**Forecasting Bike Sharing Demand**

**Effect of Weather on Ridership**

Author: Ryan Humble

Abstract

Bike-sharing programs, of which there are nearly 500 worldwide, generate very attractive sets of data, including duration of travel, departure location, arrival location, and time elapsed. Naturally, these data are often used to study mobility in a city. Yet from the perspective of the company or government that runs the program the more interesting question is to determine what drives some of these factors. Being able to forecast number of riders in a day, say, would be quite valuable in ensuring a sufficient but efficient number of bikes.

This paper seeks to answer that exact question. By using only weather data and historical usage patterns, an attempt is made to forecast bike usage on a daily basis. Various models and model types – including linear, time-series, generalized boosted regression, and random forests – are employed here to find the best predictions, as compared to set of test data, given this information. The best models, one a generalized boosted regression model and another a random forest, actually incorporated two submodels, one to model the number of casual riders and another to model the number of registered riders. The predictive ability of these models allows for the conclusion that weather, month, day of the week, and hour are all important factors in forecasting bike sharing demand.

# Introduction

Most major metropolitan areas in the western world have by now introduced a bike-sharing system as an alternative and complement to existing public and private transportation options. Each system is fundamentally comprised of a network of kiosk locations that allows users to rent and return bikes. Naturally, this system, which collects information including the time of rental, duration of travel, and departure and arrival location, is highly indicative of mobility within a city.

One useful path of inquiry then is to use this information, as well as that from other public transportation systems or systems with publicly available information, to somehow model movement within a city to better understand city dynamics and possibly glean useful social or economic conclusions. However, from the perspective of a government or company managing a bike-share program, perhaps the more economically useful question is how can the usage of the system itself be modeled instead of using it to model the subject of other research. A well-modeled system most certainly allows for a more efficient distribution of bikes across the city and over time. Particularly a model that offers some amount of forecasting ability allows those running the bike-share system to ensure that an adequate but efficient number of bikes are located in the proper locations.

A key question then is what exactly has an effect on ridership, possibly weather, demographics, or built environment characteristics to name a few. Therefore, the point of a study of this type is to determine if any of these factors provide some explanation for the level of ridership across space and time and, ideally, grant the ability to forecast ridership. For this particular paper, the task is to determine whether weather and time information can reliably be used to predict ridership.

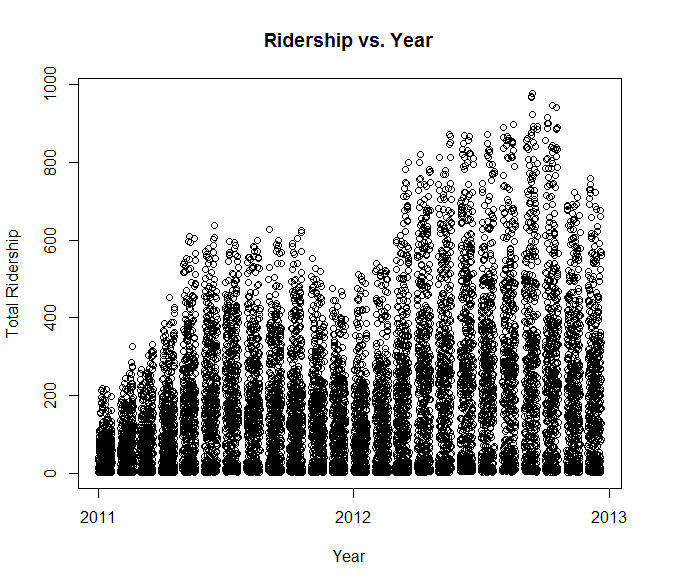
# Data Overview

Using data[[1]](#footnote-1) from the Capital Bikeshare program in Washington, D.C, an attempt is made in this analysis to forecast bike rental demand based on historical usage patterns and weather data. Specifically, the data used here provides hourly rental data for a period of two years. For each timestamp, the temperature, apparent temperature[[2]](#footnote-2), humidity, and windspeed are recorded. Additionally, each day is associated with a particular season – spring, summer, fall, or winter – and is marked as being a workingday, weekend, or holiday. An important part of any prediction modeling is the ability to test prediction against true results; therefore, this data is separated into a training and test portion. In the training portion, there are three ridership figures: casual, registered, and total. The distinction between casual and registered is merely whether the user had previously registered to use the system and will be analyzed further later. In testing predictions though, this paper only analyzes the predicted total ridership count.

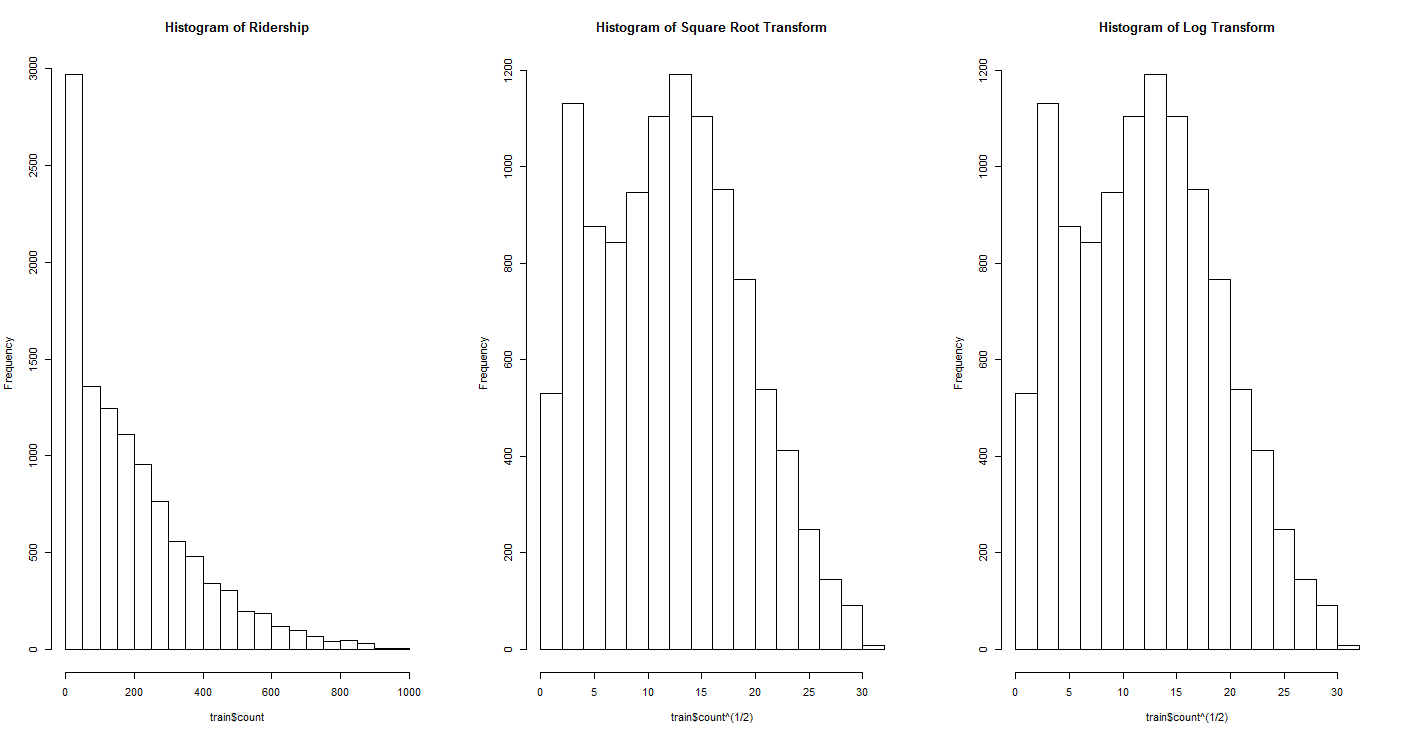
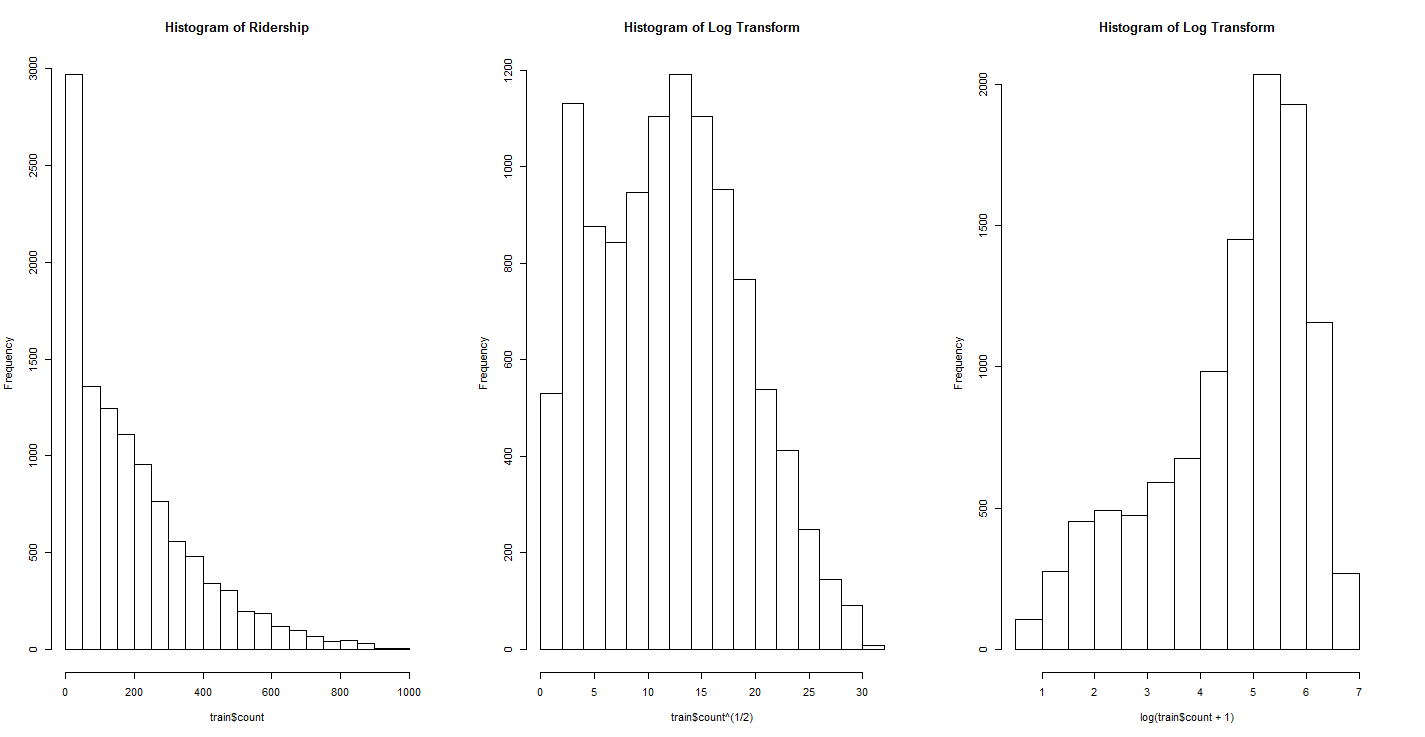
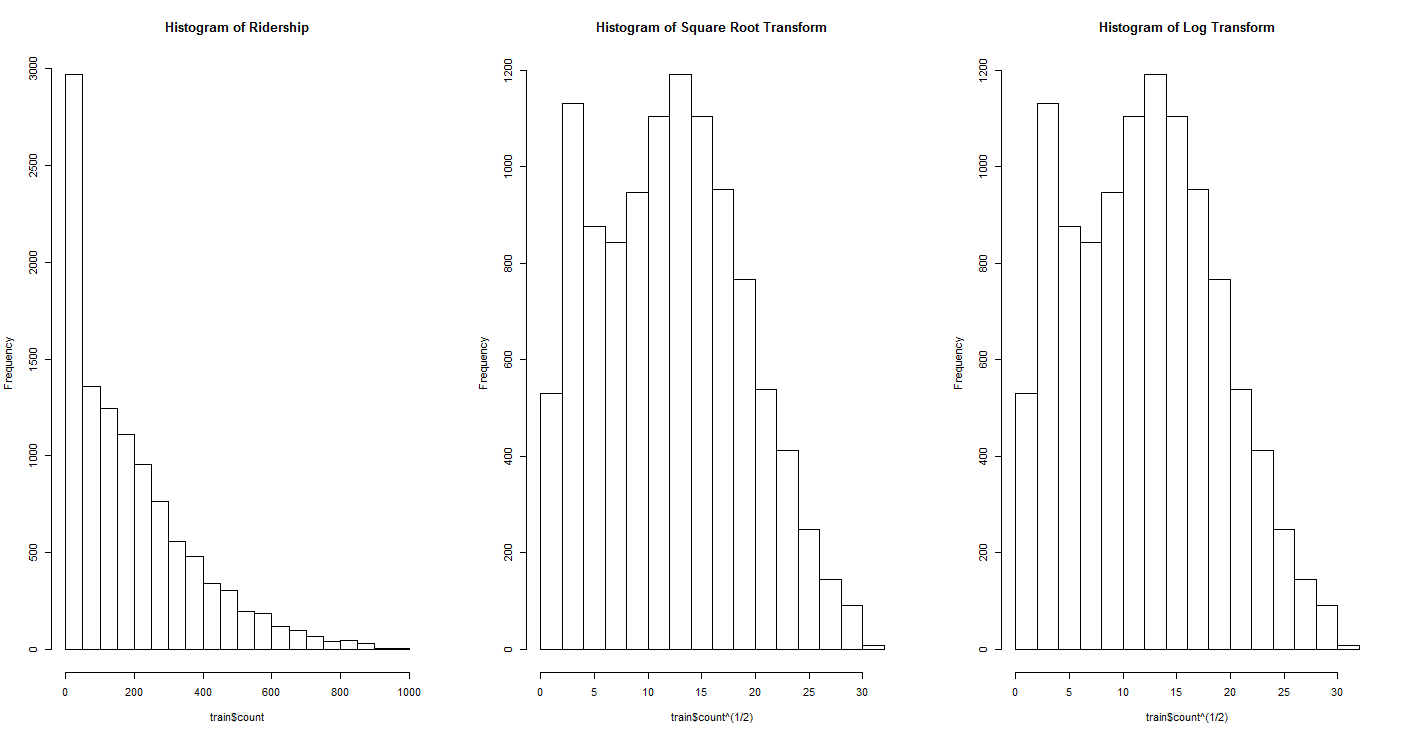
## Explanatory Variables

Before attempting any modeling of ridership based on the various explanatory variables, an individual analysis of each variable and total ridership is done. To test the individual significance of the variables, a one way analysis of variance for the categorical variables and a t-test from a simple linear regression involving individually each numerical variable is performed. The resulting p-values, which indicate the statistical significance of each variable individually in relation to total ridership, for the categorical variables indicates that season and weather are individual significant while holiday and workingday are not. The p-values from the t-test for temperature, apparent temperature, humidity, and windspeed all indicated individual significance. However, this is slightly misleading as these weather variables are not independent. Most obviously, temperature and apparent temperature are highly correlated, with a correlation of 0.985.

Lastly, the ridership values themselves should be analyzed. Since ridership inherently varies over time, Figure 1 shows ridership plotted over time. From the graph, there are a number of things to note. The first is the equispaced gaps in the data; this is a function of the training-test split. For each month, the first twenty days were used for training while the remaining days were used for test. Second, there is clearly a seasonality component to ridership in addition to an increase in ridership over time. Most importantly, Figure 1 shows a clustering of ridership figures close to zero. Figure 2 (left) is a histogram of the hourly ridership totals, Figure 2 (middle) shows a histogram of a square root transformation of ridership, and Figure 2 (right) shows a histogram of a shifted logarithmic[[3]](#footnote-3) transformation. This square root transformation yields a more normally distributed dependent variable to model. The logarithmic transformation is considered because of the special error function used in this paper. Finally, total ridership is broken down into casual and registered riders in the data. The percentage of registered riders has a very seasonal component to it; modeling casual and registered users separately then might capture the seasonality of ridership better.



*Figure 1: Total Ridership over Time*

*Figure 2: Histogram of Ridership, Square Root Transformation of Ridership, and Shifted Logarithmic Transformation of Ridership*

# Methods and Analysis

This data lends itself to very standard types of modeling. Attempting to use a number of explanatory variables to explain a single variable lends itself foremost to a linear regression model. However, as ridership is inherently a variable that depends strongly on time, a time-series model might make more intuitive sense. Beyond these simpler model types, this paper also employs generalized boosted regression models and random forests.

For each model type, there are generally six models. The first regresses the total ridership count on the various explanatory variables. The second breaks the datetime[[4]](#footnote-4) into year, month, day of week, and hour as to give more granular time information[[5]](#footnote-5). The third uses the second set of explanatory variables except there is a square root transformation on the total ridership count. The fourth additionally breaks down total ridership into casual and registered ridership and models the two separately before combining the predictions. The fifth builds upon the fourth model by adding a categorical variable for time of day[[6]](#footnote-6) and a categorical variable specifically indicating whether it is a Sunday or not. The sixth model simply changes the transformation used in the fifth model to the logarithmic transformation considered earlier. To establish a shorthand,

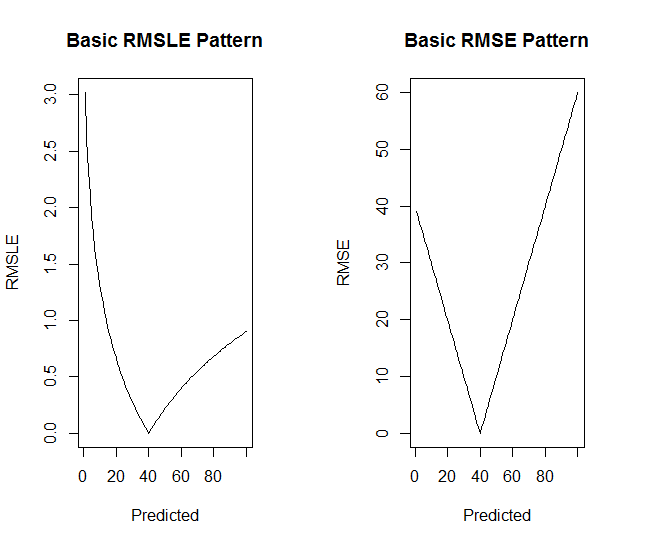
|  |  |
| --- | --- |
| Abbr. | Explanation |
| Date | Datetime Expansion |
| Sqrt | Square Root Transformation of Ridership |
| C.R. | Casual/Registered Seperation |
| Full | Datetime Expansion with Categorical Variables for Time of Day and Sunday |
| Log | Shifted Logarithmic Transformation of Ridership |

## Comparing Model Predictions

With such an array of models and model types, a simple criteria for comparing the model predictions is quite helpful. This analysis relies on comparing the Root Mean Square Logarithmic Error (RMSLE) of the various models. This criteria, where is the number of hours in the test set, is the predicted ridership, and is the actual ridership, is minimized by optimal

models: those with the smallest differences between the predicted and actual ridership. However, there is markedly different behavior of this criteria from a more traditional Root Mean Square Error (RMSE) criteria. An example, with and , for both is shown below in Figure 3. Clearly the scales of the graphs are quite different, but in choosing a model that minimizes either criteria, this scaling is rather unimportant. The key thing to note here is that RMSE penalizes equally for both over- and under-predicted values. RMSLE does not; it penalizes predications that are too small more heavily than those that are too large. The intuitive reason for why this makes sense in the context of bike share program is that having too few bikes is far more detrimental to the program than having some excess capacity. Then, the asymmetric penalty of the RMSLE criteria might be exploitable in the sense that the RMSLE might be reduced by a small, deliberate inflation of predicted ridership values.

Also, this error criterion is the motivation behind the shifted logarithmic transformation considered in the final model. Both generalized boosted regression and random forest algorithms for this data rely on minimizing a Mean Square Error (MSE). By performing the logarithmic transformation on the ridership values before running the algorithms, the algorithms are effectively still minimizing the RMSLE.



*Figure 3: Comparison of RMSLE and RMSE*

## Linear and Time-Series Models

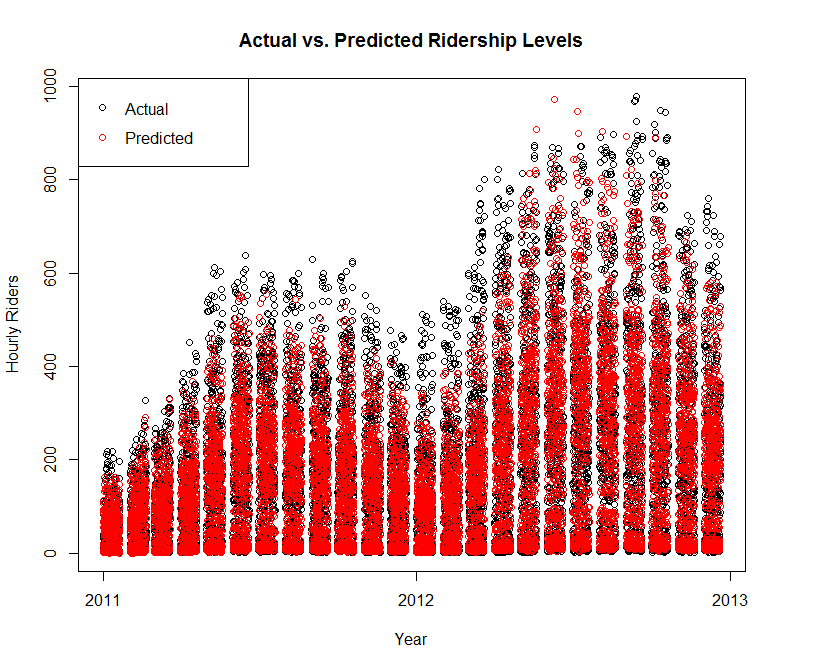
The model analysis begins with two workhorses of data analysis – linear and time-series models. For the various linear models fitted here, Table 1 shows the multiple R-squared, when fitting the training data, and RMSLE, with the test data, values. Note that each new feature introduced is beneficial. This is partially because of helpful feature engineering and partially because the size of the model increases with Date and Full. For the best linear model here, which is really a combination of two separate linear models, there are very few insignificant variables[[7]](#footnote-7), other than the linearly dependent variables[[8]](#footnote-8), so there is little opportunity to reduce the size of the model. Figure 4 shows the actual versus model-generated ridership values on the training set; it is clear that the model unfortunately underpredicts for many points. For the various time-series models, Table 2 shows important model information and statistics. The results are quite similar to those of the linear models, and a similar figure to Figure 4 is produced by the best time-series model.

|  |  |  |
| --- | --- | --- |
| Model |  | RMSLE |
| Standard | 0.3226 | 1.48807 |
| Date | 0.6947 | 1.15617 |
| Sqrt + Date | 0.794 | 0.78259 |
| Sqrt + Date + C.R. | 0.8167/0.8103 | 0.68057 |
| Sqrt + Full + C.R. | N/A[[9]](#footnote-9) | N/A |
| Log + Full + C.R. | 0.8336/0.8237 | 0.60907 |

*Table 1: Linear Model Statistics*

|  |  |  |
| --- | --- | --- |
| Model | Optimal ARMA[[10]](#footnote-10) | RMSLE |
| Standard | 5,0,5 | 1.35491 |
| Date | 4,0,3 | 1.20786 |
| Sqrt + Date | 5,0,2 | 0.79259 |
| Sqrt + Date + C.R. | 4,0,3/5,1,3 | 0.69770 |
| Sqrt + Full + C.R. | N/A[[11]](#footnote-11) | N/A |
| Log + Full + C.R. | N/A | N/A |

*Table 2: Time-Series Model Statistics*



*Figure 4: Comparison of Actual and Linear Model Generated Ridership Values*

## Generalized Boosted Regression Models

Generalized boosted regression (GBR)[[12]](#footnote-12) models are the first of the two machine learning type algorithms. Unfortunately, they are more complex than either linear or time-series models and less intuitive as well. Yet they are very good predictive models. Table 3 shows some statistics for the various generalized booted regression models. The importance of breaking down datetime into its components is made clear here by the shift in the most influential variables. Also, the inclusion of the two categorical variables in Full results in a significant boost in predictive ability even though those variables are linearly dependent; it must be that these variables are more favorable to make a decision upon.

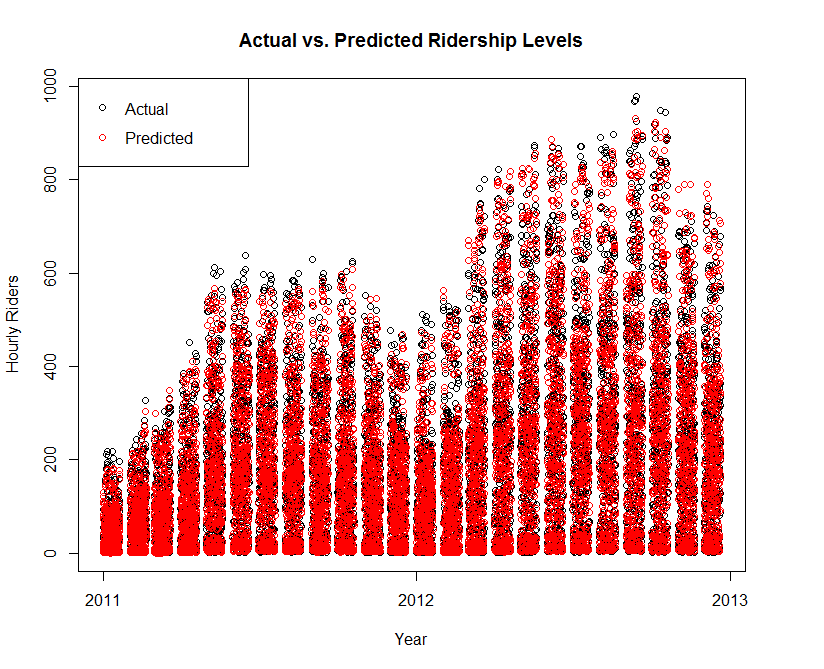
|  |  |  |
| --- | --- | --- |
| Model | Most Influential Variables | RMSLE |
| Standard | Humidity, Apparent Temp, Temp | 1.35491 |
| Date | Hour, Day, Year | 1.20786 |
| Sqrt + Date | Hour, Day, Month | 0.79259 |
| Sqrt + Date + C.R. | Hour, Temp, Day / Hour, Day, Month | 0.69770 |
| Sqrt + Full + C.R. | Hour, Temp, Month / Hour, Workingday, Month | 0.43609 |
| Log + Full + C.R. | Hour, Temp, Month / Hour, Month, Workingday | 0.41930 |

*Table 3: Generalized Boosted Regression Model Statistics*

One of the most interesting observations is the difference in the most influential variables in the final model; a side-by-side comparison of the variable importance in the submodels is shown in Table 4. Specifically, there is a notable difference in the relative influence of temperature and hour. The hour of rental is significantly more important in determining the number of registered users than the number of casual users; the opposite is true for temperature. This observation is quite easy to make in this case but was rather obscured when analyzing the linear or time-series models. Lastly, Figure 5 shows a comparison of the actual and model-generated values for ridership with the last model; the generalized boosted regression model seems to better capture the extremes of ridership.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Variable | Relative Influence |  | Variable | Relative Influence |
| Hour | 55.64007 |  | Hour | 76.04648 |
| Temp | 14.22 |  | Month | 5.166909 |
| Month | 8.356549 |  | Workingday | 4.249012 |
| Atemp | 4.330648 |  | Day | 3.762683 |
| Workingday | 4.053277 |  | Year | 3.667489 |
| Day | 3.749789 |  | Temp | 1.831522 |
| Humidity | 3.258342 |  | Humidity | 1.729017 |
| Timeday | 2.210277 |  | Weather | 1.314695 |
| Weather | 1.926755 |  | Atemp | 0.704061 |
| Year | 1.205073 |  | Season | 0.62556 |
| Windspeed | 0.808052 |  | Timeday | 0.448108 |
| Holiday | 0.15484 |  | Windspeed | 0.323554 |
| Season | 0.059585 |  | Holiday | 0.109379 |
| Sunday | 0.026737 |  | Sunday | 0.021527 |

*Table 4: Comparison of Most Influential Variables: Casual (left) and Registered (right)*



*Figure 5: Comparison of Actual and GBR Model Generated Ridership Values*

## Random Forests

The second machine learning algorithm employed here is random forest. Random forests are quite similar to generalized boosted regression, mainly in using numerous weak decision trees together to offer good prediction. Table 4 shows the important variables and predictive performance for each model. For the first four models, the most influential variables are very similar to those for the first four generalized boosted models; however, the last two models have noticeably different most influential variables. Despite this difference, the best generalized boosted regression model has nearly identical predictive ability as the best random forest. As with generalized boosted regression, as explained in the Appendix, the random forest algorithm relies on a number of parameters. A number of different sets of parameters were tried, but no significant difference in predictive ability was noticed.

|  |  |  |
| --- | --- | --- |
| Model | Most Influential Variables (by %IncMSE[[13]](#footnote-13)) | RMSLE |
| Standard | Humidity, Datetime, Apparent Temp | 1.36511 |
| Date | Hour, Year, Day | 0.50516 |
| Sqrt + Date | Hour, Day, Year | 0.49532 |
| Sqrt + Date + C.R. | Hour, Day, Temp / Hour, Day, Year | 0.46083 |
| Sqrt + Full + C.R. | Year, Humidity, Month / Year, Month, Hour | 0.42561 |
| Log + Full + C.R. | Year, Month, Humidity / Year, Hour, Month | 0.41269 |

*Table 4: Random Forest Model Statistics*

# Conclusion

This analysis has shown that weather is quite useful in modeling bike share system ridership. The time of day, week, and year are all equally or more important as well. The best models of each type involve the full explanatory variable breakdown[[14]](#footnote-14), the shifted logarithmic transformation of ridership, and the separate modeling of casual and registered riders. Although both linear models and time-series models performed well, generalized boosted regression and random forests predicted the test set the best. Since the best generalized boosted model and random forest have different most influential variables, possibly the average predicted value from the two models is better in prediction; performing such an averaging yields a RMSLE value of 4.0959 [[15]](#footnote-15). This is an improvement but not a significant one.

The dominance of these machine learning algorithms over the simpler models is mostly to be expected. The generalized boosted regression models here analyzed interaction effects, which none of the other models considered. The random forests performed well because of their “tree” and “feature” bagging[[16]](#footnote-16). The linear and time-series models could simply not capture the variability in the ridership levels.

## Further Analysis

Naturally, this report is not exhaustive in its coverage of this data or topic, and there is ample opportunity for further analysis.

1. Foremost, interaction effects were not analyzed by any model other than the generalized boosted regression models. Perhaps a random forest that considered such effects might outperform a similar a generalized boosted regression model in predictive ability.
2. However, since the fit of a generalized boosted regression or a random forest model is at least slightly dependent on its parameters and very little optimization on those parameters was performed in this analysis, the results discussed here might easily be improved.
3. Additionally, only the current day’s weather was used to fit the ridership values; including lags of major weather variables might further increase the predictive ability of the models discussed here.
4. Despite the analysis of the asymmetric penalty of the RMSLE criteria, this report does not employ any inflation or other deliberate strategy tailored to this criteria; any rudimentary attempt to use a small inflation factor here was found to yield a *worse* RMSLE.
5. Although the results of this analysis, particularly the fitted models, are quite useful in drawing conclusions about the effect of certain factors on bike-share ridership in hindsight, from the perspective of the government or private company managing such a program a more interesting analysis would be to only use the data collected prior to the period of time during which ridership was being predicted. For example, the predictions done for the very first set of days in the test dataset would only rely on the first twenty days of information. This distinction is important because although the factors affecting ridership should not vary widely from city to city the actual fitted models will[[17]](#footnote-17). Therefore, each bike share system, in order to get the most useful results, need fit its own model to its existing data in order to predict future ridership. Although there is no expectation for any of these models built up over time to be vastly different from those determined to be optimal here, an analysis of essentially dynamic models will differ from that of these essentially static models.

# Appendix

## Generalized Boosted Regression

Also known as gradient boosting, generalized boosted regression (GBR) is a machine learning technique; it produces a final, prediction model by combining a group of weak prediction models. Since this is a machine learning technique, there are a number of parameters that can be tuned and tweaked from better modeling and prediction. The main parameters tuned here are the number of trees in the model, the interaction depth, and the shrinkage factor. The first is equivalent to the number of iterations of the algorithm; the last describes how finely the model is tweaked for each iteration. The interaction depth describes how many levels of interaction the technique will analyze for significance. Some amount of trial-and-error was done in picking these parameters for the various models here; without the entire test set, it is hard to optimize over these.

## Random Forests Predictive Ability

The predictive power of random forests comes from the bootstrapping it does in selecting trees and features. Specifically, random forests use bagging to reduce variance and avoid overfitting. Bagging involves creating new “training” sets from an existing set by sampling uniformly and with replacement. Random forests use this method in two ways. First, a random sample of the training data is selecting with replacement; the decision trees of the forest are fit to that subset of the data. Second, the decision trees used in random forests are slightly different from standard decision trees; at each split in the tree, these special decision trees only consider a subset of the features available. Using this methodology, random forests often have much better predictive power than decision trees or generalized boosted regression.

## Code

An abbreviated version of the code is found below. It leaves out the generation of the predictions on the test set, plot generation, data analysis, and model summary or analysis commands.

train = read.csv(file="train.csv", stringsAsFactors=F)

test = read.csv(file="test.csv", stringsAsFactors=F)

casual = train$casual

registered = train$registered

train = train[,-c(10,11)]

train$datetime = as.POSIXct(train$datetime, format="%Y-%m-%d %H:%M:%S")

train$season = factor(train$season, labels=c("Spring", "Summer", "Fall", "Winter"))

train$holiday = factor(train$holiday, labels=c("Non-holiday", "Holiday"))

train$workingday = factor(train$workingday, labels=c("Non-workingday", "Workingday"))

train$weather = factor(train$weather, labels=c("Clear", "Cloudy", "RainSnow", "4"))

#change the 1 data point for weather = 4 to a 3

train$weather[which(train$weather==4)] = "RainSnow"

train$weather = factor(train$weather)

dateTrain = train

dateTrain$hour = as.factor(as.numeric(format(dateTrain$datetime, "%H")))

dateTrain$day = as.factor(weekdays(dateTrain$datetime))

dateTrain$month = as.factor(as.numeric(format(dateTrain$datetime, "%m")))

dateTrain$year = as.factor(as.numeric(format(dateTrain$datetime, "%Y")))

scaleFullTrain = dateTrain[,-1]

dateTrain = dateTrain[,-c(1,2,4)]

scaleTrain = train

scaleTrain$count = scaleTrain$count^(1/2)

scaleDateTrain = dateTrain

scaleDateTrain$count = scaleDateTrain$count^(1/2)

hour = as.numeric(scaleFullTrain$hour)

scaleFullTrain$timeday = "Morning"

scaleFullTrain$timeday[(hour < 10) & (hour > 3)] = "Midday"

scaleFullTrain$timeday[(hour < 16) & (hour >= 10)] = "Evening"

scaleFullTrain$timeday[(hour < 22) & (hour >= 16)] = "Overnight"

scaleFullTrain$timeday = as.factor(scaleFullTrain$timeday)

scaleFullTrain$sunday = "0"

scaleFullTrain$sunday[scaleFullTrain$day == "Sunday"] = "1"

scaleFullTrain$sunday = as.factor(scaleFullTrain$sunday)

scaleFullTrain$count = scaleFullTrain$count^(1/2)

casualScaleDateTrain = scaleDateTrain[,-7]

casualScaleDateTrain$casual = casual^(1/3)

regScaleDateTrain = scaleDateTrain[,-7]

regScaleDateTrain$registered = registered^(1/3)

casualScaleFullTrain = scaleFullTrain[,-9]

casualScaleFullTrain$casual = casual^(1/3)

regScaleFullTrain = scaleFullTrain[,-9]

regScaleFullTrain$registered = registered^(1/3)

casualLogScaleFullTrain = scaleFullTrain[,-9]

casualLogScaleFullTrain$casual = log(casual+1)

regLogScaleFullTrain = scaleFullTrain[,-9]

regLogScaleFullTrain$registered = log(registered+1)

\*\*similarly for the test data\*\*

print("Running linear models.")

train.lm = lm(count ~ ., data=train)

dateTrain.lm = lm(count ~ ., data=dateTrain)

scaleTrain.lm = lm(count ~ ., data=scaleTrain)

scaleDateTrain.lm = lm(count ~ ., data=scaleDateTrain)

scaleFullTrain.lm = lm(count ~ ., data=scaleFullTrain)

casualScaleDateTrain.lm = lm(casual ~ ., data=casualScaleDateTrain)

regScaleDateTrain.lm = lm(registered ~ ., data=regScaleDateTrain)

casualLogScaleFullTrain.lm = lm(casual ~ ., data=casualLogScaleFullTrain)

regLogScaleFullTrain.lm = lm(registered ~ ., data=regLogScaleFullTrain)

print("Running time series.")

train.xreg = model.matrix(train.lm)[,-1] #removes intercept

train.arima = auto.arima(train$count, xreg=train.xreg)

dateTrain.xreg = model.matrix(dateTrain.lm)[,-1] #removes intercept

dateTrain.arima = auto.arima(dateTrain$count, xreg=dateTrain.xreg)

scaleTrain.xreg = model.matrix(scaleTrain.lm)[,-1] #removes intercept

scaleTrain.arima = auto.arima(scaleTrain$count, xreg=scaleTrain.xreg)

scaleDateTrain.xreg = model.matrix(scaleDateTrain.lm)[,-1] #removes intercept

scaleDateTrain.arima = auto.arima(scaleDateTrain$count, xreg=scaleDateTrain.xreg)

casualScaleDateTrain.arima = auto.arima(casualScaleDateTrain$casual, xreg=scaleDateTrain.xreg)

regScaleDateTrain.arima = auto.arima(regScaleDateTrain$registered, xreg=scaleDateTrain.xreg)

print("Running generalized boosted regression models.")

train.gbm = gbm(count ~ ., data=train[,-1], var.monotone=NULL ,distribution="gaussian",n.trees=15000,shrinkage=0.01,interaction.depth=4,bag.fraction = 0.5,train.fraction = 1,n.minobsinnode = 10,cv.folds = 10,keep.data=TRUE,verbose=TRUE)

dateTrain.gbm = gbm(count ~ .,data=dateTrain, var.monotone=NULL ,distribution="gaussian",n.trees=25000,shrinkage=0.01,interaction.depth=4,bag.fraction = 0.5,train.fraction = 1,n.minobsinnode = 10,cv.folds = 10,keep.data=TRUE,verbose=TRUE)

scaleTrain.gbm = gbm(count ~ .,data=scaleTrain[,-1], var.monotone=NULL ,distribution="gaussian",n.trees=15000,shrinkage=0.01,interaction.depth=4,bag.fraction = 0.5,train.fraction = 1,n.minobsinnode = 10,cv.folds = 10,keep.data=TRUE,verbose=TRUE)

rmsle.gbm(scaleTrain.gbm, scaleTrain[,-1])

scaleDateTrain.gbm = gbm(count ~ .,data=scaleDateTrain, var.monotone=NULL ,distribution="gaussian",n.trees=20000,shrinkage=0.01,interaction.depth=4,bag.fraction = 0.5,train.fraction = 1,n.minobsinnode = 10,cv.folds = 10,keep.data=TRUE,verbose=TRUE)

scaleFullTrain.gbm = gbm(count ~ .,data=scaleFullTrain, var.monotone=NULL ,distribution="gaussian",n.trees=20000,shrinkage=0.01,interaction.depth=4,bag.fraction = 0.5,train.fraction = 1,n.minobsinnode = 10,cv.folds = 10,keep.data=TRUE,verbose=TRUE)

casualScaleDateTrain.gbm = gbm(casual ~ .,data=casualScaleDateTrain, var.monotone=NULL ,distribution="gaussian",n.trees=20000,shrinkage=0.01,interaction.depth=4,bag.fraction = 0.5,train.fraction = 1,n.minobsinnode = 10,cv.folds = 10,keep.data=TRUE,verbose=TRUE)

regScaleDateTrain.gbm = gbm(registered ~ .,data=regScaleDateTrain, var.monotone=NULL ,distribution="gaussian",n.trees=20000,shrinkage=0.01,interaction.depth=4,bag.fraction = 0.5,train.fraction = 1,n.minobsinnode = 10,cv.folds = 10,keep.data=TRUE,verbose=TRUE)

casualScaleFullTrain.gbm = gbm(casual ~ .,data=casualScaleFullTrain, var.monotone=NULL ,distribution="gaussian",n.trees=20000,shrinkage=0.01,interaction.depth=4,bag.fraction = 0.5,train.fraction = 1,n.minobsinnode = 10,cv.folds = 10,keep.data=TRUE,verbose=TRUE)

regScaleFullTrain.gbm = gbm(registered ~ .,data=regScaleFullTrain, var.monotone=NULL ,distribution="gaussian",n.trees=20000,shrinkage=0.01,interaction.depth=4,bag.fraction = 0.5,train.fraction = 1,n.minobsinnode = 10,cv.folds = 10,keep.data=TRUE,verbose=TRUE)

casualLogScaleFullTrain.gbm = gbm(casual ~ .,data=casualLogScaleFullTrain, var.monotone=NULL,distribution="gaussian",n.trees=20000,shrinkage=0.01,interaction.depth=4,bag.fraction = 0.5,train.fraction = 1,n.minobsinnode = 10,cv.folds = 10,keep.data=TRUE,verbose=TRUE)

regLogScaleFullTrain.gbm = gbm(registered ~ .,data=regLogScaleFullTrain, var.monotone=NULL, distribution="gaussian",n.trees=20000,shrinkage=0.01, interaction.depth=4,bag.fraction = 0.5,train.fraction = 1,n.minobsinnode = 10,cv.folds = 10,keep.data=TRUE,verbose=TRUE)

print("Running random forest models.")

ntree = 200

mtry = 10

importance = TRUE

train.rf = randomForest(count ~ ., data=train, ntree=ntree, mtry=mtry, importance=importance, do.trace=T)

dateTrain.rf = randomForest(count ~ ., data=dateTrain, ntree=ntree, mtry=mtry, importance=importance, do.trace=T)

scaleTrain.rf = randomForest(count ~ ., data=scaleTrain, ntree=ntree, mtry=mtry, importance=importance, do.trace=T)

scaleDateTrain.rf = randomForest(count ~ ., data=scaleDateTrain, ntree=ntree, mtry=mtry, importance=importance, do.trace=T)

scaleFullTrain.rf = randomForest(count ~ ., data=scaleFullTrain, ntree=ntree, mtry=mtry, importance=importance, do.trace=T)

casualScaleDateTrain.rf = randomForest(casual ~ ., data=casualScaleDateTrain, ntree=ntree, mtry=mtry, importance=importance, do.trace=T)

regScaleDateTrain.rf = randomForest(registered ~ ., data=regScaleDateTrain, ntree=ntree, mtry=mtry, importance=importance, do.trace=T)

casualScaleFullTrain.rf = randomForest(casual ~ ., data=casualScaleFullTrain, ntree=ntree, mtry=mtry, importance=importance, do.trace=T)

regScaleFullTrain.rf = randomForest(registered ~ ., data=regScaleFullTrain, ntree=ntree, mtry=mtry, importance=importance, do.trace=T)

casualLogScaleFullTrain.rf = randomForest(casual ~ ., data=casualLogScaleFullTrain, ntree=ntree, mtry=mtry, importance=importance, do.trace=T)

regLogScaleFullTrain.rf = randomForest(registered ~ ., data=regLogScaleFullTrain, ntree=ntree, mtry=mtry, importance=importance, do.trace=T)

1. This data, and impetus for this entire research, comes from a competition on the predictive modelling and analytics competitions platform Kaggle: <http://www.kaggle.com/c/bike-sharing-demand>. Only the training dataset has publicly available ridership values; validating models against the test set had to be done through Kaggle’s website. [↑](#footnote-ref-1)
2. The “feels like” temperature is a function of temperature, relative humidity, and wind speed. [↑](#footnote-ref-2)
3. This transformation will become clear shortly; for now, this transformation takes a variable x and computes . The scale is important because there are a large number of ridership counts that are 0. [↑](#footnote-ref-3)
4. Datetime is in a format like 2013-04-17 05:00:00 EST. [↑](#footnote-ref-4)
5. To prevent the model matrix from becoming rank deficit, both season and workingday had to be removed. However, note that any seasonality component should now be captured by the representation of month as a categorical variable. As it turns out, workingday is not significant in a standard linear regression, yet it is still captured by the combination of day of week – Monday through Sunday – as a categorical variable and the holiday variable. [↑](#footnote-ref-5)
6. The day was broken down into four segments: 3am-10am, 10am-4pm, 4pm-10pm, and 10pm-3am. [↑](#footnote-ref-6)
7. Temperature is not statistically significant in the model for the number of registered riders; all the variables are statistically significant in the model for the number of casual users. [↑](#footnote-ref-7)
8. There are numerous linearly dependent variables. Specifically there is overlap between the categorical variable holiday and day as well as between season and month; however, none of those variables is entirely linearly dependent. The two categorical variables present in Full but not in Date are entirely dependent though. [↑](#footnote-ref-8)
9. Because the two additional categorical variables added are actually linearly dependent on existing variables, this model is no different than Sqrt + Date + C.R. [↑](#footnote-ref-9)
10. The parameters here, respectively, represent the auto-regressive order, the degree of differencing, and the moving average order. Note how these models all have similar components. [↑](#footnote-ref-10)
11. The additional of the two additional categorical variables has made the model matrix be of deficient rank. [↑](#footnote-ref-11)
12. See Appendix for a short description of generalized boosted regression. [↑](#footnote-ref-12)
13. %IncMSE is a measure of how much the MSE would change if the values of the variable were randomly permuted across the entire dataset. [↑](#footnote-ref-13)
14. When permissible. Time-series required a full rank model matrix. [↑](#footnote-ref-14)
15. At the time of writing, this score was 268th out of 2853. [↑](#footnote-ref-15)
16. The appendix has more information on random forests. [↑](#footnote-ref-16)
17. Very likely the coefficients in the linear or time-series models and the relative influences of the variables in the GBR models would be different enough from those determined in this analysis as to give poor predictions if blindly used for other cities or in other scenarios. [↑](#footnote-ref-17)