Machine Learning Engineer Nanodegree: Capstone Project

Forecast Use of Capital Bikeshare Program

September 17, 2018

1. Definition

1.1 Project Overview

Washington DC is falling in love with bike share. Capital Bikeshare is metro DC's bikeshare service, with 4,300 bikes and 500+ stations across 6 jurisdictions: Washington DC; Arlington, VA; Alexandria, VA; Montgomery, MD; Prince George's County, MD; and Fairfax County, VA. Bike sharing is a means of renting bikes where the process of obtaining membership, rental, and bike return is automated via a network of kiosk locations throughout a city. Bikeshare users are able to rent a bike from one location and return it to a different location on an as-needed basis.[1] It serves both work-related and personal travel needs. It is easy, fun, flexible, affordable and environment-friendly. Since its inception in 2010, Capital Bikeshare users have ridden more than 42 million miles, or 20 million trips, which reduced 28.64 million pounds of carbon dioxide, saved 1.72 million gallons of gasoline, and burned an astonishing 1.8 billion calories, according to the District Department of Transportation (April 26, 2018)[2].

Capital Bikeshare represents a small but growing component of the regional transportation system. Predicting bikeshare demand is important for effective operation and expansion of the existing bike sharing system. Knowledge of how demand fluctuates enables the bikeshare program management to keep the right amount of stock on hand. This project hopes to shed some light on the opportunities and barriers to strengthen the role of biking in the regional transportation system.

Many studies have been carried out to explore the usage of bike share systems in the world. For example, Kaltenbrunner et al. studied the spatial and temporal patterns of bike use over the hours of the day.[3] Zhou examined the spatial-temporal pattern of bike trips in Chicago.[4] Travel patterns between weekdays and weekends as well as between subscribers and casual users tend to be very different. Students from Stanford University also built different predictive models for bike sharing demand.[5] They tried methods including basic linear regression, Generalized Linear Models with Elastic Net Regularization, Generalized Boosted Model, Principal Component Regression, Support Vector Regression, Random Forest and Conditional Inference Trees. Random Forest and Conditional Inference Trees models reported the smallest RMSLE.

[1] https://www.kaggle.com/c/bike-sharing-demand
[2] https://ddot.dc.gov/release/capital-bikeshare-celebrates-20-million-trips-and-highest-daily-ridership-record
[3] http://www.dtic.upf.edu/~akalten/kaltenbrunner_etal2010PMC.pdf
[4] http://journals.plos.org/plosone/article?id=10.1371/journal.pone.0137922
[5]http://cs229.stanford.edu/proj2014/Jimmy%20Du,%20Rolland%20He,%20Zhivko%20Zhechev,%20Forecasting%20Bike%20Rental%20Demand.pdf

1.2 Problem Statement

This project focuses on predicting the number of bike rentals for the Capital Bikeshare program, as part of a Kaggle competition.[6] It aims to understand key factors driving the hourly bikeshare demand using historical bikeshare usage data with weather data. The objective here is a typical demand forecasting problem. The predicted variable is the demand – number of bike rentals every hour, which is a real number rather than a class or category. Therefore this is a regression problem rather than a classification problem.

Target variable in this project is hourly number of bike rentals (demand), which will be the variable to be predicted by the model. Input variables will be factors that drive the bike rental demand, such as the weather condition (rain, wind speed, humidity, temperature, etc.), time of the day, day of the week, season, holiday or weekday, etc.

1.3 Model Performance Evaluation Metric

Root Mean Squared Logarithmic Error (RMSLE) is selected to evaluate the model performance. The RMSLE is calculated as:

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

In the RMSLE equation, n is the number of hours in the test set, and p_i and a_i are the predicted count and actual count for a given hour respectively. The "+1" here ensures that the logarithmic function is always applied to a value that is greater than 0.

The effect of introducing the logarithm function is to balance the emphasis on small and big predictive errors. RMSLE penalizes an under-predicted estimate greater than an over-predicted estimate. In our case here, that means errors during peak hours do not dominate the errors made during off-peak hours, thanks to the logarithmic-based calculation in RMSLE.

2. Analysis

2.1 Dataset

For this project, data is obtained from Kaggle's, including hourly rental data spanning two years (2011-2012) for the Capital Bikeshare program. The training set is comprised of the first 19 days of each month, while the test set is the 20th to the end of the month. The goal is to predict the total count of bikes rented during each hour covered by the test set, using only information available prior to the rental period.
In total, there are 10,886 data points with 12 variables each in the train set, and 6,493 data points with 9 variables (no target variables) in the test set. Date fields include:

- datetime - hourly date + timestamp
- season - 1 = spring, 2 = summer, 3 = fall, 4 = winter
- holiday - whether the day is considered a holiday (binary indicator)
- workingday - whether the day is neither a weekend nor holiday (binary indicator)

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

- weather – four categories include:
    1) Clear, Few clouds, Partly cloudy, Partly cloudy
    2) Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist
    3) Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds
    4) Heavy Rain + Ice Pallets + Thunderstorm + Mist, Snow + Fog
- temp - temperature in Celsius
- atemp - "feels like" temperature in Celsius
- humidity - relative humidity
- windspeed - wind speed
- casual - number of non-registered user rentals initiated (dependent variable)
- registered - number of registered user rentals initiated (dependent variable)
- count - number of total rentals

The last three variables are considered as dependent variables, while the remaining variables are independent variables. These variables are appropriate given the context of the problem. For example, the sensitivity of bike use to weather conditions has been widely discussed in the literature.[7] In theory, bike usage can be affected by cold weather, precipitation, and excessive heat. It is also known that bike share demand shows seasonal patterns – peaking in summer/fall, and with a lull around holidays (especially for registered users).

2.2 Data Exploration and Visualization
The main characteristics of key variables in the raw dataset are explored and summarized below, to bring important aspects of the data into focus for further analysis.

2.2.1 Distribution of Target Variables
Figure 1 below indicates a right skewed frequency distribution for all three target variables. For a right skewed distribution, the mean is greater than the median; the tail of the distribution on the right hand side is much longer than on the left hand side.

With such a heavily right skewed data, it is a common practice to apply a logarithmic transformation on the data so that very large and very small values do not negatively affect the performance of a learning algorithm. When applying this transformation, it has to note that the logarithm of 0 is undefined, so the values need to be translated by a small amount above 0 to apply the logarithm successfully. Here, 1 is added to avoid error created by log transformation.

[7] https://phillymotu.files.wordpress.com/2013/04/the-impact-of-weather-conditions-on-capital-bikeshare-trips.pdf

Figure 1. Frequency distribution of target variables.

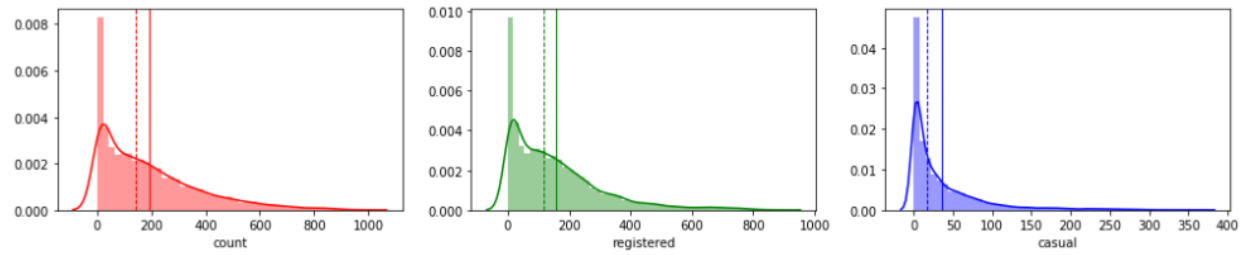
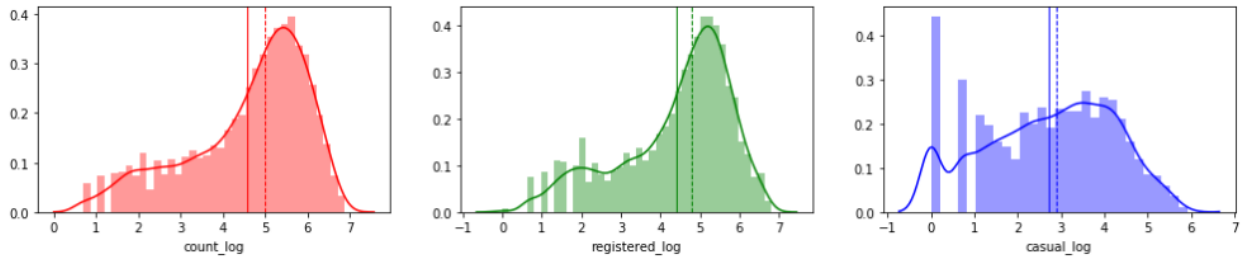


Figure 2. Frequency distribution of target variables after log transformation.

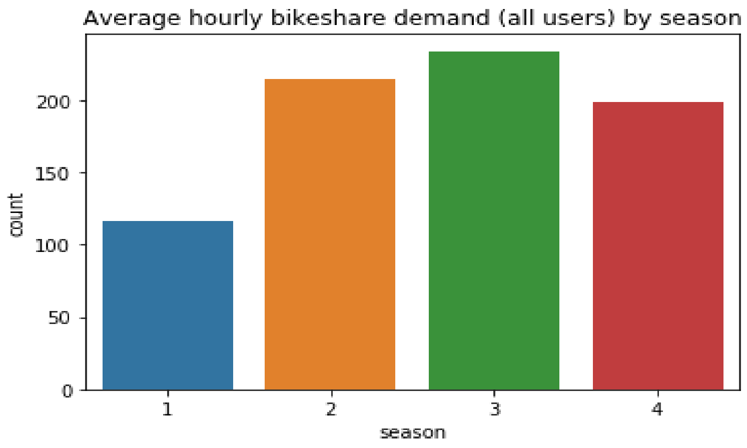


After the log transformation, the distribution of target variables looks closer to a normal distribution. Its mean (solid line) is slightly smaller than its median (dashed line) (Figure 2).

2.2.2 Seasonality Effects

Figure 3 shows that biking appears to be more popular during the summer and fall.

Figure 3. Average hourly bikeshare demand (all users) by season.

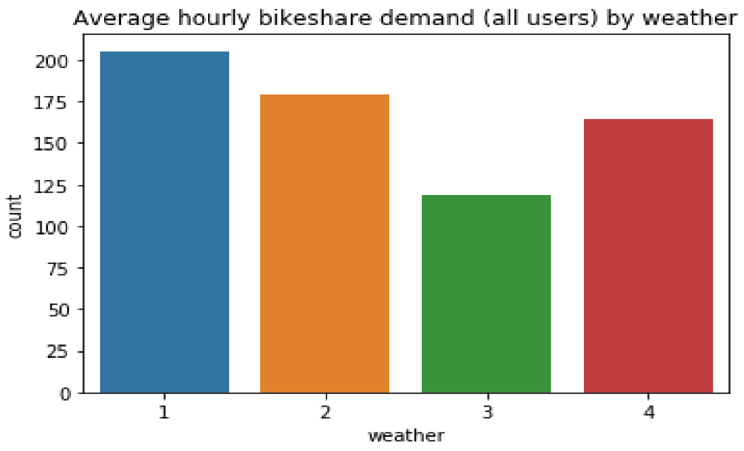


Note that seasonality can occur in any time interval. While the season variable in the raw data (Season - 1 = spring, 2 = summer, 3 = fall, 4 = winter) helps to explore the role of four seasons in explaining the spikes and dips of bikeshare demand, other types of seasonality are also worthy of examination. Multiple types of seasonality (hour of day, day of week, week of month, week of year, year) can be extracted from the datetime variable in the raw data.

2.2.3 Weather

Figure 4 indicates that weather type also plays a role in explaining bikeshare demand. When weather is nice and cozy, bikeshare demand goes up. Bikeshare use is negatively affected by cold weather and precipitation (rain and snow).

Figure 4. Average hourly bikeshare demand (all users) by weather.

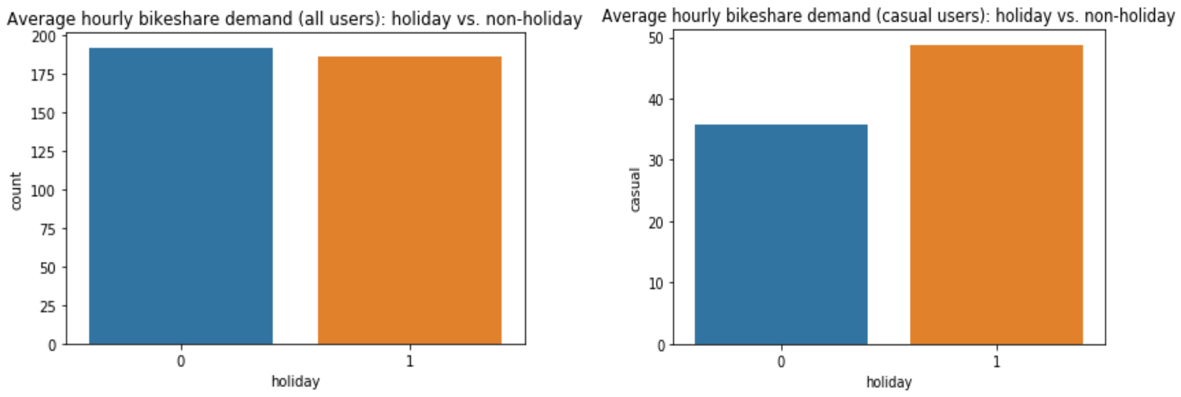


In addition to the categorical variable of weather in the data, there are four continuous weather related variables – humidity, temperature, feels like temperature, and wind speed. We create a pairwise scatter plot to visualize their relationship. At the first glance, bikeshare demand is positively related to temperature, and feels like temperature. It is also negatively related to humidity. Wind speed does not have a normal distribution and it has many zero values. Wind speed and bikeshare demand may not have a linear relationship.

2.2.4 Holiday and working day Impacts

Average hourly total bikeshare demand in non-holidays is slightly higher than it is in holidays. However, looking at casual users only, it is clear that holidays have a much stronger demand from casual users. The pattern observed in holidays is similar to non-working days (i.e. weekends).

Figure 5. Average hourly bikeshare demand (all users - left, and casual users only - right): holiday vs. non-holiday.



The holiday/non-working day impacts therefore can be understood in two-fold: first, holidays do affect total bikeshare demand that can be positive or negative depending on the nature of the holiday. For example, July 4th may enhance bikeshare demand due to the parade, concert, fireworks, etc. through the city. On the contrary, Christmas which is a typical time for families may see a steep decline of bikeshare use. Second, holidays/non-working days have an impact on the structure of the bikeshare demand. The market segmentation or the percentage breakdown of casual users and registered users tends to be very different, compared to holiday and non-holiday, working day and non-working day. Not surprisingly, demand from registered users tends to be heavy in non-holiday and working days, and fall sharply through the holidays and non-working days.

2.3 Algorithms and Techniques

2.3.1 Benchmark Model – Linear Regression

There is not an existing benchmark model. Instead, we will start with a multiple linear regression model with default parameters as a benchmark model. The performance of this model will serve as the benchmark for the more advanced approaches.

Linear regression is a linear approach to modelling the relationship between a dependent variable and one or more explanatory variables.[8] In linear regression, the relationships are modeled using linear predictor functions whose unknown model parameters are estimated from the data. We expect this linear regression model's performance would be relatively poor, as it may overfit the noise in the dataset.

A population model for a multiple linear regression model that relates an independent variable Y to independent variables X (X1,X2,…Xp-1) is written as:
$Y_i=b_0+b_1X_{i,1}+ b_2X_{i,2}+ \ldots + b_{p-1}X_{i,p-1} + e_i$

Also note some of features do not have a linear relationship with the target variable. For example, when temperature is too high or too low, people would not prefer riding a bike. A linear regression does not look like a perfect fit here.

Three additional algorithms are adopted, including Random Forest, Gradient Boosting, and Adaboost.

2.3.2 Random Forest

Random forest is an ensemble learning algorithm. The algorithm works as follows: we draw a bootstrap sample from the original data. Each sample is grown into a regression tree. At each node, rather than examining all possible feature splits among all predictors, we randomly select some subset of the features and choose the best split from the subset. We then predict new data by aggregating the predictions of all regression trees.[9] By narrowing the set of features, the learning of the trees are speeded up. In the same time, by restricting each split to a small, random sample of features, the correlation between trees in the forest can be significantly decreased. This helps to reduce the variance of the prediction without increasing the bias. The random forest algorithm is unexcelled in accuracy among many machine learning algorithms, with substantial computational efficiency.

[8] https://en.wikipedia.org/wiki/Linear_regression
[9] http://cogns.northwestern.edu/cbmg/LiawAndWiener2002.pdf

2.3.3 Gradient Boosting

Gradient boosting is a flexible, non-parametric machine learning technique for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees. Gradient boosting fits an additive model in a forward stage-wise manner. In each stage, it introduces a weak learner to compensate the shortcomings of existing weak learners, which are identified by gradients.[10] In each stage a regression tree is fit on the negative gradient of the given loss function.[11] It allows for the optimization of arbitrary differentiable loss function.

2.3.4 Adaboost

Adaboost (Adaptive Boosting) is an ensemble learning algorithm that uses the boosting principle. Adaboost is first applied to improve the performance of different learning algorithms used for classification problems, and then extended for regression problems. Unlike bagging which is a parallel ensemble method, the boosting approach uses the base models in sequential collaboration, where each new model focuses more on the examples with the higher estimator errors, by increasing the weights of the poorly predicted examples and decreasing the weights of the well predicted ones. It aims to convert a set of weak learners into a strong one.[12] AdaBoost approach is sensitive to noisy data and outliers.

Both Adaboost and Gradient Boosting follow the same idea: to boost the performance of a simple base-learner by iteratively focusing on problematic examples that are difficult to predict. In Gradient Boosting, weak learners are identified by gradients, while in Adaboost, they are identified by high-weight data points.

2.5 Stacking

Stacking (meta ensembling) is a model ensembling technique used to combine information from multiple predictive models to generate a new model, which often outperforms each of individual models due its smoothing nature and ability to highlight each base model where it performs best and discredit each based model where it performs poorly.[13] In stacking, a meta-model is trained using individual predictions in different models as the new input data. The trained meta-model is used to form the final predictions on the test set. For this project, a stacking model is trained using Random Forest and Gradient Boosting as sub models and linear regression as the final meta-model.

3. Methodology

3.1 Data Preprocessing

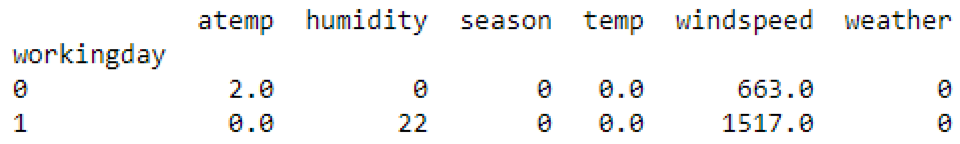
3.1.1 Weather

The initial data exploration finds that there are zeros in the weather related variables, especially wind speed. A zero in wind speed may be explained as: (1) wind speed is indeed zero at that hour; (2) wind speed is too low to be measured, therefore leaving as 0; (3) zero means not available.

[10] http://www.ccs.neu.edu/home/vip/teach/MLcourse/4_boosting/slides/gradient_boosting.pdf
[11] http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingRegressor.html
[12] http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostRegressor.html
[13] http://blog.kaggle.com/2016/12/27/a-kagglers-guide-to-model-stacking-in-practice/

However, wind speed may not be a major factor in explaining the bike share use. Unless there is a windstorm, people usually cannot tell the difference between a wind speed of 5 miles per hour and 10 miles per hour. We will not make any adjustment on wind speed values and exclude this variable in our analysis.

Figure 6. Count of zero values in weather related variables by working day type.

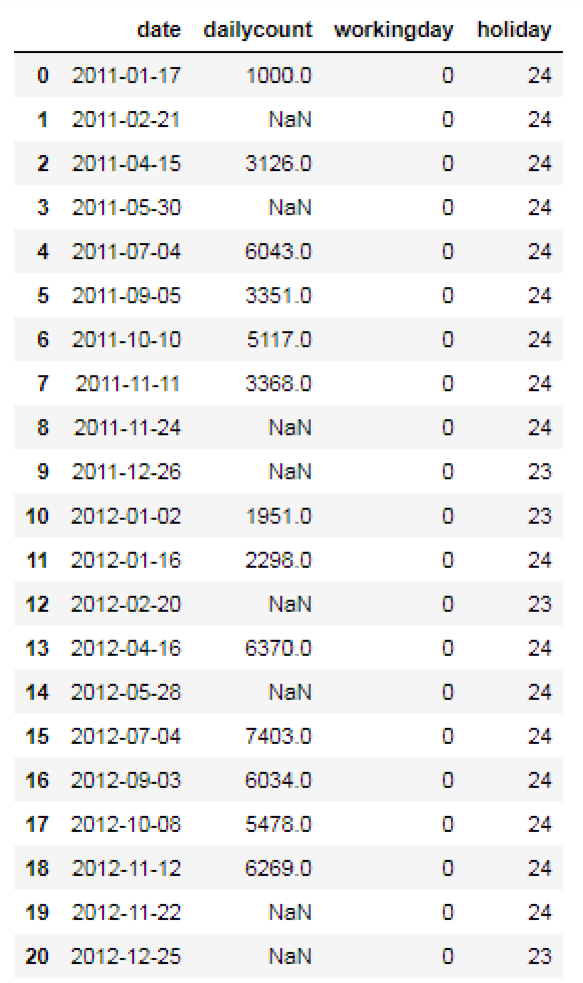


```
            atemp  humidity  season  temp  windspeed  weather
workingday
0             2.0         0       0   0.0      663.0        0
1             0.0        22       0   0.0     1517.0        0
```

3.1.2 Holiday

To understand the definition of holiday in the raw dataset, all days coded as holiday are listed below (Figure 7). In addition to Federal holidays, Tax Day is also coded as a holiday. Bikeshare use on Tax Day however is more close to a regular working day's level, after comparing to all April data. Therefore, two Tax Days (2011-04-15 and 2012-04-16) are re-coded as a working day (working day=1), not a holiday (holiday=0).

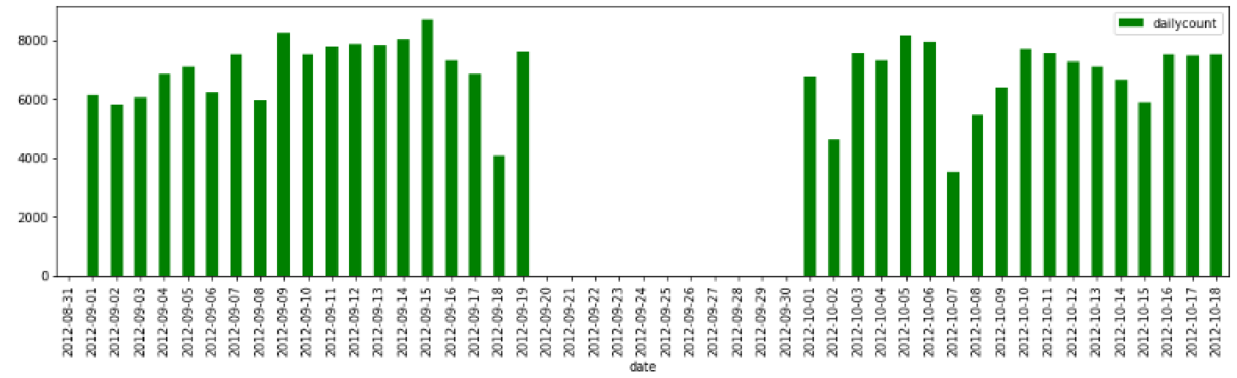
Figure 7. Total bikeshare demand (total counts) in holidays



	date	dailycount	workingday	holiday
0	2011-01-17	1000.0	0	24
1	2011-02-21	NaN	0	24
2	2011-04-15	3126.0	0	24
3	2011-05-30	NaN	0	24
4	2011-07-04	6043.0	0	24
5	2011-09-05	3351.0	0	24
6	2011-10-10	5117.0	0	24
7	2011-11-11	3368.0	0	24
8	2011-11-24	NaN	0	24
9	2011-12-26	NaN	0	23
10	2012-01-02	1951.0	0	23
11	2012-01-16	2298.0	0	24
12	2012-02-20	NaN	0	23
13	2012-04-16	6370.0	0	24
14	2012-05-28	NaN	0	24
15	2012-07-04	7403.0	0	24
16	2012-09-03	6034.0	0	24
17	2012-10-08	5478.0	0	24
18	2012-11-12	6269.0	0	24
19	2012-11-22	NaN	0	24
20	2012-12-25	NaN	0	23

Next, a further examination of daily demand data tells that days around holidays may also show a significant impact. For example, daily bikeshare demand data in September and October 2012 was extracted and visualized in Figure 8. While demand on Labor Day (9/3/2012) and Columbus Day (10/8/2012) are lower than a typical Monday, demand on Tuesday immediately after the holiday also remains lower than a typical Tuesday.

Figure 8. Daily total bikeshare demand (all users) in September/October 2012.



This pre- and post-holiday impact may be more significant for holidays like Thanksgiving, Christmas and New Year's Day. For example, many people will choose to take an extra day off on Black Friday; when Christmas falls on Tuesday, many people will choose to take Monday off to get an at least 4-day long weekend.

Notice that calendar anomalies are not only observed on holiday day, but also on days before/after each holiday, a new feature – holiday_impact (0/1 dummy) is added to model the holiday impact. Variable of holiday_impact is coded as 1 for all Federal holidays, as well as their adjacent weekdays/weekends. Specifically,

- For a holiday occurring on a Monday, the Tuesday immediately after it would be considered as a holiday_impact=1. The Saturday and Sunday immediately before it would also be considered as a holiday_impact=1.
- For a holiday occurring on a Tuesday, the Monday and weekend before, and the Wednesday after are defined as a holiday_impact=1.
- For a holiday occurring on a Wednesday, the Tuesday before and Thursday after are defined as a holiday_impact=1.
- For a holiday occurring on a Thursday, the Wednesday before, and the Friday and weekend after are defined as a holiday_impact=1.
- For a holiday occurring on a Friday, the Thursday before and weekend after are defined as a holiday_impact=1.

3.1.3 New features from the variable of datetime

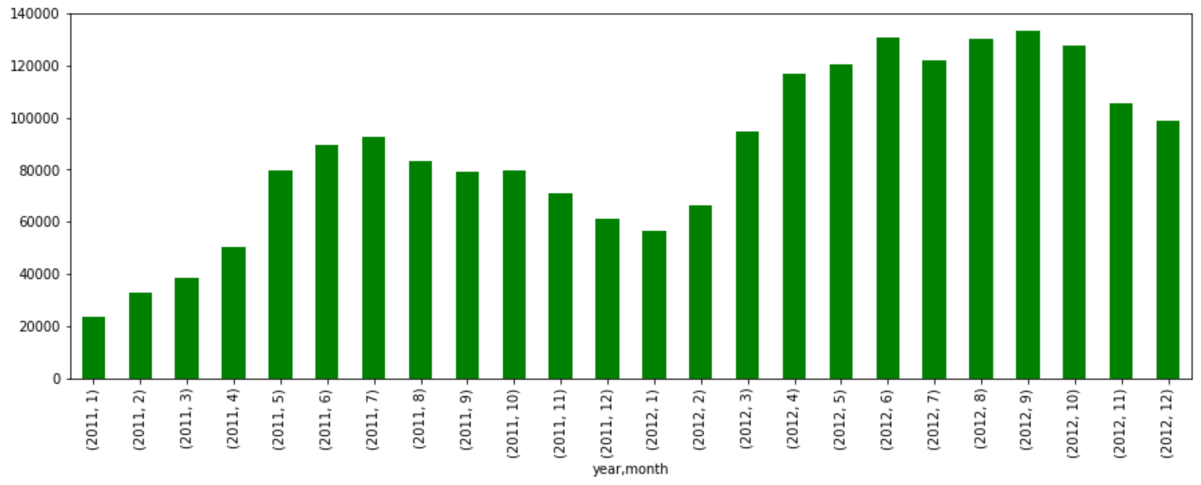
To investigate different types of seasonality, the following new features are created using Python's datetime module.

- **Date**: It is expected to see that daily bike share demand is very different, depending on major events around the city (festival, parade, conferences), weather (especially there are major weather events such as winter storm, flood, windstorm, etc.), competitive transport modes' service condition (major meltdown of Metrorail system, shutdown of a bus line, major incidents on highway).
- **Hour of day**: Traffic demand by hour in a day obviously is very different. There are typically two peak periods (AM and PM rush hours) in a day with a high traffic demand. Even AM and PM peak periods would show a different demand pattern. In the morning, most people usually have one single destination in mind: getting to work or school. In the evening, people leave work at different times, and run errands here and there (restaurant, grocery store, entertainment, etc.) that create more stop-making during the evening. These additional stops create many shorter than usual trips, which sound a good candidate for bike share trip.
- **Day in a month**: Day will work similar as Date.
- **Month**: Bike share demand, like all other travel demand, has a clear monthly/seasonal fluctuation.
- **Year**: Annual demand seems to grow in these two years.
- **Day of week**: It further helps to specify the difference between weekday and weekend. It is also widely accepted that Monday/Friday would show a different pattern compared to Tue.-Thu. The variation in daily demand can be explained by weekly cycle.
- **Week of year**: Week of year will work similar as Month/Season.

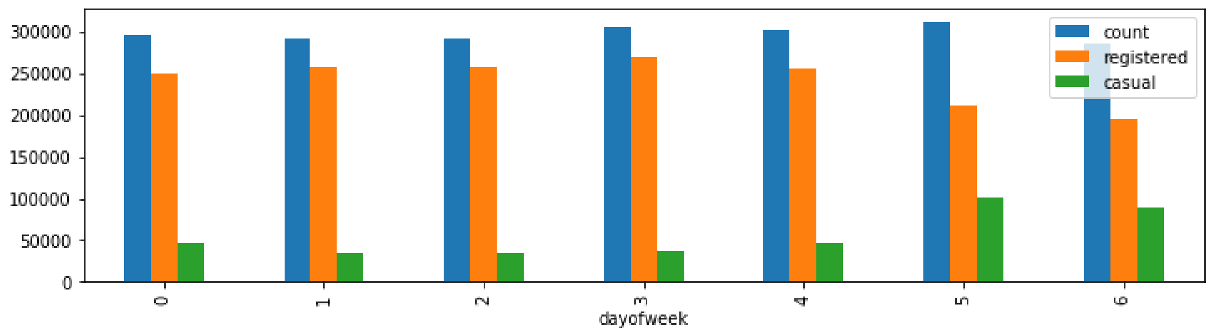
As shown below, the hourly bikeshare demand data exhibits multiple seasonal cycles of different time lengths – monthly, weekly and hourly cycles. Compared to the variable of season, these newly created features provide more flexibility to capture the seasonality effects from different perspectives.

Figure 9. Hourly bikeshare demand data exhibits multiple seasonal cycles of different time lengths.

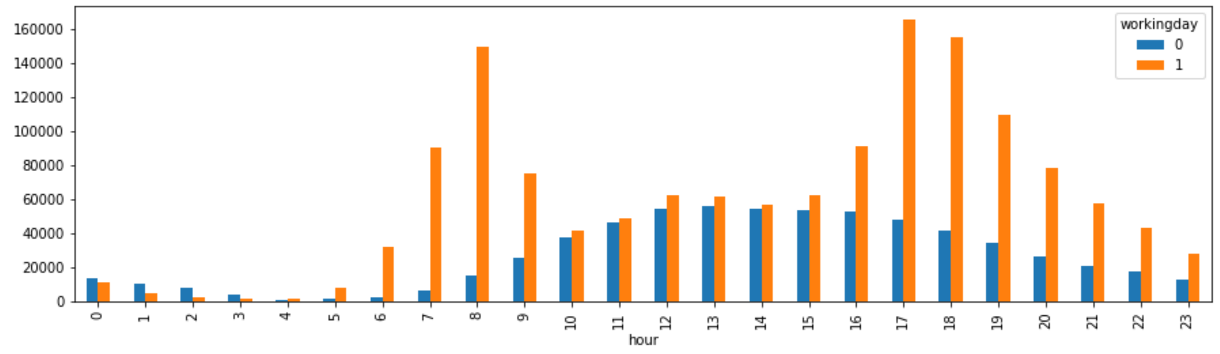
(1) Monthly cycle (total bikeshare demand, all users)



(2) Weekly cycle (bikeshare demand, all users, registered users and casual users)



(3) Hourly cycle (working days vs. non-working days)



As shown in Figure 9(3), there is a clear peak and off-peak distinction during working days. The graph shows the traditional morning and evening peaks. During non-working days, bikeshare demand shows a widespread mid-day peak period. The peak periods are often of most interest, therefore a new feature -"peak" dummy is created to indicate if an hour falls into the peak periods. Peak period is defined as: (1) in working days: 7-9 AM in the morning, 5 and 7 PM in the afternoon; (2) in non-working days: 11 AM – 5 PM.

3.1.4 New feature: Federal government close/or early dismissal/late opening

As identified earlier, registered users dominate the bikeshare demand during the working days. In the same time, daily commuting is one of the primary reasons that many registered users use Capital's Bikeshare. Another fact should not be overlooked is that the federal government is the largest single employer in the DC area.[14] The federal government accounts for more than 1 out of every 10 jobs in the DC area. Most school districts and local businesses also follow Federal government's operating status when major weather events (snow storm, wind storm, hurricane, earthquake, etc.) happen. Federal government's operating status may significantly affect the bikeshare demand.

The United States Office of Personnel Management (OPM) records all the operating status changes in the past.[15] In 2011 and 2012, Federal government offices in the DC area were closed, opened late, or departureed early on 7 days.

- **1/18/2011**: Federal government delayed arrival due to winter storm
- **1/26/2011**: Federal government early departure due to the Carmageddon Snowstorm

[14] https://wtop.com/business-finance/2018/02/exactly-many-washingtonians-work-federal-government/
[15] https://wtop.com/weather-news/2015/01/history-disastrous-snow-storms-d-c-area/

- **1/27/2011**: Federal government delayed arrival due to the Carmageddon Snowstorm
  "A snowstorm raced toward Washington from southwest to northeast on Wednesday, January 26, 2011. Heavy thundersleet and thundersnow overtook the region during the mid-afternoon. Federal workers were released two hours early, causing an early uptick in traffic volumes, coincident with the heaviest snowfall. Commuters reported sitting in traffic for over six hours. Some drivers were stranded on the George Washington Parkway into the early morning hours." [16]
- **8/23/2011**: Federal government early dismissal due to the M5.8 Earthquake
  "The White House, the Capitol, and various other buildings were evacuated. The afternoon traffic rush hour was affected, with some traffic lights inoperative, and the Washington Metro system's trains operated at reduced speeds while tracks and tunnels were inspected. District of Columbia Public Schools were shut down the day after while inspections of the schools were conducted."[17]
- **1/23/2012**: Federal government delayed arrival due to winter storm
- **10/29/2012**: Federal government closed due to hurricane Sandy
- **10/30/2012**: Federal government closed due to hurricane Sandy
  "The effects of Hurricane Sandy in Maryland and Washington, D.C. included tropical storm force sustained winds with isolated gusts to hurricane force, plus significant mountain snow and coastal flooding. On October 26, Washington, D.C. Mayor Vincent Gray declared a state of emergency.That same day the governors of Pennsylvania, Maryland, and Virginia also declared states of emergency in preparation of the approaching storm. The United States Office of Personnel Management announced federal offices in the Washington, D.C. area would be closed to the public on October 29–30. In addition, Washington, D.C., Metro service, both rail and bus, was canceled on October 29 due to expected high winds, the likelihood of widespread power outages, and the closing of the federal government. President Obama declared a state of emergency for the District of Columbia. The Smithsonian Institution closed for the day of October 29."[18]

A new feature – Federal government close (0/1 dummy) is created to capture this impact. If a day falls into any of dates listed above, it will be coded as federal_gov_close=1.

In the same time, the variable of working day is also adjusted for these 7 days. These 7 days are coded as non-working days now.

3.1.5 New feature: Cherry Blossom Festival

For two weeks in late March and early April, millions of visitors come to metro DC to take in the blossoms as they decorate some of our most famous national landmarks.[19] The enduring attraction of the cherry trees and the National Cherry Blossom Festival make the bloom period a great time to get around by bike. All types of traffic spike including biking.

[16] https://wtop.com/weather-news/2015/01/history-disastrous-snow-storms-d-c-area/
[17] https://en.wikipedia.org/wiki/2011_Virginia_earthquake
[18] https://en.wikipedia.org/wiki/Effects_of_Hurricane_Sandy_in_Maryland_and_Washington,_D.C.
[19] https://www.capitalbikeshare.com/attractions/cherry-blossoms

To capture Cherry Blossom Festival's impact, a new feature – cherry (0/1) dummy is created. If a day falls into the Cherry Blossom Festival period, it is coded as cherry=1. National Cherry Blossom Festival 2011 starts from 3/26/2011 to 4/10/2011. In 2012, the festival starts from 3/20/2012 to 4/27/2012.

3.1.6 Encode category variables

Two variables weather and season are converted from numeric to string categorical, by creating a mapping dictionary that contains each variable to process as well as a dictionary of the values to translate.

3.1.7 One-hot encoding to convert categorical variables to dummies

One-hot encoding is used to transform various categorical features to binary (0/1) features. Dummy variables for four seasons, four weather conditions, and twelve months are created.

3.2 Implementation and Refinement

All steps described above are implemented in Jupyter notebook on Python 3. After all data is in good shape, the focus is on the actual model building. Data is split into train and test model. Count_log is the target variable, while feels like temperature, humidity, whether it is a working day, day of week, week of year, hour of day, holiday impact, whether federal government is closed, peak period, day in a month, four seasons dummies and four weather types dummies are included as independent variables.

The data is shuffled into a random order when creating the training and testing subsets to remove any bias in the ordering of the dataset (train_test_split in sklean.cross_validation). Training/Testing subsets splitting and cross-validation are tools to reduce or prevent overfitting. Overfitting indicates the model fails to find good properties of training set that will generalize well to testing set. The training set is used to train or build a model. Once a model is built on training data, we need to find out the accuracy of the model on unseen data. For this, we use the testing set to provide an unbiased evaluation of the model fit on the training set. Splitting a dataset into some ratio of training and testing subsets (test size=0.25 here) for a learning algorithm helps to ensure a better predictive model assessment and possibly model refinement.

As discussed earlier, we select Root Mean Squared Logarithmic Error (RMSLE) as the key evaluation metrics for all models. This metric is not included in sklearn package, so we build a function to define Root Mean Squared Logarithmic Error (RMSLE), using the formula listed in section 1.3. We also report R squared value for each model to evaluate the model performance.

3.2.1 Linear Regression

The linearregression from the sklearn.linear_model module was used to develop the benchmark linear regression model. The linear regression object is stored in a variable called lreg. Fit() function is used to implement the linear regression. In the training set, we learn the relationship between the target variable and all independent variables selected. In doing so, we then use the learned function to estimate the hourly bikeshare demand and apply it to the test set.

Initially, all dummy variables created using one-hot encoding were used to fit the linear regression model. However, this made the model fail due to the perfect multicollinearity. To avoid the dummy variable trap, in general we have to create k-1 dummy variables if we want to analyze the effect of a categorical variable with k classes in a linear regression model. For example, there are four seasons in a year, we drop the dummy variable of winters, as we pick the winter season as the reference.

Again, the benchmark serves as a comparison for the final model. We do not fine tune the linear regression model further, and all parameters are default values. The implementation process therefore remains quite straightforward. The linear regression model returns a RMSLE value of 0.204 for training set and 0.206 for test set. The final model is expected to score a lower RMSLE value.

3.2.2 Random Forest
Second, a basic random forest regression model is developed using RandomForestRegressor from sklearn. Random Forest largely simplifies the data wrangling portion of analytic work, thanks to its robustness to outliers, scale tolerance, ability to select and rank features.

For the initial model, we set max_depth as 5, n_estimators as 100, random_state as 42, n_jobs as -1. N_estimators is the number of trees to build in the forest before taking the averages of predictions. Max_depth is the maximum number of levels in each decision tree. The basic random forest regression model returns a RMSLE value of 0.121 for the training set and 0.128 for test set.

One of the drawbacks of Random Forest is that Random Forest models work like a black box approach with little control on what exactly the model does. To fine tune the model further, different n_estimators are tested. n_estimators represents the number of trees in the forest - the higher the number of trees the better to learn the data. However, adding a lot of trees can slow down the training process considerably, therefore we do a parameter search to find the right spot. The parameter search suggests that we may stop at 100 trees as increasing the number of trees further does not help improve the performance further.

Finally, grid search is conducted to tune all parameters automatically and find the best combination. We set up a range of 5 to 20 for max_depth, a range of 40 to 200 for n_estimator. A 5-fold cross validation is performed. The optimized parameters reported are the max_depth at 19 and n_estimators at 180. We run the random forest model using the optimized parameters. RMSLE reduces to 0.037 for training set and 0.096 for test set, clearly improved from the initial model.

One of the biggest challenges for conducting a grid search is that with a very vague idea of the best hyperparameters, it is difficult to narrow our search range for each hyperparameter and determine the optimal setting with a limit of time and computational power available. If time allows, another method such as random search should be used instead. More hyperparameters should also be tested to find a better optimization setting.

3.2.3 Gradient Boosting

Using GradientBoostingRegressor from sklean, Gradient Boosting model adopts a learning rate of 0.05, max depth of 5, minimum samples leaf of 10, minimum samples of split of 10, n_estimators of 200.

Min_samples_split defines the minimum number of samples which are required in a node to be considered for splitting, which is used to control over-fitting. Min_Samples_leaf defines the minimum samples required in a terminal node or leaf, which is also used to control over-fitting. Max_depth is the maximum depth of a tree. N_estimators is the number sequential trees to be modeled. Learning rate determines the impact of each tree on the final outcome. Gradient Boosting model starts with an initial estimate which is then updated using the output of each tree. The learning rate controls the magnitude of the change in the estimate. Lower values are generally preferred as they make the model robust to the specific characteristic of tree and thus allowing it to generalize well.

With more hyperparameters in the Gradient Boosting model, we face a bigger challenge to get a proper optimization result using Grid Search method. Due to the limit of computational power, grid search is not conducted for the Gradient Boosting model. We report the code in the notebook though.

The Gradient Boosting model returns a RMSLE of 0.079 for the training set and 0.093 for the test set, which is slightly better compared to it is in Random Forest. We also notice the improvement is not significant, however it takes much longer to build and tune the Gradient Boosting model (especially the grid search part). This raises another concern about the tradeoff between model accuracy and model training time.

3.2.4 AdaBoost

From sklearn.tree and sklearn ensemble, we use the DecisionTreeRegressor and AdaBoostRegressor respectively to create the AdaBoost regression model. We use the fit() function to implement the model, then obtain the predicted values for the target variable.

A decision tree model is boosted using AdaBoost, with a learning rate of 0.1, loss of square, n_estimators of 1000. Loss refers to the loss function to be minimized in each split.

Like what we have experienced in the Gradient Boosting model, grid search is not conducted for the AdaBoost model, due to the limitation of computational power and time.

The model returns a RMSLE value of 0.166 for the training set and 0.171 for the test set, worse than the models we tested before. Major disadvantages of AdaBoost model related may include: first, it usually provides a suboptimal solution; second it is sensitive to noisy data and outliers.

3.2.5 Stacking

A meta-model (Linear regression) is created to combine information from two predictive models above - Random Forest and Gradient Boosting, which have the smallest RMSLE. The meta-model returns a RMSLE of 0.027 for the training set and 0.102 for the test set. RMSLE for the test set is slightly higher here compared to Random Forest and Gradient Boosting models.

When creating the new model, a new training dataset is constructed from the predictions of the sub-models. Each row represents one row in the training set. The first column shows predictions for each row in the training set made by the first sub-model – Random Forest; the second columns shows predictions for each row in the training set made by the second sub-model – Gradient Boosting. Similarly, the new test set is also created. A linear regression algorithm then is trained on this new set of dataset. The new model learns how to best combine the predictions from multiple sub-models, taking advantage of two types of ensemble methods.

4. Results

All our models help to establish significant relationships between the target variable (bike share demand) and various independent variables. We focus on 5 different models (linear regression, Random Forest, Gradient Boosting, AdaBoost and Stacking). While the benchmark model – linear regression reports a RMSLE of 0.206 (test set), random forest and gradient boosting models both significantly improve the performance by reducing the RMSLE value further. Random Forest and Gradient Boosting produce similar results with a RMSLE (test set) of 0.096 and 0.093 respectively. Their R squared values are also highest among all models. AdaBoost does not perform well compared to other two models.

Below are the RMSLE error values obtained for each model.

Model	**RMSLE (train)**	**RMSLE (test)**	**R2 (train)**	**R2 (test)**
Linear Regression (baseline)	**0.204**	**0.206**	**0.634**	**0.639**
Random Forest	0.037	0.096	0.989	0.929
Gradient Boost	0.079	0.093	0.946	0.930
Adaboost	0.166	0.172	0.736	0.730
Stacking	0.027	0.102	0.994	0.919

Decision-tree methods appear to be naturally more suitable for our case here. Both Random Forest and Gradient Boosting work better than the baseline model – the linear regression model, because (1) both Random Forest and Gradient Boosting use regression trees as the base regressor, which enable the modeling of more complex non-linear interaction between the features. (2) The linear regression model may suffer the strong correlation between some features, while Random Forest and Gradient Boosting do not. (3) Random Forest and Gradient Boosting methods are not much affected by the presence of outliers.

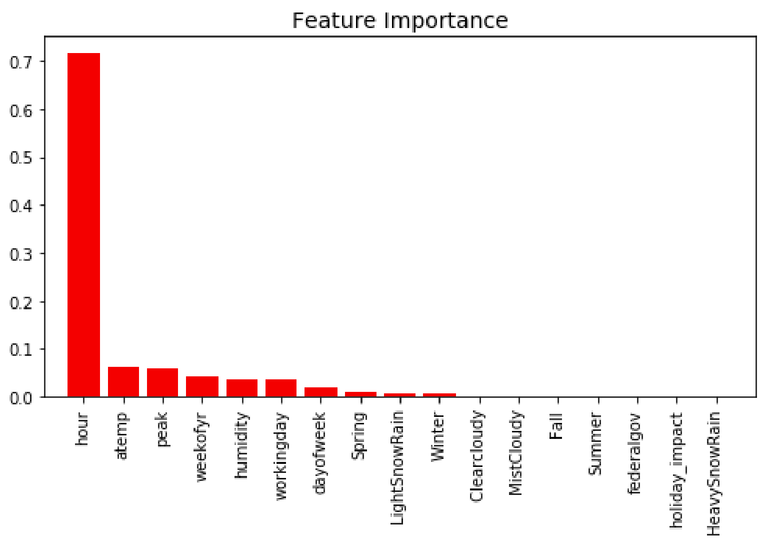
Also, RMSLEs on training set and test set for each model appear to be close to each other. As discussed earlier, dataset is split into training set and testing set. While the training set is used to build the predictive model, the testing set is unseen data when building models. Cross validation and grid search techniques are also adopted to ensure a proper model performance. The values reported here support that our models generalize well to unseen data and the results can be trusted.

Both RMSLE and R squared values above demonstrate the capability of tree regressors in real world applications for similar transportation datasets. The error range seems appropriate for the real life application to use the predictions. While Random Forest and Gradient Boosting models obtain very similar performance, Random Forest is faster and much easier to turn on this dataset.

The final meta-model (stacking) does not work as expected. Stacking may give poorer than expected performance relative to the sub-models. Also, stacking tends to do better with a larger number of input models than with a smaller number of ensemble models. This suggests some future work to build an ensemble of various models, not only tree based models.

Random Forest model further reports the feature importance. The most important features to predict bike share demand is hour of day, feels like temperature, whether it is a peak period, week of the year, humidity, whether it is a working day, and day of the week. Holiday impact is not as important as expected, which is probably due to the fact that holidays do not directly drag down the total bike share demand, but more importantly, they change the market segmentation or the percentage breakdown of casual users and register users.

Figure 10. Feature Importance



The variable of peak is extracted from the variable of hour of day. Peak period is defined as: (1) in working days: 7-9 AM in the morning, 5 and 7 PM in the afternoon; (2) in non-working days: 11 AM – 5 PM. These peak periods properly reflect the traffic density in general and bike share demand levels. Weather condition does play a key role in explaining bike share demand. Among all weather related features, temperature and humidity are two most important features. Week of year and day of week reflect the seasonality impact in bike share demand.

V. Conclusion

This project aims to understand key factors driving the hourly bikeshare demand using historical bikeshare usage data and weather data. To tackle this typical demand forecasting problem, various machine learning algorithms are developed and compared. The final stacking model combines information from multiple predictive models to generate a new model and report the best performance. The final model and solution fit our expectations for the problem, and it can be used in a general setting to solve these types of problems.

The most interesting aspect in this project is how to capture the seasonality impact properly in the data. Seasonality can occur in any time interval. To explore its role in explaining the spikes and dips of bikeshare demand, multiple types of seasonality (hour of day, day of week, week of month, week of year, year) are extracted from the datetime variable in the raw data. Holiday impact is also an interesting feature. Depending on the nature of each holiday, its impact could be positive and negative on bikeshare demand. Depending on which day the holiday falls on, its impact could be different year by year. Though holiday impact turns out to be not as important as expected, this could be an interesting topic for future work and more feature engineering effort that can bring better features to represent the structures inherent in the data.

There are further improvements and future work that could be made in this project.

(1) Experiment with other state of the art methods, such as SVM (this model is very time consuming, so only the code is listed.) and KNN, not only tree based models.
(2) Hyper-parameter optimization other than grid search. Grid search is considered as one of the most widely used way of performing hyper-parameter optimization. It is an exhaustive searching through a manually specified subset of the hypermeter space of a learning algorithm. It needs strong computation power and is extremely time consuming in this case. If time allows, I would try other approaches such as random search and gradient based optimization.