Bike Sharing Demand: Forecasting the Use of a City Bikeshare System

Meghana Ginjpalli
Department of Computer Science
University of California, Los Angeles
mginjpalli@ucla.edu

Abstract

Bike sharing systems are automated services that allow people to obtain membership, rent, or return bicycles for a short period of time from various kiosks located throughout a city. Due to these systems becoming vastly popular, there are currently over 500 bike sharing programs worldwide which provides a rich dataset for further analysis. Essentially, this data acts as a sensor network and allows analysts to learn about the mobility of a city and predict bike sharing demand within that particular city. The goal of this machine learning competition is to implement a model that can predict the hourly bike rental demand in the Capital Bikeshare program in Washington, D.C. given historical usage patterns along with weather data. My approach concludes that the Gradient Boosting Machine model results in the highest prediction accuracy and has a Root Mean Squared Logarithmic Error (RMSLE) of 0.41691 which puts me in the top 12% of the leaderboard.

1 Introduction

A bike sharing system is an automated service that allows people to obtain membership, rent, or return bicycles for a short period of time from various kiosks located throughout a city. With this system, people can use a network of kiosks to rent a bike from a particular location and return it at another location based on their needs. As these systems are becoming vastly popular and promote alternative public transportation, there are currently over 500 bike sharing programs worldwide which provides a rich dataset for further analysis.

In addition, the data that these automated kiosks generate prove to be very interesting due to the fact that we can learn about the duration of the rental, departure, and arrival locations. With these bike sharing systems in place, analysts can learn about the mobility of a city since this system acts as a sensor network and can potentially predict bike sharing demand in a particular city. In this machine learning competition which is hosted by Kaggle, the goal is to implement a model that can predict the hourly bike rental demand in the Capital Bikeshare program in Washington, D.C. given historical usage patterns along with weather data. My research shows that using decision trees as the base classifier works well with these types of classification problems. Therefore in this project, I implemented several classification approaches including Random Forests and Gradient Boosting Machine. Various pre-processing and feature selection methods of the input features are analyzed and tested to discover the best indicators that can predict bike sharing demand. In the end, the model that results in the highest accuracy is a Gradient Boosting Machine which has a Root Mean Squared Logarithmic Error (RMSLE) of 0.41691 and put me in the top 12% of the leaderboard.

This paper is organized as follows. Section 2 covers the background and related work on bike sharing systems. Section 3 introduces the Kaggle data used in this project and its various features. Section 4 briefly discusses the software tools used to implement various supervised machine learning models. Section 5 goes over how each of these models are evaluated. Section 6 gives an overview of the project approach and exhibits the results of the various approaches that were implemented. Lastly, Section 7 concludes with a discussion of the results.

2 Background

Bike sharing systems are becoming increasingly popular due to their numerous advantages. By promoting the short term bike rental system, people can leave the stress of congested downtown traffic, tourists can enjoy the city without dealing with multiple taxis or buses, and the public population can face less pollution and more exercise.

A few examples of these bike sharing systems are the Velib in Paris which is the largest in the world, the Hanhzhou Public Bicycle in China which is the second largest in the world, and Citi Bike in New York which is the largest in the US. For example, Citi Bike has a total of 6,000 bikes, 507 stations, and over 25,000 daily rentals.

As we can observe, bike sharing systems provide a vast amount of data that we can use to study further. A previous paper published by Hadi Fanaee Tork uses this same bike sharing dataset to propose an event labeling system that relies on an ensemble of detectors and background knowledge. Generally, event labeling refers to the process of marking events in unlabeled data. Early procedures of event labeling involved human experts which can be expensive and time consuming. Tork suggests that an alternative method of referring to human experts is using background knowledge which is just as effective for labeling events. Furthermore, Tork uses this particular bike sharing dataset as it contains historical and weather data which allows it to be a good example of using background knowledge in an event labeling system.

3 Data

The data is provided by Hadi Fanaee Tork and is hosted on the UCI Machine Learning Repository which contains both the training and testing dataset. This dataset spreads across two years and ranges from January 2011 to December 2012. The training set contains the data from the first two-thirds or first 19 days of each month while the testing set contains the last third or ranges from the 20th to the end of the month. There are 10,886 instances in the training set and 6,493 instances in the testing set. While the training data provided contains a total of 9 features and 3 labels as shown below, the testing data contains only the 9 features without the labels.

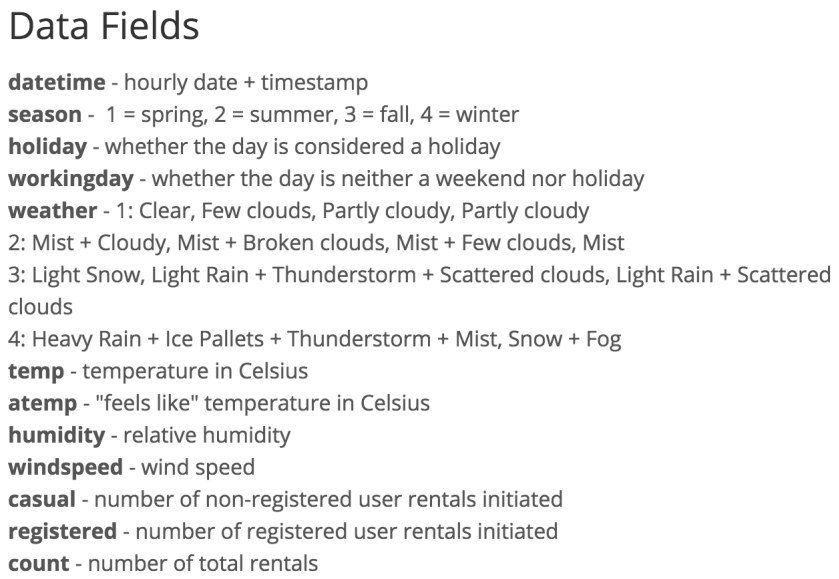


Figure 1: Features in the Bike Sharing Demand Dataset

The labels provided are casual, registered, and count. While the goal of the competition is to predict the count variable, count is just the sum of casual and registered users. From the table above, we can also observe that the dataset contains categorical, integer, decimal, and datetime values.

4 Software Tools

For this project, a combination of R and Python is used. R is primarily used to create the initial plots and visualizations using matplotlib which facilitates data visualization and yields insights into how the data is related. Python is used for a majority of the data processing, given the large number of libraries that are publicly available. Skikit-learn has a number of built-in machine learning algorithms. Numpy is very useful for working with matrices and different feature vectors. Pandas has a high performance data analysis tool used to analyze large amounts of data.

5 Evaluation

To evaluate each model, the trained model is applied to the testing data and is compared to the classifying algorithms based on performance. The metrics used to evaluate the performance of each of the algorithms is the Root Mean Squared Logarithmic Error (RMSLE). The RMSLE can be calculated as shown below.

$$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(\log(p_i+1)-\log(a_i+1))^2}$$

Where:

- n is the number of hours in the test set
- p_i is your predicted count
- a_i is the actual count
- $\log(x)$ is the natural logarithm

Figure 2: Root Mean Squared Logarithmic Error (RMSLE)

While RMSLE is just the square root of the Mean Square Error (MSE), the Root Mean Squared Error (RMSE) is the distance of a data point from a fitted line. Generally, the smaller the RMSLE, the better the accuracy of the model.

6 Approach

6.1 Visualizations

To get a better understanding of the data, plots are generated in R using ggplot to determine the correlation of various features. Since the goal of this project is to determine the hourly bike rental demand, there must be some sort of correlation between the datetime and count feature. After plotting these two features, a clear difference in bike sharing demand between the weekdays and weekends is shown in Figure 3. During the weekdays, people tend to rent bicycles in the morning or evening which must be due to the weekday commuting hours. However in the weekends, people rent bicycles during the day.

After plotting the time of day against the number of bike rentals, temperature is also included in this plot to see if it has an effect on hourly bike rentals. Figure 4 shows the time plotted against the count and temperature on weekdays. This figure implies that people rent bikes during the mornings and evenings when the weather is warm enough to ride a bike.

Figure 5 plots the time against the count and temperature on weekends. This figure shows that people rent more bikes during the day when the weather is warm enough.

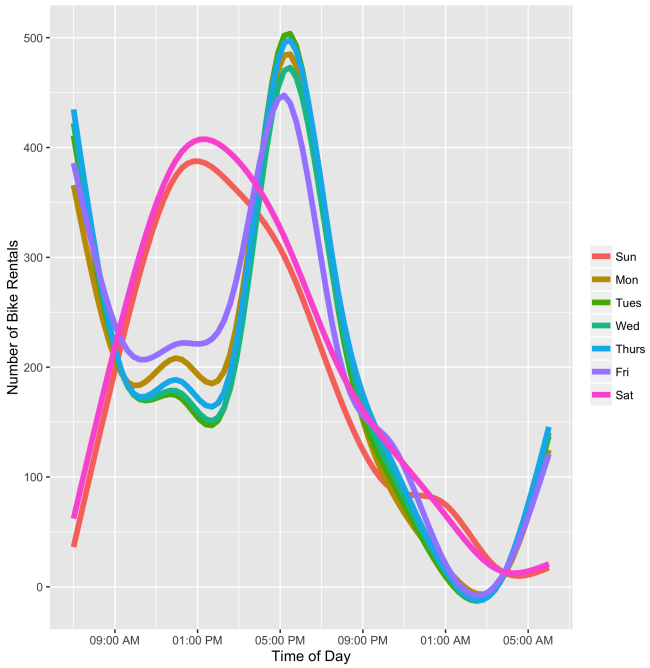


Figure 3: Bike rentals by time of day

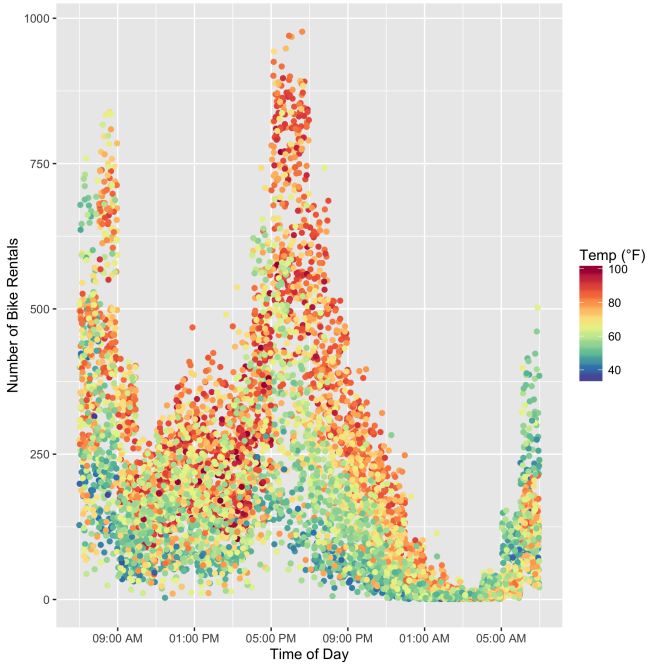


Figure 4: Bike rentals by time and temperature on weekdays

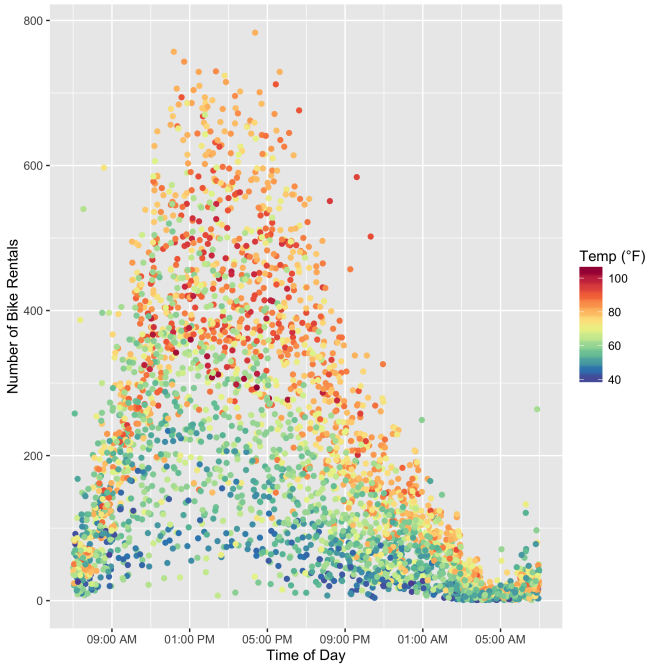


Figure 5: Bike rentals by time and temperature on weekends

Creating these visualizations provides insights into how these features impact bike rental demand. I learned that time of day and temperature definitely influences the bike rental count and probably factors as important features when creating a supervised machine learning model.

6.2 Feature Engineering

Feature engineering is a process in which we use the existing data to create new features that can potentially improve the accuracy of a particular model. The following shows what was done to transform the data.

1. Individual parts of the datetime column can be very useful to predict the bike rental count so this feature is split into new features such as year, month, weekday, and hour.
2. The count column is just the sum of the registered and casual users. Since users who are registered can behave differently than casual users, training separate models for the registered and casual users to predict bike rental demand might yield better results.
3. Since the model is being evaluated under the RMSLE, this means that everything is evaluated under a logarithmic scale. Therefore, targets counts need to transform into a logarithmic domain as well.

6.3 Random Forest

After researching about the most effective supervised machine learning models for predictive analysis, Random Forests and Gradient Boosting Machine seemed to be promising in terms of their performance. Since both models use decision trees as the base classifier, this makes sense as decision trees are good methods to apply on event labelling systems. The Random Forests model uses a bootstrapping method for training and testing by generating random samples from the dataset and uses decision trees for the prediction. In other words, Random Forests uses multiple weak learners on different subsets of the data and combines them to form a strong learner. The Gradient Boosting Machine is an additive model which means that it makes predictions by combining decisions from a sequence of base models which can be as simple as a decision tree. Comparing the two models, Random Forests are generally easier to tune due to fewer number of hyperparameters and harder to

overfit. While the Gradient Boosting Machine can perform better than Random Forests, it has more hyperparameters to tune.

In the first attempt, a simple Random Forests model is applied to the dataset. After reading in the training and testing data using the csv library in Python, the features array is initialized to contain all the features. Then, the values in the datetime feature are parsed so that they now contain year, month, hour, and weekday as separate features. The Random Forests model contain a hyperparameter called n_estimators, which is basically the number of trees in the forest. In order to tune this hyperparameter, a range of numbers from 1 to 500 was applied on the training dataset to train this model. After evaluating against the testing dataset, the RMSLE is calculated to determine the most optimal value. In this case, when n_estimators was closest to 100, the model became the most accurate. Overall, the Random Forest model received an RMSLE of 0.6683 which put me in the bottom 28% of the leaderboard.

```
rf = RandomForestClassifier(n_estimators = 100)
rf = rf.fit(train_features, train_count)
result = rf.predict(test_features)
```

Figure 6: Random Forest Classifier in Python

6.4 Feature Selection

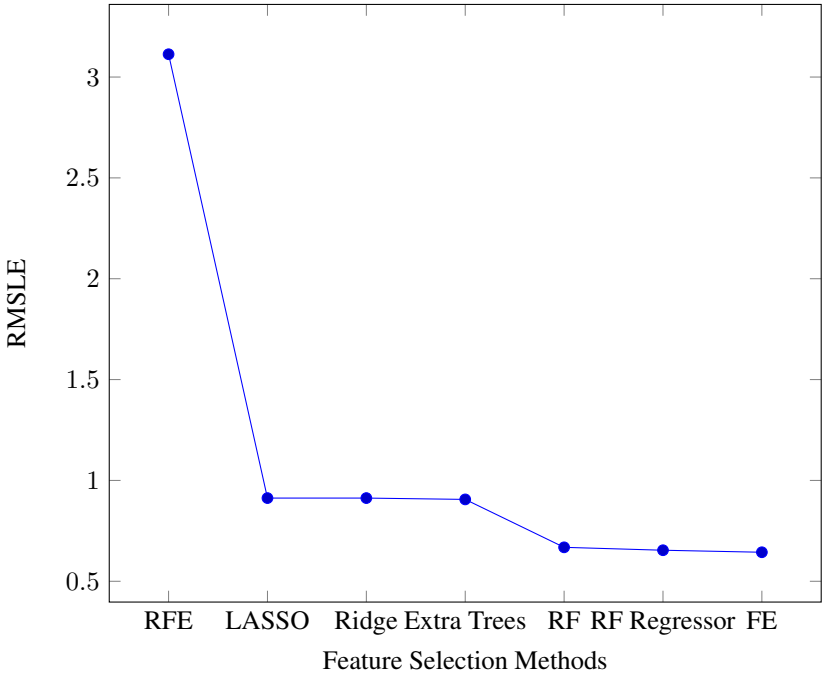
After applying the Random Forest model using all the features which resulted in a RMSLE of 0.6683, there is a possibility that a subset of features would lead to a better accuracy. Several different methods of determining feature importance were researched and discussed next.

1. **Recursive Feature Elimination**
   The first feature selection method is Recursive Feature Elimination (RFE) with Logistic Regression. RFE determines feature importance by recursively considering smaller sets of features. Initially, weights are assigned to each initial set of features based on training the dataset with the Logistic Regression estimator. Features with the smallest weights are removed and the estimator is applied again and assigns new weights. This procedure continues until it reaches the desired number of features. This method suggested to use a subset of season, workingday, and weather which resulted in a RMSLE of 3.11311. This decreased the original accuracy drastically and did not prove to be a good way to determine feature importance.
2. **LASSO and Ridge Regression**
   The next feature selection method includes LASSO and Ridge Regression. Both LASSO and Ridge Regression are forms of regularized linear regression. While the coefficients in the Ridge Regression are normally distributed, the coefficients in LASSO are Laplace distributed. LASSO is the least absolute shrinkage and selection operator and coefficients can be zero which makes it easier to eliminate some features. Both suggested using time, season, temp, and atemp which resulted in a RMSLE of 0.91249.
3. **Extra Trees**
   The next method is Extra Trees which is an ensemble of decision trees. This method uses a number of randomized decision trees on different subsets of the data to improve accuracy and reduce overfitting. This method suggested to use time, windspeed, humidity, and temp which resulted in a RMSLE of 0.9059.
4. **Random Forest Regressor**
   The next method includes the Random Forest Regressor. The Random Forest Regressor uses the Random Forest model to predict real valued outputs with continuous labels. In this case, since we are trying to predict the future bike rental demand, we do not know all possible outputs. This method suggested to use time, workingday, temp, and humidity which resulted in a RMSLE of 0.6539 which is a slight improvement but not significant enough to make a difference.
5. **Feature Elimination**
   The last approach eliminates each feature completely from the dataset and then trains on the original Random Forest model. After eliminating each feature, the RMSLE is calculated and

observed to see whether it improved or not. Based on these results, each feature is ranked by its importance and is determined that time, workingday, and season were significant. Using these three features with the Random Forest model resulted in a RMSLE of 0.64388 which is an improvement from the original score of 0.6683. However, this improvement is still not good enough as this model only put me in the top 72%. Since none of these feature selection methods has a significant improvement in the RMSLE using Random Forests, the Gradient Boosting Machine method is applied next to see whether this model might result in better accuracies.

Feature Selection Method	**RMSLE**
RFE	3.11311
LASSO	0.91249
Ridge Regression	0.91249
Extra Trees	0.9059
Random Forests	0.6683
RF Regressor	0.6539
Feature Elimination	0.64388

RMSLE for Various Feature Selection Methods



6.5 Gradient Boosting Machine

As mentioned previously, the Gradient Boosting Machine is a model where subsequent samples depend upon weights given to records in previous samples which did not predict correctly. These are also known as weak learners. The Gradient Boosting Machine has several hyperparameters that need to be tuned in order to perform optimally. The first is num_iterations which is the number of trees that are generated in the model. Generally, more trees means higher accuracy but it also means it has a longer training and prediction time. Other hyperparameters are max_depth which restricts the depth of each tree to prevent overfitting, learning_rate which shrinks the contribution of each tree by this rate, and the min_samples_leaf which are the minimum number of samples required to be at a leaf node.

In this approach, feature engineering is first applied by converting the datetime column into year, month, hour, and weekday. The casual, registered, and total count columns is also converted to be in the natural logarithm since we are dealing with the RMSLE. Once preprocessing was completed,

the next step is to tune the hyperparameters. The training data is split such that 90% is used for the training set and 10% for the validation set.

Using scikit-learn?s GridSearchCV, an optimal set of hyperparameters can be determined using cross validation. First, only min_samples_leaf, max_depth, and learning_rate are tuned, leaving the n_estimators to tune later. The optimal values are 0.1 for learning_rate, 5 for max_depth, and 30 for min_samples_leaf. To determine the optimal value for n_estimators, a range of numbers from 1 to 500 are used. The error rate of both the training and validation set are plotted as the number of estimators increased to see at which point the difference in error between the two sets starts to increase. This point occurs when the number of estimators was close to 85. After the optimal hyperparameter values are found, the Gradient Boosting Machine model can now be applied on the total training data. The RMSLE resulted in 0.42122 which is a significant improvement over the Random Forest model.

As mentioned previously, there are three labels in the training set: registered, casual, and count. The count feature is just the sum of the registered users and casual users. If there is a significant change in behavior between the casual and registered users, this could potentially influence the model?s accuracy when trained separately. Therefore, the casual and registered users are trained separately and then combined together to see if it can make any additional improvements. However, this approach did not improve the RMSLE which therefore left me with my highest score of 0.42122.

```
gbm = GradientBoostingRegressor(n_estimators=85, learning_rate=0.1, max_depth=5, min_samples_leaf=30)
gbm.fit(train[features], train['count_log'])
result = np.expm1(gbm.predict(test[features]))
```

Figure 7: Gradient Boosting Regressor in Python

7 Conclusion

In conclusion, I started this project to predict the bike rental demand in Washington, D.C. given historical and weather data. Given that this data represents an event labelling system, I compared various supervised machine learning models such as Random Forests and Gradient Boosting Machine which both use decision trees as the base classifier. Additionally, I pre-processed and tested various combinations of the provided features using feature selection. In the end, I am pleased to produce results that are high enough to be in the top 12% of the leaderboard. From starting with Random Forests which placed me in the top 72%, I was able to improve my model using Gradient Boosting Machine which got me to the top 12%.

From this project, I gained a few takeaways and future directions. First of all, manipulating a lot of data can be difficult, but using the right tools, finding the right features, and organizing it beforehand make a huge difference. Secondly, there are many methods to determine feature selection and taking the time to understand each method can potentially improve a model?s accuracy. Although feature selection did not drastically improve my model, I would like to compare and contrast various feature selection methods to see which one, if any, can improve my model in the future. Furthermore, tuning hyperparameters and finding the optimal hyperparameters can make for a more powerful classification algorithm.

In conclusion, I learned quite a lot of interesting ways to manipulate large amounts of data and find important features out of the dataset. In addition, I was able to make a good analysis of what types of supervised machine learning models performed better than others and which features contribute the most in improving the performance of my models.

Acknowledgments

I would like to thank my advisor, Professor Wei Wang, and the University of California, Los Angeles for their time and resources. In addition, I would also like to thank Kaggle for hosting this machine learning competition.

References

[1] Fanaee-T, Hadi, and Gama, Joao, Event labeling combining ensemble detectors and background knowledge, Progress in Artificial Intelligence (2013): pp. 1-15, Springer Berlin Heidelberg.

[2] Natekin, Alexey, and Alois Knoll. “Gradient Boosting Machines, a Tutorial.”Frontiers in Neurorobotics 7 (2013): 21. PMC.

[3] Breiman, L. “Random Forests.” Machine Learning (2001) 45: 5. doi:10.1023/A:1010933404324

[4] Isabelle Guyon and Andre Elisseeff. An introduction to variable and feature selection. The Journal of Machine Learning Research, 3:1157-1182, 2003.