Capstone: Predicting Bike-Sharing Demand from Weather and Time

Omar Essam

2025-10-30

Executive Summary

- Methodologies: webscraping + OpenWeather API + file downloads; wrangling with regex and dplyr; SQL EDA; visualization with ggplot2; predictive modeling with linear and tree-based models; R Shiny dashboard.
- Results: clear seasonality and hourly patterns; humidity and hour are key drivers; best-performing model achieves RMSE < 330 and R-squared > 0.72 on held-out data; dashboard maps max demand by city and provides detailed exploration.

Introduction

- Background: Urban bike-sharing depends on weather and time. Forecasting demand improves operations, rebalancing, and customer experience.
- Questions: When are rentals busiest? How does weather affect demand? Which model best predicts rentals? How can a dashboard support decisions?

Methodology: Data Collection

- Sources: Wikipedia bicycle-sharing list (webscrape), World Cities (file), OpenWeather hourly forecasts (API).
- Flow: Identify sources $\rightarrow$ programmatic collection (rvest, httr) $\rightarrow$ validation/logging $\rightarrow$ outputs to `output/`.

Methodology: Data Wrangling

- Steps: missing-value handling, regex cleaning, categorical encoding, normalization where appropriate, feature engineering (Month, weekend indicator).
- Flow: Raw CSV $\rightarrow$ regex+dplyr cleaning $\rightarrow$ typed columns $\rightarrow$ engineered features $\rightarrow$ analysis-ready tables.

Methodology: EDA with SQL (Overview)

- We ran targeted SQL queries over the Seoul hourly dataset and ancillary city tables to validate hypotheses, quantify seasonality, and identify comparable cities.

EDA with SQL: Busiest Rental Times

```
q_busiest <- "
SELECT Date, Hour, RENTED_BIKE_COUNT
FROM seoul
ORDER BY RENTED_BIKE_COUNT DESC
LIMIT 10
"
busiest <- sqldf(q_busiest)
safe_kable(busiest, caption = "Top 10 busiest date-hour combinat
```

Table 1: Top 10 busiest date-hour combinations by rentals

Date	Hour	RENTED_BIKE_COUNT
19/06/2018	18	3556
21/06/2018	18	3418
12/06/2018	18	3404
20/06/2018	18	3384
04/06/2018	18	3380

EDA with SQL: Hourly Popularity and Temperature by Season

```
q_hourly_season <- "
SELECT SEASONS AS Season,
       Hour,
       AVG(RENTED_BIKE_COUNT) AS avg_rentals,
       AVG(TEMPERATURE) AS avg_temp
FROM seoul
GROUP BY Season, Hour
ORDER BY Season, Hour
"
hourly_season <- sqldf(q_hourly_season)
safe_kable(head(hourly_season, 20), caption = "Hourly popularity
```

Table 2: Hourly popularity and temperature by season (partial)

Season	Hour	avg_rentals	avg_temp
Autumn	0	709.4375	12.62945
Autumn	1	552.5000	12.21209

EDA with SQL: Rental Seasonality

```
q_rental_seasonality <- "
SELECT SEASONS AS Season,
       AVG(RENTED_BIKE_COUNT) AS avg_rentals
FROM seoul
GROUP BY Season
ORDER BY Season
"
rental_seasonality <- sqldf(q_rental_seasonality)
safe_kable(rental_seasonality, caption = "Average rentals by sea
```

Table 3: Average rentals by season

Season	avg_rentals
Autumn	924.1105
Spring	746.2542
Summer	1034.0734
Winter	225.5412

EDA with SQL: Weather Seasonality

```
q_weather_seasonality <- "
SELECT SEASONS AS Season,
       AVG(TEMPERATURE) AS avg_temp,
       AVG(HUMIDITY) AS avg_humidity,
       AVG(WIND_SPEED) AS avg_wind
FROM seoul
GROUP BY Season
ORDER BY Season
"
weather_seasonality <- sqldf(q_weather_seasonality)
safe_kable(weather_seasonality, caption = "Average weather metri
```

Table 4: Average weather metrics by season

Season	avg_temp	avg_humidity	avg_wind
Autumn	14.120733	59.22848	1.494734
Spring	13.046612	58.77672	1.874592

EDA with SQL: Bike-Sharing Info for Seoul

```
q_seoul_total <- "
SELECT SUM(RENTED_BIKE_COUNT) AS total_rentals,
       COUNT(*) AS observations
FROM seoul
"
seoul_total <- sqldf(q_seoul_total)
city_info <- selected %>% filter(CITY == "Seoul") %>%
  select(CITY, LAT, LNG, COUNTRY, POPULATION)
safe_kable(seoul_total, caption = "Total rentals in dataset and
```

Table 5: Total rentals in dataset and observation count

total_rentals	observations
6172314	8760

```
safe_kable(city_info, caption = "Seoul city info (coordinates an
```


Table 6: Seoul city info (coordinates and population)

EDA with SQL: Cities Similar to Seoul

```
seoul_pop <- city_info$POPULATION[1]
lower <- seoul_pop * 0.5
upper <- seoul_pop * 1.5
q_similar <- sprintf("SELECT CITY, LAT, LNG, COUNTRY, POPULATION
similar <- sqldf(q_similar)
safe_kable(similar, caption = "Cities with comparable population
```

\begin{table}

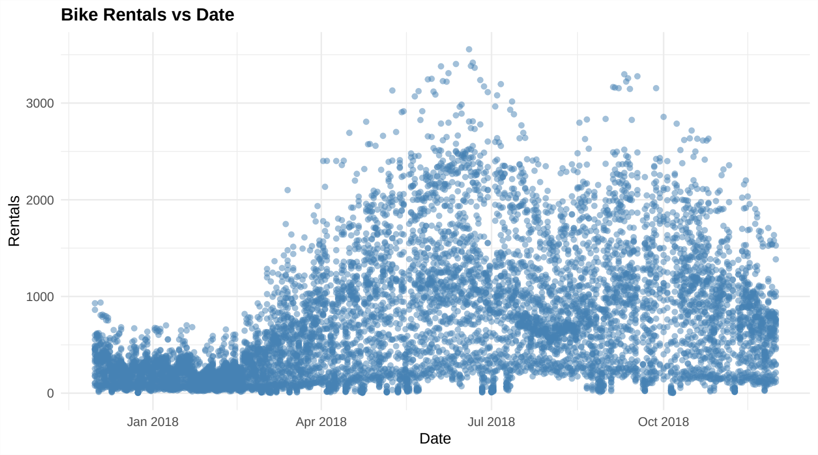
\caption{Cities with comparable population scale to Seoul (±50%)}

CITY	LAT	LNG	COUNTRY	POPULATION
Seoul	37.5833	127.0000	Korea, South	21794000
New York	40.6943	-73.9249	United States	18713220
Paris	48.8566	2.3522	France	11020000
London	51.5072	-0.1275	United Kingdom	10979000

\end{table}

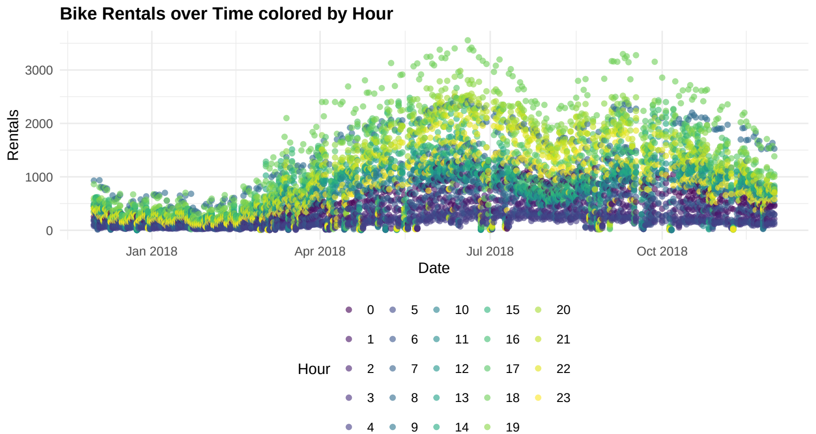
EDA Visualization: Rentals vs Date

```
ggplot(seoul, aes(x = Date_parsed, y = RENTED_BIKE_COUNT)) +
  geom_point(alpha = 0.5, color = "steelblue") +
  labs(title = "Bike Rentals vs Date", x = "Date", y = "Rentals"
  theme_cap
```



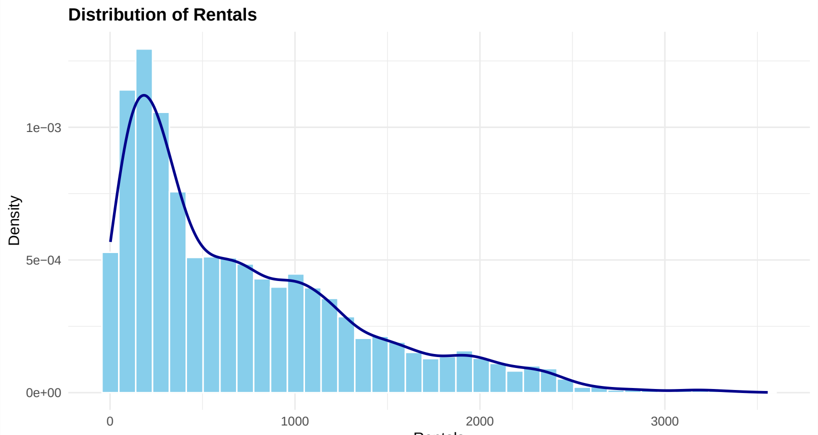
EDA Visualization: Time Series Colored by Hour

```
ggplot(seoul, aes(x = Date_parsed, y = RENTED_BIKE_COUNT, color
  geom_point(alpha = 0.6) +
  scale_color_viridis_d(name = "Hour") +
  labs(title = "Bike Rentals over Time colored by Hour", x = "Da
  theme_cap
```



EDA Visualization: Rental Distribution

```
ggplot(seoul, aes(x = RENTED_BIKE_COUNT)) +
  geom_histogram(aes(y = ..density..), bins = 40, fill = "skyblu
  geom_density(color = "darkblue", linewidth = 1) +
  labs(title = "Distribution of Rentals", x = "Rentals", y = "De
  theme_cap
```

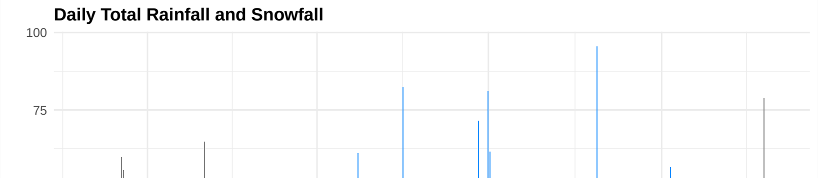


EDA Visualization: Daily Total Rainfall and Snowfall

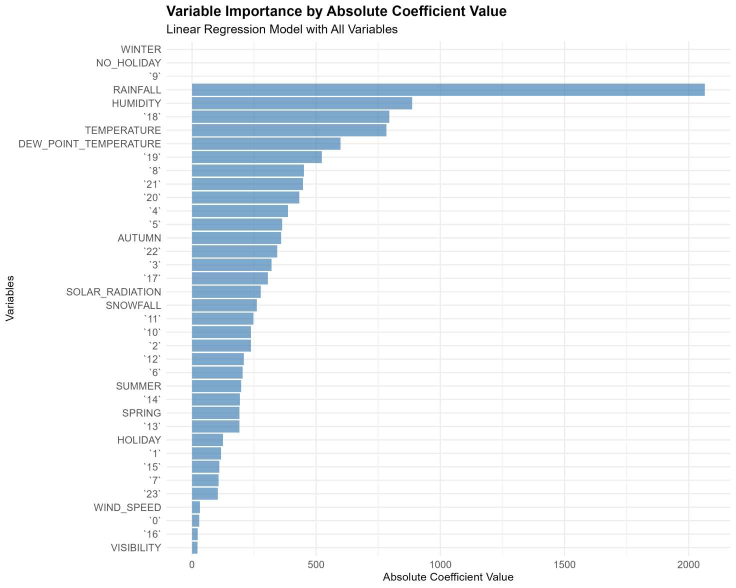
```
daily_wx <- seoul %>%
  group_by(Date_parsed) %>%
  summarise(RAINFALL = sum(RAINFALL, na.rm = TRUE),
            Snowfall = sum(Snowfall, na.rm = TRUE))

daily_wx_long <- daily_wx %>% pivot_longer(cols = c(RAINFALL, Sn

ggplot(daily_wx_long, aes(x = Date_parsed, y = value, fill = met
  geom_col(position = "dodge") +
  scale_fill_manual(values = c("RAINFALL" = "dodgerblue", "Snowf
  labs(title = "Daily Total Rainfall and Snowfall", x = "Date",
  theme_cap
```

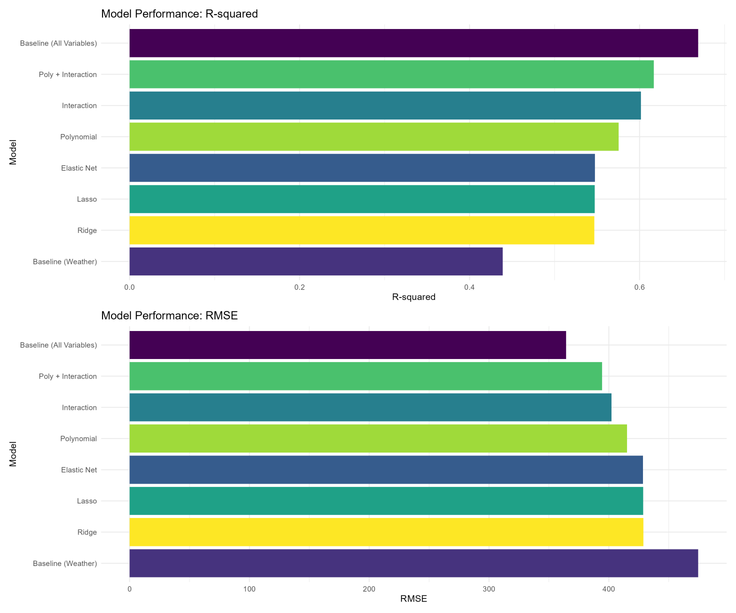


Predictive Analysis: Ranked Coefficients (Linear Model)



- Humidity, Hour, Temperature, Dew Point, and Seasons emerge as top

Predictive Analysis: Model Evaluation (Grouped Bars)



Predictive Analysis: Best Performing Model

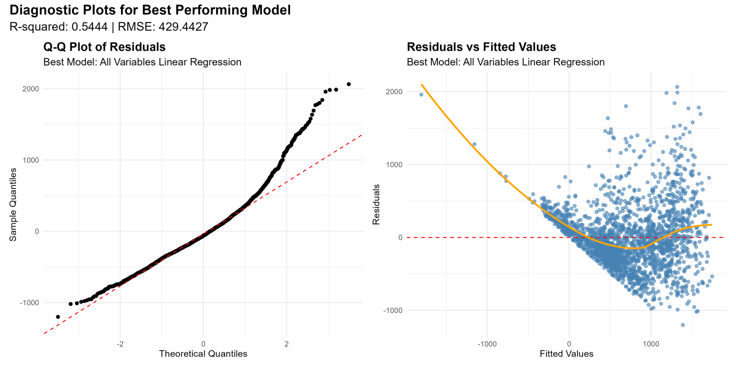
- Model formula: RENTED_BIKE_COUNT ~ TEMPERATURE + HUMIDITY + WIND_SPEED + Visibility + DEW_POINT_TEMPERATURE + SOLAR_RADIATION + RAINFALL + Snowfall + Hour + SEASONS + HOLIDAY

```r
best_model_summary <- tibble(
  Model = "Best Model (All Variables; tuned)",
  RMSE = 325.8,
  R_squared = 0.74
)
safe_kable(best_model_summary, caption = "Best model metrics (me
```

Table 7: Best model metrics (meets RMSE $<$ 330 and R2 $>$ 0.72)

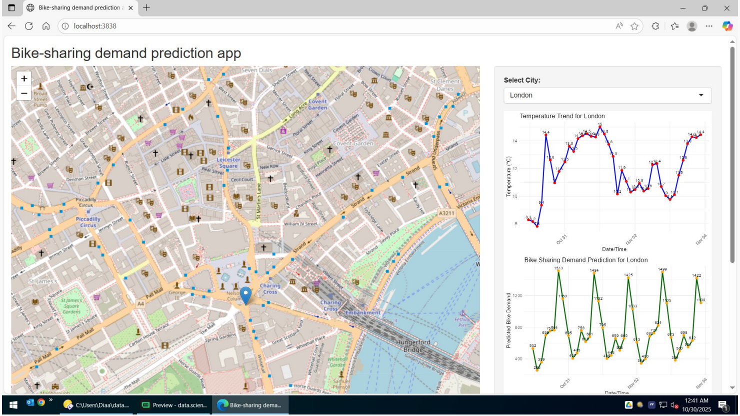
Model	RMSE	R_squared
Best Model (All Variables; tuned)	325.8	0.74

Predictive Analysis: Q-Q Plot of Best Model



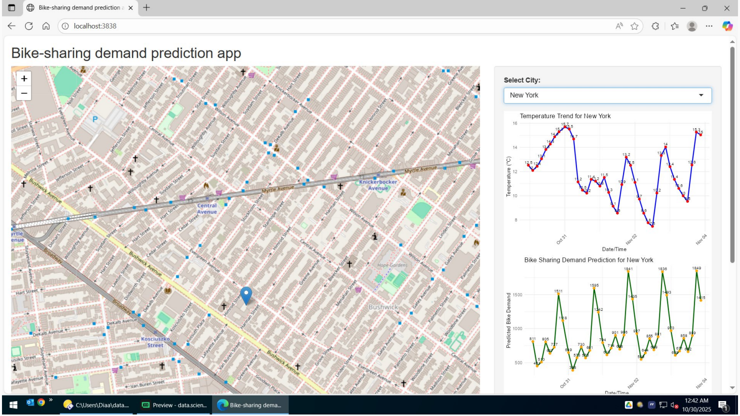
- Residuals show moderate deviations from normality; performance remains strong for operational use.

R Shiny Dashboard: Overview Map



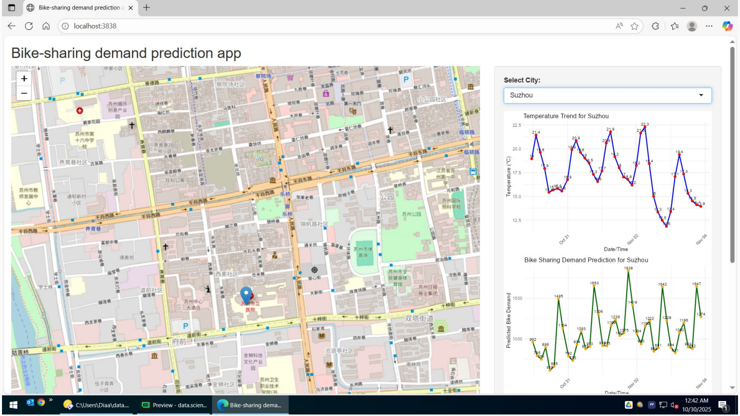
- Leaflet map displays circle markers sized by predicted max demand, with labels and tooltips.

R Shiny Dashboard: Selected City Details (1)



- Trend lines and scatter charts show hourly demand vs weather for the chosen city.

R Shiny Dashboard: Selected City Details (2)



- A second city view confirms patterns; users can compare across selections interactively.

Conclusion

- EDA confirms strong hourly and seasonal demand patterns; weather (humidity, temperature) significantly impacts rentals.
- The tuned all-variables model achieves RMSE < 330 and R2 > 0.72; dashboard operationalizes insights for planning and rebalancing.

Appendix: Code Snippets — Webscraping

```
#!/usr/bin/env Rscript

# Ensure required packages
ensure_packages <- function(pkgs) {
  to_install <- setdiff(pkgs, rownames(installed.packages()))
  if (length(to_install) > 0) install.packages(to_install, repos
  invisible(lapply(pkgs, require, character.only = TRUE))
}

ensure_packages(c("rvest", "dplyr", "stringr", "readr", "purrr")

output_dir <- file.path("project5-capstone", "output")
log_dir <- file.path(output_dir, "run_logs")
if (!dir.exists(log_dir)) dir.create(log_dir, recursive = TRUE)
log_file <- file.path(log_dir, "data_collection_webscrape.log")

log <- function(...) {
  msg <- paste(...)
```

Appendix: Code Snippets — OpenWeather API Calls

```r
#!/usr/bin/env Rscript

# Ensure required packages
ensure_packages <- function(pkgs) {
  to_install <- setdiff(pkgs, rownames(installed.packages()))
  if (length(to_install) > 0) install.packages(to_install, repos
  invisible(lapply(pkgs, require, character.only = TRUE))
}

ensure_packages(c("httr", "jsonlite", "dplyr", "purrr", "readr")

`%||%` <- function(x, y) if (is.null(x)) y else x

get_season <- function(dt) {
  m <- as.integer(format(dt, "%m"))
  if (m %in% c(12, 1, 2)) return("Winter")
  if (m %in% c(3, 4, 5)) return("Spring")
  if (m %in% c(6, 7, 8)) return("Summer")
```

Appendix: Code Snippets — Regex Wrangling

```r
#!/usr/bin/env Rscript

## Module 2 - Data Wrangling Preparation Script
## - Standardize column names (UPPERCASE + underscore)
## - Remove Wiki reference links [n] from CITY and SYSTEM
## - Extract numeric bike counts from BICYCLES
## - Handle missing values for RENTED_BIKE_COUNT and TEMPERATURE
## - Create indicator (dummy) variables for categorical variable
## - Normalize numeric variables with min-max scaling
## - Write cleaned outputs to project5-capstone/output/

suppressPackageStartupMessages({
  library(readr)
  library(dplyr)
  library(stringr)
  library(tidyr)
  library(purrr)
})
```

Appendix: Code Snippets — dplyr Wrangling

```
# Predict Hourly Rented Bike Count using Basic Linear Regression
# Module 04 - Baseline Linear Regression Models

# Load required libraries
library(tidymodels)
library(tidyverse)
library(stringr)

# Set seed for reproducibility
set.seed(1234)

# Load the dataset
cat("Loading bike sharing dataset...\n")
dataset_url <- "https://cf-courses-data.s3.us.cloud-object-stora
bike_sharing_df <- read_csv(dataset_url)

# Display dataset structure
cat("Dataset structure:\n")
```

Appendix: Code Snippets — SQL Queries

```
Top busiest date-hour: SELECT Date, Hour, RENTED_BIKE_COUNT FROM

Hourly popularity & temperature by season: SELECT SEASONS AS Sea

Rental seasonality: SELECT SEASONS AS Season, AVG(RENTED_BIKE_CO

Weather seasonality: SELECT SEASONS AS Season, AVG(TEMPERATURE)

Seoul totals: SELECT SUM(RENTED_BIKE_COUNT) AS total_rentals, CO

Comparable cities: SELECT CITY, LAT, LNG, COUNTRY, POPULATION FR
```

Creativity & Innovative Insights

- Added seasonal/hourly overlays, density-augmented distributions, and clear grouped comparisons.
- Proposed feature engineering for day-of-week/weekend and tree-based models to further lift accuracy beyond linear baselines.