Machine Learning Engineer Nanodegree

Capstone Project

Naoko Shimada

November 6th, 2018

I. Definition

Project Overview

As more people are aware of environmental issues, it is no surprise that they begin seeking a way to live less oil dependent life: supporting renewable energy plant, driving an electric car, or riding a bicycle.
I happen to live in one of the most bike friendly city int the US. With city's bike friendly road development, the number of bicyclists has increased drastically and now a days "sharing" bike (aka rental bike business) has been flourishing. What surprised me is that when seeing more people are riding rental bikes, I noticed that not all renters seemed tourists. Riding an orange rental bike at 7am with casual suite? Seeing this situation made me wonder if there was a different trend for bike demand due to a different interest of renting a bike such as commute vs. sightseeing. If we can forecast when bikes are most needed, or when customers want to ride a bike for long distance, then the rental shops can plan for the demand by stocking bikes in needed location, or stocking different types of bikes (ex, an electric bike for long distance demand.) Having an accurate model for demand forecasting will certainly help bike rental business which intern help our environment as well. Moreover, there is another application for the model. This bike demand model can be applied to another sharing vehicle business like electric scooters which is catching up in my city.

In this project I'm going to build a forecasting model for bicycle demand based on the data from one of Kaggle projects (https://www.kaggle.com/c/bike-sharing-demand/data) .

Problem Statement

Any economy has a demand and supply relationship. And rental bike business is not an exception. Being able to forecast how many bikes will be needed and when these bikes are needed, will help bike rental business.

I will start with identifying feature(s) which have higher correlation with the bike rental count. Using these features, an ensemble algorithm model (Adaboost with Decisiontree regressor) will be trained. This model will be able to provide a solution to the problem: that is, to be able to forecast how many bikes will be needed at each hour so that bike shop can plan to supply the demand. In other words, if we know seasonality information (hour of day, day of week, month) and weather conditions, the model will be able to predict the number of bikes needed for the specified time period . For example, if a bike shop owner needs to plan for the number of bike stock for the next week, say Tuesday, November 20th at 7am, the owner can estimate a number of bikes needed for that time slots by feeding into

seasonality information (7am, workingday, and Fall) and weather forecast (rain or shine, possible temperature) into the model.

Metrics

The metrics used to measure performance of a model is the Root Mean Squared Logarithmic Error(RMSLE).

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

Where y_i is an actual bike rental count and y_hat is a predicted bike rental count for i=1 to total number of sample. This measurement is used at the Kaggle project where I obtain the data.

The Adaboost can be sensitive to outliers. Thus, to penalize large errors due to outliers, a metric which apply relatively higher weight to large errors is needed. For this reason, the absolute value based error functions like Mean Absolutevalue Error (MAE) is not chosen. The Root Mean Squared Error (RMSE) and the Root Mean Squared Log Errors (RMSLE) can be a good choice since both apply higher weights to larger errors and less prone to outliers. However, RMSLE is chosen since the previous participants in the same Kaggle project uses RMSLE and I want to compare my result to their result.

II. Analysis

Data Exploration

The dataset used for this project is from Kaggle (https://www.kaggle.com/c/bike-sharing-demand/data). This data is collected through a network of kiosk locations operated by Capital Bikeshare program in Washington, D.C.. (At the kiosk, people can obtain bike membership, rental, and return.) The data contains the number of bike usage by every hour (24/7) over 2 year period, staring from Jan/2011 until Dec/2012. It also includes weather and temperature information of the time/day at which the bike usage was counted.

One caveat of this data is that the data is already split into two parts: testing and training. The training set contains all features, but is comprised of the first 19 days of each month. The testing set contains all features, except the number of bikes (which we need to predict), and is comprised of the 20th to the end of each month. Therefore, for this project, only the training dataset is used as a whole dataset.

<Sample data>

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	count
0	2011-01-01 00:00:00	1	0	0	1	9.84	14.395	81	0.0	3	13	16
1	2011-01-01 01:00:00	1	0	0	1	9.02	13.635	80	0.0	8	32	40
2	2011-01-01 02:00:00	1	0	0	1	9.02	13.635	80	0.0	5	27	32

<Data Features>

- datetime hourly date + timestamp
- season 1 = spring, 2 = summer, 3 = fall, 4 = winter
- holiday 1=holiday, 0=not (whether the day is considered a holiday)
- workingday 1=Working day(Mon-Friday), 0=Nonworking day
- weather 1: Clear, Few clouds, Partly cloudy, Partly cloudy
  2: Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist
  3: Light Snow, Light Rain + Thunder + 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
- registered number of registered user rentals initiated
- count number of total rentals

< Descriptive statistics for non categorical features>

	temp	atemp	humidity	windspeed	casual	registered	count
count	10886	10886	10886	10886	10886	10886	10886
mean	20.23086	23.655084	61.88646	12.799395	36.021955	155.552177	191.574132
std	7.79159	8.474601	19.245033	8.164537	49.960477	151.039033	181.144454
min	0.82	0.76	0	0	0	0	1
25.00%	13.94	16.665	47	7.0015	4	36	42
50.00%	20.5	24.24	62	12.998	17	118	145
75.00%	26.24	31.06	77	16.9979	49	222	284
max	41	45.455	100	56.9969	367	886	977

The data has no missing values so there is no need for extra preparation. Based on the descriptive statistics, there is no alarming abnormality in the data since the standard deviation of each feature is not too large relative to min/max values. However, based on further analysis, outliers will be eliminated.

Exploratory Visualization

(a) Scatter plot for Bike counts vs. Non-categorical features.

From the scatter plot below(see Figure 1), it is clear that atemp (feel- temperature) is a redundant feature since atemp and temp plots show a almost identical pattern. The plots for humidity and winspeed are spread out uniformly. Thus, these feature may not contribute enough information for predicting bike counts.

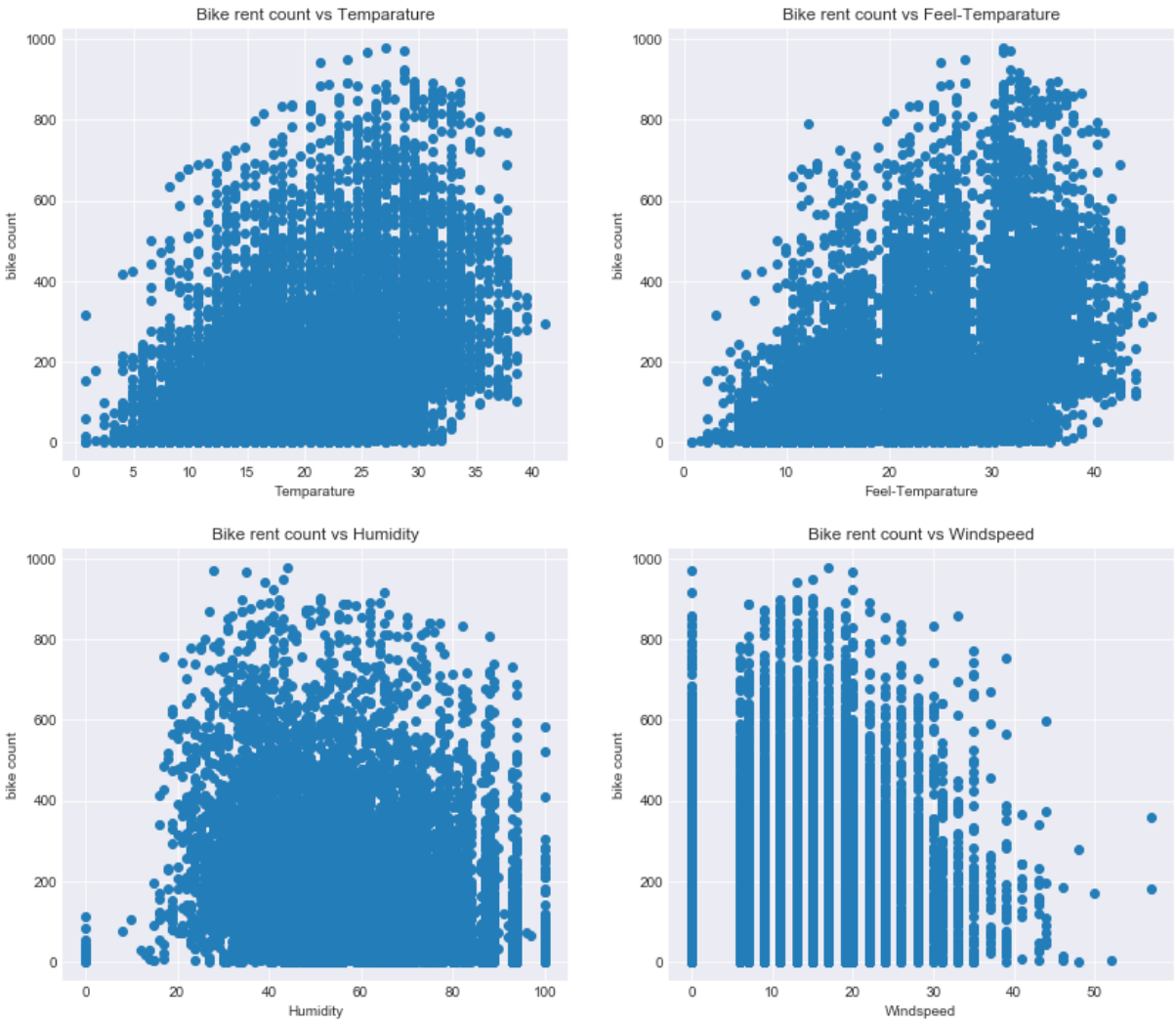

Figure 1

(b)Boxplot for Bike counts vs Categorical features

Scatter plots are not appropriate for categorical data to see its distribution. So for the categorical features, the box-plot is used (See Figure 2).

The boxplot with season indicates that Summer and Fall seasons have more bike rental than Spring, and Spring seems to have more outliers than any other seasons. However, season may need to be broken down into a month level to see if there is more seasonal trend. On the other hand, the boxplot with holiday and workingday don't show much differences. Again, not having any differences in workingday may need to look into more granular level by creating a new feature of Day of Week to see if there is a hidden trend.

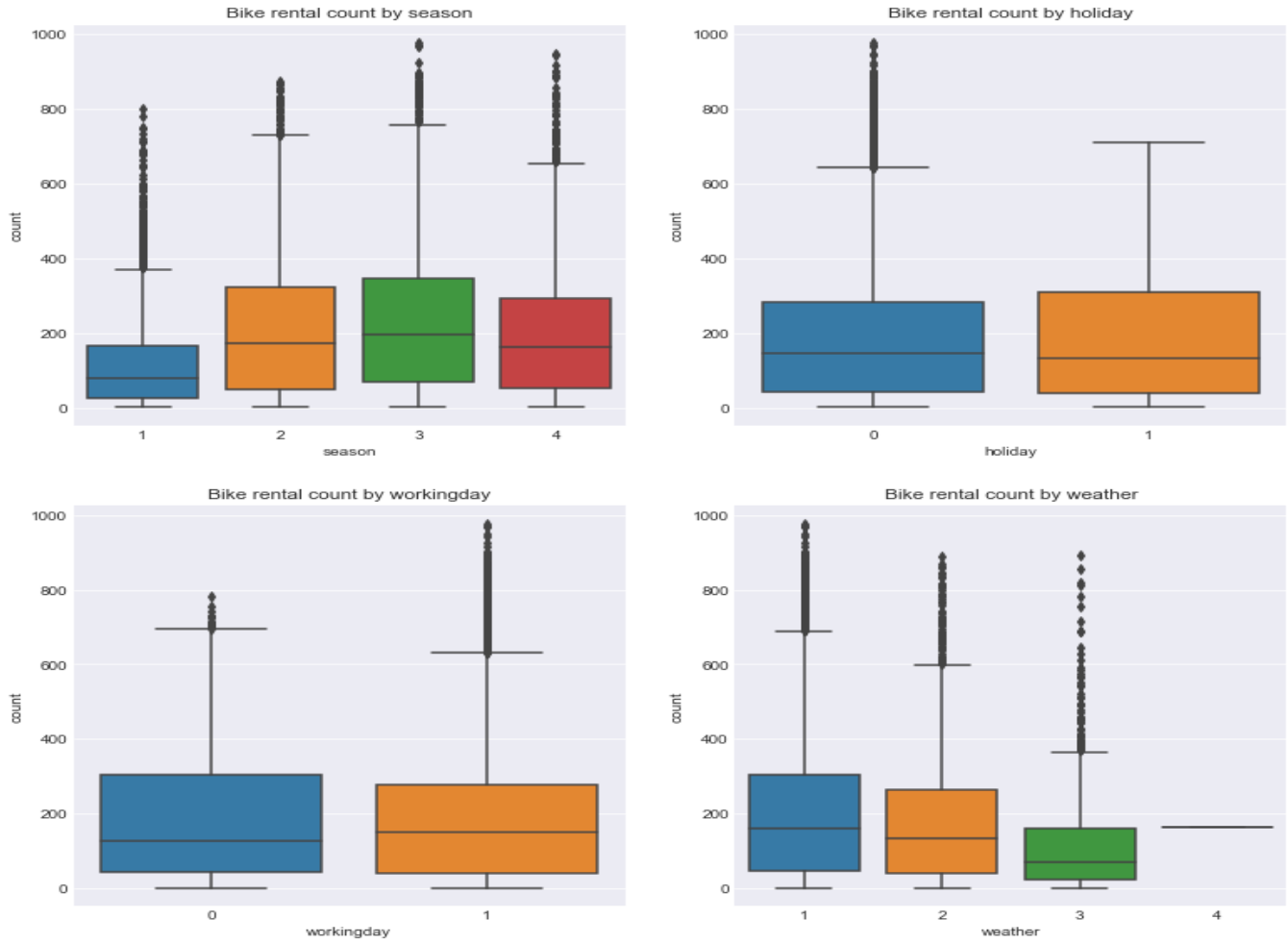

Figure 2

(c) Distribution plot for Bike counts

To see if count (bike rental counts) has a similar distribution to normal distribution or more skewed. The following are distribution plot and Q-Q (quantile) plot of the original feature count and the same feature without outliers. The outliers are defined by outside of 3 standard deviation of count. (See Figure 3)

The output feature is skewed. To make the data more normal distribution, it is possible to apply log-transformation, but the Adaboost algorithm does not require normality of data. So I'll not re-normalize the data. But, the Adaboost is sensitive to outlines so these outliers are removed.

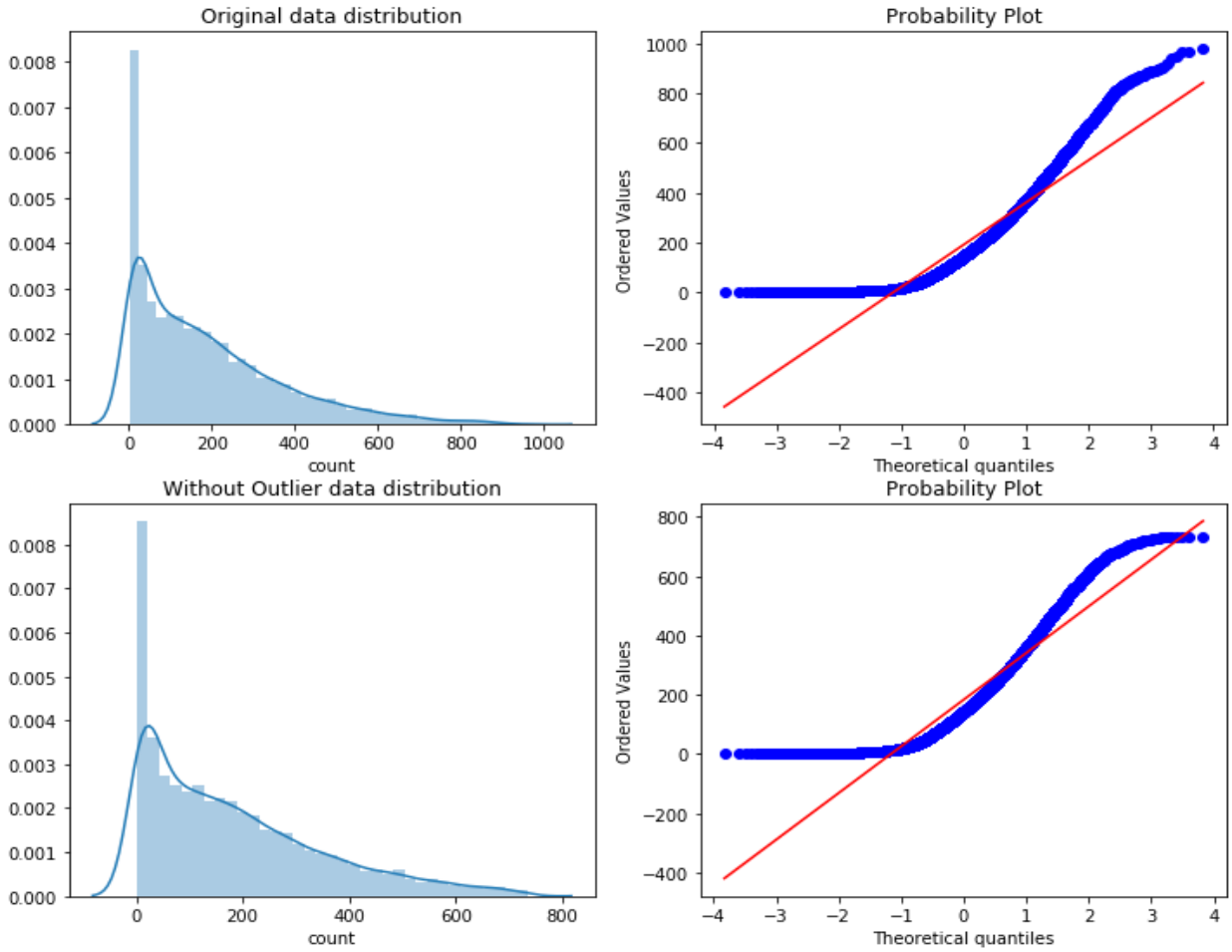

Figure 3

Algorithms and Techniques

The goal of the model is to forecast the number of bikes which will be needed based on given futures. Thus, a model should be a supervised learning model and it will be regression instead of categorical.

The base model is choose to be the Decisiontree regressor since it is suitable to handle dataset that contain both numeric and categorical features. As stated in the data Explore section, our data contains numeric features such as temperature and humidity and categorical features like season (spring, summer, fall, winter) so that this seems a reasonable choice.

To boost this base model, the Adaboost algorithm is chosen since this algorithm best fit with decisiontree as the base estimator. Also, if the data is noizy or has many features, Adaboost is prone to over fitting, but the data for this project is fairly clean and has less than 10 features. This is another reason to choose Adaboost.

The Adaboost is an ensemble algorithm which combines a number of week (not accurate, or slightly better than pure guess) learners to create a strong (more accurate) learner algorithm.
The basic mathematical logic can be summarized as follows:

Let $\mathbf{x_i}$ be an ith element of the training dataset **X,** $\mathbf{y_i}$ in {-1,1} be a desired output corresponding to each $\mathbf{x_i}$, and $\mathbf{h_t(x_i)}$ be week learners at iteration **t** of Adaboost for i=1,....,n and n is the total number of training elements.
For each $\mathbf{x_i}$ in the dataset, we assign weight and the initial weights is set to

$$D_t(i) = \frac{1}{n}$$

for i =1,2, ……., n and t=1,2,….,T (iteration t)

Each time weak learner algorithm $\mathbf{h_t}$ is applied using weighted sample, the weight for each $\mathbf{x_i}$ is updated using miss-classification rate $\boldsymbol{\varepsilon}$ by comparing $\mathbf{h_t(x_i)}$ and $\mathbf{y_i}$, and it generates a new weak prediction rule. Updated weight is defined by

$$D_{t+1}(i) = \frac{D_t(i)\exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

where $\mathbf{D_t(i)}$ is weight at previous level, $\boldsymbol{\alpha_t}$ is weighted error rate of weak learner given by $\boldsymbol{\alpha_t = 0.5*ln((1-\varepsilon)/\varepsilon)}$ where $\boldsymbol{\varepsilon = sum(D_t(i) *| h_t(x_i) - y_i |)}$ for all **i ,** and $\mathbf{Z_t}$ is normalizer given by $\mathbf{Z_t = sum(D_t(i))}$ for all **i .**

We continuing this iteration until the training set is predicted perfectly or a maximum number of models are added. Then at the end of T iteration, combining all the weighted weak learners to construct a highly accurate prediction **H(x)** which is denoted by

$$H(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$$

Benchmark

The benchmark model is chosen from the same bike-sharing demand project in Kaggle (https://www.kaggle.com/casalicchio/tuning-with-mlr). This model has the Root Mean Squared Logarithmic Error of 0.37977 using Xgboost in R.

Also, DecisionTreeregression model(from sklearn.tree package) is fitted with the training data set (which is used to train an actual model as well) as the benchmark. This model has RMSLE of 0.76.

III. Methodology

Data Preprocessing

(a) Creating Additional Features

Based on the Data Exploration, more granular level of time dependent features are created: `hour` (hour of day), `DOW` (day of week: 0=Monday, 6=Sunday) and `month` (1=January, 12=December). The following boxplots compare `season` vs. `month` , and `workingday` vs. `DOW` to see if more granular level of features can capture better trend of bike counts.

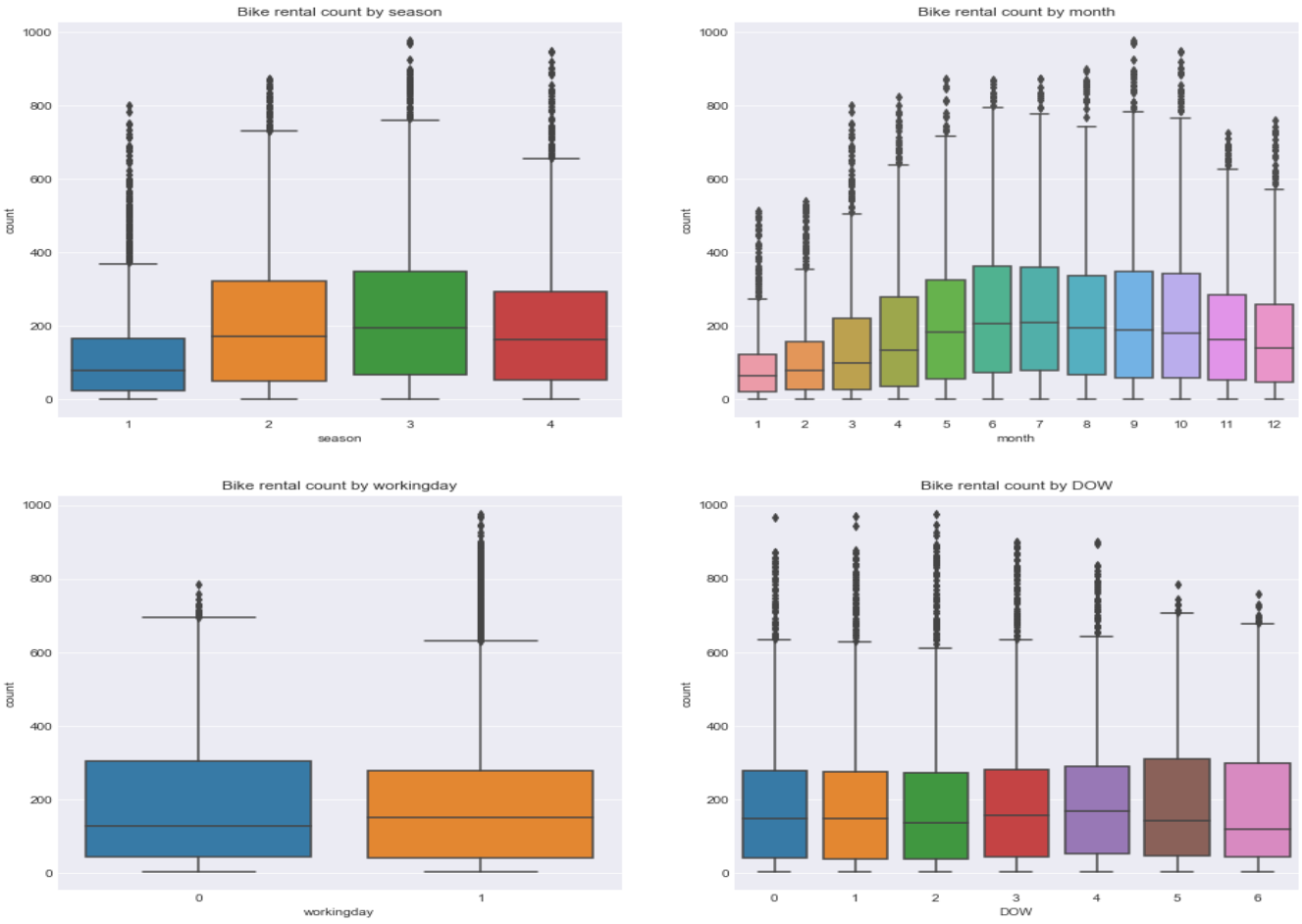

Figure 4

This side by side comparison plots (see Figure 4) indicate that more granular level of `DOW` may not contribute more information than just using `workingday` . However, `month` feature may be better than `season` because the boxplot for `month` shows how each month is different from one another.

Moreover, the hour plot below (see Figure 5) clearly indicates the importance of this feature. There is a distinctive trend during commuter hours (7am to 9am and 4pm to 7pm) where the number of rental bikes increases and there is no outlier associated with these time slots.

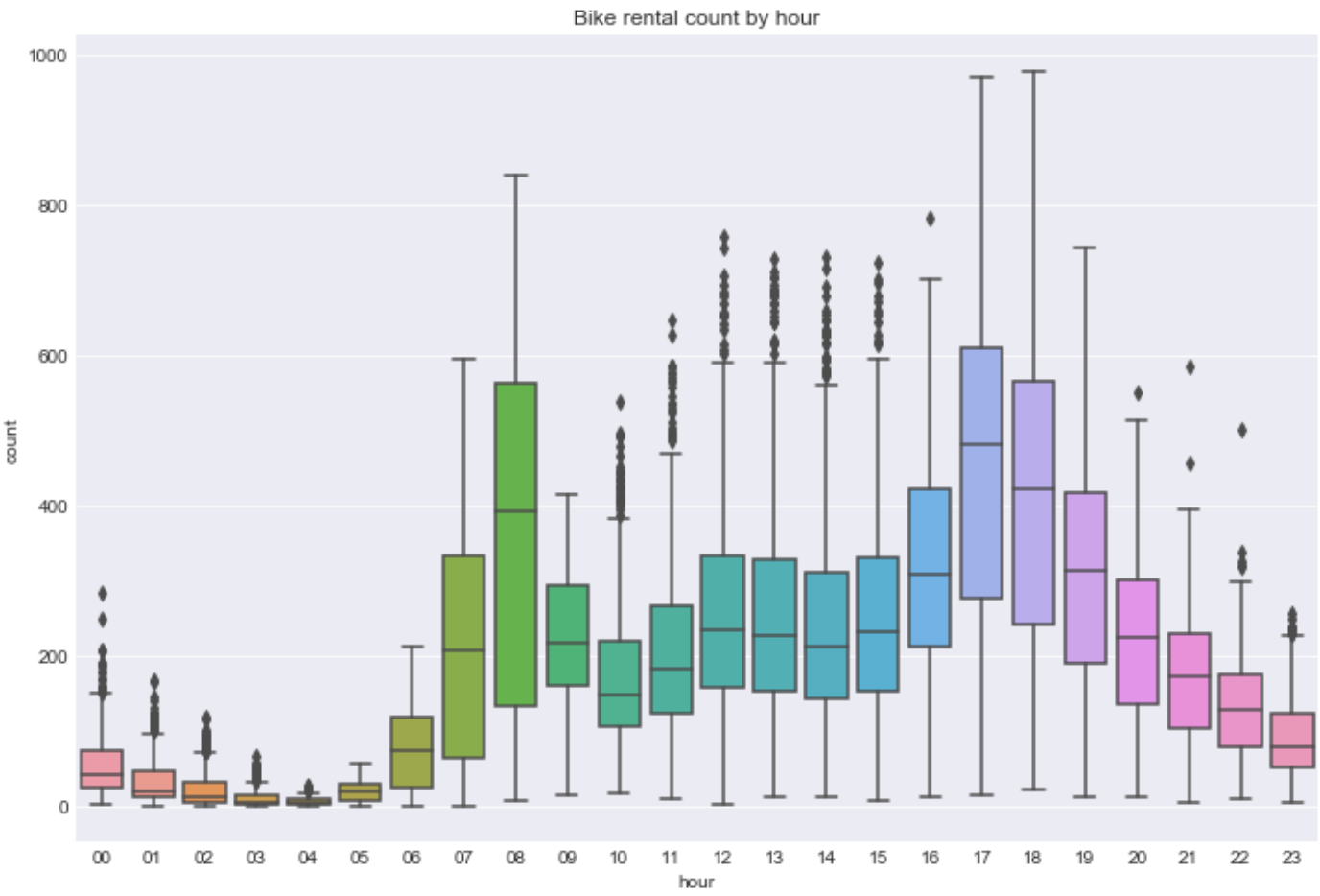

Figure 5

This new feature hour led to produce the following line plots(see Figure 6): the average of bike counts per hour vs. month and DOW. The first line plot shows how month and hour features drive bike count changes. The next line plot shows two distinctive patterns: one is workingday (Monday through Friday) pattern which has the similar trend as the hourly box plot above. The other one is weekends (Saturday and Sunday) pattern, which has a very distinctive pattern. There is no commuter hour peaks, but rather gradual mountain slope. The second plot does confirm that DOW is not necessary.

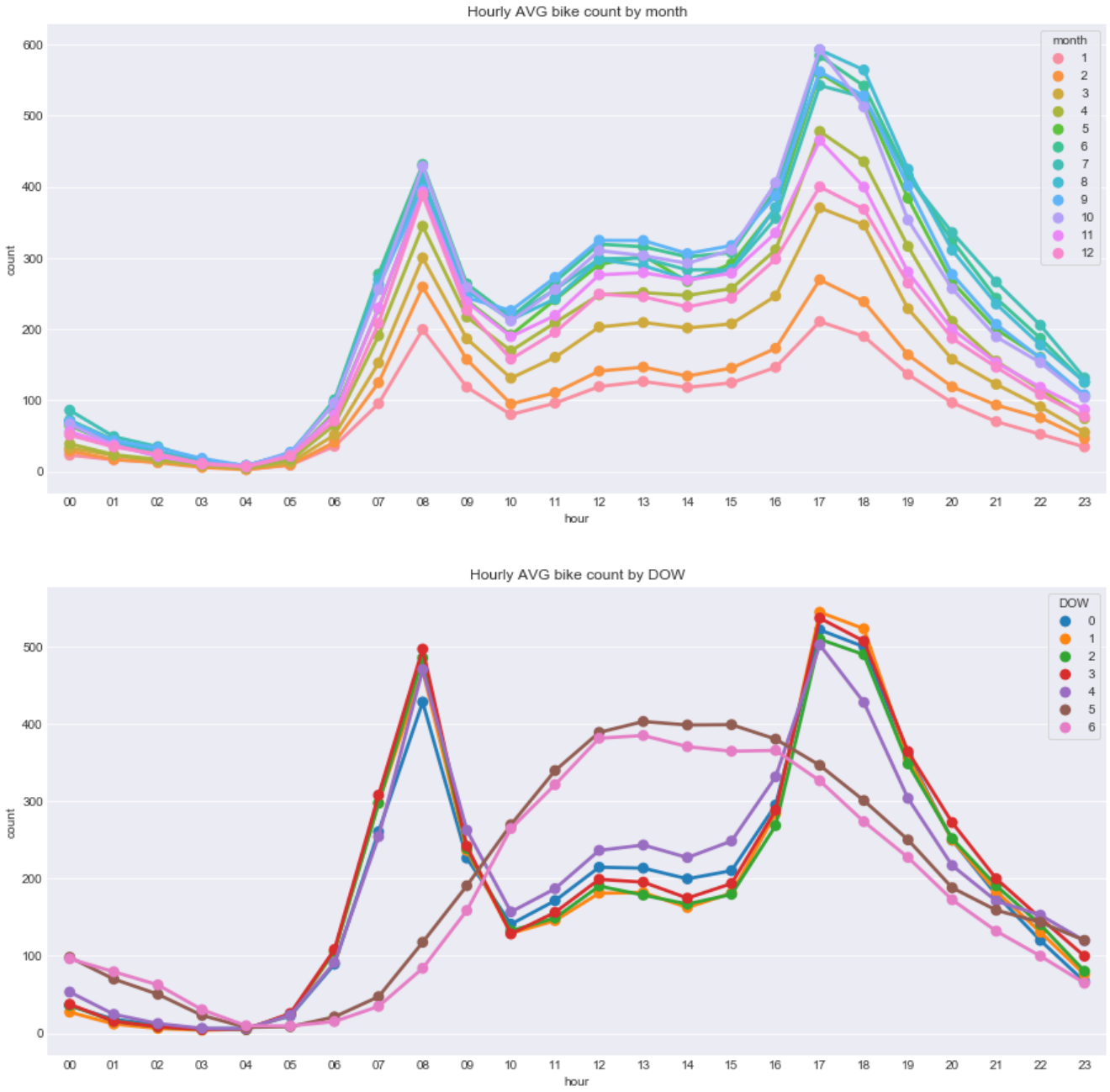

Figure 6

(b) Feature selection

Based on the analysis from the Data Exploration and (a) Creating Additional Feature section, the final features are selected. To choose highly correlated features (such as workingday and DOW) and to eliminate non-correlated features to the outcome feature count , a correlation matrix among each of all features are calculated (see Figure 7) If the absolute value of the correlation score between 2 features is above 0.5, one of them which illustrate bike rental counts better, is selected. Also, if a feature has less than +/- 0.001 correlation to count , the feature is eliminated from modeling process.

The final features are

output (y): count

input (X):

- month (instead of season : they have a correlation of 0.97)
- hour (its importance is clear from the previous section)
- workingday (instead of DOW : they have a correlation of -0.7)
- temp
- weather
- humidity
- windspeed

Eliminated features are

- datetime : datatime is break down into hour/ day/ and month.
- season : instead month feature is created
- DOW : workingday is better
- atemp : redundant to temp
- casual it is a leakage variable since count = casual + registered
- registered it is a leakage variable since count = casual + registered

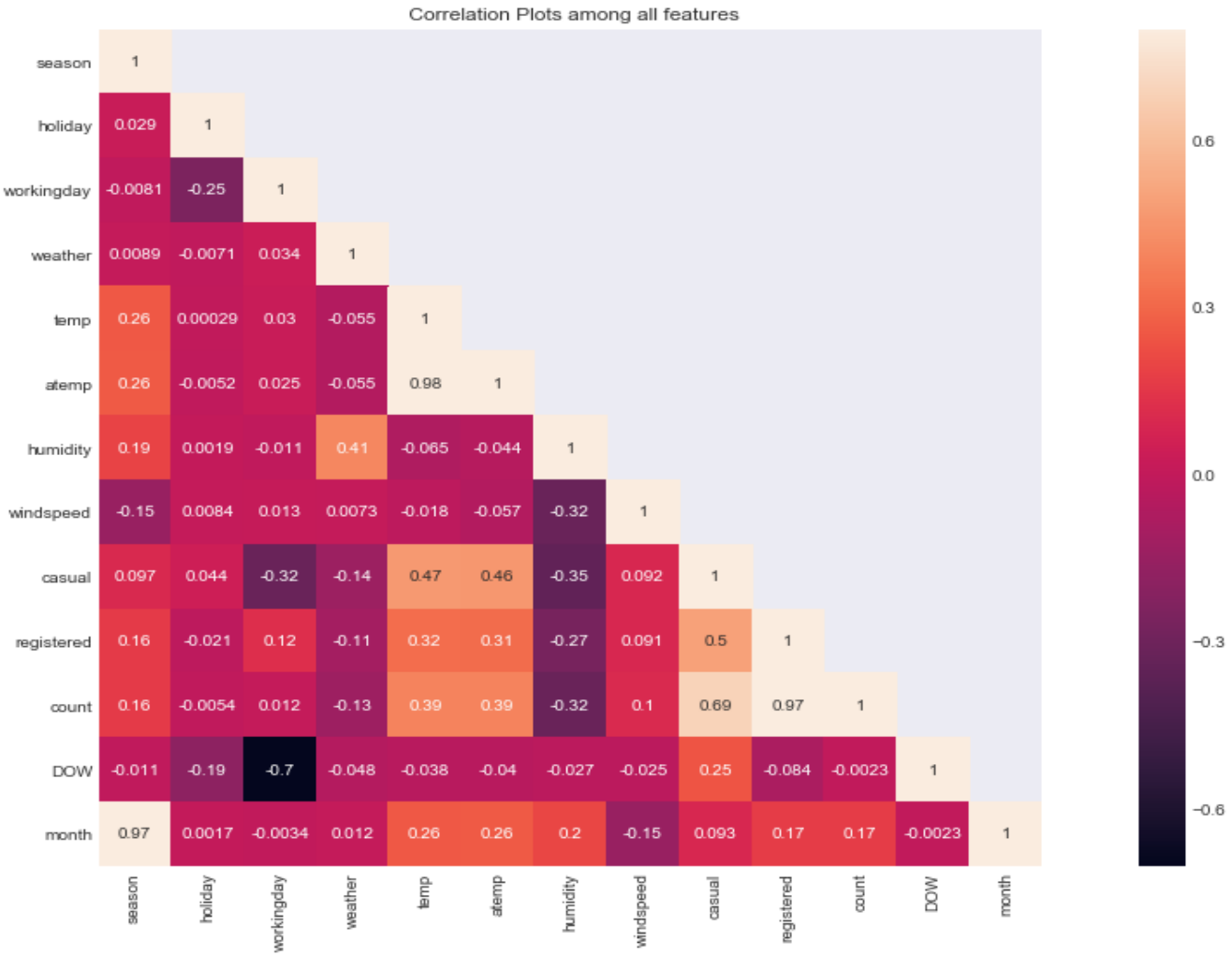

Figure 7

(c) Removing outliers for the output feature

When count value exceeds 3 standard deviation of count, then, those values and corresponding input features are eliminated.

(d) Splinting data

For the model evaluation purpose, the data needs to be split into two sets: training and test dataset. Since the data set is a time- series dataset, a usual technique of random splitting cannot be used. Also, it is important to keep all the seasonality information (each hour of day, day of week and all 12 months) in each dataset. Therefore, the data is split by year: the training set consists of year 2011 and the testing data consists of the year 2012. To ensure the quality of each data, visualization is utilized (see Figure 8)

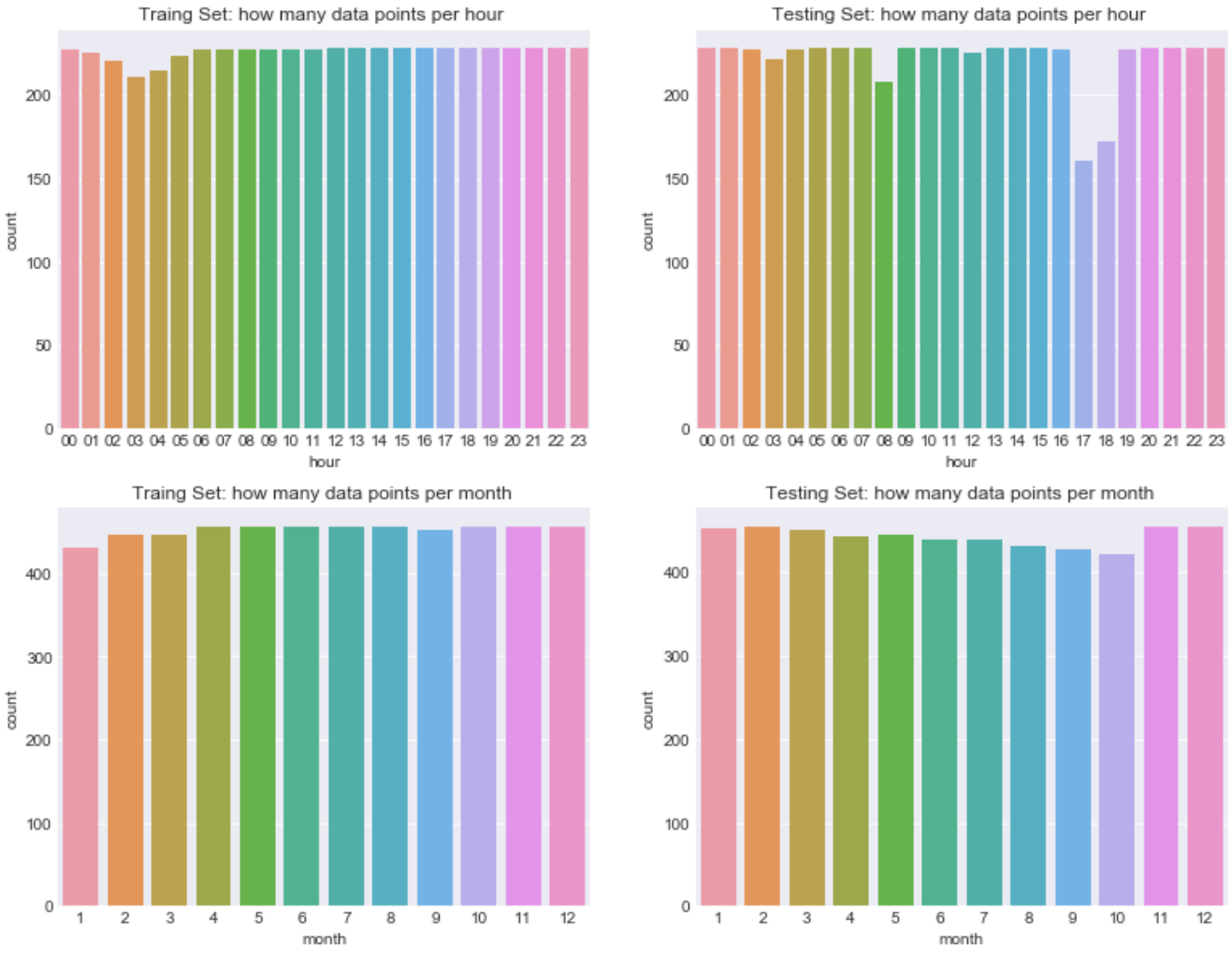

Figure 8

Implementation

All algorithms are obtained from “sklearn” (in Python) packages and code is written in JupyterNotebook with Python3 kernel. The base estimator for the model is a decision tree regressor (*DecisionTreeRegressor from sklearn.tree*) with default parameters: max_depth=3 and the criterion to split is the mean squared error. And the boosting algorithm is Adaboost (*AdaBoostRegressor from sklean.ensamble*). For the booster, the linear loss function is used when updating the weights after each boosting iteration. But, the maximum number of estimator and learning rate will be adjusted to achieve an optimal solution. First, the Adaboosted decision tree regressor algorithm is applied to the training data set to train the model and evaluate an accuracy of the model with RMSLE. Once the model is trained and obtained a satisfactory result, then fit the model with the testing data set, check RMSLE and re-evaluate. One thing to be aware is that when creating training and testing data set, random split is not appropriate for this data set since this is a time-seies dataset. Also, splinting the data with using one of time dependent feature is not acceptable either. This is because the outcome feature count depends not only on hour of day, but also day of week and even each month so that each data set (training and testing) must contain all the time related features. Thus, for this project “year” is used to split the data: the year of 2011 is training set and the year of 2012 is a testing dataset.

Refinement

Two parameters (n_estimator and learning rate) are adjusted to improve the model. Starting with adjusting one parameter: the max number of estimator (n_estimator.). The default value is set to 50 and with this condition, fit a model, and calculate RMSLE as the benchmark. Gradually increase the number of n_estimator, fit the model, and calculate the corresponding RMSLE. Repeat the process 3 times and take the average of RMSLE to be a representation of the different n_estimator. The summary of the result is shown below(TABLE1).

TABLE1: RMSLE for different n_estimator values

	n_estimator	N=50	N=100	N=300	N=1000	N=10000
RMSLE	try1	0.990	0.912	0.882	0.967	1.135
	try2	0.899	0.925	0.975	0.915	1.016
	try3	1.029	0.973	0.962	0.924	0.988
	Average	0.973	0.937	0.939	0.935	1.046

As the number of n_estimator increases, RMSLE becomes smaller, but increasing n_estimator beyond 1000 doesn’t seem to lower the error value. Thus, choose n_estimator=1000 for the optimal value. Next, adjusting the learning rate for the model with n_estimator set to be 1000. The default learning rate is 1.0, and this value is gradually decreased while the corresponding RMSLE is calculated every time the learning rate is changed. Like before, the model is fit 3 times and RMSLE is calculated 3 times, then the average of the 3 trials is used to evaluate the learning rate. The following table (TABLE2) is a summary of the result.

TABLE2: RMSLE for different learning rates

	Learning rate	R=0.5	R=0.1	R=0.01	R=0.001	R=0.0001
RMSLE	try1	0.865	0.833	0.739	0.740	0.739
	try2	0.875	0.830	0.741	0.739	0.739
	try3	0.881	0.830	0.742	0.740	0.739
	Average	0.874	0.831	0.741	0.740	0.739

The change of the learning rate improves the model, but the improvement slowes down between R=0.001 and R=0.0001. Therefore, the final choice of the learning rate is 0.0001 with n-estimator=1000.

IV. Results

Model Evaluation and Validation

The regression model is chosen since this is a regression problem where the output value is a real value (the number of bikes rented) and using a model to learn a mapping from input features to the output feature. To boost the model, Adaboost is choose and its parameters (n_estimator and learning rate) are evaluated on iterative testing which optimize RMSLE (see TABLE1 and TABLE2 in the previous section). So, the final model is the Adaboosted decision tree regressor algorithm with n_estimator=1000 and leaning rate of 0.0001.

To evaluate the robustness of the model, the testing data set is fed into the model and the corresponding RMSLE is calculated three times. The first, second and third RMSLE are 0.76,0.766, and 0.768, respectively. If the model was over-fitted, RMSLE for the testing data set would be much higher than the RMSLE for the trained model. But, the result was very close to the trained model. This ensure that the model is working properly.

Justification

The final model has RMSLE of 0.739 (average of three) and the Benchmrk model has RMSLE of 0.379. Also, my benchmark model of decisionTree has RMSLE of 0.76. Although my final model doesn't achieve the project benchmark RMSE nor significantly better than without the boost, RMSLE values form my model has improved via refinement process. The original RMSLE is 0.94 on average, but the final model produces 0.739 on average. Also, the final model is trained with 50% smaller data set than the benchmark model. (This is due to the original data provided in Kaggle, which discussed in the Data Exploration section.) Thus, it is expected that my model may not achieve the same RMSLE.

V. Conclusion

Free-Form Visualization

The model evaluation is done by using RMSLE, but the following plots show a comparison between actual outcome `count` and predicted outcomes per each hour. As you can see (Figure 9), why the model with n_estimator=300 has the lowest error rate. From 9am to 7pm, all models over estimate the bike count, but n_estimator=300 model become closer to actual count values.

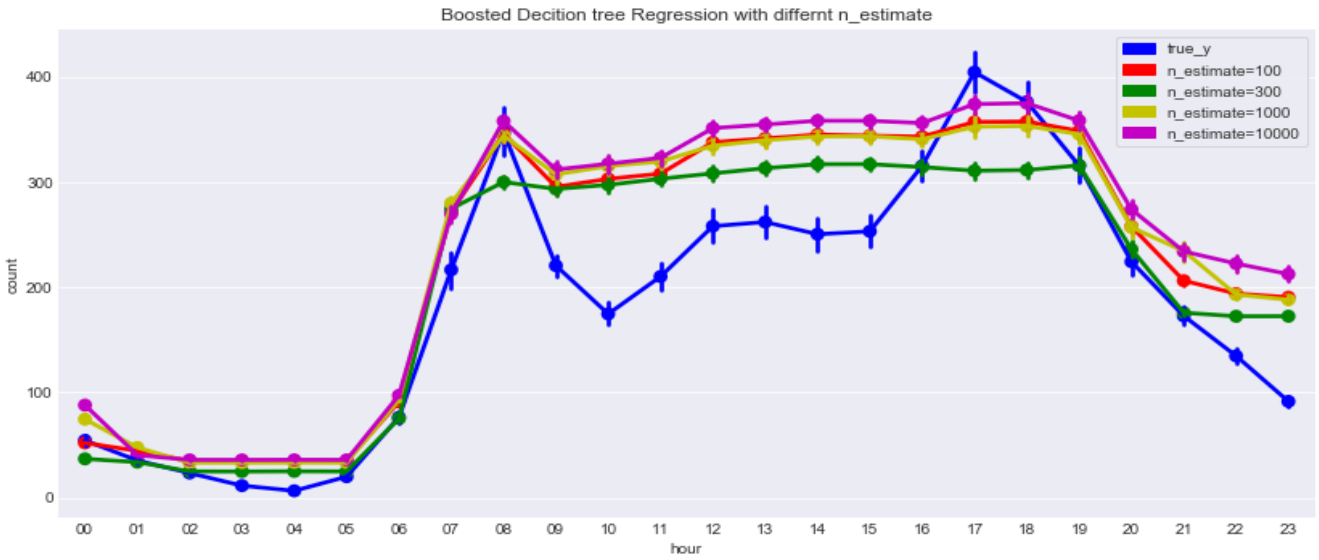

Figure 9

The plot below (Figure 10) is the same outcome comparison plot, but n_estimator is fixed at 300 with a different learning rate. The final model of learning rate=0.0001 predicts the count reasonably well,

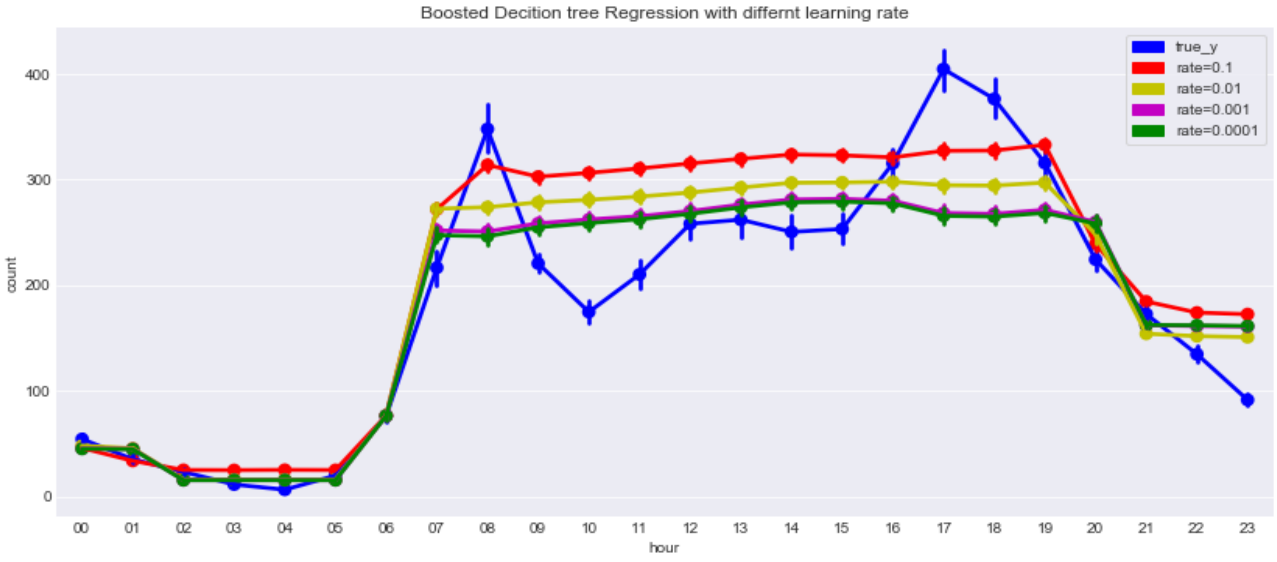

Figure 10

except during 7am-8am and 5pm to 7pm. This result makes me think about how to improve the model. (more discussion is in the improvement section)

Reflection

The process used for this project can be summarized as the following:

(1) Each feature is analyzed using descriptive statistics and visualization.

(2) Based on the initial analysis, additional features are created and evaluated against the existing features to see if the new features are improvement or not.

(3) After evaluating each features via visualization and correlation matrix, the best features are selected to fit a model.

(4) Starting with default parameters, a model is fit with training data set (which is 80% of the main data) and using RMSLE to evaluate the optimal parameters for the model.

(5) Visualization of predicted outcome is utilized to choose the final model.

(6) The testing data is then fed into the final model for evaluating the robustness of the model.

From the data exploration, it is clear that the rental bike counts have a seasonality pattern (time of day, day of week, and month). When there is a clear and predictable pattern, we should be able to build a reasonably accurate model. Unfortunately my final model needs some improvement so that it can predict bike counts during the morning and evening peak time. But, once it is improved, the model should be applicable to other rental bike business in other cities since the pattern is associated with human daily behavior. And basic human behavior is similar across cities.

One challenging part of this project is that bike count in the given data is pre-summzrized by hour, and there is no way to create more granular buckets of time duration. I think granular time duration especially during high peak of bike rental counts, will help the model. I didn't realize this until I started visualizing model outcome and see at which the model was strangling to predict.

Improvement

There are three things I want to try to improve the model.

(1) If I have access to the original data (that is, the bike count is not pre-summarized by hour), I would like to create 30min time interval instead of an hour. This is because the increase of the bike count from 7am to 8am, or 5pm to 6pm is significantly different from any other time period. The number of bike counts double or tripled within an hour, and this huge increase only happened during peak time.

(2) I used the total bike counts, `count` as output. However, in hindsight, it could have been better to use `casual` (the number bike count by non-registered user) and `registered` (the bike count by registered user) as output features and built two models first and then combine the prediction later.

This is because casual user's bike rental pattern is similar to weekends, and the registered user's pattern is same as the workingday pattern. (see Figure 11)

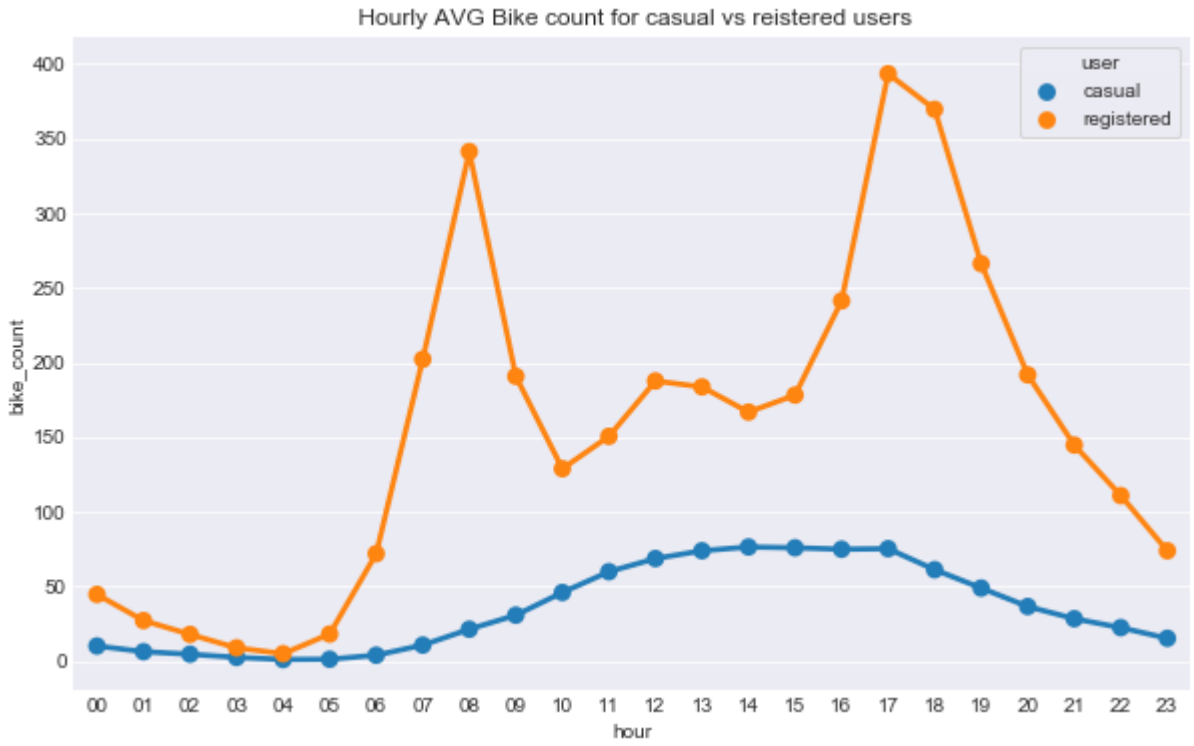

Figure 11

(3) I didn't have enough time to explore a different algorithm, but I would like to apply a different booster algorithm such as xgboost which is used for the benchmark project in Kaggle.

As I stated at "free-form visualization" section, the final model is having difficulty of predicting bike rental count during rental peak time period. So, I think that improvement plan of (1) and (2) will significantly improve the accuracy of the prediction. Also applying xgboost may work better thatn Adaboost.