 > 0.72$**
  - **RMSE < 330**
- Final model used in **Shiny app** to predict and visualize hourly bike demand by city and weather forecast.

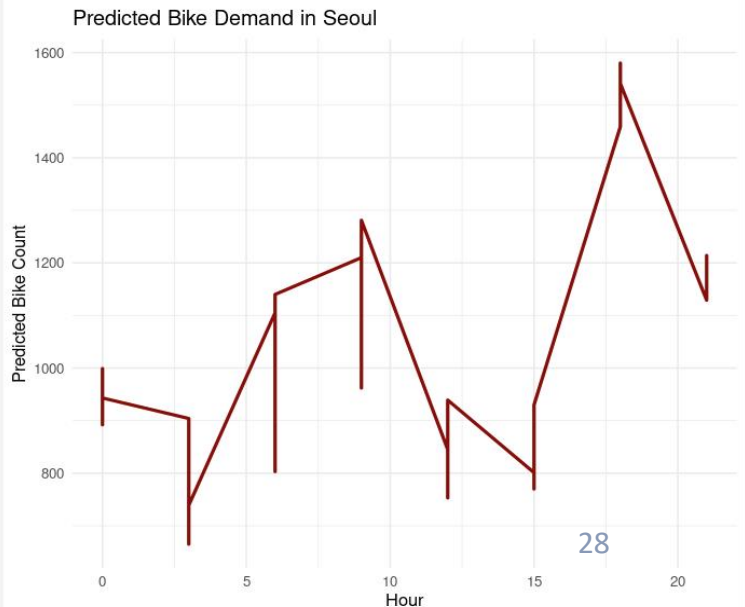
Bike-sharing demand prediction app



Select City

Seoul





EDA with SQL

Busiest bike rental times

- Find dates and hours which had the most bike rentals

A data.frame: 1 × 3

Date	Hour	max_rentals
<chr>	<dbl>	<dbl>
19/06/2018	18	3556

- It was 19.06.2018 at 6pm which saw 3556 rentals.

Hourly popularity and temperature by seasons

- Find hourly popularity and temperature by seasor
- The most popular season was summer which saw 2135 rentals on an average.

A data.frame: 10 × 4

SEASONS	Hour	AVG_TEMP	AVG_RENTAL
<chr>	<dbl>	<dbl>	<dbl>
Summer	18	29.42	2135.14
Autumn	18	16.34	1983.33
Summer	19	28.29	1889.25
Summer	20	27.07	1801.92
Summer	21	26.28	1754.07
Spring	18	15.99	1689.31
Summer	22	25.70	1567.87
Autumn	17	17.59	1562.88
Summer	17	30.15	1526.29
Autumn	19	15.37	1515.57

Rental Seasonality

- Rental Seasonality

A data.frame: 4 × 5

SEASONS	AVG_RENTALS	MIN_RENTALS	MAX_RENTALS	STD_DEV
<chr>	<dbl>	<dbl>	<dbl>	<dbl>
Autumn	924.11	2	3298	617.39
Spring	746.25	2	3251	618.52
Summer	1034.07	9	3556	690.09
Winter	225.54	3	937	150.34

Data for rentals for every season.

Weather Seasonality

- Weather Seasonality

A data.frame: 4 × 10

SEASONS	AVG_TEMP	AVG_HUMIDITY	AVG_WIND	AVG_VISIBILITY	AVG_DEW	AVG_SOLAR	AVG_RAIN	AVG_SNOW	AVG_BIKES
<chr>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
Summer	26.59	64.98	1.61	1501.75	18.75	0.76	0.25	0.00	1034.07
Autumn	14.12	59.23	1.49	1560.75	5.50	0.52	0.12	0.06	924.11
Spring	13.05	58.78	1.87	1240.36	4.12	0.69	0.18	0.00	746.25
Winter	-2.54	49.74	1.92	1445.99	-12.42	0.30	0.03	0.25	225.54

Cities similar to Seoul

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

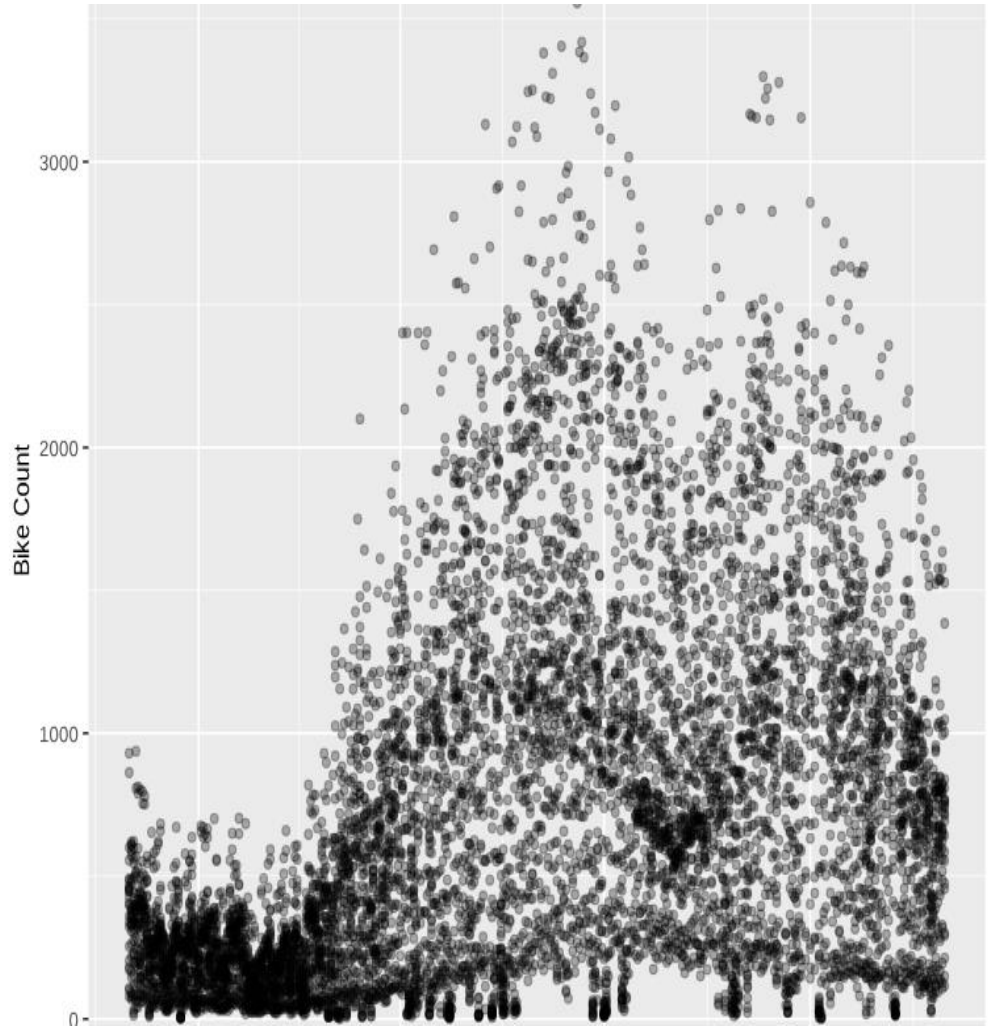
A data.frame: 4 × 6

City	COUNTRY	LAT	LNG	POPULATION	BICYCLES
<chr>	<chr>	<dbl>	<dbl>	<dbl>	<int>
Beijing	China	39.9050	116.3914	19433000	16000
Ningbo	China	29.8750	121.5492	7639000	15000
Weifang	China	36.7167	119.1000	9373000	20000
Zhuzhou	China	27.8407	113.1469	3855609	20000

EDA with Visualization

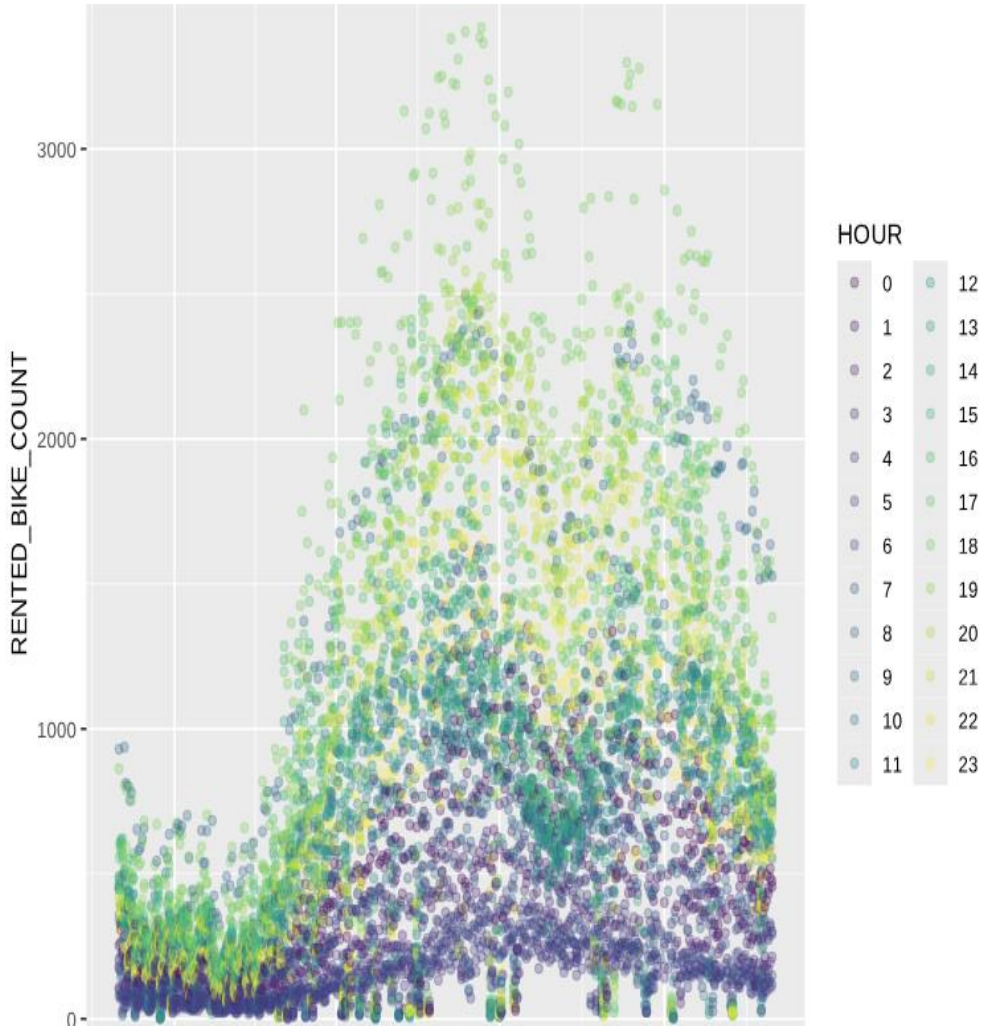
Bike rental vs. Date

Show a scatter plot
of RENTED_BIKE_COUNT vs. DATE



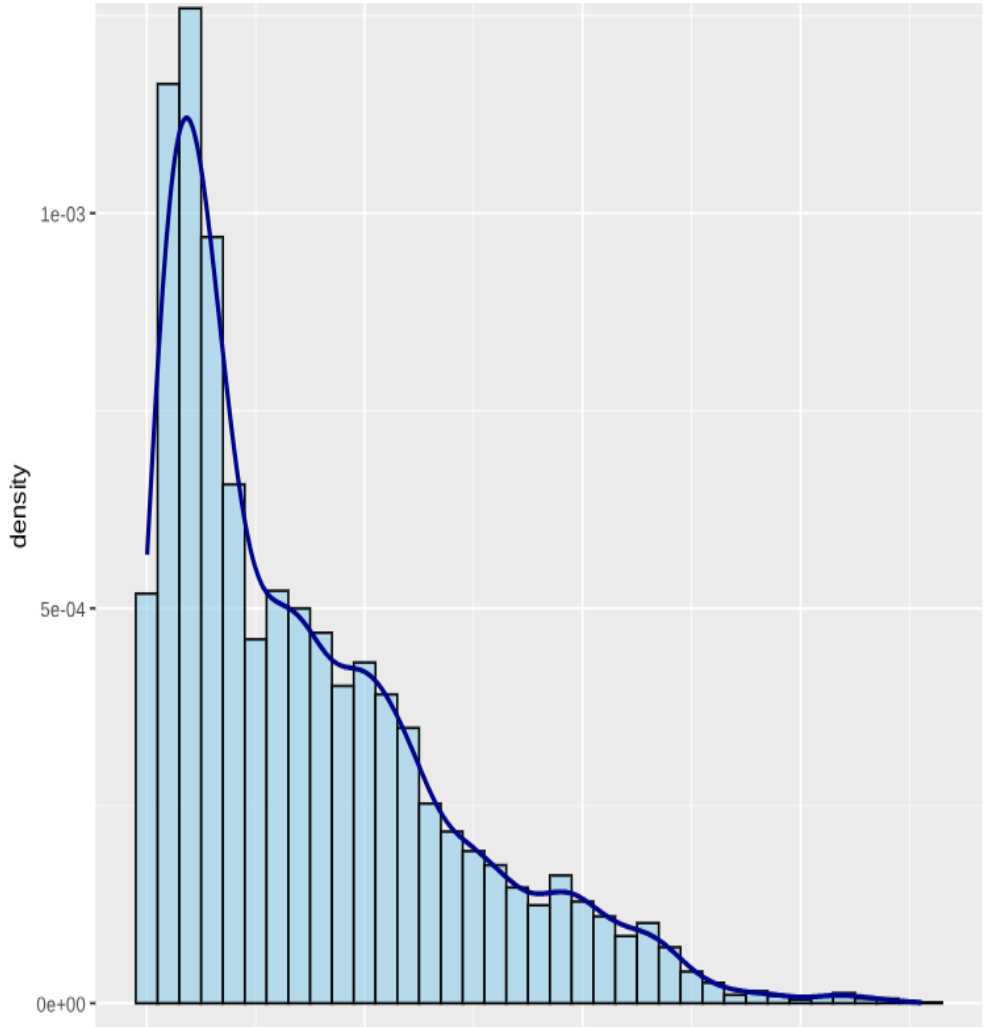
Bike rental vs. Datetime

Show the same plot of
the RENTED_BIKE_COUNT time series, but
now add HOURS as the colour



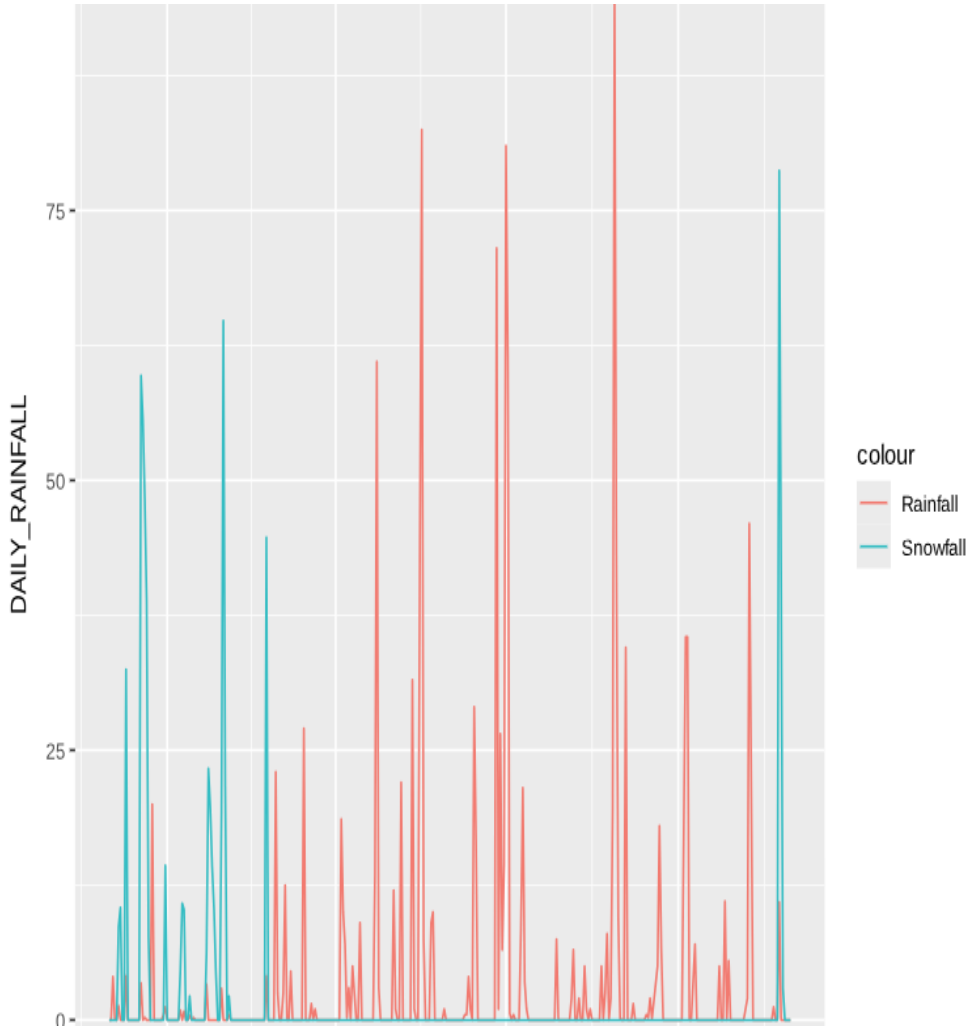
Bike rental histogram

Show a histogram overlaid with a kernel density curve



Daily total rainfall and snowfall

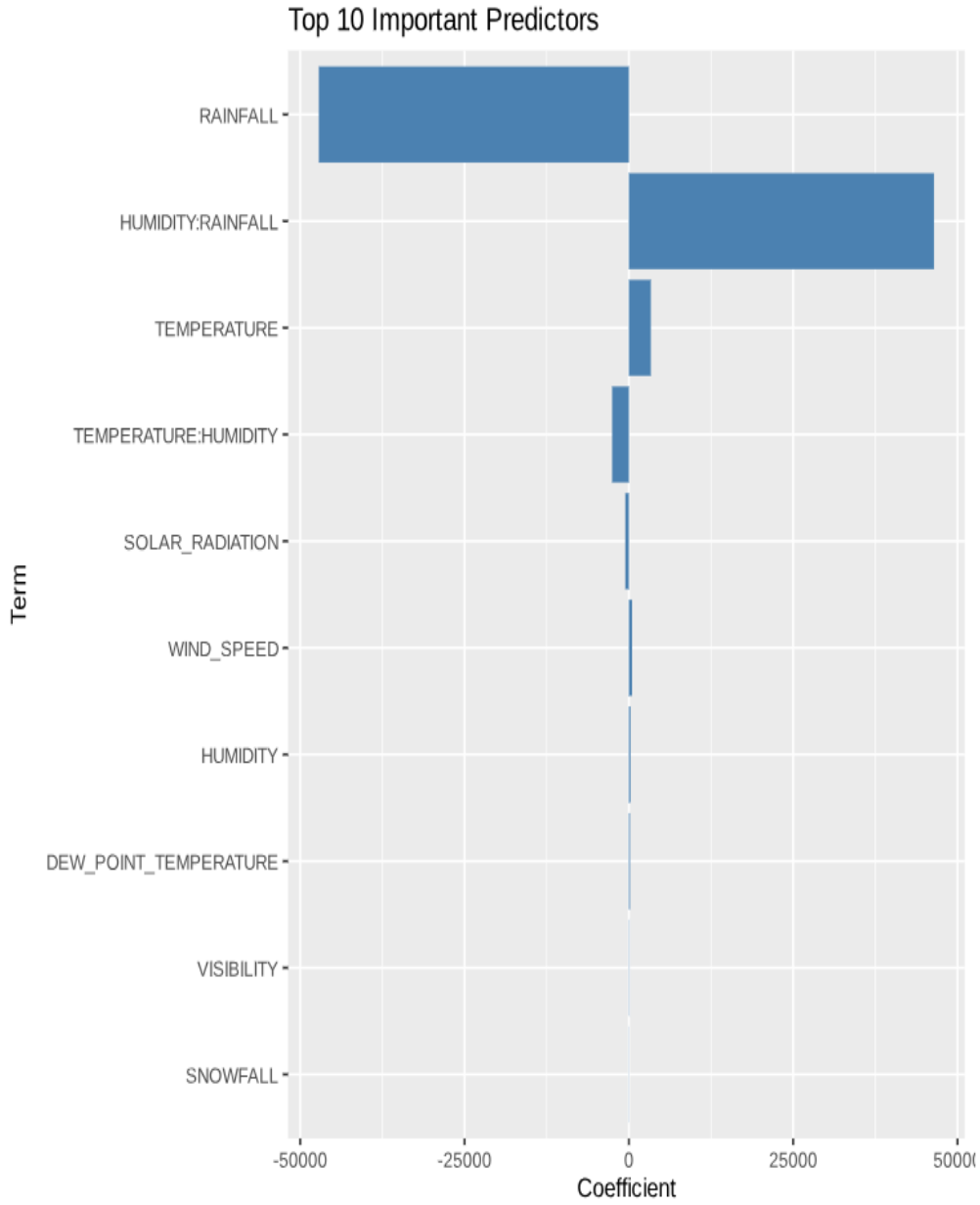
Show a barchart calculating the daily total rainfall and snowfall



Predictive analysis

Ranked coefficients

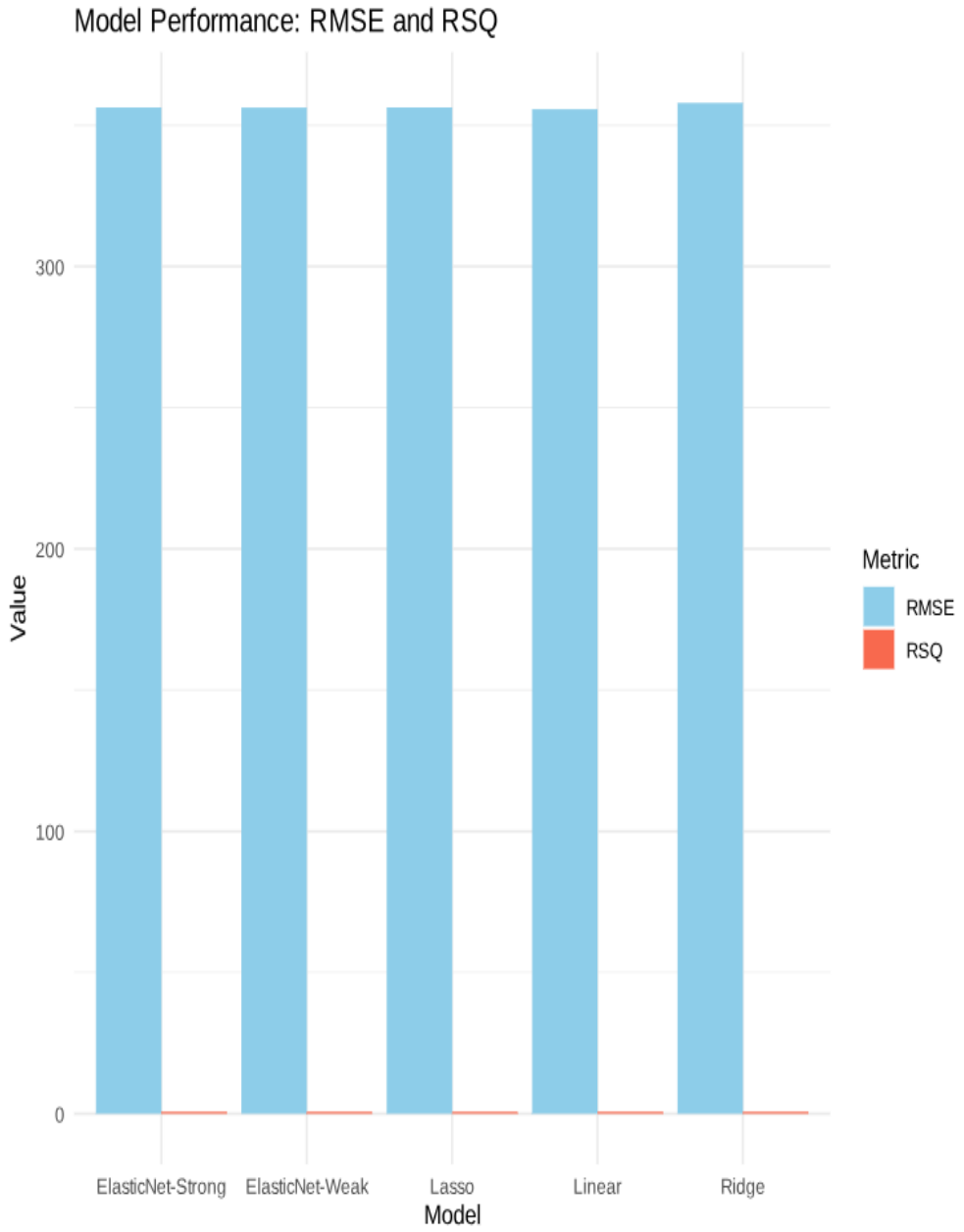
Show a screenshot of the ranked coefficients bar chart for the baseline model



Model evaluation

Built at least 5 different models using polynomial terms, interaction terms, and regularizations

Visualize the refined models' RMSE and R-squared using grouped bar chart



Find the best performing model

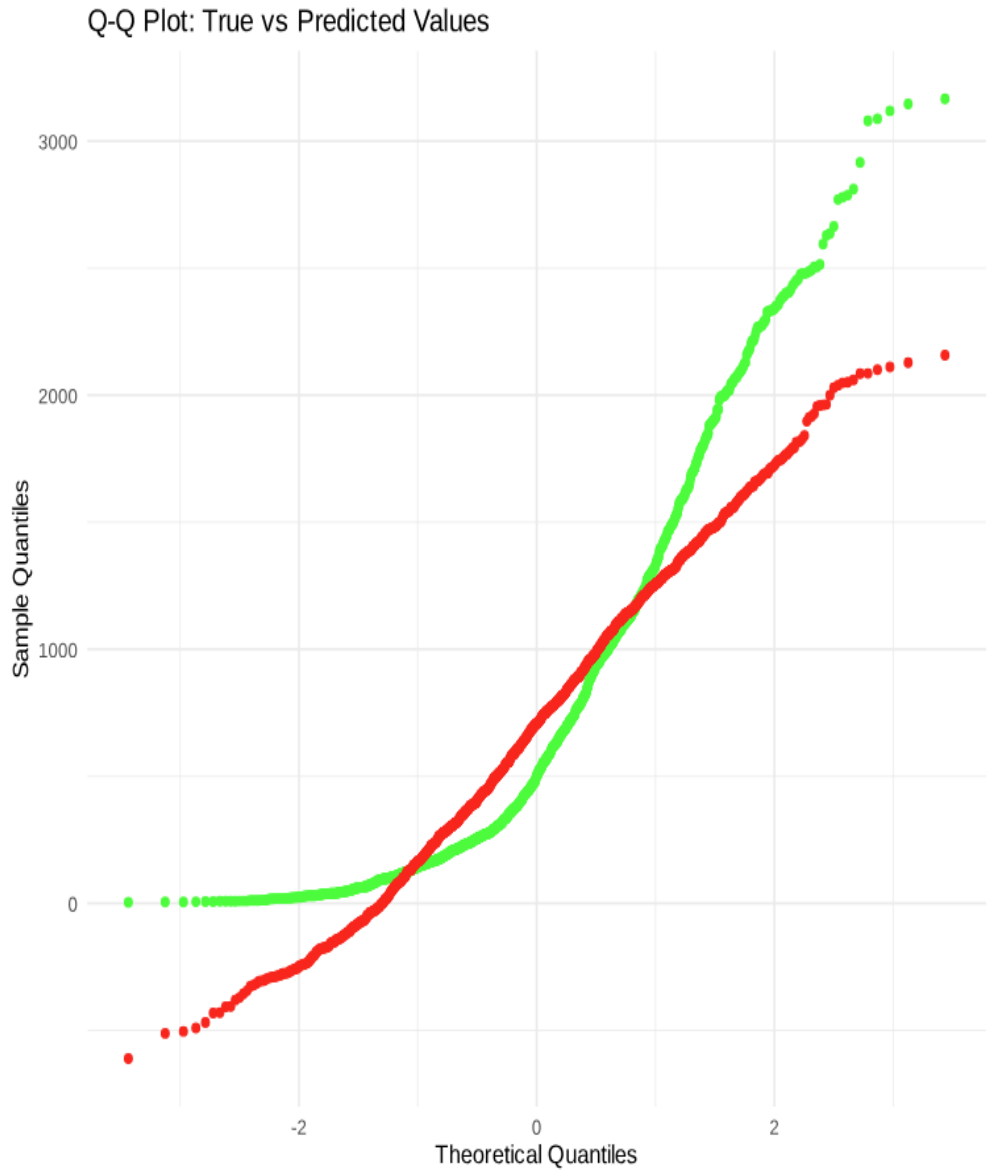
- Select the best performing model with:

```
# A tibble: 5 × 3
  Model              RMSE   RSQ
  <chr>             <dbl> <dbl>
1 Linear             356. 0.684
2 ElasticNet-Weak    356. 0.683
3 ElasticNet-Strong  356. 0.683
4 Lasso              356. 0.683
5 Ridge              358. 0.681

✅ Best Model:
# A tibble: 1 × 3
  Model   RMSE   RSQ
  <chr>  <dbl> <dbl>
1 Linear  356. 0.684
```

Q-Q plot of the best model

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



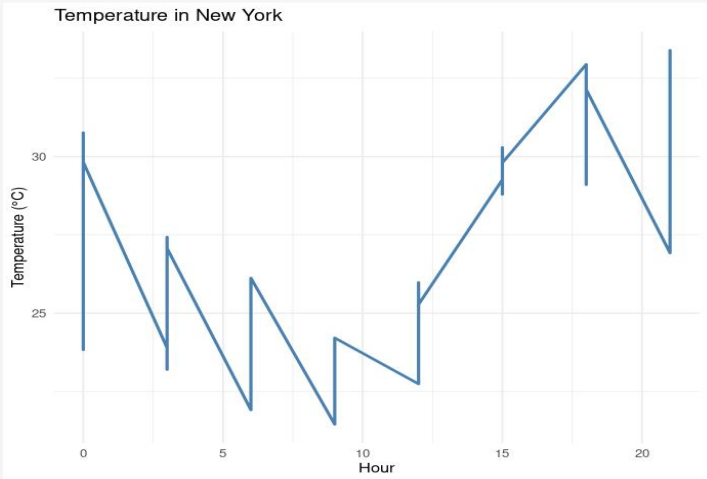
Dashboard

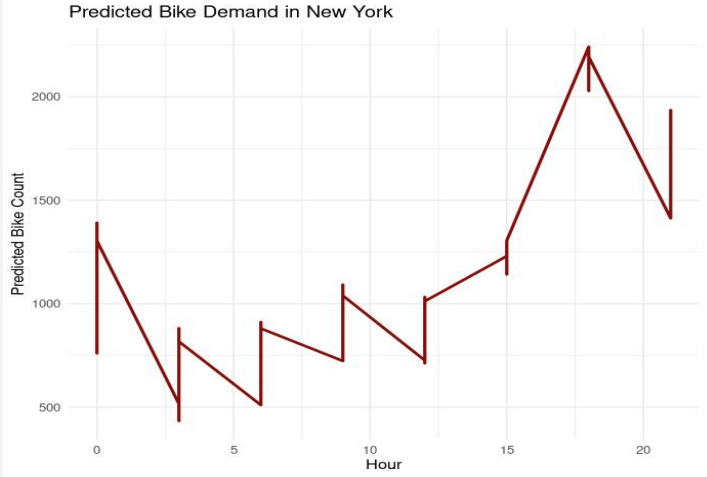
cities' max bike-sharing prediction on a map

Bike-sharing demand prediction app

Select City

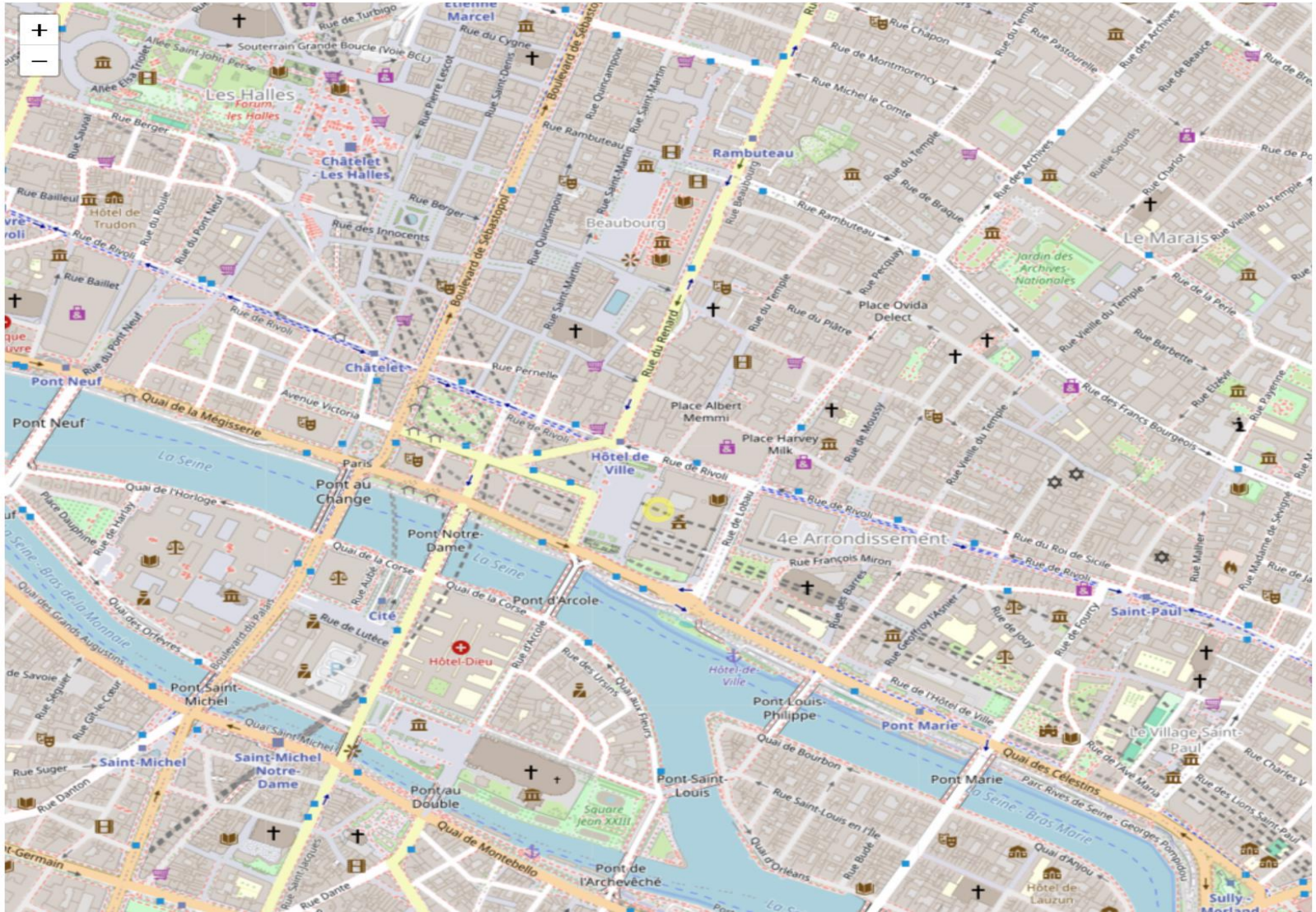
New York





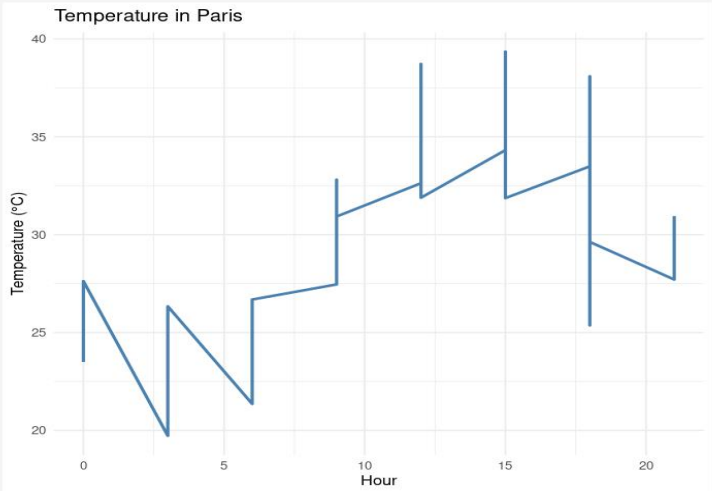
Showing one specific city

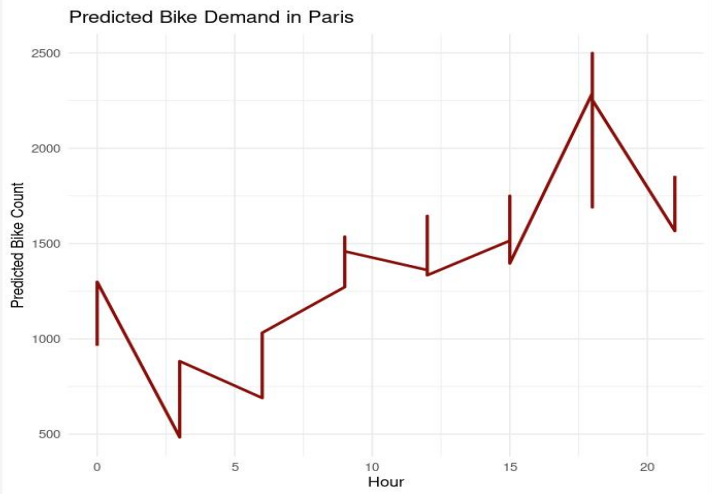
Bike-sharing demand prediction app



Select City

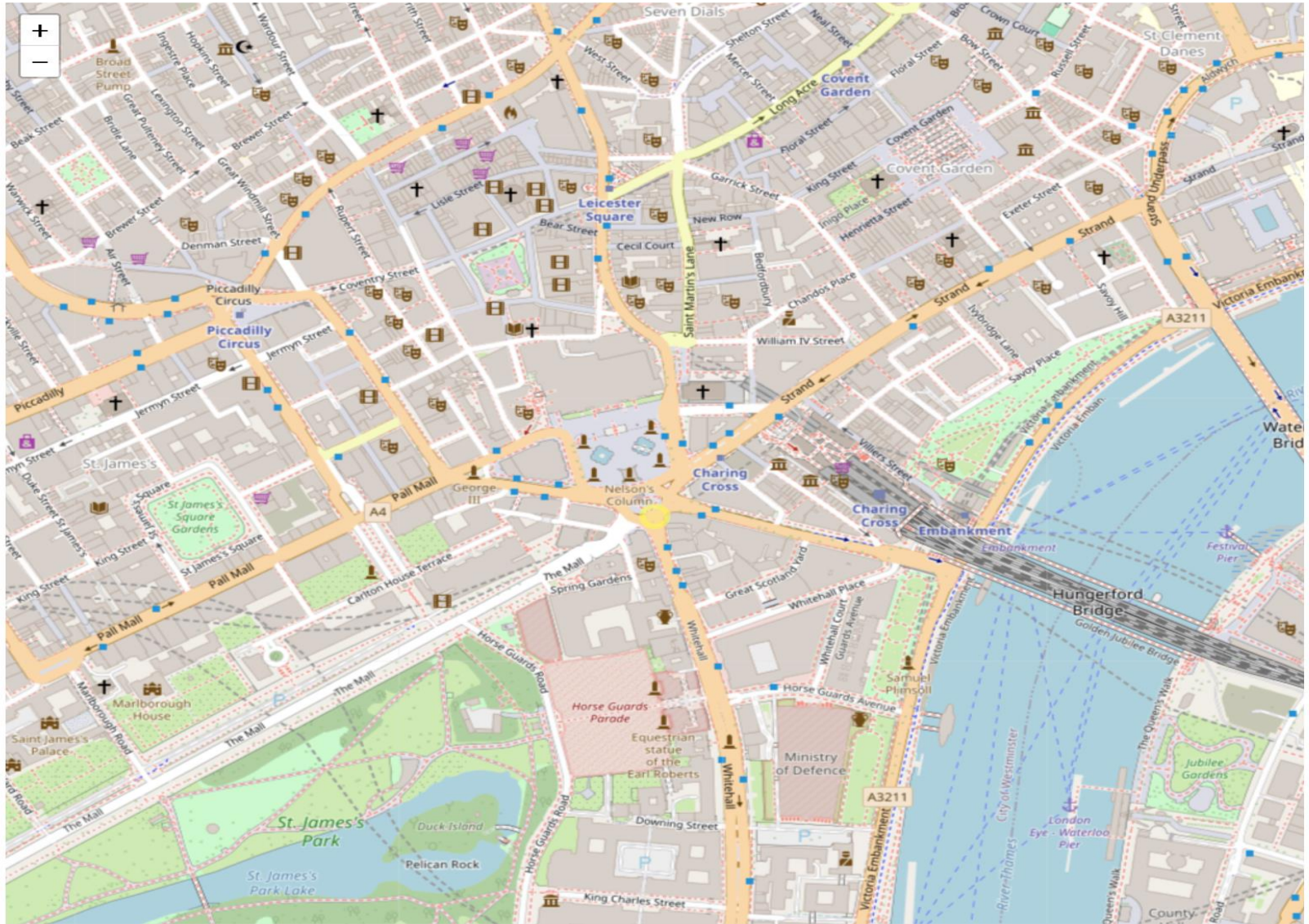
Paris





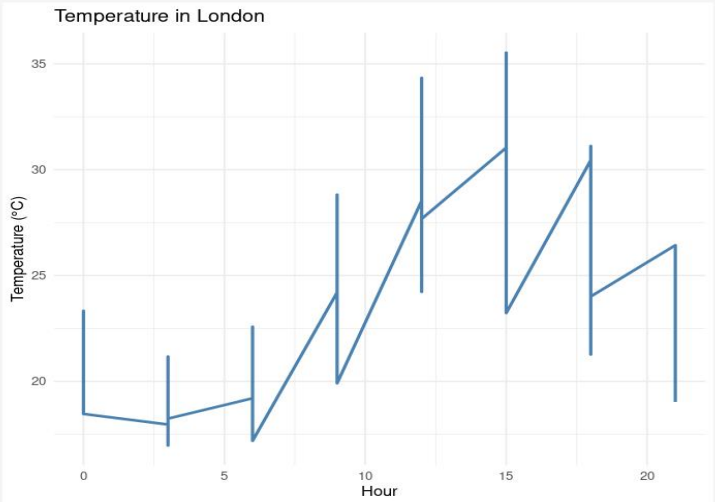
Showing one specific city

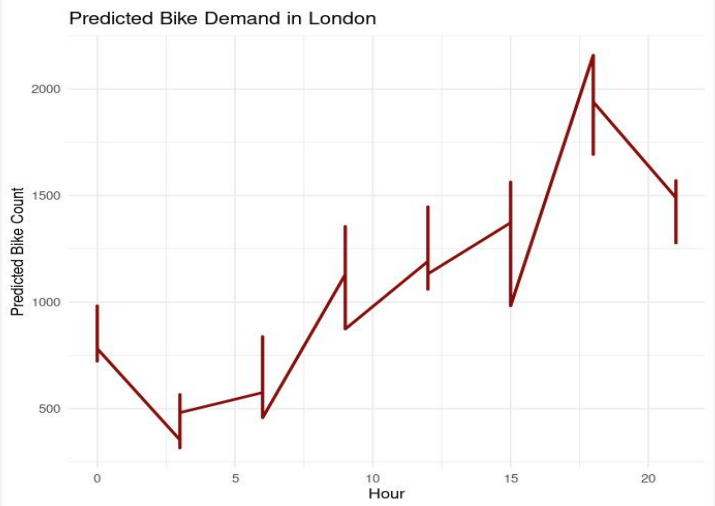
Bike-sharing demand prediction app



Select City

London





CONCLUSION

- We developed and compared **five regression models** to predict hourly bike rentals using features like temperature, humidity, and rainfall.
- **Model Evaluation Metrics:**
  - Used **Root Mean Squared Error (RMSE)** and **R-squared (R^2)** for performance evaluation.
  - Regularization and feature engineering (polynomials, interactions) were tested for performance gains.
- 🔍 **Best Performing Model:**
  - **[Insert model name here: e.g., "Ridge Regression" or "Elastic Net"]** achieved the **lowest RMSE** and/or **highest R^2**, indicating the best generalization on unseen data.
- 📌 **Key Takeaways:**
  - Incorporating polynomial and interaction terms can improve predictive accuracy.
  - Regularization (Ridge/Elastic Net) helps manage multicollinearity and overfitting.
  - The relationship between weather conditions and rental demand is non-linear and complex.

APPENDIX

```
# Load required libraries
require(shiny)
require(leaflet)

# Create a RShiny UI
shinyUI(
  fluidPage(padding = 5,
            titlePanel("Bike-sharing demand prediction app"),
            sidebarLayout(
              mainPanel(
                leafletOutput("city_bike_map", height = 1000)
              ),
              sidebarPanel(
                selectInput("selected_city", "Select City",
                            choices = c("All", "Seoul", "New York", "Paris", "London", "Suzhou")),
                plotOutput("weather_plot"),
                plotOutput("bike_plot")
              )
            )
  )
)
```