Seoul Bike Sharing Demand Prediction

Abhash Jain, Mounika Dontula
Data science Trainees
AlmaBetter, Bangalore

Abstract:

This documents presents a rule-based regression predictive model for bike sharing demand prediction. A bike-sharing system provides people with a sustainable mode of transportation and has beneficial effects for both the environment and the user. In recent days, Pubic rental bike sharing is becoming popular because of is increased comfortableness and environmental sustainability. Data used include Seoul Bike and Capital Bikeshare program data. Data have weather data associated with it for each hour. For the dataset, we are using linear regression model were train with optimize hyperparameters using a repeated cross validation approach and testing set is used for evaluation. Multiple evaluation indices such as 2, Root Mean Square error are use to measure the prediction performance of the regression models. The performance of the model is vary with the time interval used in transforming data.

Keywords:machine learning,

1. Problem Statement

Currently Rental bikes are introduced in many urban cities for the enhancement of mobility comfort. It is important to make the rental bike available and accessible to the public at the right time as it lessens the waiting time. Eventually, providing the city with a stable supply of rental bikes becomes a major concern. The crucial part is the prediction of bike count required at each hour for the stable supply of rental bikes.

The main objective is to build a predictive model, which could help them in predicting the rental bikes supply proactively. This would in turn help them in matching the avilibility of rental bikes with the right customers quickly and efficiently.

2. Introduction

Bike sharing systems are a means of renting bicycles where the process of obtaining membership, rental, and bike return is automated via a network of kiosk locations throughout a city. Using these systems, people are able rent a bike from one location and return it to a different place on an as-needed basis. Currently, there are over 500 bike-sharing programs around the world.

Several bike/scooter rides sharing facilities (e.g., Bird, Capital Bike share, Citi Bike) have started up lately especially in metropolitan cities like San Francisco, New York, Chicago and Los Angeles, and one of the most important problems from a business point of view is to predict the bike demand on any day. While having excess bikes results in wastage of resource (both with respect to bike maintenance and the land/bike stand required for parking and security), having fewer bikes leads to revenue loss (ranging from a short term loss due to missing out on immediate customers to potential longer term loss due to loss in future customer base), Thus, having an estimate on the demands would enable efficient functioning of these companies.

The goal of this project is to combine the historical bike usage patterns with the weather data to forecast bike rental demand. The data set consists of hourly rental data spanning one year.

Data Description

The dataset contains weather information (Temperature, Humidity, Wind speed, Visibility, Dewpoint, Solar radiation, Snowfall, Rainfall), the number of bikes rented per hour and date information.

Attribute Information:

- Date: year-month-day
- Rented Bike count - Count of bikes rented at each hour
- Hour - Hour of he day
- Temperature-Temperature in Celsius
- Humidity - %
- Wind speed - m/s
- Visibility - 10m
- Dew point temperature - Celsius
- Solar radiation - MJ/m2
- Rainfall - mm
- Snowfall - cm
- Seasons - winter, spring, summer, autumn
- Holiday - Holiday/No holiday
- Functional Day – NoFunc (Non Functional Hours), Fun (Functional hours

3. Steps involved:

- **Feature Engineering**

After loading the dataset we performed this method. The provided data in its raw form wasn't directly used as an input to the model. Several feature engineering was carried out where few features were modified, few were dropped, and few were added.

- **Null values Treatment**

Our dataset contains a large number of null values which might tend to disturb our accuracy hence we fill it by mean, median or dropped them at the beginning of our project in order to get a better result.

- **Exploratory Data Analysis**

Before we start modeling, let us first get an idea on how the number of bike rentals depend on the various features provided to us one by one. By comparing our target variable with other independent variables. This process helped us figuring out various aspects and relationships among the target and the independent variables. It gave us a better idea of which feature behaves in which manner compared to the target variable.

- **Encoding of categorical columns**

We used One Hot Encoding to produce binary integers of 0 and 1 to encode our categorical features because categorical features that are in string format cannot be understood by the machine and needs to be converted to numerical format.

- **Standardization of features**

Our main motive through this step was to scale our data into a uniform format that would allow us to utilize the data in a better way while performing fitting and applying different algorithms to it.

The basic goal was to enforce a level of consistency or uniformity to certain practices or operations within the selected environment.

- **Fitting different models**

For modelling we tried various classification algorithms like:

1. Linear Regression –L1&L2
2. Linear Regression with Elastic net
3. Decision tree.
4. Random ForestRegressor
5. Gradient Boosting Regressor

- **Tuning the hyperparameters for better accuracy**

Tuning the hyper parameters of respective algorithms is necessary for getting better accuracy and to avoid over fitting in case of tree based models like Random Forest, Decision tree and Gradient Boosting Regressor.

4. Algorithms:

1. Linear Regression:

Linear Regression is a machine learning algorithm based on supervised learning. It performs a regression task. Regression models a target prediction value based on independent variables. It is mostly used for finding out the relationship between variables and forecasting. Different regression models differ based on – the kind of relationship between dependent and independent variables they are considering, and the number of independent variables getting used.

Linear regression performs the task to predict a dependent variable value (y) based on a given independent variable (x). So, this regression technique finds out a linear relationship between x (input) and y(output). Hence, the name is Linear Regression. In the figure above, X (input) is the work experience and Y (output) is the salary of a person. The regression line is the best fit line for our model.

Hypothesis function for Linear Regression

$$y = \theta_1 + \theta_2 . x$$

While training the model we are given:
x: input training data (uni variate – one input variable (parameter))
y: labels to data (supervised learning)
When training the model – it fits the best line to predict the value of y for a given value of x. The model gets the best regression fit line by finding the best θ1 and θ2 values.
θ1:intercept
θ2: coefficient of x

Once we find the best θ1 and θ2 values, we get the best fit line. So when we are finally using our model for prediction, it will predict the value of y for the input value of x.

L1: A regression model that uses L1 regularization technique is called Lasso Regression.
Lasso Regression (Least Absolute Shrinkage and Selection Operator) adds “absolute value of magnitude” of coefficient as penalty term to the loss function.

$$\sum_{i=1}^{n}(Y_i - \sum_{j=1}^{p} X_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} |\beta_j|$$

Cost function

Again, if lambda is zero then we will get back OLS whereas very large value will make coefficients zero hence it will under-fit.
L2: A regression model that uses L1 regularization technique is called *Ridge* Regression. Ridge regression adds “squared magnitude” of coefficient as penalty term to the loss function. Here the highlighted part represents L2 regularization element.

$$\sum_{i=1}^{n}(y_i - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} \beta_j^2$$

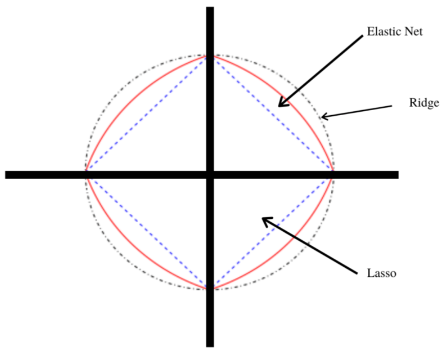
Cost function

Here, if lambda is zero then you can imagine we get back OLS. However, if lambda is very large then it will add too much weight and it will lead to under-fitting. Having said that it’s important how lambda is chosen. This technique works very well to avoid over-fitting issue.

The **key difference** between these techniques is that Lasso shrinks the less important feature's coefficient to zero thus, removing some feature altogether. So, this works well for **feature selection** in case we have a huge number of features.

2. Linear Regression with Elastic net:

Elastic net linear regression uses the penalties from both the lasso and ridge techniques to regularize regression models. The technique combines both the lasso and ridge regression methods by learning from their shortcomings to improve the regularization of statistical models.

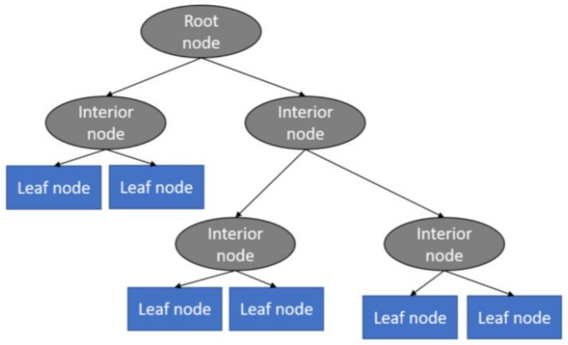


The elastic net method improves lasso's limitations, i.e., where lasso takes a few samples for high dimensional data. The elastic net procedure provides the inclusion of "n" number of variables until saturation. If the variables are highly correlated groups, lasso tends to choose one variable from such groups and ignore the rest entirely.
To eliminate the limitations found in lasso, the elastic net includes a quadratic expression (||β||2) in the penalty, which, when used in isolation, becomes ridge regression. The quadratic expression in the penalty elevates the loss function toward being convex. The elastic net draws on the best of both worlds – i.e., lasso and ridge regression.
In the procedure for finding the elastic net method's estimator, two stages involve both the lasso and regression techniques. It first finds the ridge regression coefficients and then conducts the second step by using a lasso sort of shrinkage of the coefficients.
This method, therefore, subjects the coefficients to two types of shrinkages. The double shrinkage from the naïve version of the elastic net causes low efficiency in predictability and high bias. To correct for such effects, the coefficients are rescaled by multiplying them by (1+λ2).

3. Decision tree:

Decision Tree is one of the most commonly used, practical approaches for supervised learning. It can be used to solve both Regression and Classification tasks with the latter being put more into practical application.

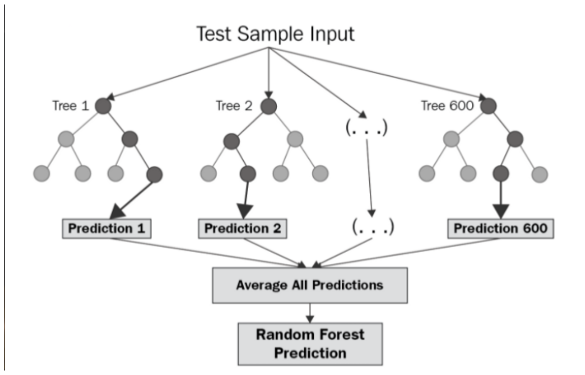
It is a tree-structured classifier with three types of nodes. The Root Node is the initial node which represents the entire sample and may get split further into further nodes. The Interior Nodes represent the features of a data set and the branches represent the decision rules. Finally, the Leaf Nodes represent the outcome. This algorithm is very useful for solving decision-related problems.



With a particular data point, it is run completely through the entirely tree by answering *True/False* questions till it reaches the leaf node. The final prediction is the average of the value of the dependent variable in that particular leaf node. Through multiple iterations, the Tree is able to predict a proper value for the data point.

4. Random Forest Regressor

Random Forest Regression is a supervised learning algorithm that uses ensemble learning method for regression. Ensemble learning method is a technique that combines predictions from multiple machine learning algorithms to make a more accurate prediction than a single model.



The diagram above shows the structure of a Random Forest. You can notice that the trees run in parallel with no interaction amongst them. A Random Forest operates by constructing several decision trees during training time and outputting the mean of the classes as the prediction of all the trees. To get a better understanding of the Random Forest algorithm, let's walk through the steps:

1. Pick at random k data points from the training set.
2. Build a decision tree associated to these k data points.
3. Choose the number N of trees you want to build and repeat steps 1 and 2.
4. For a new data point, make each one of your N-tree trees predict the value of y for the data point in question and assign the new data point to the average across all of the predicted y values.

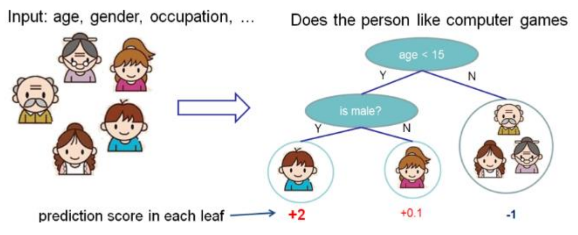
A Random Forest Regression model is powerful and accurate. It usually performs great on many problems, including features with non-linear relationships. Disadvantages, however, include the following: there is no interpretability, overfitting may easily occur, we must choose the number of trees to include in the model.

5. Gradient Boosting Regressor

Gradient Boosting algorithm is used to generate an ensemble model by combining the weak learners or weak predictive models. Gradient boosting algorithm can be used to train models for both regression and classification problem. Gradient Boosting Regression algorithm is used to fit the model which predicts the continuous value.

- **Gradient Boosting-**

Gradient boosted trees consider the special case where the simple model is a decision tree


Input: age, gender, occupation, ...
Does the person like computer games
age < 15
Y
N
is male?
Y
N
prediction score in each leaf
+2
+0.1
-1

In this case, there are going to be 2 kinds of parameters P: the weights at each leaf, w, and the number of leaves T in each tree (so that in the above example, T=3 and w=[2, 0.1, -1]).
When building a decision tree, a challenge is to decide how to split a current leaf. For instance, in the above image, how could I add another layer to the (age > 15) leaf? A 'greedy' way to do this is to consider every possible split on the remaining features (so, gender and occupation), and calculate the

new loss for each split; you could then pick the tree which most reduces your loss.

5. Hyper parameter tuning:

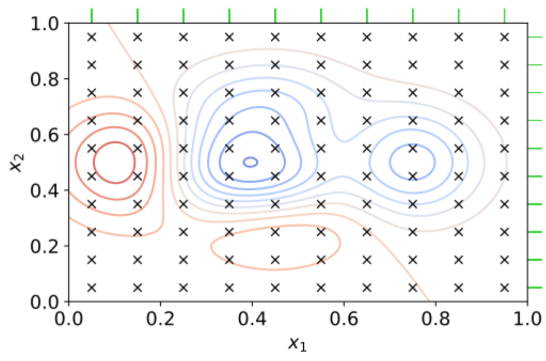
Hyperparameters are sets of information that are used to control the way of learning an algorithm. Their definitions impact parameters of the models, seen as a way of learning, change from the new hyperparameters. This set of values affects performance, stability and interpretation of a model. Each algorithm requires a specific hyperparameters grid that can be adjusted according to the business problem. Hyperparameters alter the way a model learns to trigger this training algorithm after parameters to generate outputs.

We used Grid Search CV for hyperparameter tuning. This also results in cross validation and in our case we divided the dataset into different folds.

Grid Search CV-

Grid Search combines a selection of hyperparameters established by the scientist and runs through all of them to evaluate the model's performance. Its advantage is that it is a simple technique that will go through all the programmed combinations. The biggest disadvantage is that it traverses a specific region of the parameter space and cannot understand which movement or which region of the space is important to optimize the model.

Grid Search uses a different combination of all the specified hyperparameters and their values and calculates the performance for each combination and selects the best value for the hyperparameters. This makes the processing time-consuming and expensive based on the number of hyperparameters involved.



GridSearchCV() method is available in the scikit-learn class model_selection. It can be initiated by creating an object of GridSearchCV():

clf = GridSearchCv(estimator, param_grid, cv, scoring)

Primarily, it takes 4 arguments i.e. estimator, param_grid, cv, and scoring. The description of the arguments is as follows:

1. estimator – A scikit-learn model

2. param_grid – A dictionary with parameter names as keys and lists of parameter values.

3. scoring – The performance measure. For example, 'r2' for regression models, 'precision' for classification models.

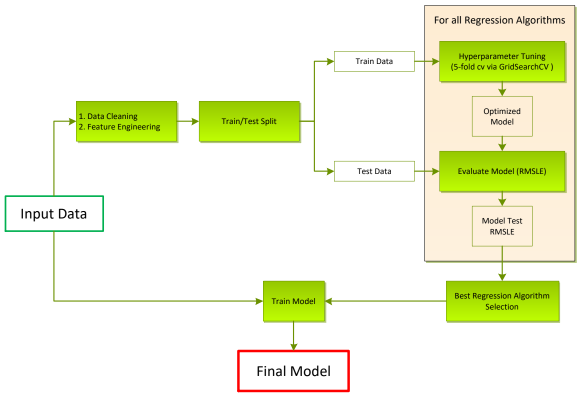
4. cv – An integer that is the number of folds for K-fold cross-validation.

GridSearchCV can be used on several hyperparameters to get the best values for the specified hyperparameters.

6. Modeling Overview

The overall procedure followed to obtain the Final Model. The provided data is first cleaned and transformed using Feature Engineering. We then split the data into

Train set (for Hyperparameter tuning) and Test set (for Model Evaluation).



7. Conclusion

- Hour of the day holds most importance among all the features for prediction of dataset.
- It is observed that highest number bike rentals counts in Autumn/fall Summer Seasons and the lowest in Spring season.
- We observed that the highest number of bike rentals on a clear day and the lowest on a snowy or rainy day.
- As we can see the top 5 important features of our dataset are: Season_winter, Temperature, Hour, Season_autumn and Humidity.
- Peoples do not use rented bikes in no functioning day.
- People tend to rent bikes when the temperature is between -5 to 25 degrees.
- People tend to rent bikes when the visibility is between 300 to 1700.
- The above experiments we can conclude that gradient boosting and random forest regressor with using hyperparameters we got the best results.

References-

1. MachineLearningMastery
2. GeeksforGeeks
3. Analytics Vidhya