**Bike-sharing demand prediction**

Project Report



# **PART I: INTRODUCTION AND PURPOSE**

## **Business Understanding**

Bike-sharing is a new share economy business which provides means of renting bicycles effortlessly to users. After paying deposits, on an as-needed basis, people are able to rent a bike from one location and return it to another. Currently, there are over 500 bike-sharing programs around the world. Bike-sharing has provided an alternative other than the traditional public transportations for citizens to get around. On one hand, it gives users almost 24/7 access bike vehicles without needing to own one or bring one around. On the other, it also benefits the municipal environments management in ways such as easing the overall crowdedness on streets and reducing carbon emission.

As the whole bike-sharing ecosystem is newly established, from the point of view of a bike-sharing company operator or potential venture capital investor, it is critical to understand the most crucial operating metrics, bike user count, and the different factors influencing the number of users.

For companies, gaining insights into the bike user count figure and being able to predict them will not only help with predicting the overall revenue performance, but will also help the management to understand the future demand of rental bikes thus enabling themselves to make better corporate decisions such as the amount of bike to be put into a particular region in the future. For researchers, focusing on studies regarding urban mobility, the user number of share-bikes will also contribute greatly to their understanding of citizens’ habits of commuting.

## **Purpose**

Aiming at finding the factors influencing bike user count, we have acquired a dataset from kaggle.com with over 10,000 entries specifying the pattern of users in the “Capital Bikeshare program” in Washington, D.C. We believe that by studying this dataset, we will be able to build a machine learning model to forecast the bike user amount in a specific hour of a day with certain conditions applied.

# **PART II: INTRODUCTION AND PURPOSE**

We found the dataset from a competition - “Bike Sharing Demand” on Kaggle, retrieved from the following link: <https://www.kaggle.com/c/bike-sharing-demand>

## **Data Description & Summary**

There are in total 10,886 data instances, consisting of 12 columns in our dataset. They are *DATETIME*, *SEASON*, *HOLIDAY*, *WORKINGDAY*, *WEATHER*, *TEMP, ATEMP, HUMIDITY, WINDSPEED*, *CASUAL*, *REGISTERED*, and *COUNT*.

Having a first look at our data, we can easily figure out that categorical attributes include *DATETIME*, *SEASON*, *HOLIDAY*, and *WORKINGDAY*, and that numerical attributes include *TEMP, ATEMP, HUMIDITY,* and *WINDSPEED*, and target variables include *CASUAL*, *REGISTERED*, and *COUNT*. Furthermore, the value of *COUNT* is equal to the sum of the value of *CASUAL* and *REGISTERED*. Therefore, we can treat *CASUAL* and *REGISTERED* as our *intermediate target variables*, and *COUNT* as our final target variable.

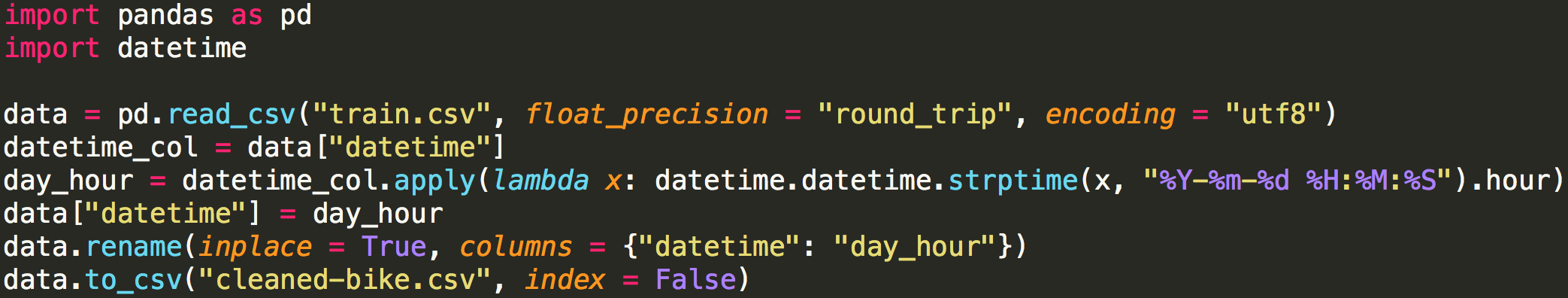
Using this idea, we can build two separate models for *CASUAL* and *REGISTERED*, using all the 9 attributes. Finally, we can sum up out the predicted value of *CASUAL* and *REGISTERED* to get the prediction of *COUNT*.

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Category** | **Description** |
| *DAY\_HOUR* | attribute (categorical) | an hourly date, in the format of a timestamp |
| SEASON | attribute (categorical) | 1 = spring, 2 = summer, 3 = fall, 4 = winter |
| *HOLIDAY* | attribute (categorical) | 1 = is holiday, 0 = is not holiday |
| *WORKINGDAY* | attribute (categorical) | 1 = is working day, 0 = is not working day |
| *WEATHER* | attribute (categorical) | 1 = clear + cloudy, 2 = mist + cloudy,  3 = light snow / rain, 4 = heavy snow / rain |
| *TEMP* | attribute (continuous numerical) | the temperature in Celsius |
| *ATEMP* | attribute (continuous numerical) | the sensible temperature in Celsius |
| *HUMIDITY* | attribute (continuous numerical) | relative humidity (%) |
| *WINDSPEED* | attribute (continuous numerical) | wind speed |
| *CASUAL* | intermediate target (numerical) | number of non-registered user rentals |
| *REGISTERED* | intermediate target (numerical) | number of registered user rentals |
| *COUNT* | target (numerical) | number of total rentals |

## **Data Analysis (Information Extraction)**

Before doing data processing, we need to analyze more about our dataset, considering the aspects of missing values, skewness, and outliers.

Our goal is to predict the total amount of the bike user in a specific hour of a day with certain conditions applied. Therefore, the only useful information in *DATETIME* variable is the hour information. It is very hard to extract the hour information in Azure, (format: “YYYY-MM-DD hh:mm:ss”)so we wrote a python program to clean the *DATETIME* column. We make use of the python package of pandas to read the datetime package to extract the hour information in the *DATETIME* attribute.



## **Findings & Conclusion (Details in Frequency Histogram)**

***DAY\_HOUR:*** categorical

* No missing values.
* The hours are evenly distributed in a day, which makes sense. Therefore, this attribute is already clean.

***SEASON:*** categorical

* No missing values.
* The days are evenly distributed into four seasons, which makes sense. Therefore, this attribute is already clean.

***HOLIDAY:*** categorical

* No missing values.
* There are fewer holidays in a year than non-holidays, which makes sense. Therefore, this attribute is already clean.

***WORKINGDAY:*** categorical

* No missing values.
* There are more working days in a year than non-working days, which makes sense. Therefore, this attribute is already clean.

***WEATHER:*** categorical

* No missing values.
* There are few samples with weather = 4. We do not ignore them, as special values carry important information. So, this attribute is clean.

***TEMP:*** numerical

* No missing values and no skewness.
* There are outliers for both the sub-peaks and peaks, so we need to substitute both the sub-peaks the peaks with mean.

***ATEMP:*** numerical

* No missing values and no skewness.
* There are outliers for both the sub-peaks and peaks, so we need to substitute both the sub-peaks the peaks with mean.

***HUMIDITY:*** numerical

* No missing values, and skewness to left.
* There are outliers for the sub-peaks, so we need to substitute the sub-peaks with mean.

***WINDSPEED:*** numerical

* No missing values and skewness to right.
* There are outliers for the peaks, so we need to substitute the peaks with mean.

## **Frequency Histogram**

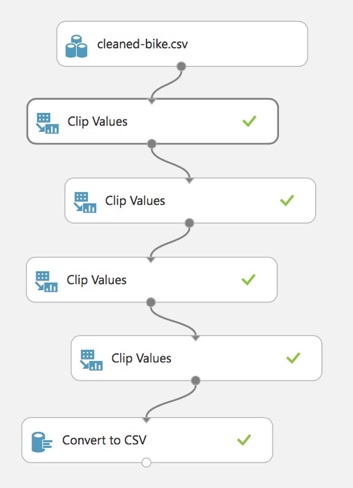
|  |  |  |
| --- | --- | --- |
| *https://lh3.googleusercontent.com/z3UH5ovh6p7DaHe6Q4A6_9rHI0osE0fdGWl0FMSzwUS5eOuXWLWMJE4NBbiv8RTb3FnfCqrNXeeyDL7mBK8kxjkfHQs5G8jL5exBB-EIDUzw0EZkQveBx_jLsUXkFrloeFkxDYl_*  *DAY\_HOUR* | *https://lh6.googleusercontent.com/bTNyVjTeTrms6I95TR0SmzMp-07G9B8qsvg2zjtkkkWn7ZOeLoDhkRX92QmXNCyCgHuAqgI5Ahn1hZt3MSje12SJftXNXJju319Xht6NAwmlpP2JOpRjMRkqKUkuY0PUBa5FPVRM*  *SEASON* | *https://lh6.googleusercontent.com/empfD5L6ndoThJJtp0p_56ICkRW3A87BrcQE9-C0aAjdI7yGLsfNTdWcFTBuaYRYvCzmBbUksnDDOvacLiOc6ymKsjRmf8gUvFcF8dozBM2d3jfex-fDsgw4Wz2t-KOcPmJ-DgHI*  *HOLIDAY* |
| *https://lh5.googleusercontent.com/aeuapi7yvjb9JaeKFsaJ0_uwELfiQjNt7CLxbkrzS8m1eJwb3xGSRkS15kb_O1bD5NHIPjbxzJGpdLWIQDXsZeIhiy7DLevNIbNUYJsENHUo3_HlqqlpZaCwMduJNT6K8NAlJD6_*  *WORKINGDAY* | *https://lh6.googleusercontent.com/6W94Yk5DQG24RsjqyEkkB2oKduyVTzPX37Uf7hW7WQ1pMR8vsT1IsHOYrqxqkZoAmsiEY59cNPqH7DoSzbJsK3FJaUNwUQD33MshqS6C7HOoIz0b0V54F2ozF9zsndhtnCzcm9xV*  *WEATHER* | *https://lh3.googleusercontent.com/J8e4va_MUX0ywc3-nucGj29rdtb_r7rELUgTovqhcYkkSRZars-m2xUJRcYFvpZTSBPsJQMAuN_e20qsbZyauyL0mcWBV7gaFFcQSfdGDJHIe8IXtqWMdBsrLAqNvAhtHW4piJmY*  *TEMP* |
| *https://lh6.googleusercontent.com/Q-yt94ncWb8lunTwiSrcdr83JjgsZXXKCA_xfiVbyZeYyy1yOgK9GLTBmANSCPbOWTmqnKeyr0k8lAuGb2P1tccf4R_-tK_-Le9oQH_tFASBS44GAoghKUtyugD4n5E55qlFDIHg*  *ATEMP* | *https://lh4.googleusercontent.com/JwUTUJm9plkg076NAyduRwvBT3Nvbmhl0iDgwaq_3PJCoUwrc7y2ZvStEV0PbxWTM43IscWOKlPhOYXyTlY0gDOlaBtyj9ILLWlZ3IvTxRDVncCDILS7EAkTBUI34amcbBgbvYTE*  *HUMIDITY* | *https://lh6.googleusercontent.com/TH2jqhAz9XZy_sByBgAU57mObNXg6OgqZO8WGEm8ajDEu9CjC9Kv3e9aZgThpnM6NKw5GOYQvAioLUPL95XfHaSkPRJJpuVHgsK1cSIIysDjsVtyJ5UhhVs7sXYgRF9r5k9a4fxB*  *WINDSPEED* |

[Appendix 2.1 - 2.7: Scatter Plot (with x-label =](#_Appendix) *[ATTRIBUTE](#_Appendix)* [& y-label =](#_Appendix) *[CASUAL](#_Appendix)* [or](#_Appendix) *[REGISTERED](#_Appendix)*[)](#_Appendix)

## **Clean Data**

After data analysis, we can conclude that there are no missing values in the dataset and that we need to handle the outliers for *TEMP, ATEMP, HUMIDITY,* and *WINDSPEED*, by replacing the outliers with the mean of all the other values (non-outliers).

Azure Experiment



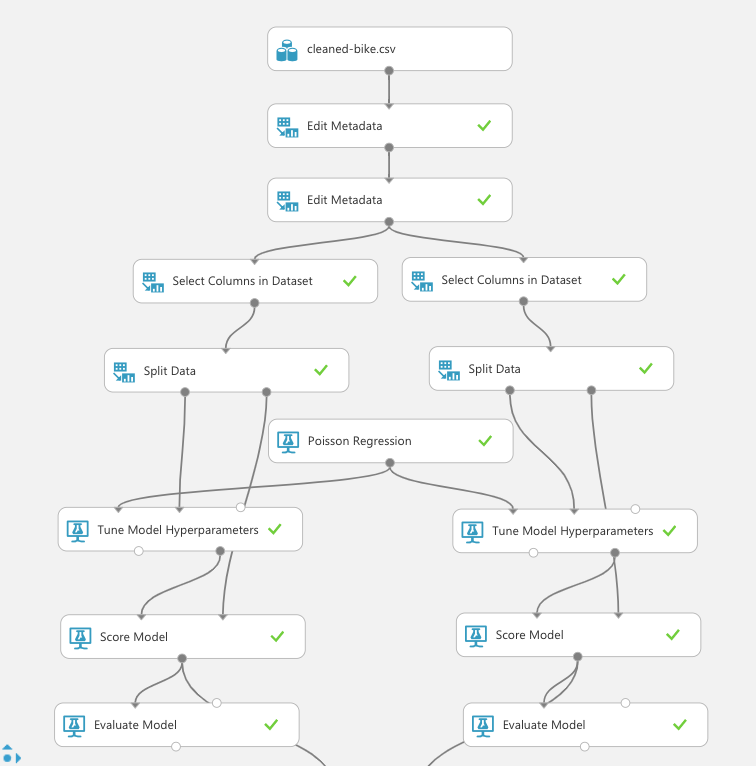
Explanations (“Clip Value”)

1. For *TEMP* and *ATEMP*, we clip the peak and sub-peak with a percentile of 99 and 1 respectively, then substitute the outliers with missing values.
2. For *HUMIDITY*, we clip the sub-peak with a percentile of 1, then substitute the outliers with missing values.
3. For *WINDSPEED*, we clip the peak with a percentile of 99, then substitute the outliers with missing values.
4. Lastly, for *TEMP, ATEMP, HUMIDITY,* and *WINDSPEED*, we replace the missing value with the mean of all the other values (non-outliers).

# **PART III: MODEL BUILDING**

## **General Idea**

The final target variable is the sum of the rentals by both casual and registered users, which could possibly have a different pattern in terms of the attributes provided. For instance, registered users are normally frequent users with daily commuting need, while casual users do bike rentals mainly for leisure activities. In this kind of situation, if we simply do a regression on the total count of rental, the prediction could contain a larger bias, and in the meanwhile, detailed patterns and information about each type of users could be lost. As a consequence, we decided to first build two models for the prediction of these two underlying intermediate target variables. After choosing the best model for *CAUSAL* and *REGISTERED* respectively, we can sum up the prediction of *CAUSAL* and *REGISTERED* as the prediction for the total rental.



# **Procedures & Explanations**

1. “Edit Metadata”: The first one is to set *HOLIDAY* and *WORKINGDAY* to be Boolean, and the second one is to set *DAY\_HOUR*, *SEASON*, and *HOLIDAY* to be categorical.
2. “Tune Model Hyperparameters”: We use this module with the entire grid mode to figure out the best set of hyperparameters using the try-them-all strategy.
3. “Select Columns in Dataset” & “Split Data”: On the left-hand side, we are choosing the *CASUAL* variable as the target variable for the model while using all other columns excluding the *REGISTERED* and *COUNT* as attributes. On the right-hand side, the target variable is *REGISTERED,* and the model is built on all other columns excluding *CASUAL* and *COUNT*. For each side, we use the same random seed to split the data into a training dataset containing 70% of the data and a testing dataset composed of the remaining 30% of rows.
4. “XXX Regression”: Different regression algorithm with a set of parameter ranges.
5. “Score Model”: This module is used to make the prediction of the test set, using the built model.
6. “Evaluate Model”: This module is used to calculate the performance of the model.

## **Trained Models & Understanding**

**Linear Regression and Poisson Regression**

|  |  |  |  |
| --- | --- | --- | --- |
| Setting for Linear Regression | Parameter Range | *CASUAL* | *REGISTERED* |
| Normalize Features | TRUE | TRUE | TRUE |
| Averaged | TRUE | TRUE | TRUE |
| Learning Rate | 0.025, 0.05, 0.1, 0.2 | 0.2 | 0.2 |
| Number of Iterations | 1, 10, 100 | 1 | 100 |
| L2 Regularization | 0.001, 0.01, 0.1 | 0.001 | 0.001 |

|  |  |  |  |
| --- | --- | --- | --- |
| Setting for Poisson Regression | Parameter Range | *CASUAL* | *REGISTERED* |
| Optimization tolerance | 0.00001, 0.00000001 | 0.00001 | 0.00001 |
| L1 regularization | 0.0, 0.01, 0.1, 1.0 | 0.1 | 0.0 |
| L2 regularization | 0.01, 0.1, 1.0 | 0.1 | 0.01 |
| Memory size for L-BFGS | 5, 20, 50 | 5 | 5 |

Both Poisson and Linear regression produce a set of feature weights that could perceivably show the direction and extent of correlation.

From the feature weights derived from these two models for *CASUAL*, we can conclude that during 1-6am, which is the ordinary sleeping hours, less casual users would rent the bike since the coefficients are all negative. In addition, it is exhibited that sensible temperature is the most important positively related factor considering casual users. This also answers to the intuition that most casual users would choose to rent a bike when the sensible temperature is so high that they do not feel like walking. Moreover, the number of casual rentals tend to decrease during working days, as most of them do not have a leisure time during working days.

Some counterintuitive findings could also be observed. For instance, the coefficient of *HOLIDAY\_True* is negative, which indicates that less casual users would rent a bike during holidays when they actually have more leisure time. After a thorough discussion, we figured that this could be a result of the increased traffic volume during holidays and most people travel a little further which require cars or buses as transportation.

Differences exist between the findings of these two models. For linear regression, it is exhibited that more casual users would choose to rent a bike during summer (*SEASON\_2\_1*) and autumn (*SEASON\_3\_2*), as well as when the weather is clear (*WEATHER\_1\_0*). Also, humidity is having a strong negative correlation with the bike rentals by casual users. On the contrary, in Poisson regression, the coefficient for all seasons and weathers are also negative, against the intuition that bike rental would be more popular during clear summer days.

These two models also produced several similar conclusions for *REGISTERED*. As shown from the feature weights, similar to the result for casual users, registered users also do not rent the bikes during sleeping hours from 1 am to 5 am. However, the day hours of 8-9am and 5-8pm have a strong positive correlation with the rental by registered users. This confirms the assumption that registered users are mostly using the rental for commuting. The weights for *ATEMP* and *TEMP* are much lower than the ones for casual users, which supports the viewpoint that registered users would need to commute to work regardless of the temperature condition, while casual users could choose to rent the bike according to how they feel about the temperature.

In brief, both models exhibit that the determinant factors for casual users are mostly temperature, weather and season while those for registered users are mostly the hour of a day (*DAY\_HOUR*). This significantly corresponds to our assumption that casual users could choose to rent for leisure activity during good days while registered users do routine commutation on specific hours of the day.

[Appendix 3.1 - 3.4: optimal feature weights for linear regression and poison regression](#_PART_VII:_Appendix)

**Decision Forest Regression and Boosted Decision Tree**

|  |  |  |  |
| --- | --- | --- | --- |
| Settings for Decision Forest Regression | Parameter Range | *CASUAL* | *REGISTERED* |
| Minimum Number of Samples Per Leave Node | 1, 4, 16 | 1 | 4 |
| Number of Random splits per node | 1, 128, 1024 | 128 | 128 |
| Max depth of the decision trees | 1, 16, 64 | 64 | 64 |
| Number of decision trees | 1, 8, 32 | 32 | 32 |

|  |  |  |  |
| --- | --- | --- | --- |
| Settings for Boosted Decision Tree | Parameter Range | *CASUAL* | *REGISTERED* |
| Maximum number of leaves per tree | 8, 32, 128 | 128 | 128 |
| Minimum number of samples per leaf node | 10, 50 | 10 | 10 |
| Learning rate | 0.01 | 0.01 | 0.01 |
| Number of trees constructed | 20, 100 | 100 | 100 |

We have also conducted 2 decision tree regression models, decision forest and boosted decision tree. As was shown by the calculated tree of the two models, we have found that attributes derived from *WORKINGDAY* and *DAY\_HOUR* are the important roots used in most trees. Specifically, *WOKINGDAY.0* is used in the decision forest model for predicting the casual user amount, *DAY\_HOUR.17* is used in the decision forest model for predicting the registered user amount and *DAY\_HOUR.18* is used for predicting the registered user amount in the boosted decision tree model. This indicated that whether the hour is on a working day and whether it is at 17:00 to 18:00 when an employee normally gets off work is crucial for predicting the user amount - citizens use the share-bikes for commuting from workplace to their home during Monday to Friday the most. While this matches with our intuitive thoughts, we can also observe that people use bikes less when they commute from their home to workplace, we think this is because as biking requires certain physical power thus riding bikes to work might not be as convenient and is not the prior choice for going to the office.

Similar to Poisson regression and linear regression, *ATEMP* is also used as the root in the boosted decision tree model for predicting casual user amount.

**Fast Forest Quantile**

|  |  |  |  |
| --- | --- | --- | --- |
| Settings for Fast Forest Quantile | Parameter Range | *CASUAL* | *REGISTERED* |
| Number of Trees | 16, 32, 64 | 32 | 32 |
| Number of Leaves | 16, 32, 64 | 64 | 64 |
| Minimum Instance Per Leaf Node | 1, 5, 10 | 10 | 10 |
| Bagging Fraction | 0.25, 0.5, 0.75 | 0.75 | 0.75 |
| Feature Fraction | 0.25, 0.5, 0.75 | 0.75 | 0.75 |
| Quantile Sample Count | 100 | 100 | 100 |

Fast forest quantile regression was also adopted as one of our data-mining techniques. In tune model hyperparameters, we used “Root of mean squared error” as the metric for measuring the performance of the regression. Mean squared error calculates the distance between the target variable and predicted values. From our results, we found that Quantile Loss of 0.5 has the lowest Root Mean Squared Error for both casual and registered users. Therefore, we will be choosing Quantile loss of 0.5 as the interval to predict our target variable.

[Appendix 3.5 & 3.6: Comparison of Models – Fast Forest Quantile for *CASUAL* & *REGISTERED*](#_Appendix)  
[Appendix 3.7 & 3.8: Settings – Bayesian Linear Regression & Neural Network](#_Appendix)

# **PART IV: PERFORMANCE EVALUATION**

## **General Idea**

We split the raw data into a training set (70%) and a test set (30%), train the model with the training set, and evaluate the model with the test set.

1. The “Tune Model Hyperparameter” Module in Azure is used to specify a set of values for every hyperparameter in a regression algorithm, every combination of hyperparameters will be a different model. We use the entire grid option, so as to test each and every combination, and the best model will be chosen at last.
2. In order to choose the best model among all combinations, the idea of “cross validation” will be used automatically in this module to prevent overfitting and to calculate the performance. During each model training, it basically separates the training dataset into several folds. For each iteration, it chooses a fold to be the validation set and all the others to be the training set. The final performance of each model is calculated by the mean of the performance of all the iterations. Eventually, the model with the best performance is chosen as the best model.
3. Every different regression algorithm will give us the best model. We evaluate these models with the test set, which has not been used in the model training process.

## **Benchmark**

To evaluate the model, we decide to use three criteria, Mean Absolute Error (MAE), Root Mean Squared Error (RMSE) and Coefficient of Determination (R2). We choose the mean value of CAUSAL and REGISTERED column as our benchmark prediction, and accordingly, we can calculate the benchmark value for criteria.

*CASUAL*: MAE = 34.628605, RMSE = 49.958182, R2 = 0.000000  
*REGISTERED*: MAE = 114.476759, RMSE = 157.707829, R2 = 0.000000  
*COUNT*: MAE = 142.711387, RMSE = 181.136134, R2 = 0.000000

[Appendix 3.9: Python Code for Calculating Benchmark](#_Appendix)

## **Ranking for *CASUAL***

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Rank | Model | MAE |  | Rank | Model | RMSR |  | Rank | Model | R2 |
| 1 | Decision Forest | 14.13 |  | 1 | Decision Forest | 24.56 |  | 1 | Decision Forest | 0.74 |
| 2 | Poisson | 14.84 |  | 2 | Poisson | 26.74 |  | 2 | Poisson | 0.70 |
| 3 | Fast Forest | 15.59 |  | 3 | Neural Network | 27.96 |  | 3 | Neural Network | 0.68 |
| 4 | Boosted DT | 17.20 |  | 4 | Fast Forest | 28.97 |  | 4 | Fast Forest | 0.65 |
| 5 | Neural Network | 20.37 |  | 5 | Boosted DT | 31.54 |  | 5 | Boosted DT | 0.59 |
| 6 | Bayesian Linear | 22.64 |  | 6 | Bayesian Linear | 32.28 |  | 6 | Bayesian Linear | 0.57 |
| 7 | Linear | 27.99 |  | 7 | Linear | 47.10 |  | 7 | Linear | 0.08 |
| 8 | Benchmark | 34.63 |  | 8 | Benchmark | 49.96 |  | 8 | Benchmark | 0.00 |

## **Ranking For *REGISTERED***

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Rank | Model | MAE |  | Rank | Model | RMSR |  | Rank | Model | R2 |
| 1 | Decision Forest | 44.08 |  | 1 | Decision Forest | 65.51 |  | 1 | Decision Forest | 0.80 |
| 2 | Fast Forest | 47.34 |  | 2 | Fast Forest | 75.54 |  | 2 | Fast Forest | 0.74 |
| 3 | Poisson | 57.79 |  | 3 | Neural Network | 84.46 |  | 3 | Neural Network | 0.68 |
| 4 | Boosted DT | 63.98 |  | 4 | Poisson | 85.13 |  | 4 | Poisson | 0.67 |
| 5 | Neural Network | 65.13 |  | 5 | Bayesian Linear | 92.63 |  | 5 | Bayesian Linear | 0.61 |
| 6 | Bayesian Linear | 66.63 |  | 6 | Boosted DT | 100.03 |  | 6 | Boosted DT | 0.55 |
| 7 | Linear | 108.43 |  | 7 | Benchmark | 151.03 |  | 7 | Benchmark | 0.00 |
| 8 | Benchmark | 114.48 |  | 8 | Linear | 157.71 |  | 8 | Linear | -0.12 |

With regards to the above ranking charts, we can see that Decision Forest ranks over all the other models in all the criteria. Decision Forest model generates a lot of decision trees to predict the testing data. It uses the mean value of all the predicted value to be the final predicted result. So, it will be useful to examine the most common root node of all the decision tree in the Decision Forest model.

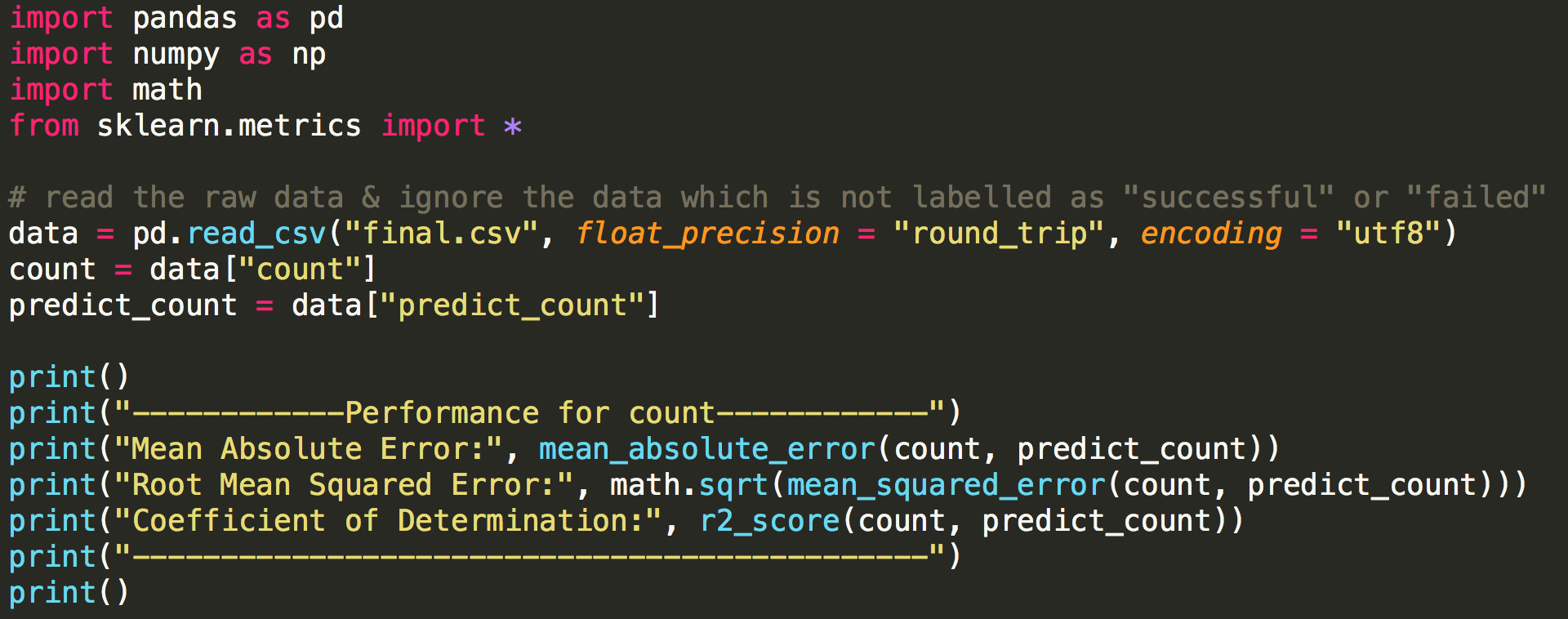
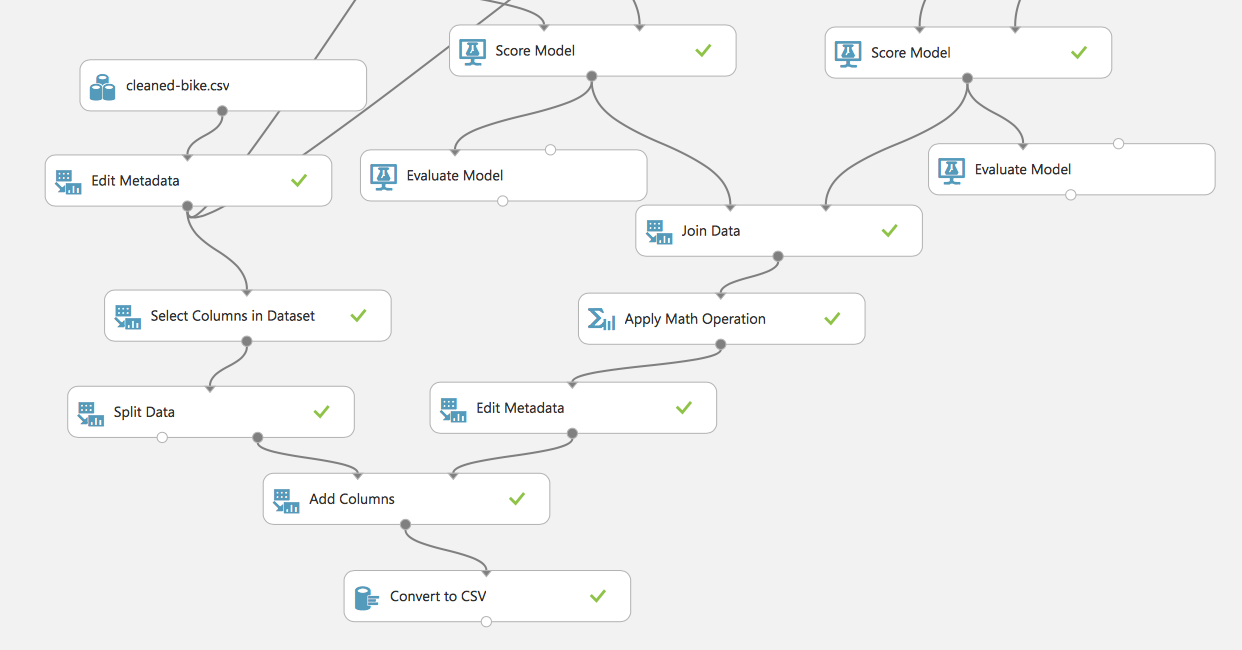
The Decision Forest model with a root of WORKINGDAY.0 is very common in *CASUAL* model. This indicates that the rental by casual users has a strong correlation with whether it is a workday or not. It is normal to think that non-registered users need to work during working days, if a lot of non-registered users are using the rental bike, it would be strange that they do not register as a member in the rental bike app.

For the registered users, on the other hand, the Decision Forest model with a root of *DAY\_HOUR.18* is the most common. This indicates that the rental records of registered users are concentrated at 18:00 of the day, which is normally the time people leave companies for home. Registered users ride bikes back home.

As exhibited through the rankings, the performances of all models except for Linear regression are better than the benchmark performance, providing instructive and meaningful findings of the bike-sharing ecosystem. Combining the findings of these two target variables, it is exhibited that the Mean Absolute Errors and Root Mean Squared Errors are much higher for registered users than casual users, an explanation could be that the number of registered rentals is much larger than casual rentals in records. On the other hand, the top three models for *REGISTERED* have a higher Coefficient of Determination than those for *CASUAL*, this is an indication that there exists a stronger pattern of activities for registered users than casual users.

## **Final *COUNT* Prediction & Performance**

Now, we can decide to use the best-built models for *CASUAL* and *REGISTERED* to run on the testing data. We use the following sub-experiment to sum up the predicted value for *CASUAL* and *REGISTERED* to get *COUNT*, and the following python code to get the performance for *COUNT*. By comparing to the benchmark, we can conclude that our model for predicting *COUNT* is very good in terms of Mean Absolute Error, Root Mean Squared Error, and Coefficient of Determination.



|  |  |  |  |
| --- | --- | --- | --- |
|  | Mean Absolute Error | Root Mean Squared Error | Coefficient of Determination |
| Our Model | 58.728966 | 83.229623 | 0.786675 |
| Benchmark | 142.711387 | 181.136134 | 0.000000 |

# **PART V: CONCLUSIONS**

In conclusion, the problem we want to tackle is to predict the bike users amount in a specific hour of a day with certain conditions applied, such as timing, weather, temperature, etc. We found out that the Decision Forest model had the best overall performance for both *CASUAL* and *REGISTERED*. Through this project, we have gained valuable insights into how different factors would influence the number of users of share-bikes, the root node *WORKINGDAY.0* and *DAY\_HOUR.18* are the most common root nodes for the two models respectively, which means that they provide the most information gain.

To sum up, indications are that the most important attributes for casual and registered users are *WORKINGDAY* and *DAY\_HOUR* respectively. Thus, if we choose a specific hour in a day, the prediction of user amount can be easily computed. Useful forward-looking information will be generated and will definitely help companies and share-bike investors to gain a deeper understanding of how citizens like to commute thus being able to better make corporate decisions or investment decisions regarding the bike-sharing ecosystem.

To reflect on our current approach to this issue, we would also like to point out some limitations and propose some potential improvements for our current approach.

1. Current attributes of the dataset are limited and can be expanded when gathering the data in the future. For example, information about location, user time elapsed, etc. are possible relevant attributes for user amount prediction.
2. Current dataset is based on the bike-sharing program in Washington D.C., though it provided insights into general user habit, the patterns might differ from city to city.

# **PART VI: References**

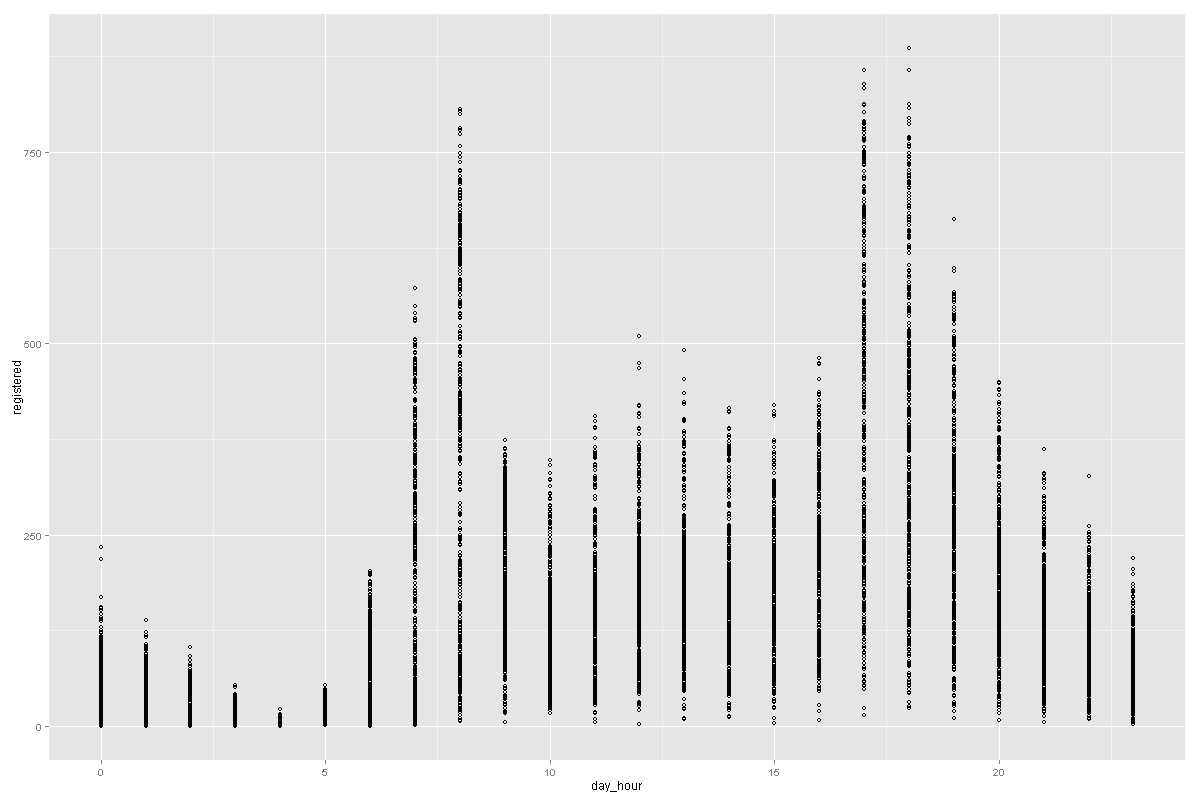
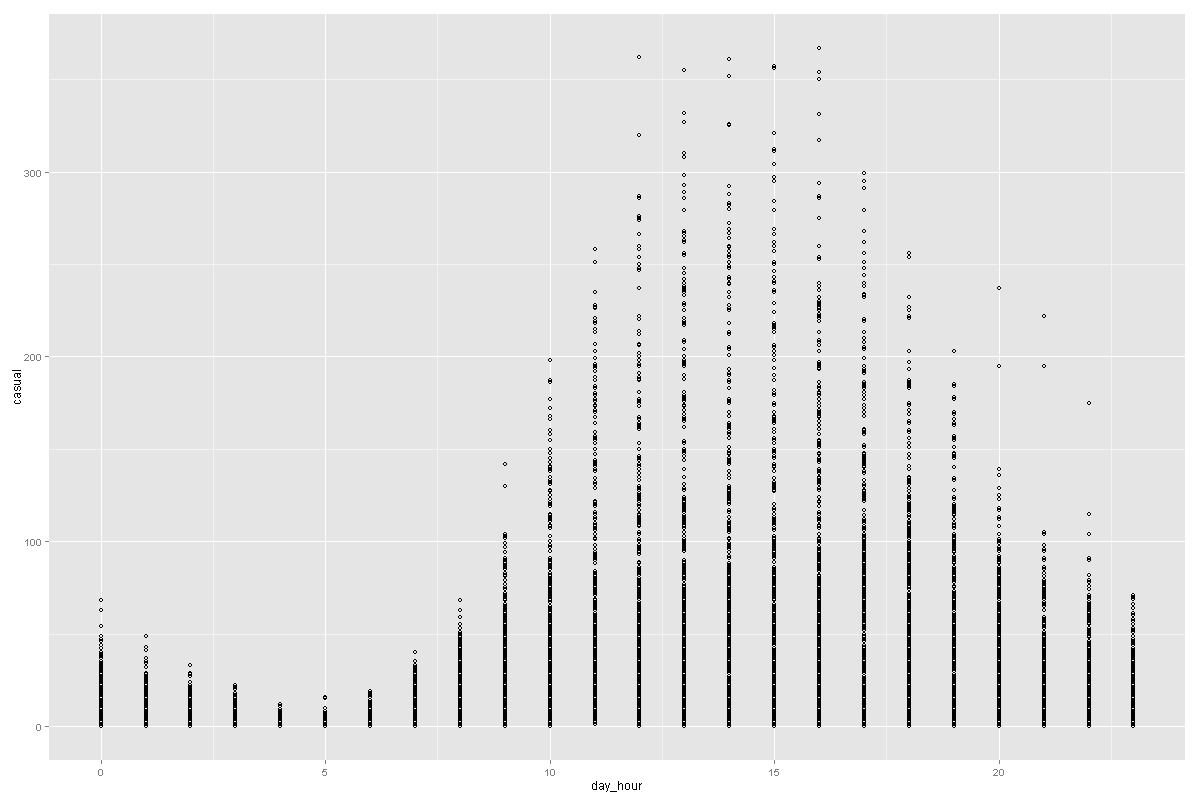
Python:  
<https://pandas.pydata.org/pandas-docs/stable/generated/pandas.read_csv.html>  
<https://docs.python.org/3/library/datetime.html>  
<https://scikit-learn.org/stable/modules/model_evaluation.html#regression-metrics>

Azure:  
<https://gallery.azure.ai/CustomModule/Create-Scatterplot-1>

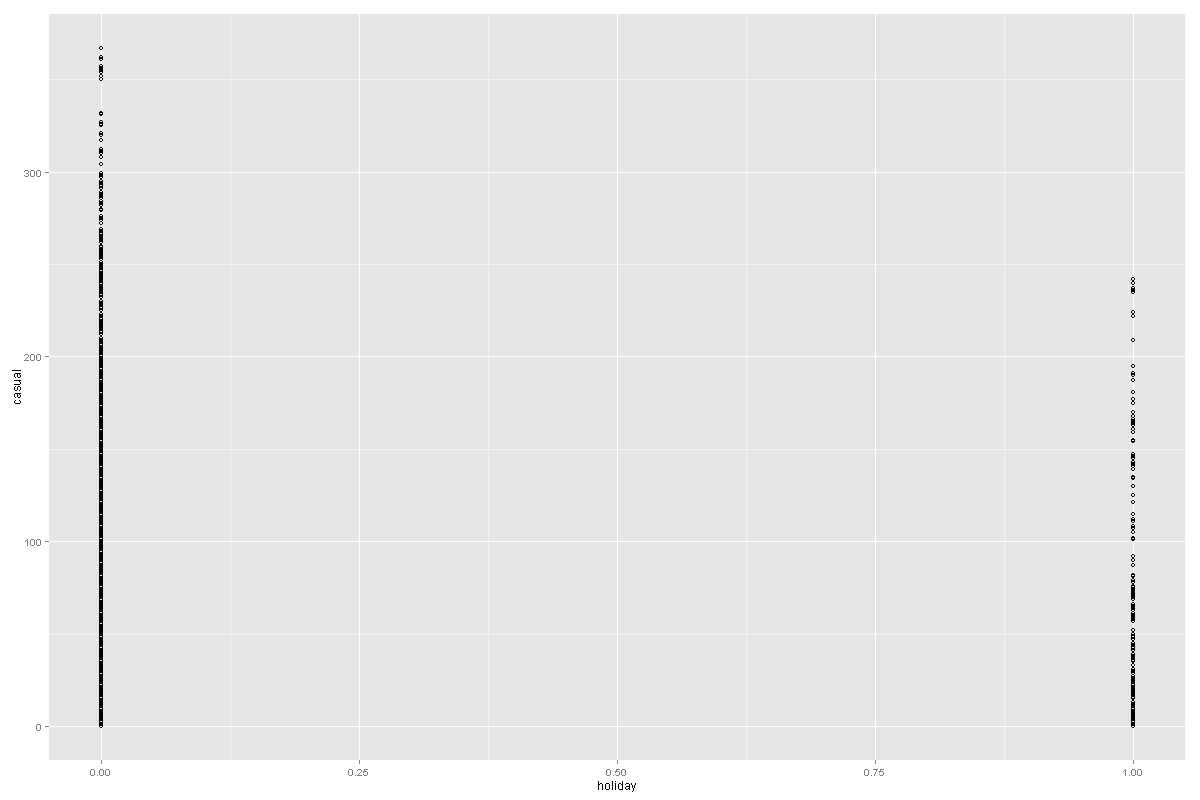
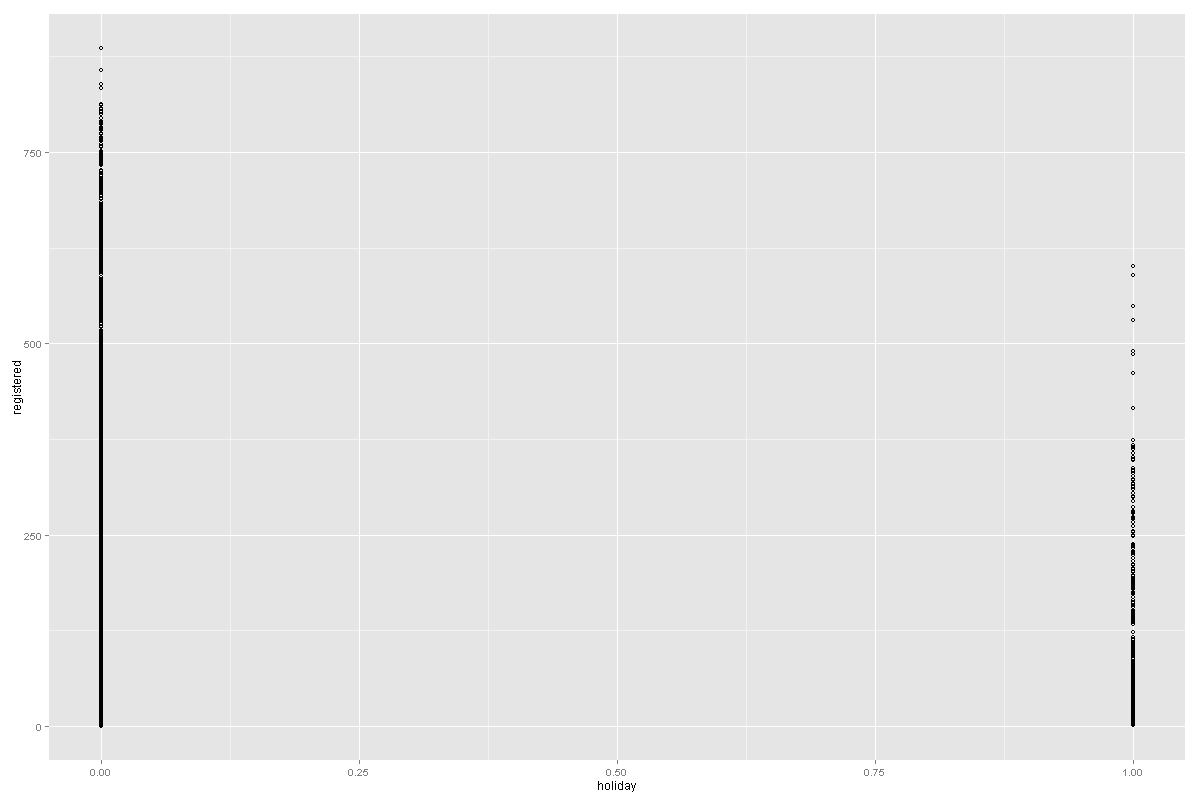
<https://docs.microsoft.com/en-us/azure/machine-learning/studio-module-reference/machine-learning-initialize-model-regression>

# **PART VII: Appendix**

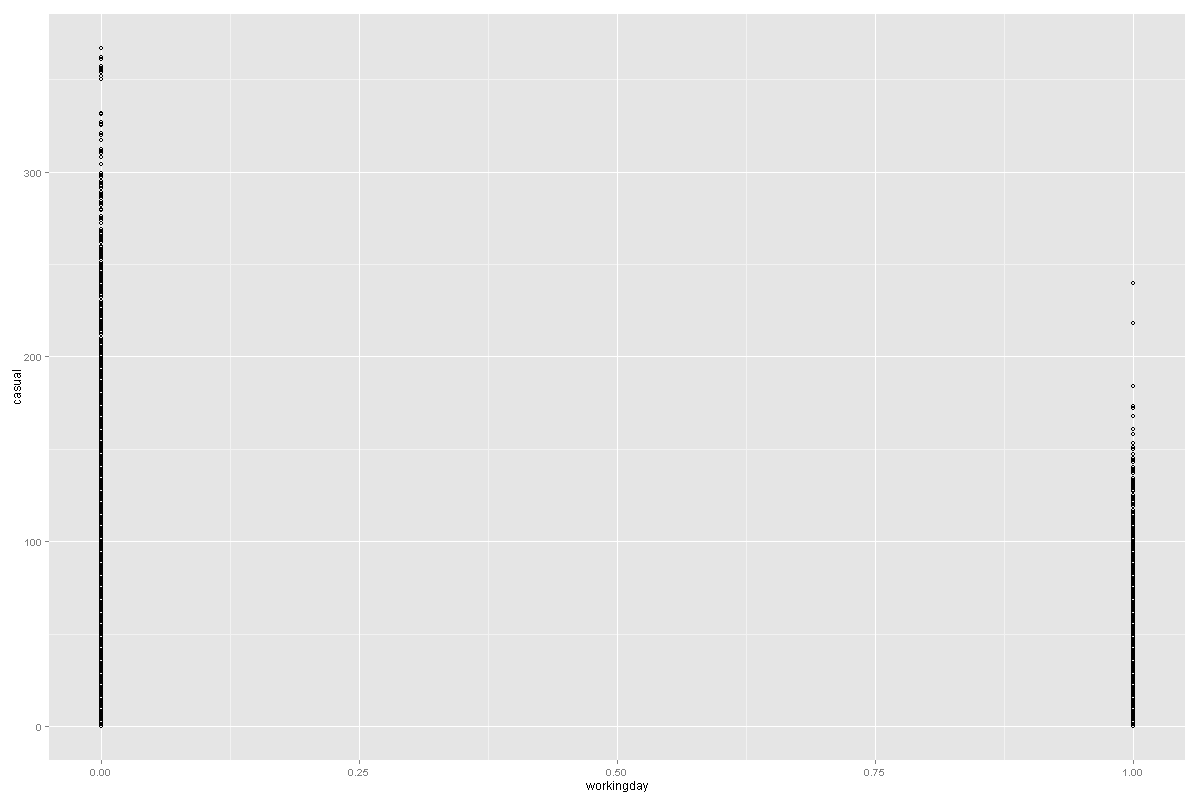
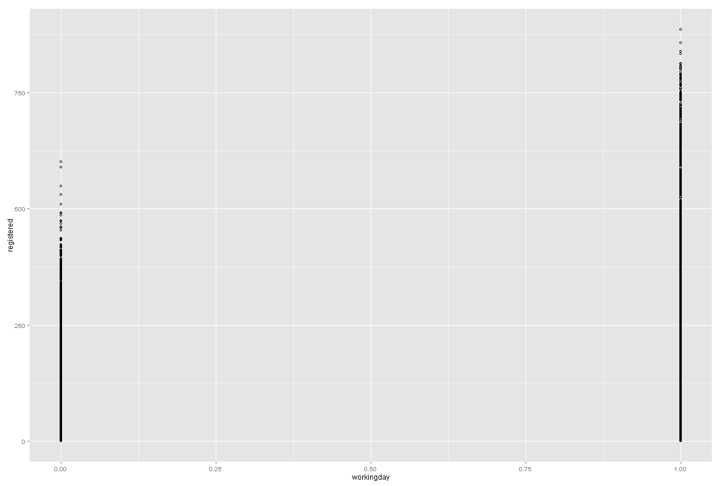
Appendix 2.1 Scatter Plot for *DAY\_HOUR* (with y-label = *CASUAL* or *REGISTERED*):

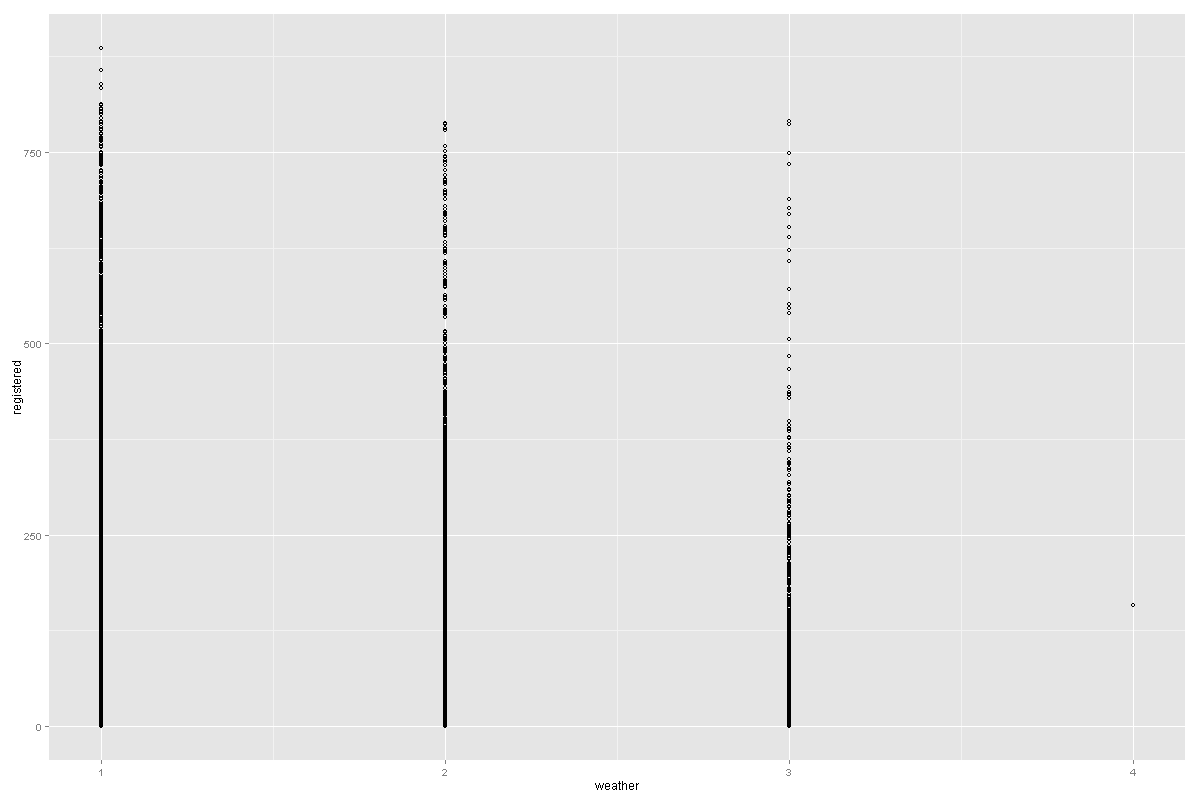
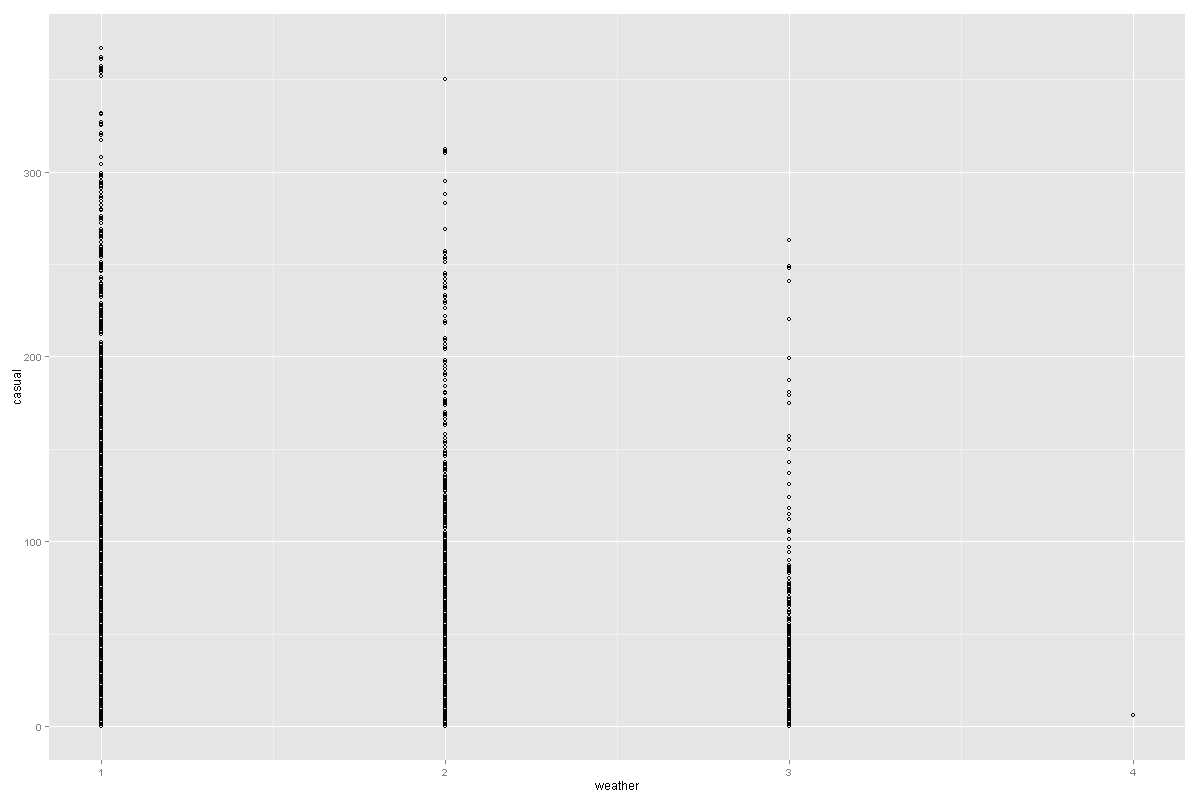


Appendix 2.2 Scatter Plot for *HOLIDAY* (with y-label = *CASUAL* or *REGISTERED*):

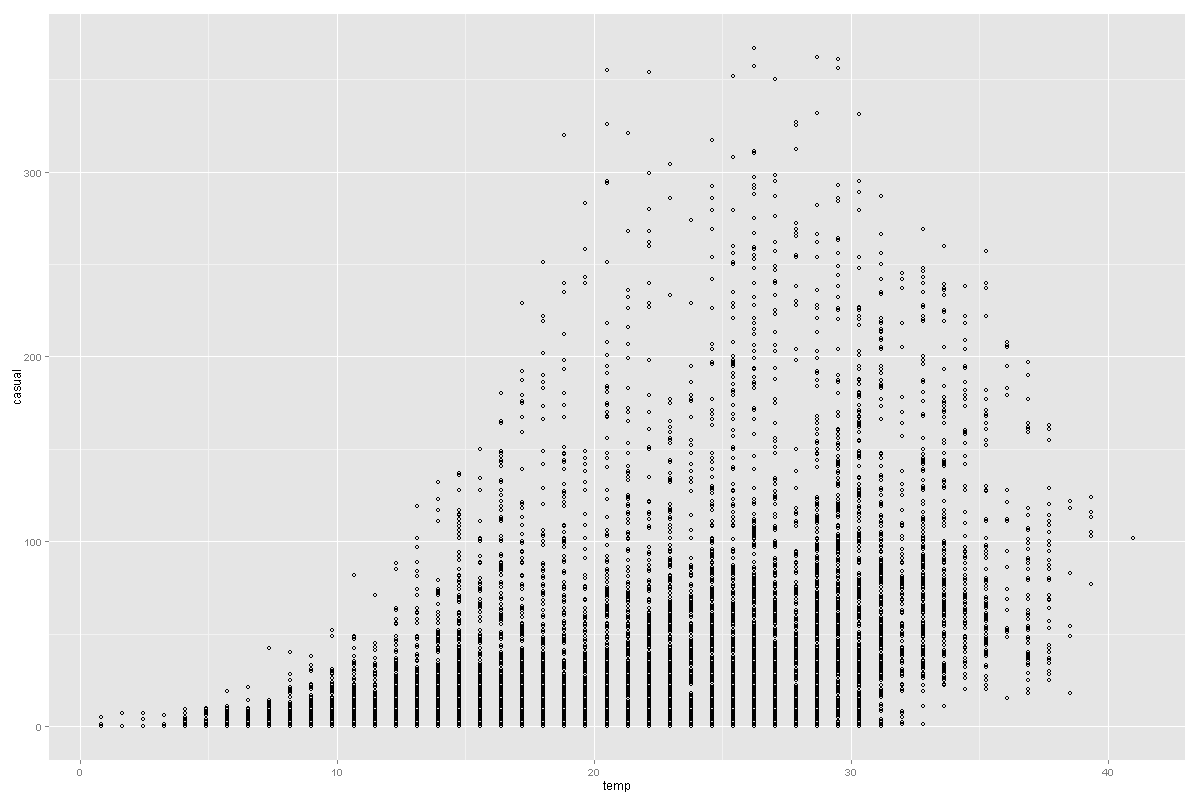
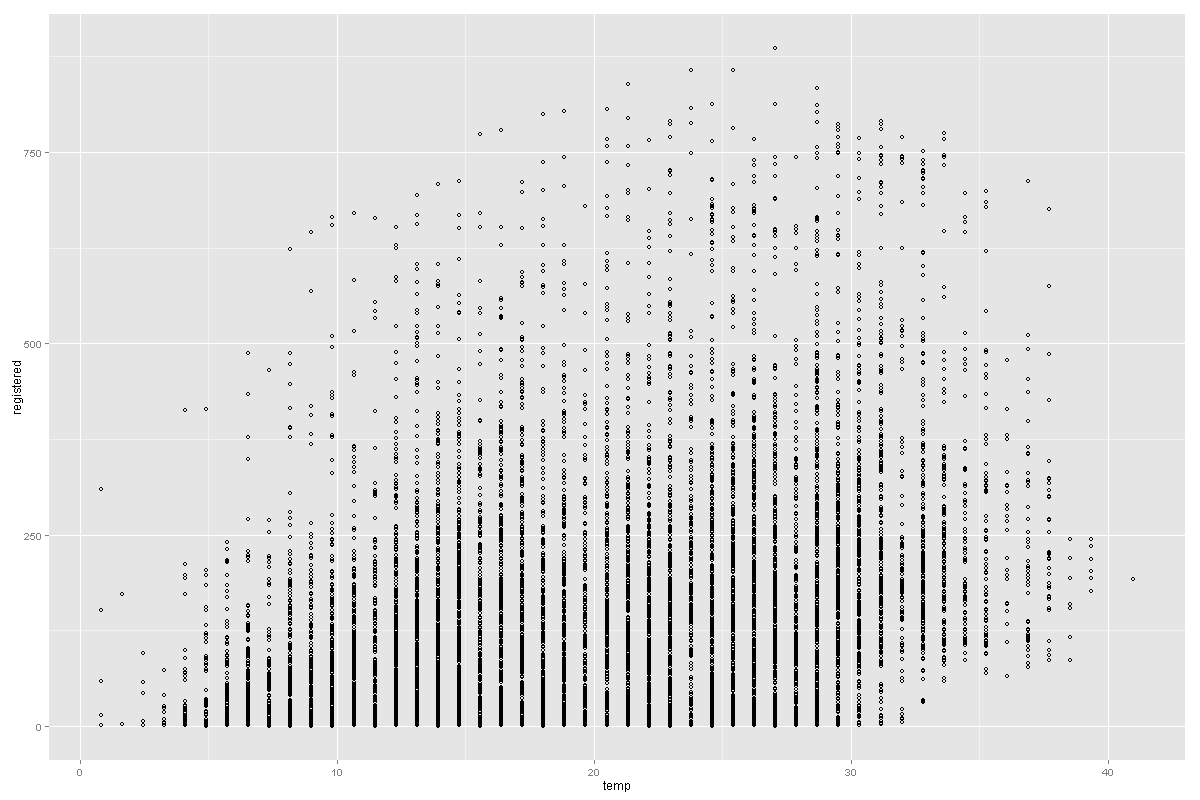
 

Appendix 2.3 Scatter Plot for *WORKINGDAY* (with y-label = *CASUAL* or *REGISTERED*):

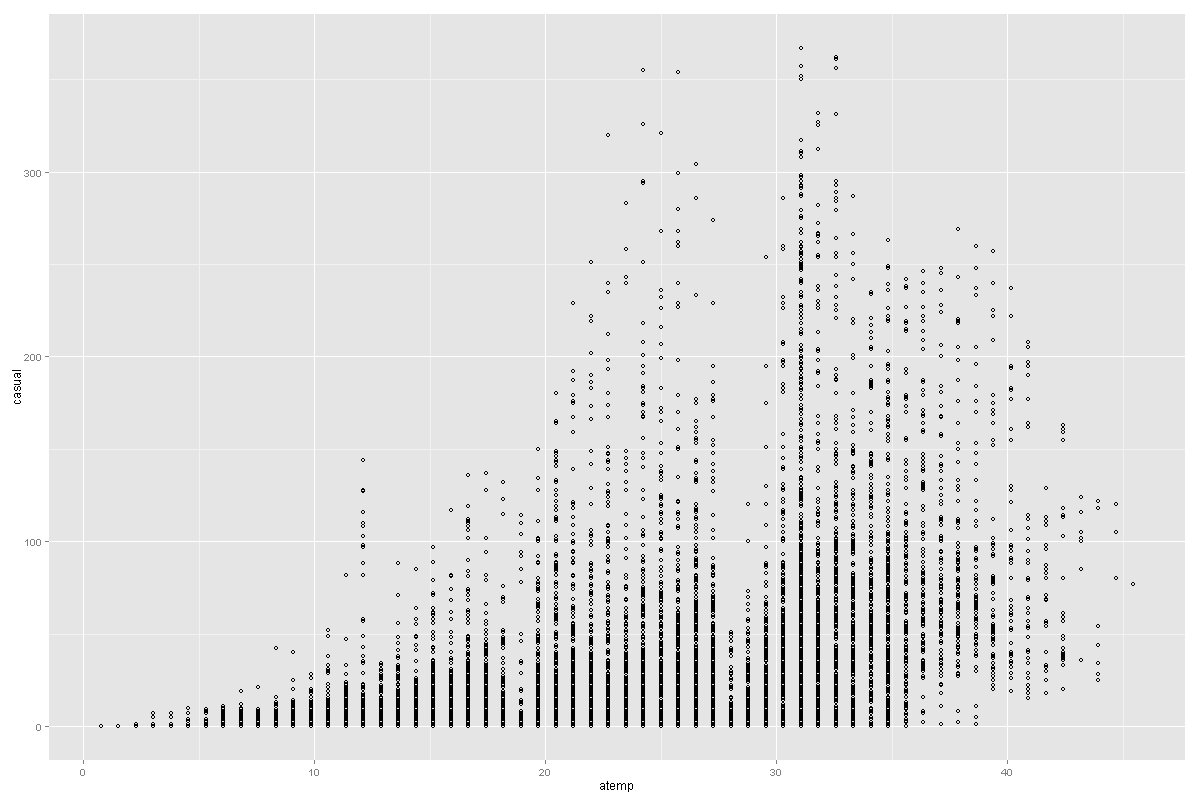
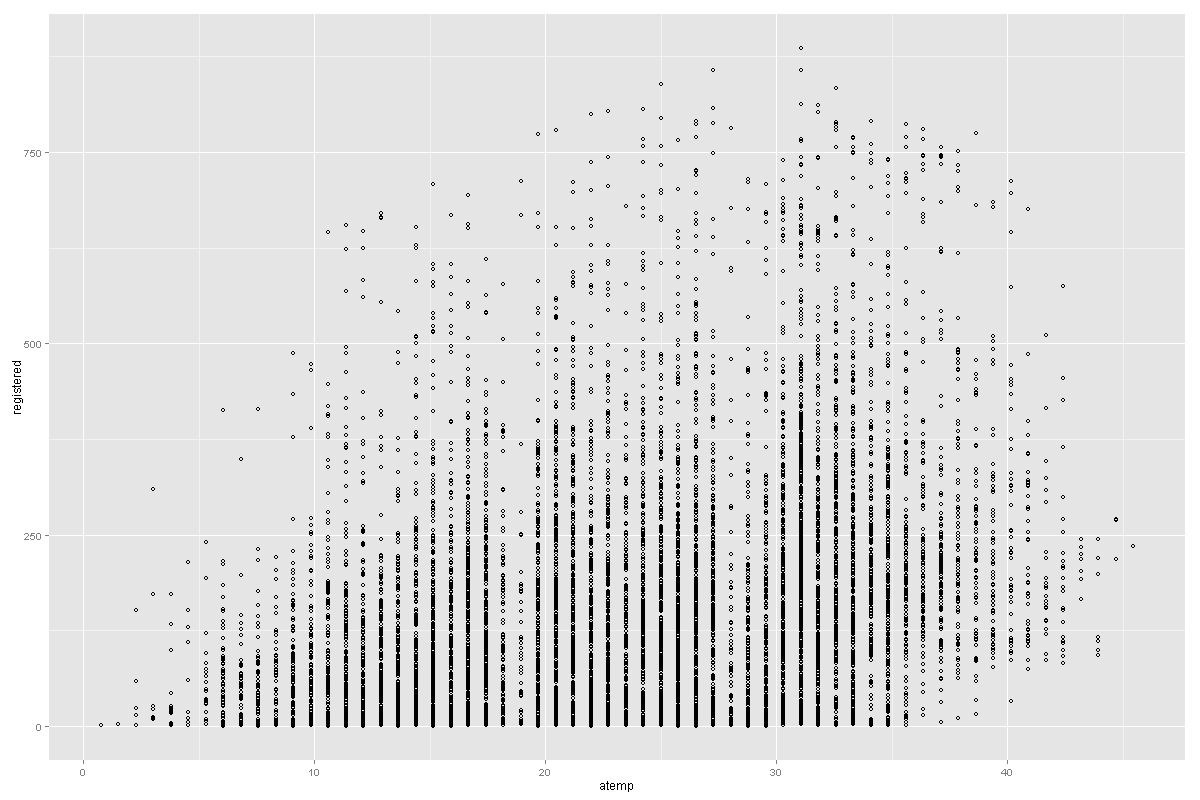
 

Appendix 2.4 Scatter Plot for *WEATHER* (with y-label = *CASUAL* or *REGISTERED*): 

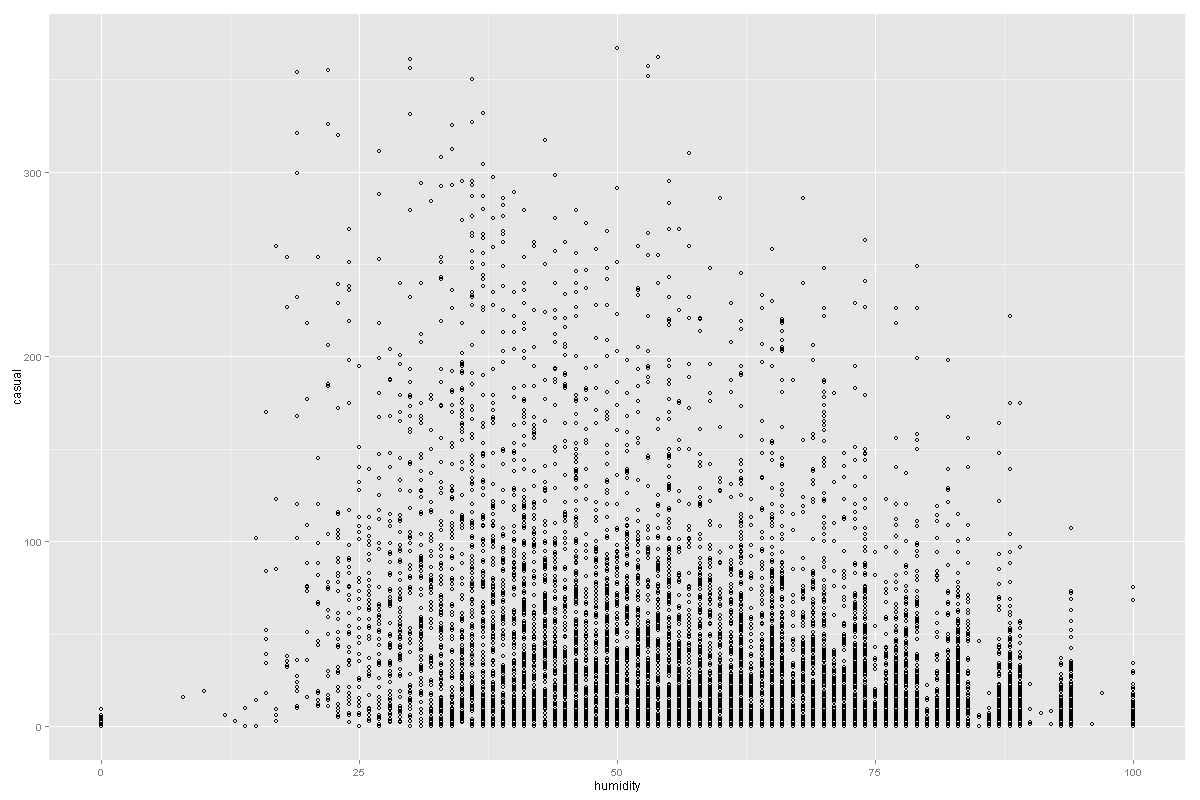
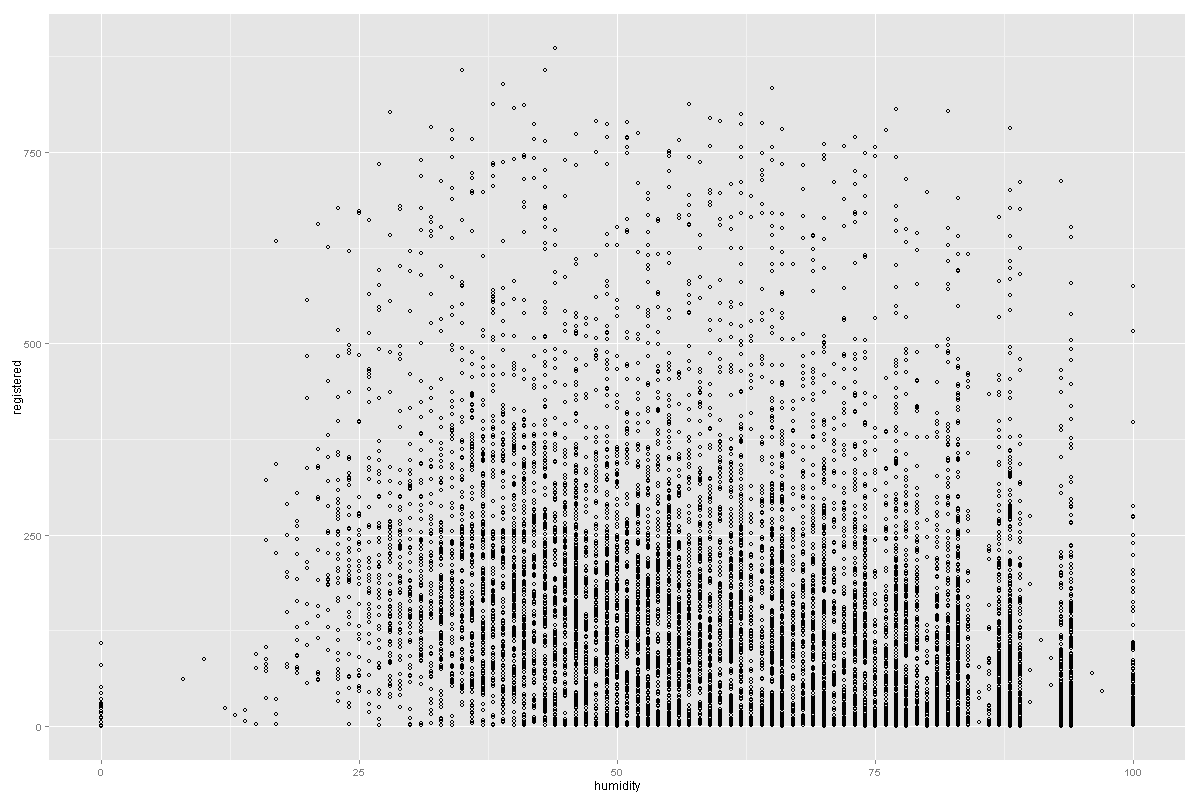
Appendix 2.5 Scatter Plot for *TEMP* (with y-label = *CASUAL* or *REGISTERED*):

Appendix 2.6 Scatter Plot for *ATEMP* (with y-label = *CASUAL* or *REGISTERED*):

Appendix 2.7 Scatter Plot for *HUMIDITY* (with y-label = *CASUAL* or *REGISTERED*):

Appendix 3.1 Feature Weights - Linear Regression for *CASUAL*

|  |  |  |  |
| --- | --- | --- | --- |
| **Feature** | **Weight** | **Feature** | **Weight** |
| temp | 12.0738 | day\_hour\_18\_18 | 3.71124 |
| workingday\_0\_0 | 11.5912 | day\_hour\_0\_0 | -3.63833 |
| atemp | 11.262 | day\_hour\_11\_11 | 3.57668 |
| workingday\_1\_1 | -9.23524 | windspeed | 2.95699 |
| humidity | -8.7124 | weather\_3\_2 | -2.8408 |
| season\_1\_0 | -7.7765 | day\_hour\_7\_7 | -2.74795 |
| day\_hour\_15\_15 | 5.97819 | day\_hour\_23\_23 | -2.69091 |
| season\_3\_2 | 5.96409 | day\_hour\_10\_10 | 2.64719 |
| season\_2\_1 | 5.71148 | day\_hour\_22\_22 | -2.50919 |
| day\_hour\_14\_14 | 5.59068 | holiday\_0\_0 | 2.11903 |
| day\_hour\_13\_13 | 5.20559 | day\_hour\_19\_19 | 2.11485 |
| day\_hour\_17\_17 | 5.18329 | day\_hour\_8\_8 | -1.62232 |
| day\_hour\_16\_16 | 5.14245 | season\_4\_3 | -1.54313 |
| weather\_1\_0 | 5.10455 | day\_hour\_21\_21 | -1.10687 |
| day\_hour\_12\_12 | 4.85362 | Bias | 1.10381 |
| day\_hour\_4\_4 | -4.72549 | day\_hour\_9\_9 | -0.804421 |
| day\_hour\_3\_3 | -4.68338 | holiday\_1\_1 | 0.236916 |
| day\_hour\_5\_5 | -4.58188 | day\_hour\_20\_20 | 0.132905 |
| day\_hour\_6\_6 | -4.35596 | weather\_2\_1 | 0.0950255 |
| day\_hour\_1\_1 | -4.21652 | weather\_4\_3 | -0.0028312 |
| day\_hour\_2\_2 | -4.09751 |  |  |

Appendix 3.2 Feature Weights - Linear Regression for *REGISTERED*

|  |  |  |  |
| --- | --- | --- | --- |
| **Feature** | **Weight** | **Feature** | **Weight** |
| Bias | 88.5154 | day\_hour\_6\_6 | -1.60265 |
| season\_1\_0 | -5.0362 | season\_4\_3 | 1.52989 |
| day\_hour\_17\_17 | 4.62706 | season\_2\_1 | 1.49484 |
| day\_hour\_8\_8 | 4.1823 | day\_hour\_23\_23 | -1.47401 |
| temp | 4.05548 | day\_hour\_7\_7 | 0.937666 |
| atemp | 3.79185 | day\_hour\_22\_22 | -0.802287 |
| day\_hour\_18\_18 | 3.249 | windspeed | 0.782486 |
| humidity | -3.01205 | day\_hour\_12\_12 | 0.775298 |
| day\_hour\_4\_4 | -2.95939 | day\_hour\_13\_13 | 0.617269 |
| workingday\_1\_1 | 2.9192 | day\_hour\_10\_10 | -0.589339 |
| workingday\_0\_0 | -2.8373 | day\_hour\_9\_9 | 0.576373 |
| day\_hour\_3\_3 | -2.55227 | day\_hour\_20\_20 | 0.535393 |
| day\_hour\_2\_2 | -2.47129 | day\_hour\_15\_15 | 0.302753 |
| day\_hour\_5\_5 | -2.40201 | holiday\_0\_0 | 0.251441 |
| day\_hour\_1\_1 | -2.25092 | day\_hour\_21\_21 | -0.180095 |
| weather\_1\_0 | 2.25048 | holiday\_1\_1 | -0.169547 |
| season\_3\_2 | 2.09337 | weather\_2\_1 | -0.134606 |
| weather\_3\_2 | -2.03436 | day\_hour\_11\_11 | -0.112653 |
| day\_hour\_0\_0 | -1.98977 | day\_hour\_14\_14 | 0.0639742 |
| day\_hour\_19\_19 | 1.97248 | weather\_4\_3 | 0.00037505 |
| day\_hour\_16\_16 | 1.629 |  |  |

Appendix 3.3 Feature Weights - Poisson Regression for *CASUAL*

|  |  |  |  |
| --- | --- | --- | --- |
| **Feature** | **Weight** | **Feature** | **Weight** |
| day\_hour\_4\_4 | -2.34683 | holiday\_True\_1 | -0.726945 |
| day\_hour\_5\_5 | -2.25298 | humidity | -0.684641 |
| day\_hour\_3\_3 | -1.93014 | day\_hour\_0\_0 | -0.679634 |
| day\_hour\_6\_6 | -1.47603 | day\_hour\_19\_19 | 0.65914 |
| Bias | -1.45677 | day\_hour\_10\_10 | 0.633461 |
| day\_hour\_2\_2 | -1.40091 | day\_hour\_7\_7 | -0.548403 |
| atemp | 1.15169 | holiday\_False\_0 | -0.497366 |
| workingday\_True\_1 | -1.12128 | season\_3\_2 | -0.405169 |
| day\_hour\_1\_1 | -1.10465 | day\_hour\_20\_20 | 0.40351 |
| day\_hour\_13\_13 | 0.976793 | season\_4\_3 | -0.355207 |
| day\_hour\_17\_17 | 0.973042 | day\_hour\_9\_9 | 0.338015 |
| day\_hour\_16\_16 | 0.937769 | day\_hour\_23\_23 | -0.310924 |
| day\_hour\_15\_15 | 0.933247 | weather\_2\_1 | -0.280124 |
| day\_hour\_14\_14 | 0.916963 | weather\_1\_0 | -0.256721 |
| day\_hour\_12\_12 | 0.906708 | season\_2\_1 | -0.227176 |
| temp | 0.881583 | workingday\_False\_0 | -0.226315 |
| season\_1\_0 | -0.840232 | day\_hour\_21\_21 | 0.216453 |
| day\_hour\_18\_18 | 0.832261 | windspeed | -0.13126 |
| day\_hour\_11\_11 | 0.807115 | day\_hour\_8\_8 | 0.067625 |
| weather\_3\_2 | -0.738155 |  |  |

Appendix 3.4 Feature Weights - Poisson Regression for *REGISTERED*

|  |  |  |  |
| --- | --- | --- | --- |
| **Feature** | **Weight** | **Feature** | **Weight** |
| day\_hour\_4\_4 | -2.65415 | holiday\_True\_1 | -0.507314 |
| day\_hour\_3\_3 | -2.36875 | season\_1\_0 | -0.49865 |
| day\_hour\_2\_2 | -1.75736 | day\_hour\_13\_13 | 0.473023 |
| day\_hour\_5\_5 | -1.70309 | day\_hour\_15\_15 | 0.429718 |
| day\_hour\_17\_17 | 1.26312 | workingday\_True\_1 | -0.395809 |
| day\_hour\_1\_1 | -1.25797 | day\_hour\_21\_21 | 0.337193 |
| day\_hour\_8\_8 | 1.25483 | day\_hour\_14\_14 | 0.334348 |
| day\_hour\_18\_18 | 1.19949 | day\_hour\_11\_11 | 0.320545 |
| Bias | -1.041 | day\_hour\_23\_23 | -0.316561 |
| day\_hour\_19\_19 | 0.885725 | season\_3\_2 | -0.287726 |
| day\_hour\_7\_7 | 0.777929 | day\_hour\_6\_6 | -0.279516 |
| day\_hour\_0\_0 | -0.75056 | humidity | -0.275596 |
| day\_hour\_16\_16 | 0.732419 | season\_2\_1 | -0.229003 |
| workingday\_False\_0 | -0.641257 | weather\_2\_1 | -0.213584 |
| day\_hour\_9\_9 | 0.640847 | day\_hour\_10\_10 | 0.201698 |
| weather\_3\_2 | -0.61334 | weather\_1\_0 | -0.178899 |
| day\_hour\_20\_20 | 0.569309 | windspeed | -0.102706 |
| day\_hour\_12\_12 | 0.534889 | day\_hour\_22\_22 | 0.095821 |
| holiday\_False\_0 | -0.52975 | weather\_4\_3 | -0.031241 |
| atemp | 0.517365 | season\_4\_3 | -0.021683 |
| temp | 0.511652 |  |  |

Appendix 3.5 Comparison of Models – Fast Forest Quantile for *CASUAL*

|  |  |  |  |
| --- | --- | --- | --- |
| **Performance Measure** | **0.25** | **0.5** | **0.75** |
| **Mean Absolute Error** | 22.751430 | 15.586803 | 20.925263 |
| **Root Mean Squared Error** | 41.692382 | 28.969914 | 31.134509 |
| **Coefficient of Determination** | 0.279476 | 0.652120 | 0.598192 |

Appendix 3.6 Comparison of Models – Fast Forest Quantile for *REGISTERED*

|  |  |  |  |
| --- | --- | --- | --- |
| **Performance Measure** | **0.25** | **0.5** | **0.75** |
| **Mean Absolute Error** | 67.012964 | 47.34384568 | 65.205650 |
| **Root Mean Squared Error** | 108.336499 | 75.53659848 | 86.876055 |
| **Coefficient of Determination** | 0.470374 | 0.742525248 | 0.659419 |

Appendix 3.7 Settings – Bayesian Linear Regression

|  |  |  |  |
| --- | --- | --- | --- |
| **Setting** | **Parameter Range** | ***CASUAL*** | ***REGISTERED*** |
| **Regularization weight** | 1 | - | - |
| **Lambda** | - | 1 | 1 |
| **Min lambda** | - | 1.00E-06 | 1.00E-06 |
| **Noise Fraction** | - | 0.1 | 0.1 |
| **Noise Variance For Uniform** | - | 0.001 | 0.001 |
| **Allow Unknown Levels** | TRUE | TRUE | TRUE |

Appendix 3.8 Settings – Neural Network

|  |  |  |  |
| --- | --- | --- | --- |
| **Setting** | **Parameter Range** | ***CASUAL*** | ***REGISTERED*** |
| **Is Initialized From String** | FALSE | FALSE | FALSE |
| **Is Classification** | FALSE | FALSE | FALSE |
| **Initial Weights Diameter** | 0.1 | 0.1 | 0.1 |
| **Learning Rate** | 0.01, 0.02, 0.04 | 0.01 | 0.01 |
| **Loss Function** | SquaredError | SquaredError | SquaredError |
| **Momentum** | 0.1 | 0.1 | 0.1 |
| **Data Normalizer Type** | Gaussian | Gaussian | Gaussian |
| **Number Of Input** | - | Features 45 | Features 45 |
| **Number Of Iterations** | 20, 40, 80, 160 | 160 | 160 |
| **Number Of Output Classes** | - | 1 | 1 |
| **Shuffle** | TRUE | TRUE | TRUE |
| **Allow Unknown Levels** | TRUE | TRUE | TRUE |

Appendix 3.9: Python Code for Calculating Benchmark

