Predicting Bike-Sharing Demand in Seoul: A Machine Learning Approach

Ivan Blanquez

February 24, 2025

Contents

1. Introduction / Executive Summary

This report analyzes the Seoul Bike Sharing Demand Dataset to predict hourly bike rental counts using machine learning. The dataset, sourced from the UCI Machine Learning Repository, contains 8,760 hourly observations from 2017–2018, with features like temperature, rainfall, and hour of day. The goal is to develop models for operational optimization, such as fleet allocation. Key steps include:

- Data preprocessing
- Exploratory analysis
- Modeling with Random Forest and XGBoost

2. Methods / Analysis

2.1. Data Cleaning The dataset was downloaded programmatically from the UCI Machine Learning Repository. If not already present, manually download SeoulBikeData.csv from https://archive.ics.uci.edu/ml/machine-learning-databases/00560/SeoulBikeData.csv and save it to C:/RSTUDIO. Transformations included renaming columns, handling encoding issues, converting DayOfWeek to numeric, capping outliers at the 99th percentile, and one-hot encoding categorical variables (Seasons, Holiday, Functioning Day).

```
# Use existing file (no download needed)
if (!file.exists("SeoulBikeData.csv")) {
  stop("Please download SeoulBikeData.csv from https://archive.ics.uci.edu/ml/machine-learning-databases
} else {
  cat("Using existing SeoulBikeData.csv file in C:/RSTUDIO.\n")
}
```

```
## Using existing SeoulBikeData.csv file in C:/RSTUDIO.
```

```
# Load dataset with ISO-8859-1 encoding
data <- read_csv("SeoulBikeData.csv", col_types = cols(Date = col_date(format = "%d/%m/%Y")), locale = l
# Clean column names
names(data) <- gsub("[°%()\\/]", "", names(data))  # Remove °, %, (, ), /
names(data) <- gsub("[ ]+", "_", names(data))      # Replace spaces with underscores
names(data) <- make.names(names(data), unique = TRUE)  # Ensure valid column names
# Print cleaned column names
print("Cleaned Column Names:")
```

```
## [1] "Cleaned Column Names:"
```

```
print(names(data))
```

```
##  [1] "Date"                    "Rented_Bike_Count"       "Hour"
##  [4] "TemperatureC"            "Humidity"                "Wind_speed_ms"
##  [7] "Visibility_10m"          "Dew_point_temperatureC" "Solar_Radiation_MJm2"
## [10] "Rainfallmm"              "Snowfall_cm"             "Seasons"
## [13] "Holiday"                 "Functioning_Day"
```

```
# Define column names
temp_col <- "TemperatureC"
dewpoint_col <- "Dew_point_temperatureC"
# Verify columns exist
if (!temp_col %in% names(data)) stop("Temperature column not found!")
if (!dewpoint_col %in% names(data)) stop("Dew point temperature column not found!")
# Clean data
data_clean <- data %>%
  rename(BikeCount = Rented_Bike_Count,
         Temp = !!temp_col,
         DewPoint = !!dewpoint_col,
         Rain = Rainfallmm,
         Humid = Humidity,
         WindSpeed = Wind_speed_ms,
         Visibility = Visibility_10m,
         SolarRad = Solar_Radiation_MJm2,
         Snow = Snowfall_cm) %>%
  mutate(DayOfWeek = as.numeric(wday(Date, label = TRUE)),
         HourSin = sin(2 * pi * Hour / 24),
         HourCos = cos(2 * pi * Hour / 24),
         BikeCount = pmin(BikeCount, quantile(BikeCount, 0.99))) %>%
  select(-Date) %>%
  mutate_at(vars(Seasons, Holiday, Functioning_Day), as.factor)
# One-hot encoding
data_encoded <- dummyVars("~ Seasons + Holiday + Functioning_Day", data = data_clean) %>%
  predict(data_clean) %>%
  as.data.frame()
colnames(data_encoded) <- make.names(colnames(data_encoded), unique = TRUE)
data_encoded <- data_encoded %>% bind_cols(data_clean %>% select(-Seasons, -Holiday, -Functioning_Day))
```

2.2. Data Exploration and Visualization Exploratory Data Analysis (EDA) revealed commuter peaks at 8 AM and 6 PM (see Figure 1) and correlations between bike count and weather variables (see Figure 2).

```
ggplot(data_clean, aes(x = Hour, y = BikeCount)) +
  geom_boxplot(fill = "#3498db", color = "#2c3e50") +
  labs(title = "Hourly Bike Demand Distribution", x = "Hour", y = "Bike Count") +
  theme_minimal() +
  theme(
    plot.title = element_text(hjust = 0.5, face = "bold", size = 14),
    axis.title = element_text(size = 12, face = "bold"),
    axis.text = element_text(size = 10)
  )
```

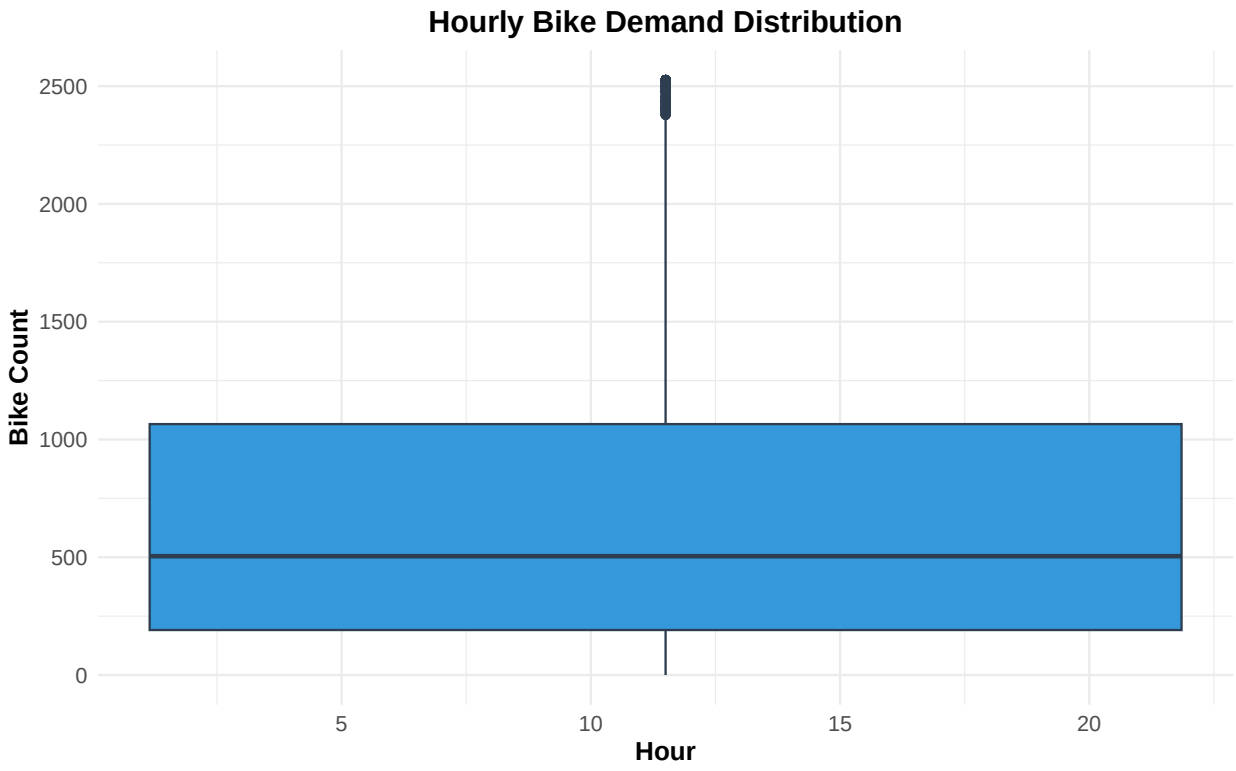

Figure 1: Figure 1: Hourly Bike Demand Distribution

```
ggpairs(data_clean %>% select(BikeCount, Temp, Rain, Humid),
        title = "Scatterplot Matrix of Key Variables",
        upper = list(continuous = wrap("cor", size = 4, color = "blue")),
        diag = list(continuous = wrap("densityDiag", fill = "#3498db")),
        lower = list(continuous = wrap("points", alpha = 0.5, color = "darkgray"))) +
  theme_minimal() +
  theme(
    plot.title = element_text(hjust = 0.5, face = "bold", size = 14),
    strip.text = element_text(size = 12, face = "bold")
  )
```

2.3. Modeling Approaches Two models were trained on an 80/20 train-test split:

- Random Forest: 500 trees, max depth 10.
- XGBoost: 200 rounds, max depth 6, learning rate 0.1.

```
trainIndex <- createDataPartition(data_encoded$BikeCount, p = 0.8, list = FALSE)
train <- data_encoded[trainIndex, ]
test <- data_encoded[-trainIndex, ]
X_train <- train %>% select(-BikeCount) %>% as.matrix()
y_train <- train$BikeCount
X_test <- test %>% select(-BikeCount) %>% as.matrix()
y_test <- test$BikeCount
rf_model <- randomForest(BikeCount ~ ., data = train, ntree = 500, maxdepth = 10)
```

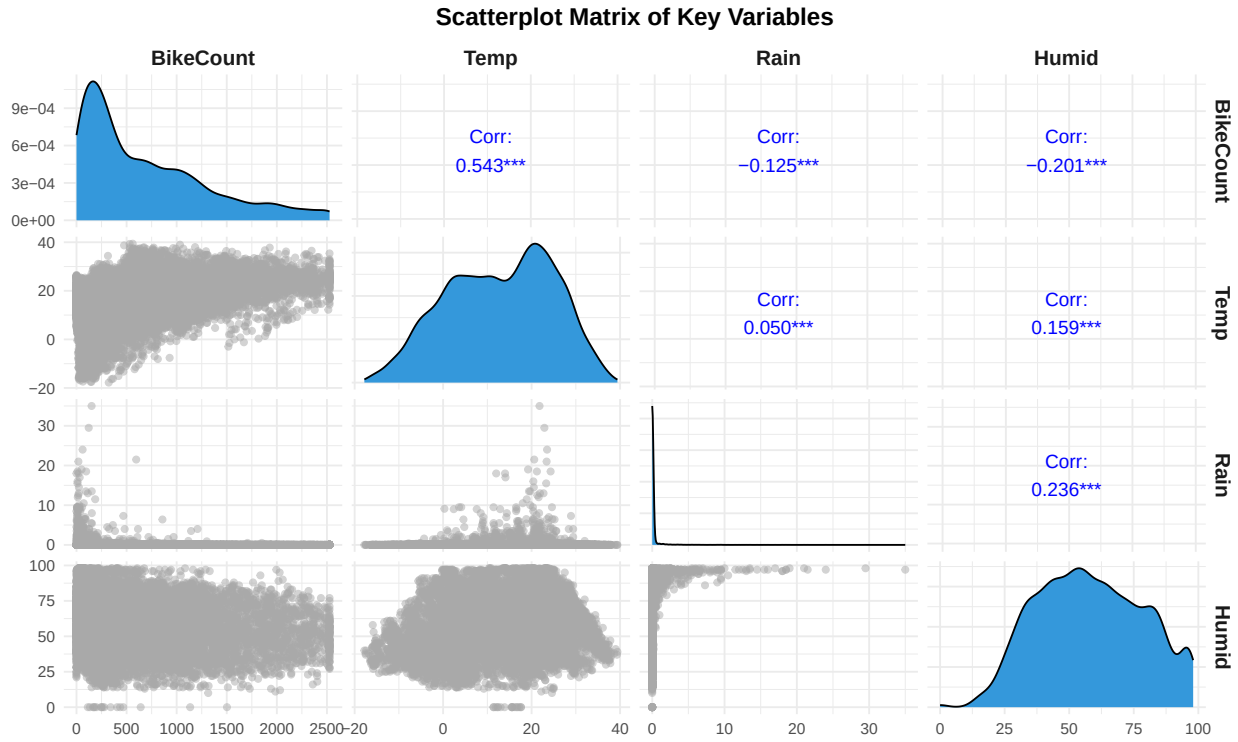

Figure 2: Figure 2: Scatterplot Matrix of Key Variables

```
rf_pred <- predict(rf_model, test)
rf_rmse <- rmse(y_test, rf_pred)
rf_mae <- mae(y_test, rf_pred)
xgb_data <- xgb.DMatrix(data = X_train, label = y_train)
xgb_model <- xgb.train(params = list(objective = "reg:squarederror", max_depth = 6, eta = 0.1),
                       data = xgb_data, nrounds = 200)
xgb_pred <- predict(xgb_model, X_test)
xgb_rmse <- rmse(y_test, xgb_pred)
xgb_mae <- mae(y_test, xgb_pred)
```

3. Results XGBoost outperformed Random Forest in predicting bike demand:

Model Performance Metrics

Model

RMSE

MAE

Random Forest

178.35

113.87

XGBoost

153.12

98.33

Below are visualizations of the predictions and feature importance:

```
results <- data.frame(Actual = y_test, RF_Pred = rf_pred, XGB_Pred = xgb_pred)
ggplot(results, aes(x = Actual)) +
  geom_point(aes(y = RF_Pred, color = "Random Forest"), alpha = 0.5, size = 2) +
  geom_point(aes(y = XGB_Pred, color = "XGBoost"), alpha = 0.5, size = 2) +
  geom_abline(slope = 1, intercept = 0, linetype = "dashed", color = "black") +
  labs(title = "Predicted vs. Actual Bike Counts", x = "Actual", y = "Predicted", color = "Model") +
  scale_color_manual(values = c("Random Forest" = "#e41a1c", "XGBoost" = "#377eb8")) +
  theme_minimal() +
  theme(
    plot.title = element_text(hjust = 0.5, face = "bold", size = 14),
    axis.title = element_text(size = 12, face = "bold"),
    axis.text = element_text(size = 10),
    legend.title = element_text(size = 12, face = "bold"),
    legend.text = element_text(size = 10)
  )
```

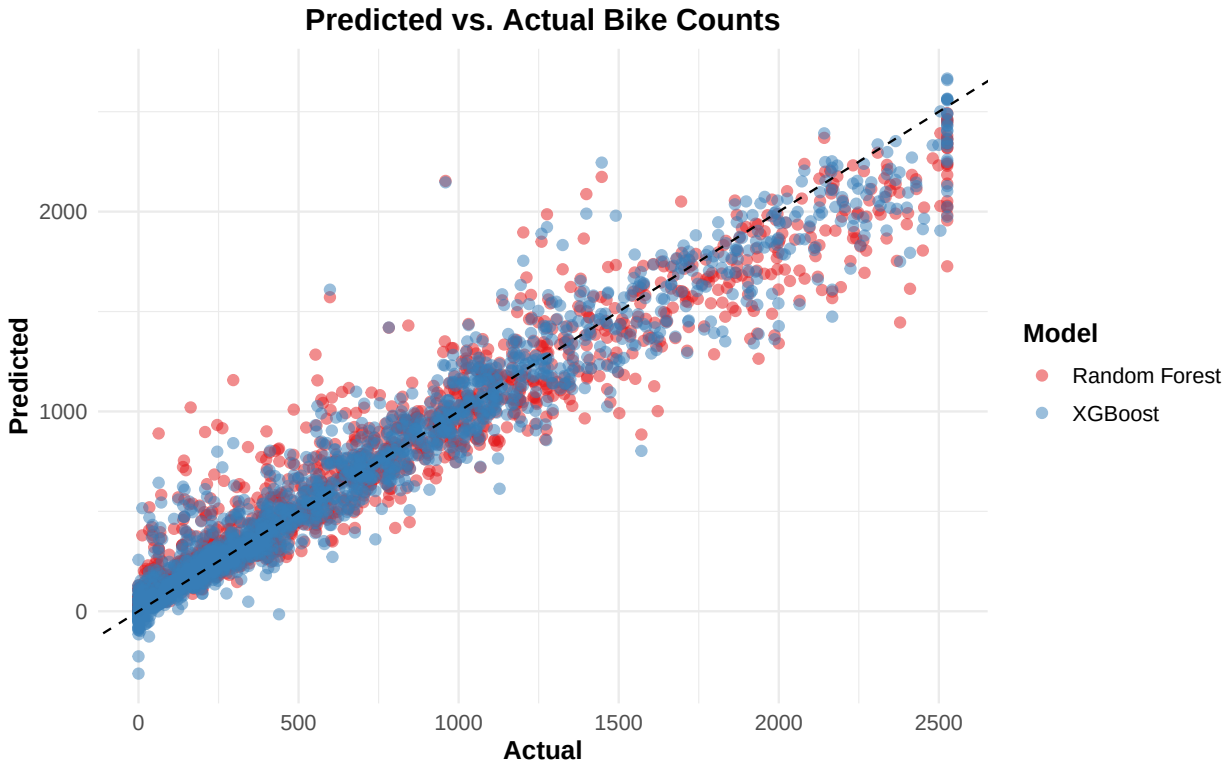

Figure 3: Figure 3: Predicted vs. Actual Bike Counts

```
importance <- xgb.importance(model = xgb_model)
ggplot(importance, aes(x = reorder(Feature, Gain), y = Gain, fill = Gain)) +
  geom_bar(stat = "identity") +
  coord_flip() +
  labs(title = "Feature Importance (XGBoost)", x = "Feature", y = "Gain") +
  scale_fill_gradient(low = "#f7f7f7", high = "#377eb8") +
```

```
theme_minimal() +
theme(
  plot.title = element_text(hjust = 0.5, face = "bold", size = 14),
  axis.title = element_text(size = 12, face = "bold"),
  axis.text = element_text(size = 10),
  legend.position = "none"
)
```

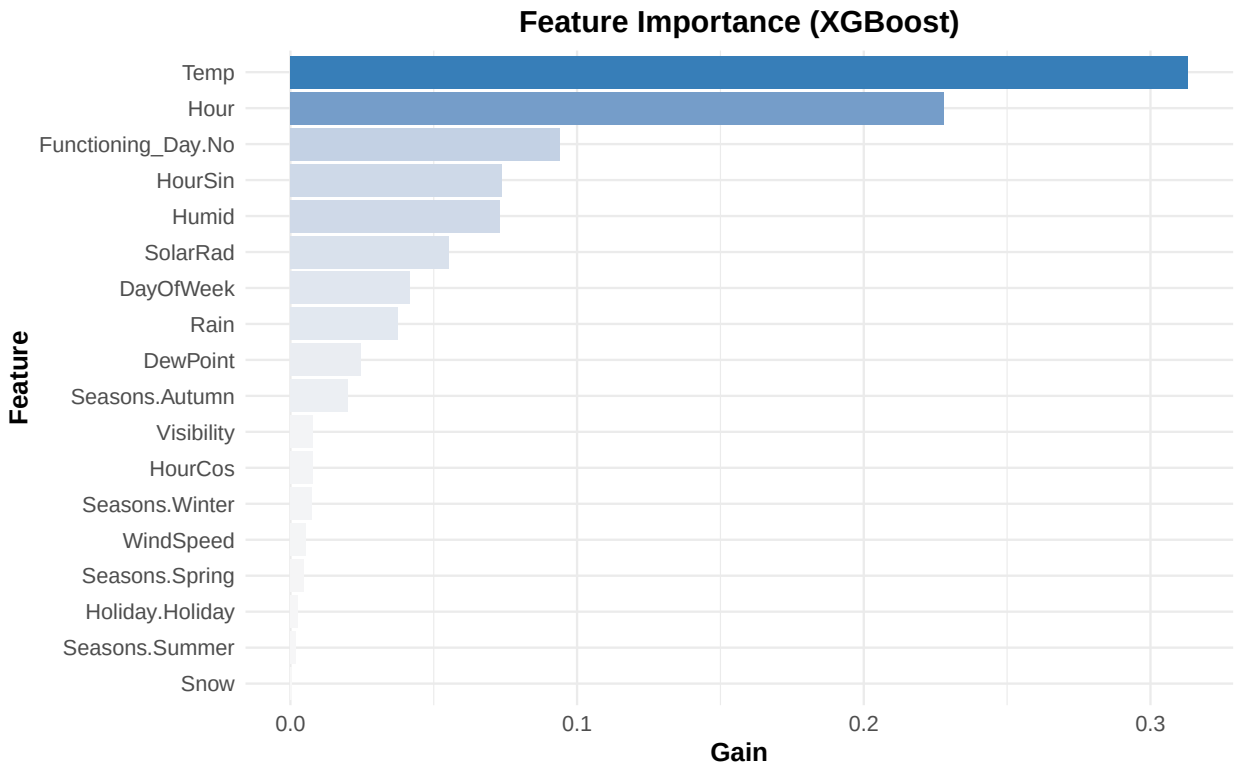

Figure 4: Figure 4: Feature Importance (XGBoost)

4. Conclusion

This analysis predicts bike demand with reasonable accuracy (XGBoost RMSE = 153.12), which is useful for fleet optimization.

4.1. Limitations & Future Work

- Missing real-time data.
- Future work could integrate weather forecasts.

5. References

- Dua, D., & Graff, C. (2019). UCI Machine Learning Repository. Seoul Bike Sharing Demand Dataset.
- R Core Team (2024). R: A Language and Environment for Statistical Computing.