Bike-sharing Demand Prediction Using Machine Learning Methods

Setareh Changizrezaei

(Dated: August 6, 2021)

Abstract

One of the popular transportation for short trips is bike sharing system. Using these systems, people can rent a bike from one location and return it to a different place. This provides a convenient, efficient and environmental-friendly transportation; however, there are some issues in the system that must be addressed. One of the major problems is that the bicycles might not evenly distributed across the city, some bike stations might not have enough bicycles and some of them might provide more bicycles that is unnecessary. Therefore, in order to help the users to plan their travel, it is useful to predict the bike sharing system demand. In the present work, different machine learning (ML) models are used to predict the number of bike rentals in the bicycle system serving in Washington D.C. We used Python libraries to perform data analysis and ML algorithms, and our code can be found in the GitHub[1]. We performed data preprocessing and identified relationships between features by visualization tools. Then we trained several different models on the data and tuned the corresponding required hyper parameters. Finally, we found that ensemble models such as random forest and XGBoost have better performance by comparing CV-error, test error and R^2 score of each model.

I. INTRODUCTION

In the past few years, the public bike sharing systems are preferred by many travelers for short-distance trips in major cities due to their low price, health and environment benefits, and reduce in congestion, noise, and air pollution [2]. In order to provide high quality services to the users, operators must be prepared for the daily bike rental demands in advance and have the information to anticipate the future demand for the system stations. Therefore, a good forecast of daily bike demand is necessary in order to have effective and efficient operation and system planning. As a result, there are rental systems to record the data on accessing and returning bikes in most of the bike-sharing system stations [3]. These systems record the real-time of each bike usage such as the start and end times and also provide the data on available bicycles at each station so that the users can conveniently have an access to the availability information of the bikes. Recording data enables the data scientist and researchers to help the operators to improve their system performance and maintenance by prediction of bike-sharing usage. For example in Ref. [4], the bike demand forecast modeled by linear regression is studied by considering taxi usage, weather and spatial variables as covariates to predict bike demand during morning rush hours in the weekdays and analyze the effect of precipitation and day of week. Furthermore, the bike-sharing system is studied in Ref. [5]. They examine the influence of meteorological data, temporal characteristics, bicycle infrastructure, land use and built environment attributes on arrival and departure flows at the station level using a multilevel approach to statistical modeling.

In this work, we aim to examine machine learning(ML) regressors to predict the bike rental usage in Washington. Before applying the ML models on the data, we did data preprocessing to make sure if all the missing values and outliers are removed and then try to understand better the effect of the features on the target variable which is the number of bike rentals through visualization techniques. The performance of applied ML models is presented in the results and discussion section.

II. DATA

The bicycle sharing system serving Washington D.C, is called Capital Bikeshare. The data generated by these systems makes them attractive for researchers because the duration of travel, departure location, arrival location, and time elapsed is explicitly recorded. Bike sharing systems therefore function as a sensor network, which can be used for studying mobility in a city. The Capital Bikeshare data was downloaded from [6] and we combined historical usage patterns with weather data in order to forecast bike rental demand.

The training data has 10886 observations and each observation consists of 11 features: date time, season, holiday, working day, weather, temp, atemp (the temperature it feels like), humidity, wind speed, casual (number of non-registered user rentals initiated), registered(number of registered user rentals initiated). The data set is comprised of the first 19 days of each month. The date time feature gives the date and hour as a string, holiday is a binary variable showing if the day was a holiday or not, working day is also a binary variable representing if the day was a non-holiday and the weather feature shows different weather conditions through 4 integers. The target of the data set is count which is the sum of casual and registered values and is the response that we want to predict.

III. METHODS

A. Data Preprocessing

In order to extract year, month, and day of week, the date time feature was converted from string to Python Date time object. Therefore, three new features were added and the date time variable was removed. As we want to visualize the data to get more insight out of it, we decided to convert the variables season, weather, holiday and working day variables to their actual string values and converted them to categorical variables. Then we converted all the categorical variables into dummy variables for implementing the ML algorithms. We also found the correlation between the pairs of the numerical variables using a correlation matrix and then this matrix is represented by heatmap plot to understand how strong the pair of variables are correlated. We concluded that the target value of count has a strong correlation with registered and casual variables, so we dropped these two features in our data set. Also, we checked whether there are any missing values in the data, and there were no missing values.

Then we worked on outliers detection in the observations. The outliers are extreme values that deviate from other observations and they diverge from an overall pattern on a sample. The outliers may show experimental errors or a variability in a measurement. There are two types of outliers: univariate and multivariate. The univariate outliers can be detected when considering at a distribution of values in a one-dimensional feature space; however, the multivariate outliers can be detected in a n-dimensional feature space. We can discover the outliers through visualization tools. One of a useful visualization tool for describing the behavior of the data in the middle as well as at the ends of the distributions is the box plot[7]. The box plot uses the median and the lower and upper quartiles (defined as the 25th and 75th percentiles). The lower quartile is Q1 and the upper quartile is Q3, and the difference (Q3 - Q1) is called the interquartile range or IQR. If there are outliers in the data, they will be shown as points in box plot but other population will be grouped together and are displayed as boxes. In our data set, we plotted the box plot for the target variable (count) and we observed that there are outliers that are not included in the box and far from the quartiles. We also see multivariate outliers in the box plot of count vs. season. In order to extract a list of outliers and then remove them from the data, we calculated IQR, which is similar to Z-score in terms of finding the distribution of data, and then determined a threshold to identify the outliers. We found out that %2.75 of the target values are outliers and then removed these points from the original data points.

Also, we plotted the distribution of the count variable and found that it is skewed towards to the right. As most of machine learning algorithms need the target to be close to normal distribution, we used the log transformations[8] on count variable to transform skewed data to conform normality so that the distribution of transformed data follows close to normal distribution. The data looks better after the transformation but still not exactly following the normal distribution. We used $\log(1 + y)$to transform the data because if a certain value of count is zero it does not converge to infinity.

B. Exploratory Data Analysis

Let's first look at the data before building prediction models using visualization tools. The figures are shown in appendix section. It can be seen that the time of the day is an important factor in bike rental demand. Figure 1 shows the plot of the hour of the day versus the bike rental count. The plot reveals that in working days there is a peak in number of rentals in the morning round 8am and also in the evening around 5pm to 6pm, which makes sense as the people go to work in the morning and come home from work in the evening so there is more demand in these times of the day; however, the highest rental counts occurs around noon in non-working days. To investigate the effect of month variable, the bar plot of month against count is shown in figure 2. The plot illustrates that there is a high bike rental demand between May to August corresponding to the summer season. Moreover, as it can be seen in figure 3, people rented more bikes in year 2012 comparing to 2011.

We also plotted the bar plot for the working day variable vs. count and holiday vs. count. The figure 4a reveals that there is a higher demand in working days. Also, the figure 4b is in agreement with fig. 4a showing that there is very high bike rental count in non-holidays. To illustrate how the weather affects the bike rental demand, we plotted the bar plot for weather variable against count as it is represented in figure 5. The plot shows that users rent mush higher bikes when the weather is clear. On the other hand there is no bike rental when there is thunderstorm.

IV. EVALUATION METRICS

In the present work, as we implement log transformation on the count variable, which was discussed in data preprocessing section, we use Root Mean Squared Logarithmic Error (RMSLE) to evaluate the model performance which is computed as follows:

$$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(\log(\hat{y}_i+1)-\log(y_i+1))^2} \tag{1}$$

where n is the number of observations, $\hat{y}_i$ is the predicted count, and y_i is the actual count. We split the data into train and test set, so that we can implement the prediction on test set and compute the test error as the final evaluation. We also use cross-validation method on the train set to evaluate generalization performance which is more stable and has less variance than a single train-test set split [9]. The most commonly used version of cross-validation is k-fold cross-validation, where k is a user-specified number, usually 5 or 10. Each split is called a fold. In our work we use 10 folds. The ML model is trained on k-1 folds with one held back and tested in the held back fold. This process is repeated so that each fold of the dataset is given a chance to be held back test set. After the process is done, we end up with k different performance scores that can be summarized using a mean or standard-deviation. We also used R^2 metric as an indication of "goodness of fit" of the prediction to actual value. This is a value between 0 and 1 corresponding to no-fit and perfect fit.

V. MODELING

We implemented the following methods to predict the target variable which is the bike rental count. The relevant hyper parameters are tuned through a 10-fold cross-validation.

A. Linear Regression

First, we used simple linear regression[10] describing a linear relationship between the features(independent variables) and the target(dependent variable). The motive of the linear regression algorithm is to find the best values for the slope and the intercept of the line fitted to the data. We use this model to get some baseline prediction.

B. Ridge and Lasso Regression

Lasso and Ridge Regression are the classical statistical algorithms which have been known for a long time. They are widely used and as an example, in Ref.[11] they are used in conjunction with a high dimensional feature space. In Ridge and Lasso regression models a penalty is applied on the coefficients so that they use regularization to control overfitting. In ridge regression, a penalty is applied on the sum of squares of coefficients, whereas in lasso, a penalty is applied on the absolute values of the coefficients[12]. This penalty is tuned to change the dynamics of the model fit. In fact, the goal of Ridge regression is to minimize the magnitude of coefficients, whereas lasso tries to eliminate them.

C. Polynomial Regression

The linear model also includes polynomial regression[10] in which the degree of features is equal to or greater than 2. In fact, the model is linear in the parameters. The only difference is that in the polynomial regression, the equation generates a curved line, instead of a straight line. In our work, we implemented a second-degree parabolic regression model which can be estimated by introducing the second-degree term in regression model.

D. Decision Tree

Decision trees use binary rules to calculate a target value. In these models, the predictor space is segmented into a number of simple regions. Multiple algorithms are used to split a node into two or more sub-nodes [10, 13]. The homogeneity of the resultant sub nodes increases as a result of each segmentation meaning that the purity of the node increases with respect to the response. One of the common problem of decision tress is overfitting issues. In order to reduce the overfitting problem, pruning is used by growing the tree completely and reduces the size of the decision tree which slightly increases the training error but drastically decrease the testing error. Decision trees have low a bias and a high variance error.

E. Random Forest

In Random Forest algorithm[14] a large number of decision trees are combined and each decision tree is made on bootstrapped samples of the data. This process is the same as bagging algorithm, which is an ensemble technique applied on decision trees but there is a difference. In bagging algorithm, all of the features as independent variables are selected for each sample so that all the trees look similar because during each independent tree sampled, significant variables always came first in the top layer of splitting. However, in random forest method only a subset of the features are selected. Therefore, random forests has low bias and low variance error.

F. Gradient Boosting

One of the most powerful techniques for building predictive models is Gradient Boosting. Gradient boosting[15] can be considered as a numerical optimization problem where the objective is to minimize the loss of the model by adding weak learners using a gradient descent like procedure. The regression trees are used as the weak learners in this algorithm whose output can be added together allowing subsequent models outputs to be added and correct the residuals in the predictions. Trees are constructed in a greedy manner, and choosing the best split points is based on purity scores like Gini or to minimize the loss. Trees are added one at a time, and existing trees in the model are not changed. A gradient descent procedure is used to minimize the loss when adding trees. We use XGboost to implement gradient boosting algorithm.

VI. RESULTS

We implemented 10-fold cross-validation on our models. In the models that the hyper parameters are tuned using GridSearchCV in python, we reported the minimum CV-error corresponding to the optimal hyper parameter values. The following table represents the 10-fold CV error, test error, and R^2 value for each model.

TABLE I:

Models	CV error	Test error	R^2
Linear	0.9594	0.9483	0.4981
Ridge	0.9594	0.9483	0.4981
Lasso	0.9594	0.9489	0.4975
Polynomial(2nd degree)	0.8183	0.7927	0.6425
Decision Tree	0.4210	0.4446	0.8873
Random Forests	0.3264	0.3327	0.9362
XGBoost	0.2657	0.2672	0.9581

VII. DISCUSSION

As a baseline model, we started with a linear model. As this model is considered to be simple and inflexible and may overfit the noise, we expected it to have a poor performance. As it is reported in table I, there is a low R^2 value for the linear model and there is no big difference between CV error and test error. We also tried Ridge and Lasso regression models and we found that their errors and R^2 values are the same as the linear model and there is no improvement in their performances which makes sense. The best value of α for

Lasso and Ridge model is 0.001 which makes the model similar to linear regression. We often use these two models when there is a really large number of variables which results in overfitting; however, the original data has only 11 features with 10886 observations and there is a very low possibility to have overfitting.

Then we performed a 2nd-degree polynomial regression on the data set, and found that it has a lower CV error and test error and higher R^2 comparing to the previous models, therefore it performs better than the linear models.

The next model is decision tree. The minimum CV error is found for the decision tree with maximum depth of 11. The error is lower than the described models and the R^2 score is equal to 0.8873 showing that there is an improvement in the predictive model's performance. We achieved better performance when we implemented random forest algorithm using 1000 estimators with maximum depth of 11. This model produced a significant improvement from our models. We could tune the number of estimators using GridSearchCV, but as it is computationally expensive, we only tried 1000 estimators with the optimal maximum depth found in decision tree model. Therefore, multiple number of estimators can be used to get lower CV-error.

As the final model, we used XGBoost model which has the lowest CV error and test error and highest R^2 score and it has the best performance to predict the bike rental count. The reason that XGBoost and random forest models perform better can be related to this point that the tree-based models might be able to capture non-linear interaction effects between the variables. In all the mentioned models, the test error is less than CV-error, showing overfitting does not occur.

It's worthwhile mentioning that the data contains the information over the first 19 days of each month, therefore it might be useful to perform time-series analysis to get realistic generalized bike-rental prediction. Also in order to only identify more relationships in the data, we can implement unsupervised learning models such as clustering in addition to applying the supervised learning predictive models as we did in our project.

VIII. APENDIX

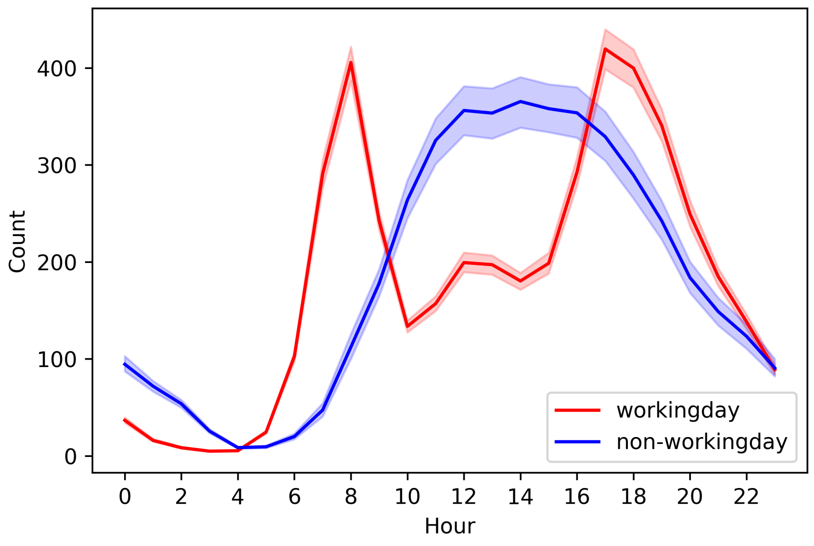


FIG. 1: Plot of hour of the day against the bike rental count for working and non-working days.

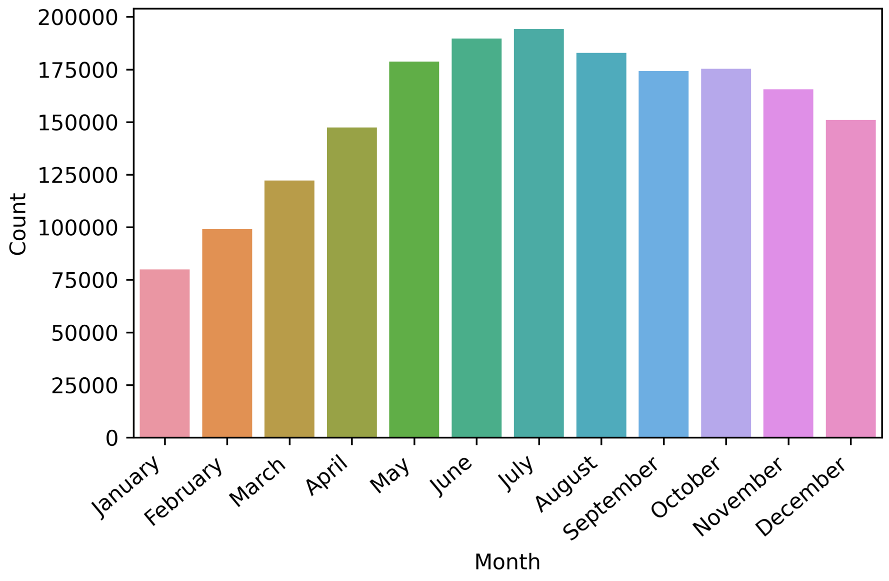


FIG. 2: Bar plot of month vs. count.

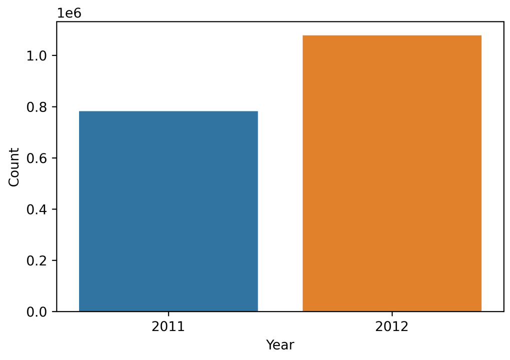
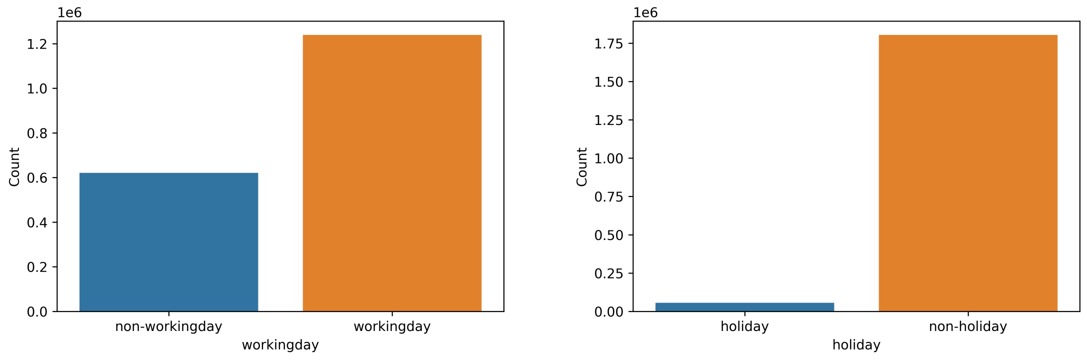


FIG. 3: Bar plot of year vs. count.



a)

b)

FIG. 4: Bar plot of a) working day and b)holiday vs. count.

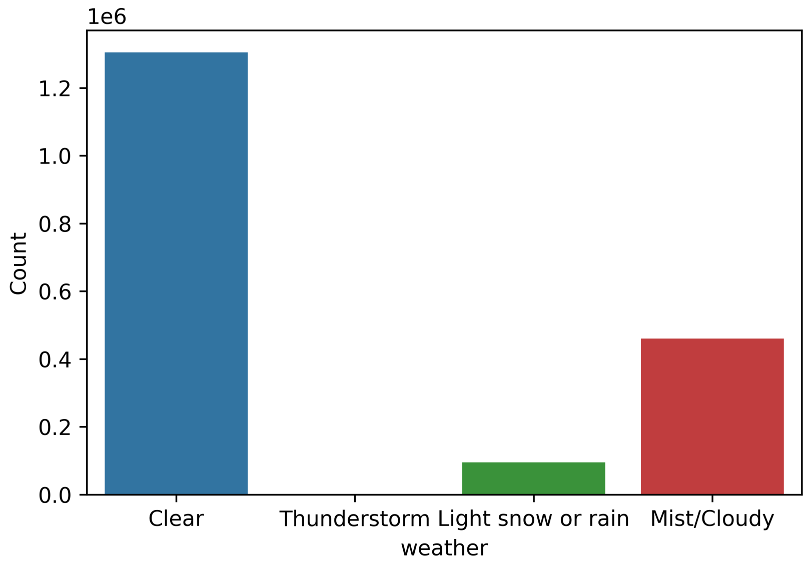


FIG. 5: Bar plot of weather vs. count.

[1] https://github.com/Setareh1/Bike-sharing-Demand-Prediction-Using-Machine-Learning-Methods/blob/main/Project.ipynb.

[2] S. A. Shaheen, S. Guzman, and H. Zhang, Transportation Research Record **2143**, 159 (2010), https://doi.org/10.3141/2143-20, URL https://doi.org/10.3141/2143-20.

[3] T. Raviv and O. Kolka, IIE Transactions **45**, 1077 (2013), https://doi.org/10.1080/0740817X.2013.770186, URL https://doi.org/10.1080/0740817X.2013.770186.

[4] D. Singhvi, S. Singhvi, P. Frazier, S. Henderson, E. O'Mahony, D. Shmoys, and D. B. Woodard, in *AAAI Workshop: Computational Sustainability* (2015).

[5] A. Faghih-Imani, N. Eluru, A. M. El-Geneidy, M. Rabbat, and U. Haq, Journal of Transport Geography **41**, 306 (2014), ISSN 0966-6923, URL http://www.sciencedirect.com/science/article/pii/S0966692314000234.

[6] https://www.kaggle.com/c/bike-sharing-demand/overview/.

[7] R. R. Wilcox, *Fundamentals of Modern Statistical Methods* (Springer, 2010).

[8] C. Feng, H. Wang, N. Lu, T. Chen, H. He, Y. Lu, and X. M. Tu, Shanghai Arch Psychiatry **26**, 105 (2014).

[9] A. C. Mller and S. Guido, *Introduction to Machine Learning with Python* (OReilly, 2017).

[10] S. Shalev-Shwartz and S. Ben-David, *Understanding Machine Learning: From Theory to Algorithms* (Cambridge University Press, 2014).

[11] H. Drucker, C. J. C. Burges, L. Kaufman, A. Smola, and V. Vapnik, in *ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS 9* (MIT Press, 1997), pp. 155–161.

[12] T. H. Gareth James, Daniela Witten and R. Tibshirani, *An Introduction to Statistical Learning: With Applications in R* (Springer, 2017).

[13] L. Rokach and O. Maimon, *DATA MINING WITH DECISION TREES Theory and Applications* (World Scientific Publishing Co. Pte. Ltd., 2014).

[14] A. Liaw and M. Wiener, R News **2(3)**, 18 (2002).

[15] J. Brownlee, *XGBoost With Python: Gradient Boosted Trees with XGBoost and scikit-learn* (Machine Learning Mastery, 2016).