IE 542 Project 1

Predicting Seoul Bike Sharing Demand

Jastej Singh

Prof. Jeoungyoun Ahn

Abstract—[1]This report presents an exploratory data analysis of Seoul's bike-sharing system using hourly rental and weather data from one year to predict the demand for bicycles in Seoul using (a) Multiple Linear regression and (b) Gradient Boosting Algorithm. The primary objective is to identify patterns and factors that influence rental demand, such as time, temperature, and precipitation. Through data visualisation, transformation techniques, and statistical testing, the analysis reveals significant temporal and weather-driven trends in usage. The findings suggest opportunities to improve demand forecasting and operational planning. Future work includes building predictive models and incorporating external factors for a more comprehensive understanding of usage behaviour.

Keywords—*Linear Model, Gradient Boosting, Performance, EDA, Feature Selection, Bike Demand*

1. Introduction

This project analyses bike rental data in Seoul to uncover patterns and factors affecting demand. Urban bike-sharing systems offer eco-friendly and flexible transport solutions. Understanding how demand varies across time and weather conditions can help city planners and service providers optimise operations, predict usage, and allocate resources more effectively. Therefore, this research aims to use machine learning and data mining-based algorithms to predict the required number of rental bikes at each hour.

2. Data Description

The dataset used for this project, titled SeoulBikeData.csv, was collected over the course of a full calendar year and contains hourly records of bike rental counts in Seoul. It provides a rich set of features that allow for the exploration of both temporal and environmental factors influencing rental demand.

2.1. Structure and size

The dataset contains a total of 8,760 observations, corresponding to 24 hourly entries for each day across 365 days. Each row represents the number of bikes rented during a specific hour of a given day, along with associated weather and time-related information.

2.2. Target Variable

Rented Bike Count: This is the primary variable of interest. It represents the total number of bikes rented during a given hour. Initial visual inspection showed a right-skewed distribution, indicating a large number of low-demand periods with a few hours of very high usage.

2.3. Feature Variable

The data set includes a variety of predictor variables. Table 1 shows the list of variables and that there are no missing data for any predictor variable. I use abbreviations as mentioned in the table for the code, and later on, do appropriate transformations on the categorical variables and highly skewed data.

2.4. Data Preparation and EDA

rent_count: We plot a histogram plot(Fig. 1 and 2) for our target variable rent_count which is highly skewed to the right, thus we do log transformation of $y = \log(100+y)$. The reason for this transformation is to make the data look more normal. It helps stabilise variance, reduces skewness, and can improve model performance by linearising relationships. There is no specific reason to choose 100; using smaller values like 1 to 10 did not improve the skewness a lot because of the many 0s in the target variable, so we use an arbitrarily large number.

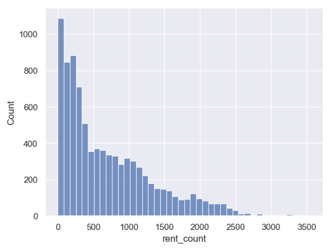


Figure 1. Histogram of rent_count

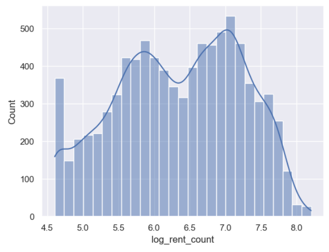


Figure 2. Log-transformed rent_count.

Predictor Variables: We use 12 variables: *hour, temp, humid, wind_speed, visib, dew_temp, radiation, rainfall, snowfall, seasons, Holiday, Fday* of which 3 are categorical, and we make 5 dummy binary variables for the analysis of categorical variables and there are 9 count variables. The box plot and a scatter plot of the count variables are shown in Fig. 3. The data for visib, radiation, rainfall, and snow

Table 1. Data variables and description.

Feature	Abbreviation	Type	Measurement	Count
Date	date	Date (yyyy-mm-dd)	–	8760
Rented Bike Count	rent_count	Continuous	0, 1, ..., 3556	8760
Hour	hour	Continuous	0 to 23	8760
Temperature	temp	Continuous	°C	8760
Humidity	humid	Continuous	%	8760
Wind Speed	wind_speed	Continuous	m/s	8760
Visibility	visib	Continuous	10 m	8760
Dew Point Temperature	dew_temp	Continuous	°C	8760
Solar Radiation	radiation	Continuous	MJ/m^2	8760
Rainfall	rainfall	Continuous	mm	8760
Snowfall	snowfall	Continuous	cm	8760
Season	Seasons	Categorical	Spring, Summer, Fall, Winter	8760
Holiday	Holiday	Categorical	Yes, No	8760
Functioning Day	Fday	Categorical	Yes, No	8760

Note: All features had complete data with no missing values. Some features were transformed or encoded for analysis.

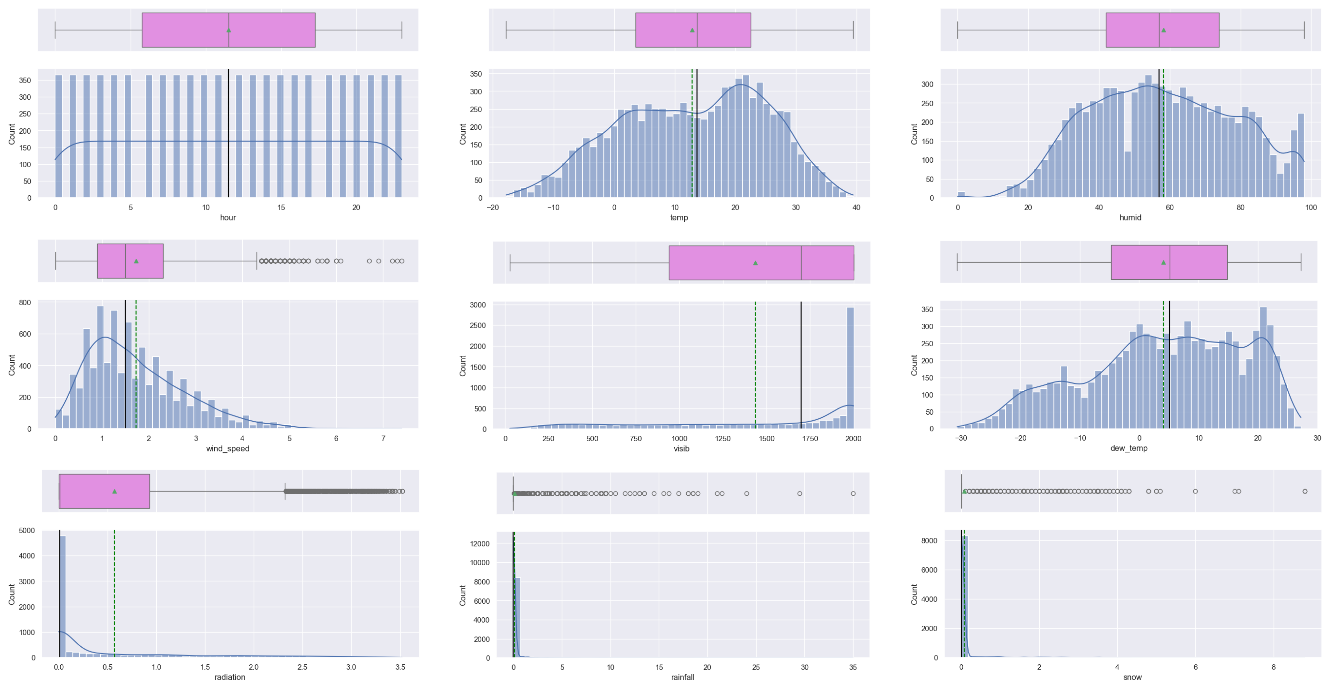


Figure 3. 3 × 3 scatterplot and boxplot matrix using individual plots.

is very skewed. For safety purposes and better model prediction, we remove the tail 3% data for each count variable. After the removal of these outliers, we are left with 7326 data points.

We then analyse the correlation between the count variables which is shown in Fig. 4. There is a high correlation between the variables *temperature* and *dew_temp* around 0.91, which suggests that we should only take one of them in our feature space. Taking both of them will lead to a bigger confidence interval of the coefficients of these variables. In the next section, we will try to rank the importance of variables using random forest and PCA techniques.

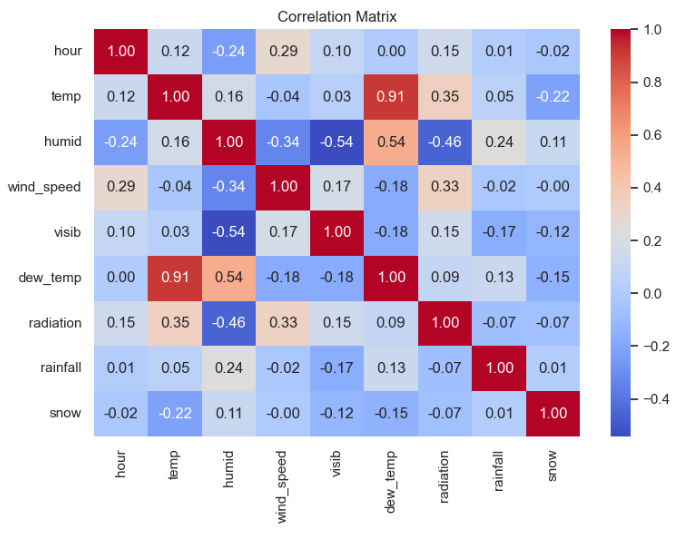


Figure 4. Correlation Matrix between count variables

Next, we work on converting the categorical data into dummy variables. For example, a variable like *Seasons* has four possible outcomes. In regression, we will encounter the problem of multicollinearity if we take all four dummy variables. Thus, the autumn season is taken as the base dummy variable, and for the remaining 3 seasons (*summers, winters, spring*), binary dummy variables are created and encoded with 0-1. A similar procedure is done for variables *Holiday, Functioning Day* to make a binary dummy variable which equals when the feature is True. We finally made a list of 14 variables for our analysis.

3. Feature Engineering

We use two methods to find the best features for the linear model that explain most of the variation in R^2 values. The first one is a very old model called Principal Component Analysis (PCA), and the other is Random Forest Regression.

3.1. Prinicipal Component Analysis

Principal Component Analysis (PCA) is a statistical technique used for dimensionality reduction. It transforms a large set of correlated variables into a smaller set of uncorrelated variables called principal components, which capture the most significant patterns (or variance) in the data. The goal of PCA is to reduce the number of variables while preserving as much information (variance) as possible. Fig.2. We can observe that both dew_temp and temp variables are the most important features in the data because they are highly correlated; it is important to choose only one of them for analysis.

#	Feature	Value
1	dew_temp	0.529693
2	temp	0.504468
3	Season_Winter	0.433296
4	Season_Summer	0.384348
5	humid	0.270888
6	snow	0.161331
7	radiation	0.097866
8	wind_speed	0.091393
9	visib	0.068453
10	rainfall	0.058290
11	holiday_No Holiday	0.055283
12	functioning_day_Yes	0.030553
13	Season_Spring	0.005612
14	hour	0.005011

Table 2. Feature Importance or PCA Component Weights

3.2. Random Forest Regression

A Random Forest is an ensemble of decision trees. Each tree splits the data to reduce the prediction error, and the forest combines the outputs of all the trees. Feature importance is computed based on how much each feature contributes to reducing the error (typically Mean Squared Error (MSE)) across all the trees. A feature with a high importance score means it was frequently used and made significant improvements to the model's accuracy. Fig 5 shows a graph of feature importance found using the random forest regression Trees. Again, just the top 3 features *temp, hour, functioning_day_Yes* are able to explain most of the variation in the data.

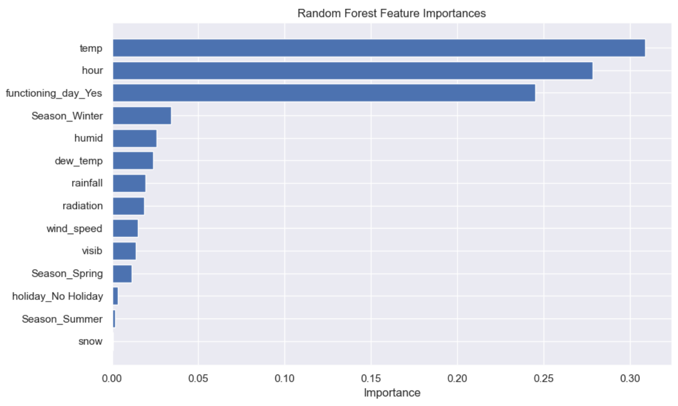


Figure 5. Feature Importance/weights using random forest

4. Methodology

4.1. Multiple Linear Regression

The Linear Regression(LM) Model is the simplest method to make predictions about some variables using a set of features X. We assume the model to hold a linear relationship of the form in eq. 1. Linear regression refers to a system where the conditional mean of Y is an affine function of X, given the value of X.

$$Y = \beta_0 + \beta X + \epsilon \tag{1}$$

X is the set of features that we described above. The coefficients β_0 and β are model parameters that describe the effect of a unit change in a feature on the predictor variable, and ϵ is the random error.

We want to measure the predictive power of this model, for which we use statistics such as R^2 and MSE (mean squared error). The linear regression model is the simplest model for interpreting the coefficients. We chronologically divide the data into train and test by an 80:20 ratio, and then we fit our whole model first using the most important 7 predictors, and then the full model.

4.2. Gradient Boosting Machine

Gradient boosting is a very popular algorithm which uses weak models to begin with and then starts giving higher weights to the points that have high residuals in the training data after each iteration. This boosting method has won many Kaggle competitions.

This method is built on Regression Trees, where we have tuned the hyperparameters using a K-fold cross-validation approach. The main hyperparameters to decide are the depth of the tree, the number of trees/iterations and the learning rate(α). For training, again the same chronological split is made with the first 80% of data used for training the model. K-fold cross-validation is performed on the training set with K=5. This cross-validation approach averages the predicted value on the training set by dividing the training set into K=5 sets and using one for testing in each iteration. For measurement of predictive power, we use the same statistics used for the Linear Regression model, i.e. R^2 and MSE (mean squared error).

5. Result and Analysis

5.1. Multiple Linear Regression Model

After getting the best features from the feature selection section. We fit both the full and reduced model and compare them on how good they explain the model and the MSE in both of them. The values are shown in Table 3. Note that the target variable y is log_rent_count and not rent_count because taking the log makes the distribution more normal. For the Reduced model, we have taken the following 7 features: *rainfall, hour, functioning_day_Yes, humid, Season_Winter, temp, radiation*. We dropped *dew_temp* because of its high correlation with *temp*. We can clearly see from the Table below that there isn't a huge difference in R^2 values obtained from reduced and full model on both training and testing data. We can see from Fig. 6 the relation between y and $\hat{y}$. There is a straight line due to 0s in the rent count data. Other than that, it approximately follows a straight line y=x for values greater than 0.

Table 3. Model's performance.

Models	Training		Testing	
	R^2	MSE	R^2	MSE
Reduced Model	0.6870	0.2148	0.7156	0.2491
Full Model	0.6933	0.2104	0.7432	0.2249

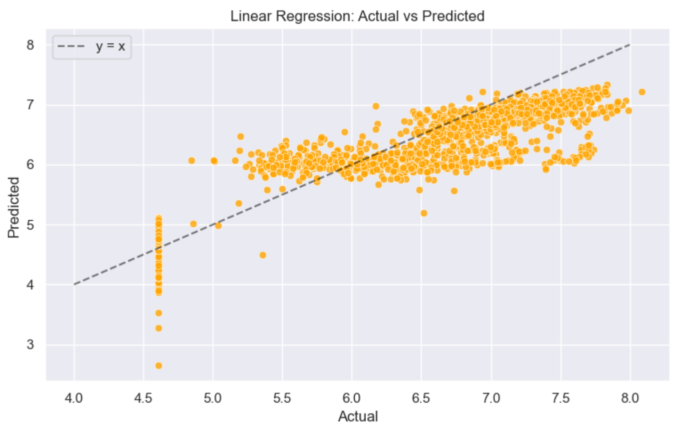


Figure 6. Plot between Actual vs Predicted on Test Data

5.2. Gradient Boosting Trees

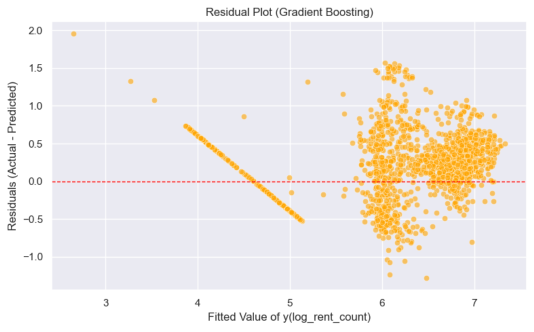
In the gradient boosting, we use the xgboost library/model to fit the training data. There are mainly three hyperparameters we want to optimise: The depth of the Tree (d), the Learning Rate (α), and the number of iterations for boosting (n). Same chronological split (80:20) is used for model training.

To find these hyperparameters, we use the K-fold Cross-Validation technique with K=5 using a library RandomizedSearchCV to find the values: $d = 3$, $\alpha = 0.2$ and $n = 121$. Using these hyperparameters, we then fit the data into xgboost model. Table 4 shows the performance of Gradient Boosting model

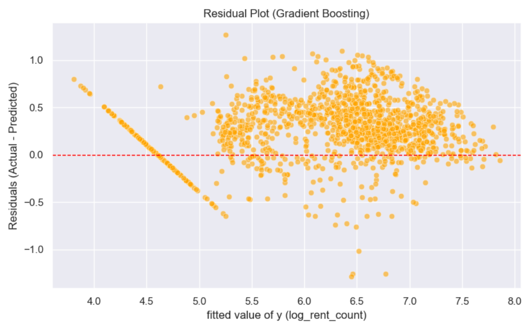
Table 4. Gradient Boosting performance

Model	Training		Testing	
	R^2	MSE	R^2	MSE
Gradient Boosting	0.9104	0.0615	0.8107	0.1657

We tried to avoid overfitting by keeping a low depth of the trees and fewer iterations in the K-fold CV. This way we got a really high R^2 value in train, which is not too different from R^2 obtained in test/-validation set. We next compare the performance of the two models.



(a) Residual plot for Linear Model



(b) Residual plot for Gradient Boosting

Figure 7

6. Comparison and Discussions

Residual plots are used to evaluate the performance of a regression model by examining the difference between the actual and predicted values (residuals). Ideally, residuals should be randomly scattered around zero, indicating that the model captures the data's patterns without systematic bias. Figure 7 shows the relation of residuals v/s the fitted values $\hat{y}$= log_rent_count. Residuals are mostly centered around zero, suggesting that the model generally performs well. In the Linear model, the variance of residuals is higher compared to the Gradient Boosting. We also plot the distribution of residuals as shown in Fig. 8, which is approximately normal in both models which is ideal, but a little better in the gradient boosting method due to lower variance of residuals.

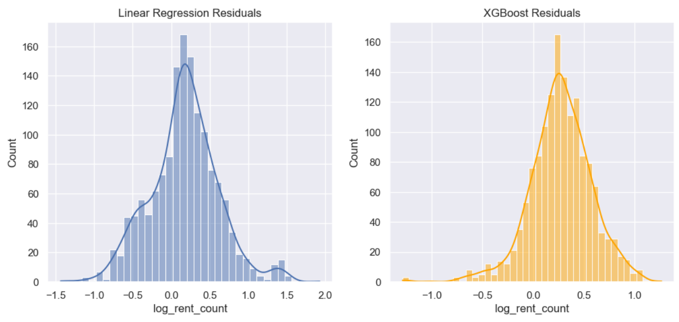


Figure 8. Distribution of Residuals

Table 5. Performance Summary

Model	**Training**		**Testing**	
	R^2	MSE	R^2	MSE
Gradient Boosting	0.9104	0.0615	0.8107	0.1657
Full Linear Model	0.6933	0.2104	0.7432	0.2249

Table 5 shows the performance of both models on the training and test data and it is evident that the Gradient Boosting method has worked better on both Train and Test Data. As gradient boosting is a tree-based algorithm we can find the importance of each feature by the overall contribution in reduction of RSS by that feature in each split. This is shown in Fig 9. The best 7 predictors we obtained from the random forest regression are similar to what the gradient boosting algorithm suggests.

We further try to interpret the coefficients of predictors in the Multiple Linear Regression Model. To do so, we fit the full Model (14 predictors) and also standardise the covariates for better interpretation. Standardised coefficients indicate the change in the outcome variable for every one standard deviation increase in the predictor variable, holding other variables constant. Thus, a larger (in magnitude) coefficient suggests that it has a large effect on the outcome variable and its sign indicates the direction of the effect. A positive value would suggest higher demand in bikes with an increase in that predictor. Fig 10 suggests that the variables *dew_temp, Functioning Day, Hour, humidity* have a huge impact on the demand of cycles. E.g. Demand of cycles on a functioning day(weekdays) is higher compared to weekends.

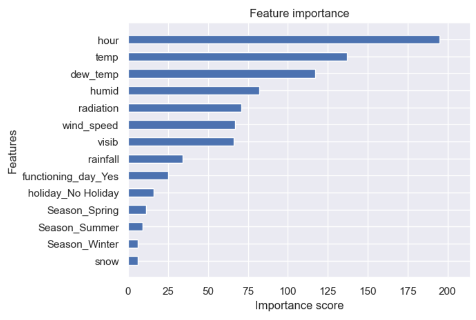


Figure 9. Feature Importance in gradient boosting

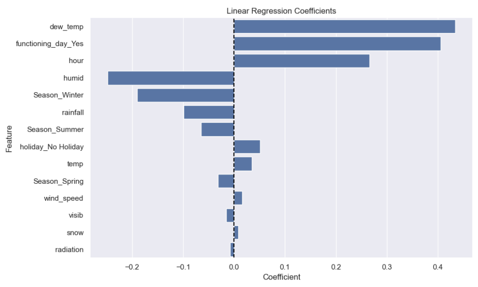


Figure 10. Coefficients from Full Linear Model

Multiple Regression model does a very good job in predicting the variables, but it can be improved with the data. Variables like hour have been assumed to be continuous, but can be better represented in groups of 6 hours to better model, but this reduces the data. Also, the demand changes with time. A linear model doesn't take this into account, that in recent years the demand is probably more than it was 5 years ago.

References

[1] S. V. E, J. Park, and Y. Cho, "Using data mining techniques for bike sharing demand prediction in metropolitan city", *Computer Communications*, vol. 153, pp. 353–366, 2020, ISSN: 0140-3664. DOI: https://doi.org/10.1016/j.comcom.2020.02.007. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0140366419318997.