Seoul Bike Sharing Demand Prediction

Mahin Arvind C
Almabetter, Bangalore

Abstract

Due to rapid urbanization, commuters have faced the critical issue of traffic congestion. While being cost-effective and healthy, bike sharing systems have risen in recent times by tackling this very problem.
Consumers are more likely to prefer such a solution as bike sharing systems are provided on an 'as-needed' basis, helping avoid the costs and responsibilities associated with bike ownership.
The caveat in such a system lies in providing the right bike supply that meets the daily demands of the people as their routines depend on these commutes while also keeping in mind to not overcrowd the city with more bikes than required.
In this project, we tackle the problem of predicting the number of bikes rented at any given hour in the city of Seoul using the Seoul Bike Share Demand Prediction dataset. The efficiency of standard machine learning techniques namely Linear Regression, Nearest Neighbors, Decision Trees, Bagging, Boosting and Stacking Ensembles are implemented and their performances are compared.

1. Problem Statement

In this project we'll be using the Seoul Bike Share Demand dataset to

1. Understand Bike Share use trends
2. Apply machine learning techniques to predict the number of bikes rented at any given hour using the city weather and date information, and
3. Provide reasonable explanations from the best predicting model to understand factors affecting bike share demands.

2. Introduction

A bike share program is a shared transport service in which bicycles are made available for shared use to individuals on a short-term basis at an affordable price. In such systems, the individual picks up their bike from a dock and drops the bike back at the dock nearest to their destination.
The bike sharing system, much like other transport services like public buses, trains and cabs caters to a group with fluctuating transport demands affected by a variety of factors.
Predicting this demand can prove to be efficacious as it allows one to stock bikes in docking stations according to user demands in advance. This allows bike sharing systems to become not just an economical and healthy mode of transport, but also a reliable mode of transport.

3. Dataset Description

3.1. Data Preparation

Data for one year (2017 December to 2018 November) is obtained from the Seoul Public Data Park website of South Korea. This dataset contains the count of public bikes rented at each hour in the Seoul Bike Sharing System with the corresponding weather data and holidays information. There are 24 logs(one for each hour of the day) of bike rental data recorded consistently for each day.
The 'Date' attribute has the date of the recording stored as a string. This attribute is converted to datetime format and features indicating day of the week, weekends, different times of the day and month are collected.
Features such as hour of the day, day of the week and month are not exclusively continuous.

These features are rather cyclic in nature and this needs to be reflected to capture routines and recurring behaviors in the data we are interested in predicting. In order to do so, the trigonometric sine and cosine of their relative values are used.
The physical data describing the weather in the city of Seoul are temperature, humidity, windspeed, visibility, solar radiation, rainfall and snowfall. The dataset contains features such as Weekend and Holiday that might explain sudden fluctuations in activity. There are 18 holidays in Seoul where one might notice a demand trend varying from a regular workday trend. All attributes are listed in Table 1.

Table 1: Attribute Description

Feature	Type	Measurement
Date	year-month-day	-
Rented Bike Count	Continuous	0,1,2,...3556
Hour	Continuous	0,1,2,..24
Temperature(°C)	Continuous	Celsius
Humidity(%)	Continuous	%
Wind speed (m/s)	Continuous	m/s
Visibility (10m)	Continuous	m
Dew point temperature(°C)	Continuous	Celsius
Solar Radiation (MJ/m2)	Continuous	mJ/m^2
Rainfall(mm)	Continuous	mm
Snowfall (cm)	Continuous	cm
Holiday	Categorical	Holiday/ No Holiday
Seasons	Categorical	Summer, Winter, Spring, Autumn
Functioning Day	Categorical	Yes/No
Month	Count	1,2,3..12
Weekend	Categorical	Yes/No
Day Phase	Categorical	Morning, Afternoon, Evening, Night
Sin Month	Continuous	[-1,1]
Cos Month	Continuous	[-1,1]
Sin Day	Continuous	[-1,1]
Cos Day	Continuous	[-1,1]
Sin Hour	Continuous	[-1,1]
Cos Hour	Continuous	[-1,1]

3.2. Exploratory Data Analysis

Exploratory Data Analysis (EDA) is the most common methodology used to summarize the data. and analyze visually using different parameters.
The Pandas package has been utilized in data-handling and Matplotlib and Seaborn have been used for data visualization. The bikes rented per hour throughout the entire year and the changes in demand on different days are depicted in **Fig. 1** and **Fig.2.**
Number of Bikes Rented increases from 5:00 am and reaches its first peak at 8:00 am. The demand starts rising again at 10:00 am to reach the second peak at 6:00 pm, marking the busiest time of the day. The demand decreases from 6 pm to 4 am the next day. The time between 4:00 am and 5:00 am is observed to be the quietest hours of an average day. The mornings are busiest on Mondays while evenings are busiest on the last working day, Friday.
From **Fig. 3,** we can tell bike sharing demands are substantially low on holidays as compared to working days, suggesting that the working class play a major role in creating bike sharing demand. The workday-holiday rental difference is reflected in **Fig. 4** where the hourly bike sharing demand is plotted and compared for weekdays and weekends.

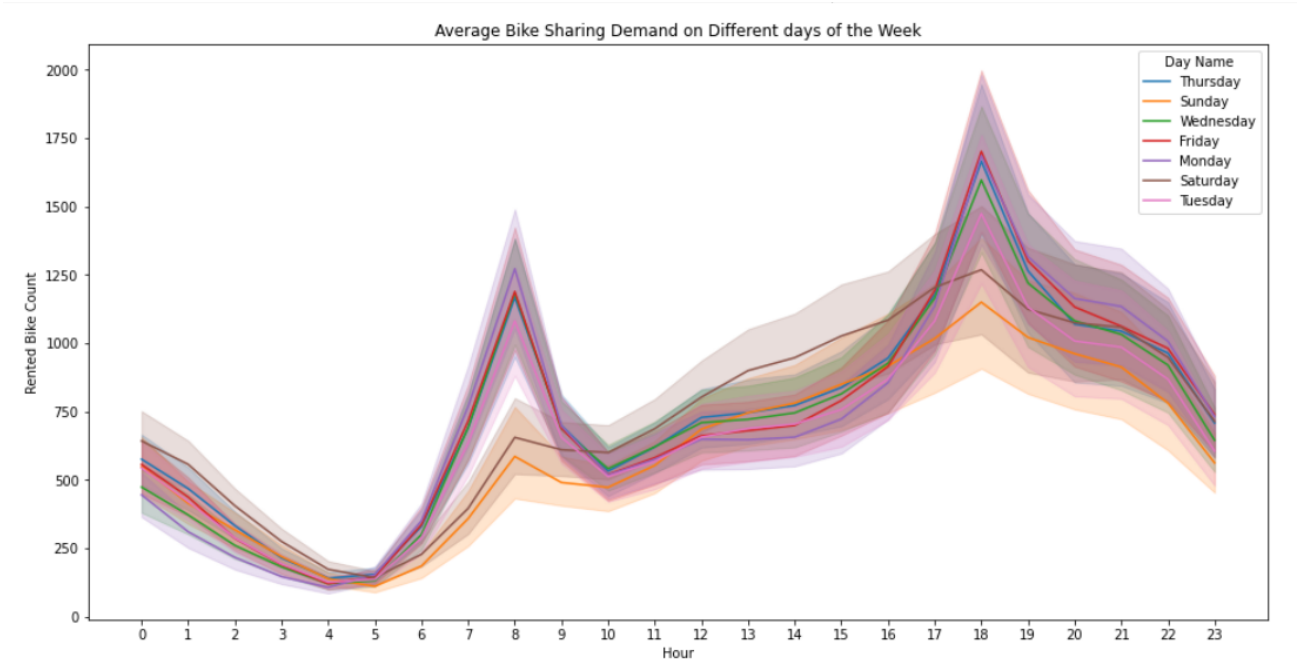


Figure 2

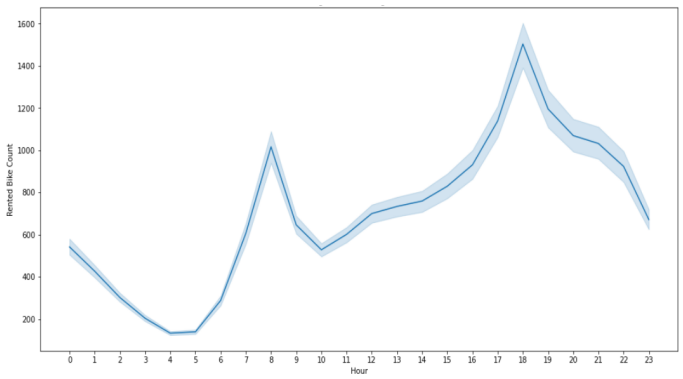


Figure 1

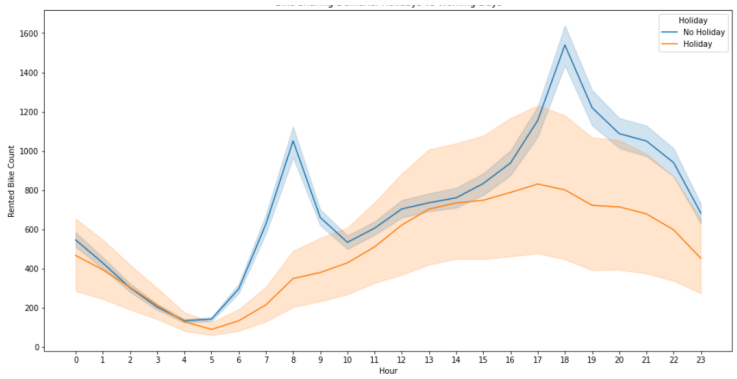


Figure 3

Fig. 1 shows the Hourly Bike Rental plot on an average day in Seoul. **Fig. 2** shows the Hourly Bike Rental plot averaged on different days of the week. **Fig. 3** compares the Hourly Bike Rental plot averaged on Holidays and a Non-Holiday (Holiday depicted in Orange and Non-Holiday depicted in Blue).

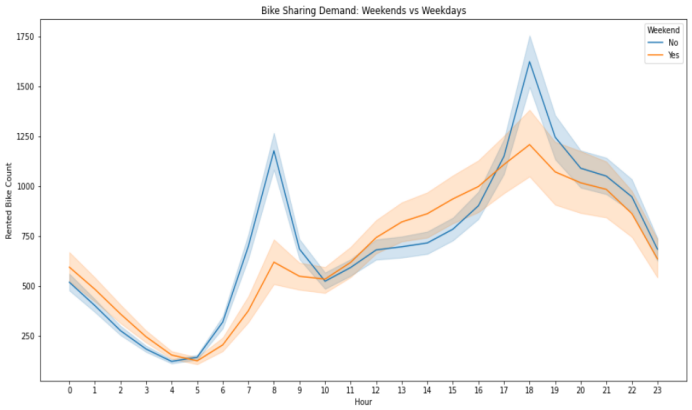


Figure 4

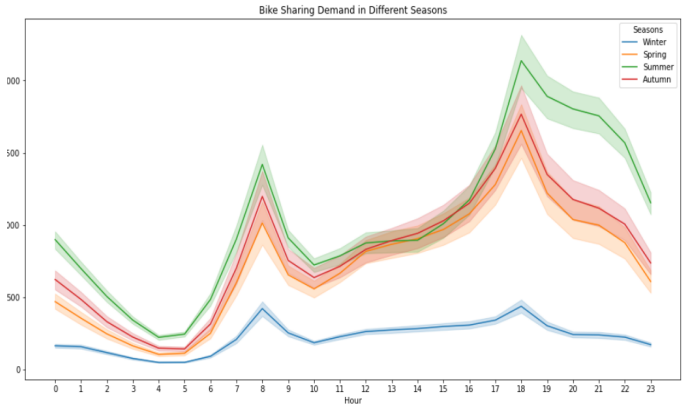


Figure 5

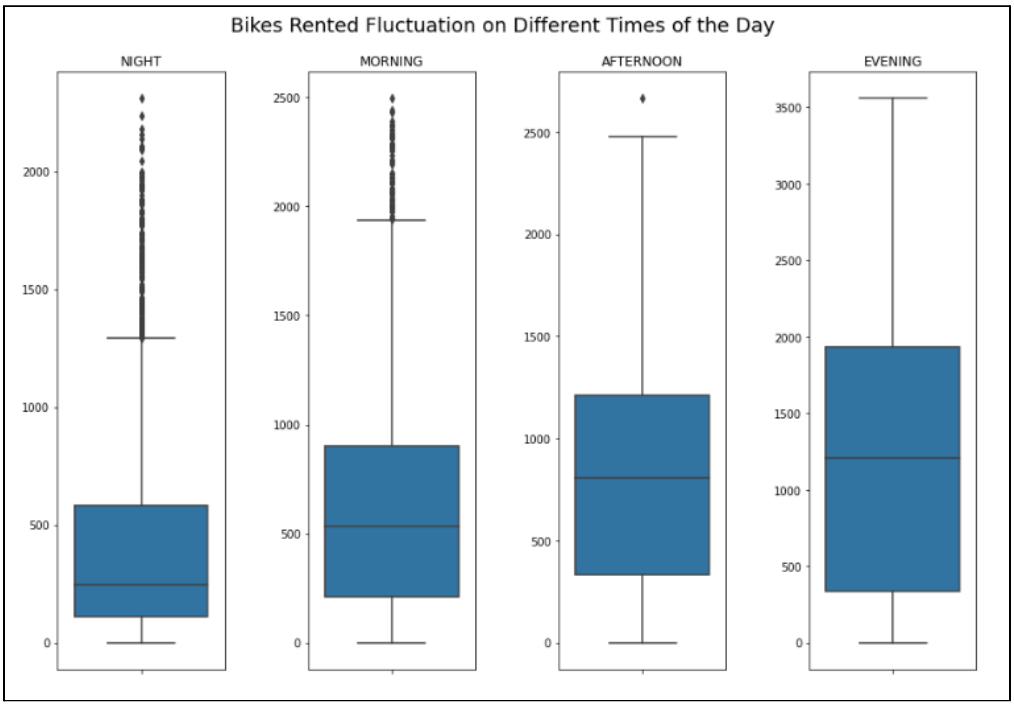


Figure 6

Fig. 4 shows the Hourly Bike Rental plot on an average weekday and an average weekend.. **Fig. 5** shows the Hourly Bike Rental plot averaged on different seasons in a year. **Fig.6.** shows Bike Rentals fluctuations during different times of the day

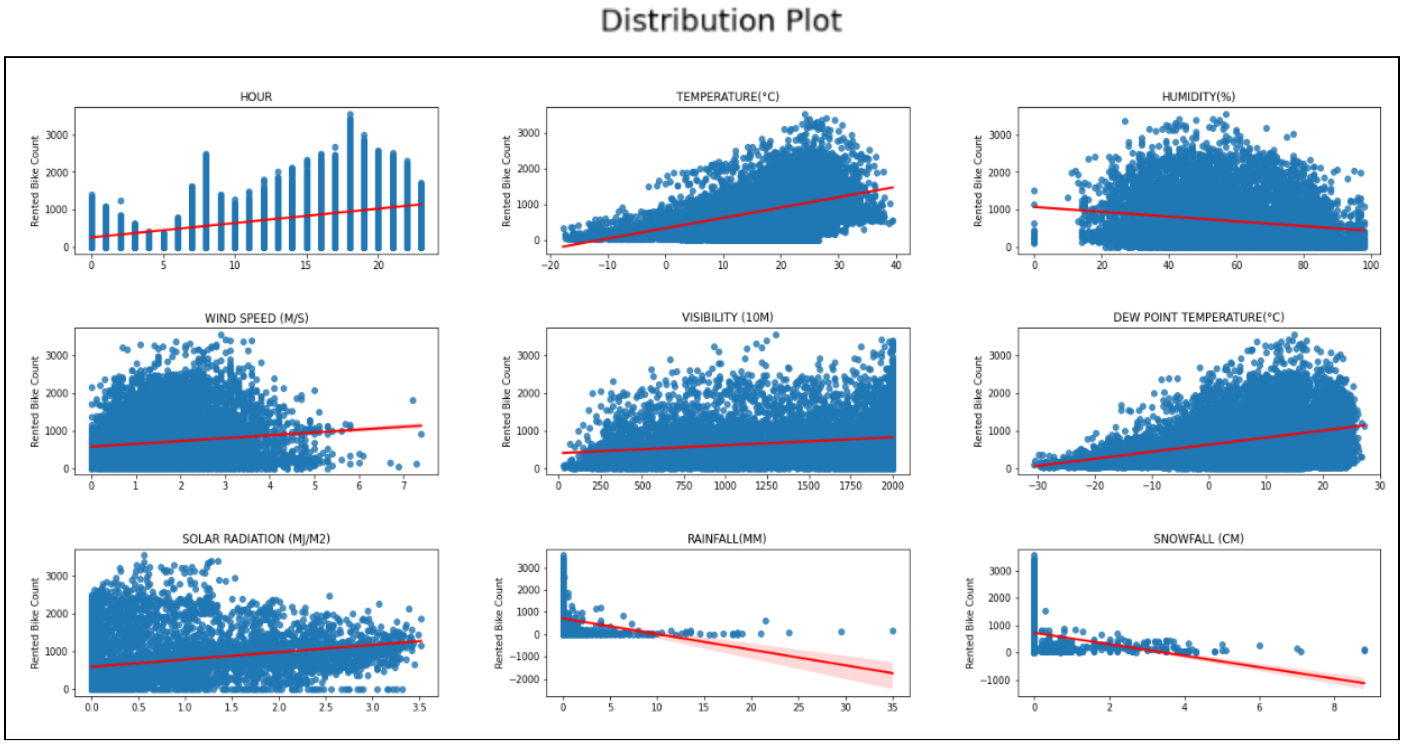


Figure 7

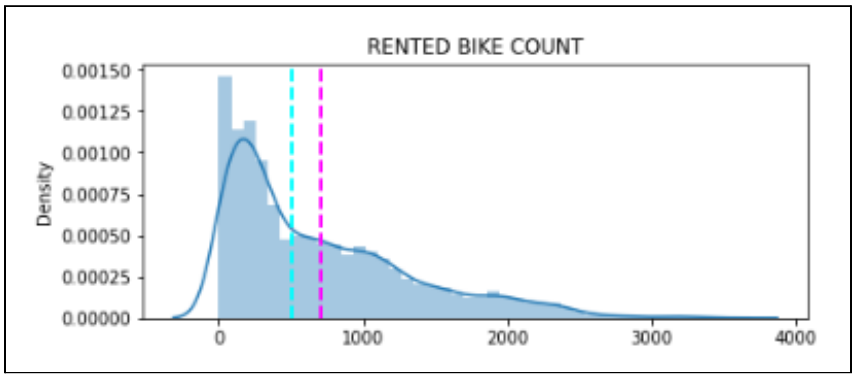


Figure 8

Fig.7. shows how Bike Rentals vary with respect to time of the day and weather related to features such as Temperature, Humidity, Windspeed, Visibility, Dew Point Temperature, Solar Radiation, Rainfall and Snowfall**.** **Fig. 8.** shows the distribution of Bike Rentals throughout the dataset. In the graph, the pink line indicates the average bike share rental and the cyan line indicates the median bike share rental count in the dataset.

Fig. 5 shows the bike sharing demand on an average day during different seasons of the year. We can observe that the bike rentals are the highest during summers while the demand is significantly low during winters. The peaks and lows in demand are analogous across all seasons suggesting that although the bike rental demands change with seasons, the rental routines of the users are consistent. **Fig. 6** shows Bike Rentals fluctuations during different times of the day and sees most bike rentals during the evening. Numerical features are plotted against the bike rental count to see if there were any relations using scatter plot graphs in **Fig. 7.** The distribution of bike rentals in the dataset are plotted along with their means and median in **Fig. 8.**

4. Methodology

The performance of ten machine learning algorithms are evaluated and compared to predict bike rental demand in the city of Seoul using the dataset at any given hour.

4.1. Linear Regression

Linear Regression is one of the simplest machine learning algorithms based on supervised learning. A linear model makes a prediction by computing the weighted sum of the input features and constant term called the bias.The simplest linear model for regression is one that involves a linear combination of the input variables.

$$y(\mathbf{x}, \mathbf{w}) = w_0 + w_1 x_1 + \ldots + w_D x_D$$

where x = $(x1,...,xD)^T$. This is known as linear regression. The key property of this model is that it is a linear function of the parameters w0,...,wD. It is also, however, a linear function of the input variables x_i. Therefore, the class of models is extended by considering linear combinations of the fixed nonlinear functions of the input variables, of the form

$$y(\mathbf{x}, \mathbf{w}) = w_0 + \sum_{j=1}^{M-1} w_j \phi_j(\mathbf{x})$$

where φj (x) are known as basis functions. By denoting the maximum value of the index *j* by $M - 1$, the total number of parameters in this model will be *M*. The parameter w_0 allows for any fixed offset in the data and is called the bias parameter. If the original variables comprise the vector x, then the features can be expressed in terms of the basis function, {φj (x)}[1].

The model minimizes the cost function by evaluating the residual sum of squares of errors and updating the parameter weights appropriately using gradient descent.

4.2. Ridge Regression

Ridge Regression is a regularized version of Linear Regression. Regularization is a technique used to reduce the errors by fitting the function appropriately on the given training set and avoid overfitting.

In ridge regression, a regularization term called the 'sum of squares', $\boldsymbol{\alpha} \Sigma \boldsymbol{\theta}^2_i$ is added to the cost function. This forces the learning algorithm to not only fit the data but also keep the model weights as small as possible. Hence, reduces overfitting by trading off variance for increased bias.

Here, **α** controls how much regularization is allowed. If **α** = 0, then the Ridge Regression is just Linear Regression. If **α** is very large, then all weights end up being very close to zero.

Ridge regression decreases the complexity of a model but does not reduce the number of variables as it never leads to a coefficient being zero, rather it only minimizes the coefficient.

4.3. Lasso Regression

Least Absolute Shrinkage and Selection Operator Regression (Lasso) is another regularized version of Linear Regression. Lasso Regularization adds a regularization term, ℓ1 norm, to the cost function which is the sum of absolute value of coefficients in the cost function or $\boldsymbol{\alpha} \Sigma \mid \boldsymbol{\theta}_i \mid$.

Like Ridge Regression ,**α** controls how much regularization is allowed. If α = 0 then the Lasso Regression is just Linear Regression. If α is very large, then all weights end up very close to zero

Lasso regression tends to make coefficients to absolute zero and acts as the model's built-in feature selector.

4.4. Decision Tree

A decision tree consists of a set of tree-structured decision tests working in a divide-and-conquer way. Each non-leaf node is associated with a feature test also called a split. Data falling into the node will be split into different subsets according to their different values on the feature test. Each leaf node is associated with a label, which will be assigned to instances falling into this node. In prediction, a series of feature tests is conducted starting from the root node, and the result is obtained when a leaf node is reached.
Decision tree learning algorithms are recursive processes. The key of a decision tree algorithm lies in selecting the splits and the feature-value pair which will cause the largest information gain is selected for the split.

4.5. Gradient Boosting

In Gradient Boosting, decision trees are used as the weak learners. These weak learners solve the problem of machine learning by transforming the data into tree representation. Gradient boosting Regression calculates the difference between the current prediction and the known correct target value each time.This difference is called the residual. Gradient boosting Regression trains a weak model that maps features to that residual. The residual predicted by a weak model is added to the existing model input and thus this process nudges the model towards the correct target. Repeating this step again and again improves the overall model prediction.

4.6. CatBoost

CatBoost is a member of the family of Gradient Boosting Decision Trees and is well-suited to machine learning tasks involving categorical, heterogeneous data. CatBoost encodes categorical values in order to alleviate the problem of target leakage and also chooses the most efficacious combinations of features during training.
CatBoost uses the Ordered TS method for encoding new features it generates from feature combinations when the combination of features includes a categorical variable[3]. CatBoost uses the following steps:
1. Dividing the records to subsets randomly,
2. Converting the labels to integer numbers, and
3. Transforming the categorical features to numerical, as:

$$avgTarget = \frac{countInClass + prior}{totalCount + 1}$$

where, 'countInClass' is the number of ones in the target for a given categorical feature, 'totalCountis' the number of previous objects and prior is specified by the starting parameter[4].

4.7. Light Gradient Boosting Machine

In LightGBM, the decision trees are grown leaf-wise, instead of checking all of the previous leaves for each new leaf. All the attributes are sorted and grouped as bins. This implementation is called

histogram implementation. LightGBM has several advantages such as better accuracy, faster training speed, and is capable of large-scale data handling as it is supported by GPU learning[4].

4.8. Bagging

Bagging algorithm consists of two key components, Bootstrapping and Aggregating. Bagging adopts the bootstrap distribution for generating different base learners. In a given training data set containing *m* number of training examples, a sample of *m* training examples will be generated by sampling with replacement. Some original examples appear more than once, while some original examples are not present in the sample. By applying the process *T* times, *T* samples of m training examples are obtained. Then, from each sample a base learner can be trained by applying the base learning algorithm.
The most popular strategies adopted by Bagging to aggregate the outputs of the base learners are voting for classification and averaging for regression.
Given a dataset D = $\{(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)\}$ with base learning algorithm, L and number of base learners T, the steps undergone to create bagging are[2]:

1. for t = 1, . . . , T :
2. ht = L(D,Dbs) % Dbs is the bootstrap distribution
3. end

Output:

$$H(\boldsymbol{x}) = \arg\max_{y \in \mathcal{Y}} \sum_{t=1}^{T} \mathbb{I}(h_t(\boldsymbol{x}) = y)$$

4.9. Random Forest

Random Forests build a predictor ensemble with a set of decision trees that grow in randomly selected subspaces of data. The final predicted result is the mean of the results predicted by each decision tree from the ensemble. They are fast, simple to implement and can handle a very large number of input variables without overfitting.
Given a dataset D = $\{(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)\}$ with a feature subset size K, the steps undergone to create a random tree in a random forest algorithm is:

1. N ←create a tree node based on D;
2. if all instances in the same class then return N
3. F ←the set of features that can be split further;
4. if F is empty then return N
5. F˜ ←select K features from F randomly;
6. N.f ←the feature which has the best split point in F˜;
7. N.p ←the best split point on N.f;
8. Dl ←subset of D with values on N.f smaller than N.p ;
9. Dr ←subset of D with values on N.f no smaller than N.p ;
10. Nl ←call the process with parameters (Dl, K);
11. Nr ←call the process with parameters (Dr, K);
12. return N

Random forest algorithm is regarded as an extension of the Bagging algorithm with the difference incorporation of random feature selection. The decision boundaries of RF and its base classifiers are more flexible, leading to a better generalization ability)}[2].

4.10. K-Nearest Neighbors

K-Nearest Neighbor is one of the simplest Machine Learning algorithms based on Supervised Learning technique. The KNN algorithm uses ‘feature similarity’ to predict the values of any new data points. This means that the new point is assigned a value based on how closely it resembles the points in the training set.
A generic K-NN algorithm undergoes the following steps before predicting values:

1. The distance between the new point and each training point is calculated.
2. The closest k data points are selected based on the distance.
3. The average of these data points is the final prediction for the new poin

K-NN is a non-parametric algorithm, which means it does not make any assumption on underlying data. It does not learn from the training set immediately instead it stores the dataset at the time of prediction and performs an action on the dataset.

5. Evaluation indices:

Metrics help gauge the performance of machine learning algorithms quantitatively. Evaluation of the deployed models was done by observing four regression performance metrics:

5.1. Root Mean Squared Error

RMSE is calculated as the square root of the mean of the squared differences between actual outcomes and predictions. Squaring each error forces the values to be positive, and the square root of the mean squared error returns the error metric back to the original units for comparison. RMSE is given by the formula,

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} \|y(i) - \hat{y}(i)\|^2}{N}},$$

where N is the number of data points, y(i) is the i-th measurement, and y^(i) is its corresponding prediction.

5.2. Mean Absolute Error

MAE is calculated as the average of the absolute error of the actual values and the predicted values.

$$\mathbf{MAE} = \frac{1}{n}\sum_{i=1}^{n} |\boldsymbol{x}_i - \boldsymbol{x}|$$

Here, $|x_i-x|$ is the absolute error in prediction from the actual value.

5.3. R-Squared

The R2 (or R Squared) metric provides an indication of the goodness of fit of a set of predictions to the actual values. In statistical literature this measure is called the coefficient of determination. R^2 is given by

$$R^2 = \frac{SSR}{SST} = \frac{\sum(\hat{y}_i - \bar{y})^2}{\sum(y_i - \bar{y})^2}$$

where $\hat{y}_i$ is the i^{th} predicted value, $\bar{y}$ is the mean of all the actual values and y_i is the i^{th} actual value. This is a value between 0 and 1, 0 for no-fit and 1 for perfect fit respectively.

5.4. Adjusted R-Squared

The adjusted R-squared is a modified version of R-squared that adjusts for the number of predictors in a regression model. It is calculated as:

$$Adjusted\ R^2 = 1 - \frac{(1 - R^2)(N - 1)}{N - p - 1}$$

where R^2 is the R squared value, N is the total sample size and p is the number of independent variables.

6. Model Development

In order to find and decrease the error values while fitting a model, it is necessary to find optimal tuning parameters for the regression algorithm. We select the top three best performing regression algorithms, CatBoosting, LightGBM and Random Forest, and tune them using the GridSearchCV package. Search

Space is the volume to be searched where each dimension represents a hyperparameter and each point represents one model configuration. GridSearchCV is a searching strategy to try and find the best combination of hyperparameters that reduces the error and thereby, increases model performance. GridSearchCV tries all the combinations of the values in the search space and evaluates the model for each combination using the Cross-Validation method. Hence after using this function, the accuracy/loss for every combination of hyperparameters is obtained, from which the best combination is picked.The hyperparameters used to optimize the model are the number of trees in the ensemble, maximum depth of trees in the ensemble and the maximum number of features to be used in each tree from the dataset.

Apart from this, the tuned performers are combined to see if they produce better results by using them with Stacking Ensemble. Stacking often considers heterogeneous weak learners, learns them in parallel, and combines them by training a meta-learner to output a prediction based on the different weak learner's predictions. A meta learner inputs the predictions as the features and the target being the ground truth values in data , it attempts to learn how to best combine the input predictions to make a better output prediction.

7. Results and Discussion

Upon training each regression model, each of the regression models had the RMSE, MAE, R^2 and adjusted R^2 calculated from their predictions on the test split (listed in **Table. 2).**

Upon evaluation, the Linear models (Linear Regression, Lasso Regression and Ridge Regression) performed nominally and increased with regularization. This improvement only explained 61.4% of the variance at best. As non-linear models like K-NN and CART were implemented, significant improvements in their R^2 scores were observed, explaining 78 % and 79 % of the variances in predictions explained by the dataset respectively.

The ensemble models produced the highest results in their predictions. The top three performing models were picked and optimized based on the highest R^2 scores and lowest RMSE and MAE values. The CatBoost, LightGBM and Random Forest produced the highest R^2 results of 0.9234, 0.9093 and 0.8977. Hyperparameter tuning increased the R^2 scores of these models to 0.9369, 0.9216 and 0.9093 respectively.

Stacking Ensemble of the three parameter-tuned models, CatBOOST, LightGBM and Random Forest, was implemented to test if their combined result can perform better than any one individual model. A linear regression model was used as the meta-learner to make predictions from the outputs of the individual models. Upon evaluation, the Stacking Regressor's score was 0.938, outperforming all the other models.

SHAP and LIME were the packages used to plot the feature importances of the models and explain their outputs. **Fig. 9, 10** and **11** shows the SHAP feature importances of the hyperparameter-tuned models. The combination of the following feature importances are used for the Stacking Ensemble model. The top features used to predict the number of bike rentals on any given hour in the city of Seoul are temperature of the city, hour of the day , solar radiation measured and if the season is winter.

Fig. 12, 13 and **14** show the LIME model explainability of a random prediction for the hyperparameter-tuned models and stacked ensemble.

In the figures the actual observed value is 486 bike rentals per hour for a given set of features. The color blue indicates negative influence in the values while the color orange indicates positive influence of the feature in predictions.

8. Conclusions

Upon Exploratory Data Analysis, we found that the bike rentals follow an hourly trend where it hits the first peak in the morning and the highest peak later in the evening. We also found that these trends are

Model Name	RMSE	MAE	R2	Adjusted R2
Stacking Regressor	160.5961	96.9616	0.9380	0.9365
Tuned CatBoost Regressor	162.0191	96.6193	0.9369	0.9354
CatBoost Regressor	178.3280	108.5684	0.9235	0.9217
Tuned LightGBM Regressor	180.5627	107.3737	0.9216	0.9197
LightGBM Regressor	194.2324	119.3688	0.9093	0.9071
Tuned Random Forest Regressor	200.6814	120.4977	0.9032	0.9008
Random Forest Regressor	206.2215	120.6033	0.8977	0.8953
Bagging Regressor	214.7691	127.2935	0.8891	0.8864
Gradient Boosting Regressor	246.4315	163.3944	0.8540	0.8505
DecisionTree Regressor	294.2235	165.7871	0.7918	0.7868
K Neighbors Regressor	301.9540	195.3866	0.7808	0.7755
Lasso Regression	400.6544	303.2638	0.6140	0.6047
Ridge Regression	400.7197	303.3573	0.6139	0.6046
Linear Regression	408.7252	312.7789	0.5983	0.5887

Table. 2. Model Performance Report

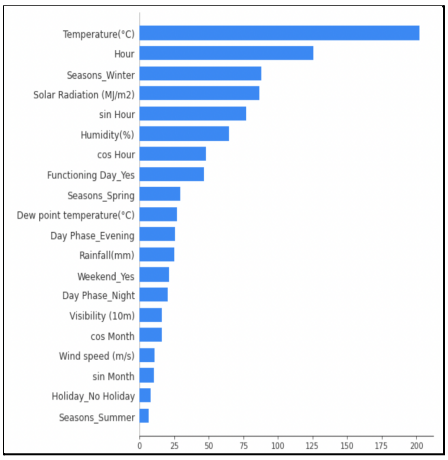


Fig. 9. Feature Importance of CatBoost

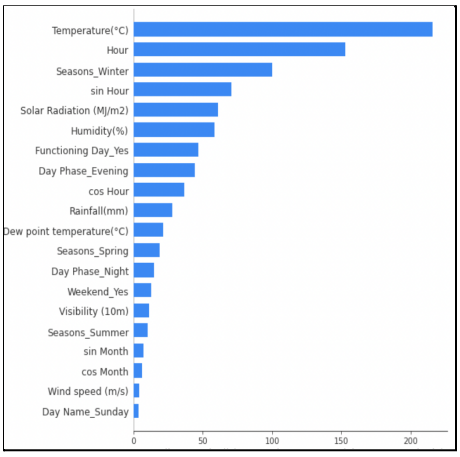


Fig. 10 Feature Importance of Random Forest

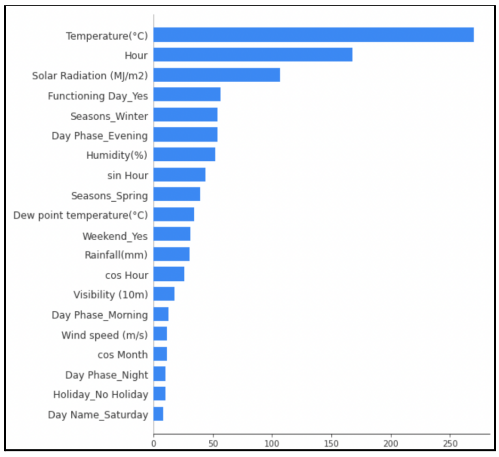
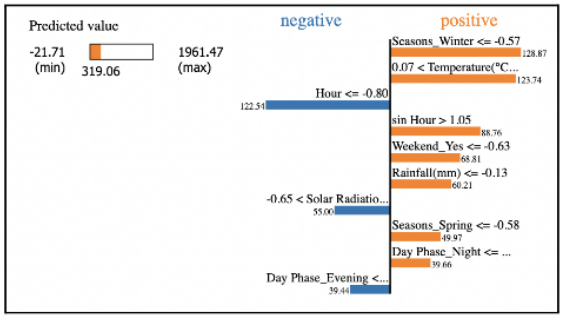
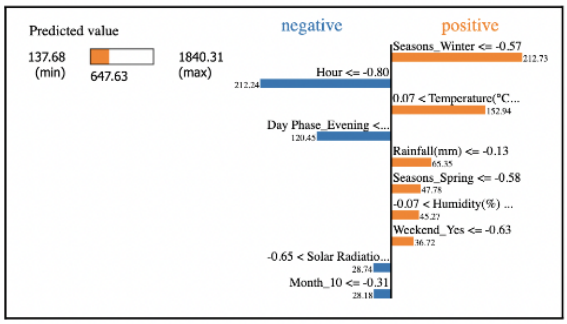


Fig. 11 Feature Importance of LightGBM



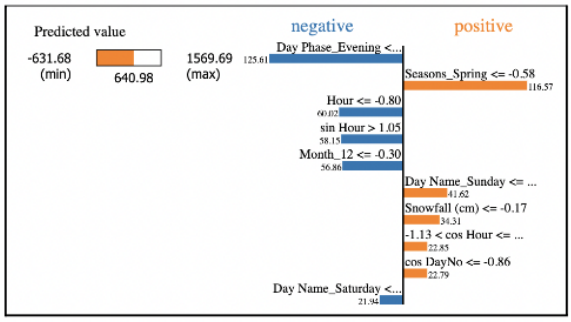
Feature	Value
Seasons_Winter	-0.57
Temperature(°C)	0.68
Hour	-0.80
sin Hour	1.44
Weekend_Yes	-0.63
Rainfall(mm)	-0.13
Solar Radiation (MJ/m2)	-0.62
Seasons_Spring	-0.58
Day Phase_Night	-0.70
Day Phase_Evening	-0.45

Fig.12 Model Explainability for CatBoost



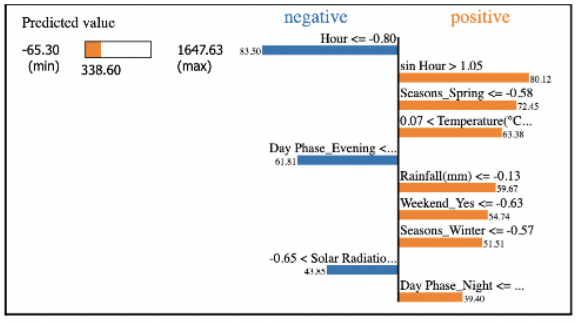
Feature	Value
Seasons_Winter	-0.57
Hour	-0.80
Temperature(°C)	0.68
Day Phase_Evening	-0.45
Rainfall(mm)	-0.13
Seasons_Spring	-0.58
Humidity(%)	0.27
Weekend_Yes	-0.63
Solar Radiation (MJ/m2)	-0.62
Month_10	-0.31

Fig.13 Model Explainability for Random Forest



Feature	Value
Day Phase_Evening	-0.45
Seasons_Spring	-0.58
Hour	-0.80
sin Hour	1.44
Month_12	-0.30
Day Name_Sunday	-0.41
Snowfall (cm)	-0.17
cos Hour	-0.14
cos DayNo	-0.86
Day Name_Saturday	-0.41

Fig.14 Model Explainability for LightGBM



Feature	Value
Hour	-0.80
sin Hour	1.44
Seasons_Spring	-0.58
Temperature(°C)	0.68
Day Phase_Evening	-0.45
Rainfall(mm)	-0.13
Weekend_Yes	-0.63
Seasons_Winter	-0.57
Solar Radiation (MJ/m2)	-0.62
Day Phase_Night	-0.70

Fig.15 Model Explainability for Stacking Ensemble

prominent only during weekdays and working days, leading us to make a safe assumption that office-goers make a notable contribution to bike sharing demand. In addition, seasons were observed to have a notable effect on bike rentals with high traffic during summer and a significantly lower demand in winter.

Upon training and evaluation of the machine learning models, the CatBoost model and the Stacked Ensemble of CatBoost, LightGBM and Random Forest models performed the best when evaluated using the R^2 metric. They produced R^2 scores of 0.9369 and 0.9380, with a root mean squared error of 162.01 and 160.59 respectively.

It was found that the top performing models made predictions based on the weather and time of the day as high weightage was given to seasons, temperature recorded, solar radiation and hour of the day. This confirms the trends observed during the exploratory data analysis stage of the project.

9. References

1. Christopher M. Bishop, “Pattern Recognition and Machine Learning”, Pg. 137-139
2. Zhi-Hua Zhou, “Ensemble Methods Foundations and Algorithms”, Pg. 57-58
3. John T. Hancock and Taghi M. Khoshgoftaar, “CatBoost for big data: An Interdisciplinary Review”
4. Essam Al Daoud, “Comparison between XGBoost, LightGBM and CatBoost Using a Home Credit Dataset”.
5. Jason Brownlee,“Machine Learning Mastery With Python, Understand Your Data, Create Accurate Models and Work Projects End-To-End”