**Bike Sharing Hourly Counts Information to Foresee Near Future Trend**

**1. Business understanding:**

* 1. The objectives of the business

This study is to identify the near future trend of bike sharing. Used Knowledge Discovery in Databases (KDD) methods and given the foreseeable trend, the business could understand the situation of bike sharing better, so that they would make good investment and adjustment to the bike sharing strategies and systems.

* 1. The situation

As known, bike sharing is getting more and more popular in recent years around the globe. There are massive of opportunities for bike sharing business. On the other hand, cycling could potentially reduce heavy burdens on public transports and be convenient for certain groups of people, and beyond. Bike sharing has peak and low hours and usages as well, such as morning rush hours, after work hours. Although the users of bike sharing are increasing and there are more and more opportunities for business, it would be better if the business could serve them even much better by inputting efforts on providing more supports, and attracting even more users. In addition, there must be popping up many other rivals in bike sharing market to defeat each other. Hence, it is good time to review the situation to seize the market shares.

* 1. Data mining objectives

In this data mining objectives, it is not going to find out that bike sharing could reduce traffic congestion. Ricci (2015) argued that there is no evidence on bike sharing reducing traffic congestion and any other environmental issues. Hence, I will not look for whether bike sharing affects other issues.

Also, the raw data in this study doesn’t contain the distance information, because users may consider to use the system based on short distance. Matrai and Toth (2016) indicated that bike sharing is only for the purpose of short distance trip, occasional travel, and it doesn’t provide alternatives for long distance commuters.

Instead, the objectives of this data mining are to determine what main fields are affecting bike sharing itself. O’Brien et al. (2014) suggested that differences between weekday and weekend usage are apparent, and peak usages at different parts of the day depend on the docking station, academic and workplace locations.

In addition, to see if environmental conditions are affecting bike sharing, such as weather condition, wind speed, humidity. Based on the data that come out with the prediction information, the business could make improvement to the user experience and to satisfy users’ enjoyment in using the services. Furthermore, it could potentially increase various groups of people in wilder range to use bike sharing with different purposes.

* 1. Project plan

The following is the relevant proposed time frame for conducting the project using BDAS (*Figure 1*) – *please do refer to Project Plan Gantt Chart attached (Project\_plan.xlsx) in the zip file for greater details*. The entire plan has total 20 days to complete. I break down the plan into two separated streams: the plan start and duration; the actual start and duration. Percent complete is the indicator comes along the way to help me monitor my progress.

|  |  |
| --- | --- |
| **Timeline** | **Item Progression** |
| 23/09/2018 – 25/09/2018 | Business understanding; Data understanding |
| 26/09/2018 – 26/09/2018 | Data preparation |
| 27/09/2018 – 28/09/2018 | Data transformation |
| 29/09/2018 – 30/09/2018 | Data-mining method(s) selection |
| 01/10/2018 – 01/10/2018 | Data-mining algorithm(s) selection |
| 02/10/2018 – 05/10/2018 | Data Mining |
| 06/10/2018 – 08/10/2018 | Interpretation |
| 09/10/2018 – 09/10/2018 | Feedback to the model |
| 10/10/2018 – 10/10/2018 | Refinement |
| 11/10/2018 – 11/10/2018 | Action |
| 12/10/2018 (DUE) | Submission |

*Figure 1: Brief Project Plan Extracted from Project\_plan.xlsx*

**2. Data understanding:**

2.1 The initial data

The data are collected by Fanaee-T and Gama (2013), who was one of the researchers under the Capital Bike Sharing System in Washington D.C., USA. The data have a variety of formats, which are publicly available by clicking [*HERE*](http://capitalbikeshare.com/system-data). For the purpose of this project and the software I use, CSV data format would be the best option, because the following software I use would help organize the data. If I use Excel instead, it would probably have incompatible issues with other software as we know that the Excel application itself incorporating many features.

The data come with two separate collection sets: one is bike sharing counts based on daily basis; the other one is bike sharing counts based on hourly basis. I try to sum up all the hours into a day and it is exactly the same as the daily data. In addition, hourly data will be much more in details rather than the daily one. Hence, I choose the hourly basis for the project.

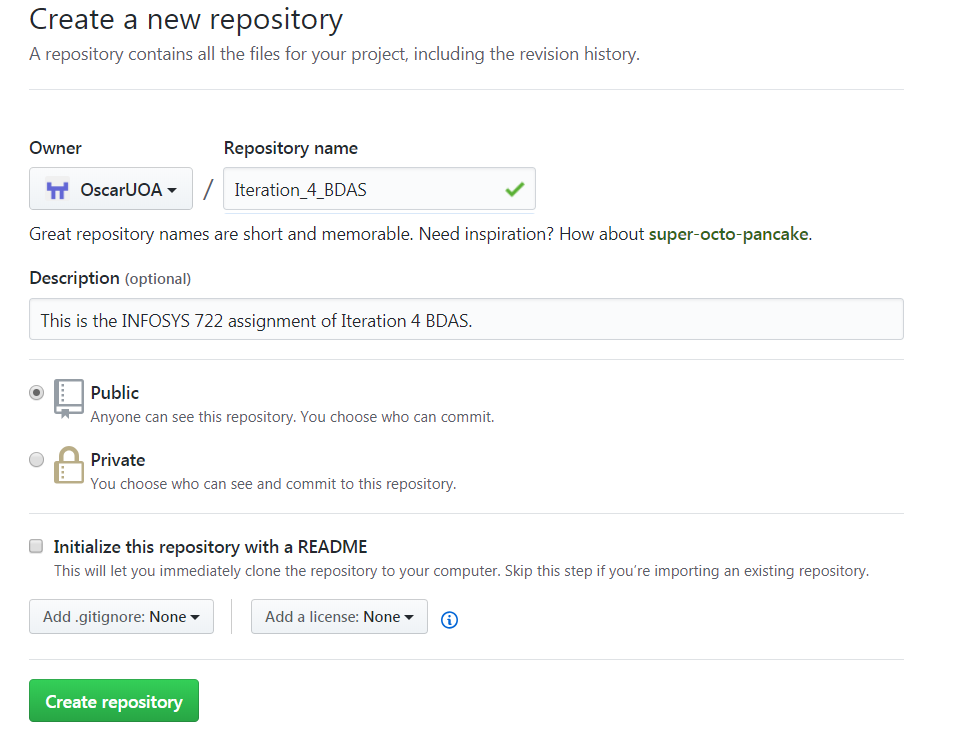
2.2 About the data

The data are collected within two-year historical log corresponding to years 2011 to 2012 from Capital Bike Sharing System in Washington D.C., USA. Also, data contain weather conditions, precipitation, day of week, season, hour of the day, etc.

Most importantly, it has the counts of users in every single hour per day. There are total 17379 hours. A glance at the CSV file from top to down, there is no missing values spotted. As known, CSV contains the raw data with comma to separate themselves. It doesn’t have powerful feature as compared to Excel. The good thing is that CSV file can be associated and compatible with many different free software and tools. Hence, conclusion of the existing data are good to go for further processes.

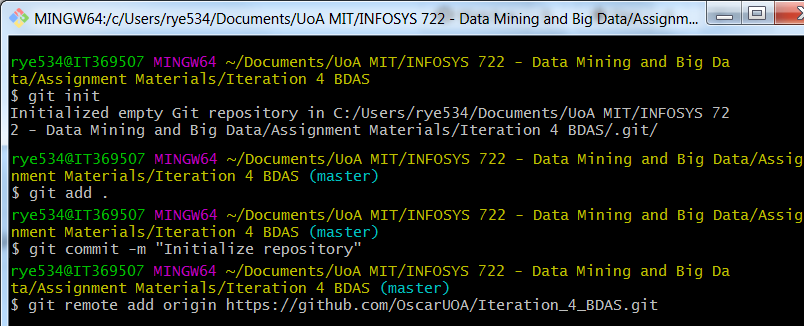
2.3 Explore the data

The data file is in CSV format, I then will duplicate a new one CSV data file as a back-up file in case the unexpected issues that damage my source file. Now, I am going to use the combination of software to manipulate the data. To begin with, I need to create a new remote repository so that I could synchronize all my local work to the remote repository. Logon to my GitHub account, create a new repository and type in the following information (*Figure 2*). Click Create repositoryto complete.



*Figure 2: Create New Online Repository in GitHub*

Then, I create a local repository that contains all my work for this study of Iteration 4 BDAS*.* Start git command prompt, and type in the followings to initialize the folder, add the source files, commit and put first comment (*Figure 3*) and click *Enter* to add the work into remote repository. For the first time to do so, a prompt will ask for username and password to authenticate the login. The GitHub now should have all the files that are the same as my local.



*Figure 3: git Initialize the Repository and Add Source Files remotely*

Next, start *AWS* virtual machine and login to it by using PuTTY – *this should be done in the lab and I am not going to repeat the installation steps here.* Type in the following commands to clone the remote repository to my virtual machine (*Figure 4*). I can see the folder Iteration\_4\_BDAS that is cloned successfully.



*Figure 4: Clone the Repository into the AWS Virtual Machine*

Now, all the folder structures and synchronization work have been done and I can start the real work. Type jupyter notebook and click *Enter,* I copy the link (- *note to change the IP address that indicates in my AWS instance*), paste into my browser and locate the workspace of juptyer. Click *New* and select Python 3 to create a new file so that I could put my scripts on it. First of all, I change the file name to Explore the entire data – *please note that all the script files will be put under different folders accordingly to the section I work on.*

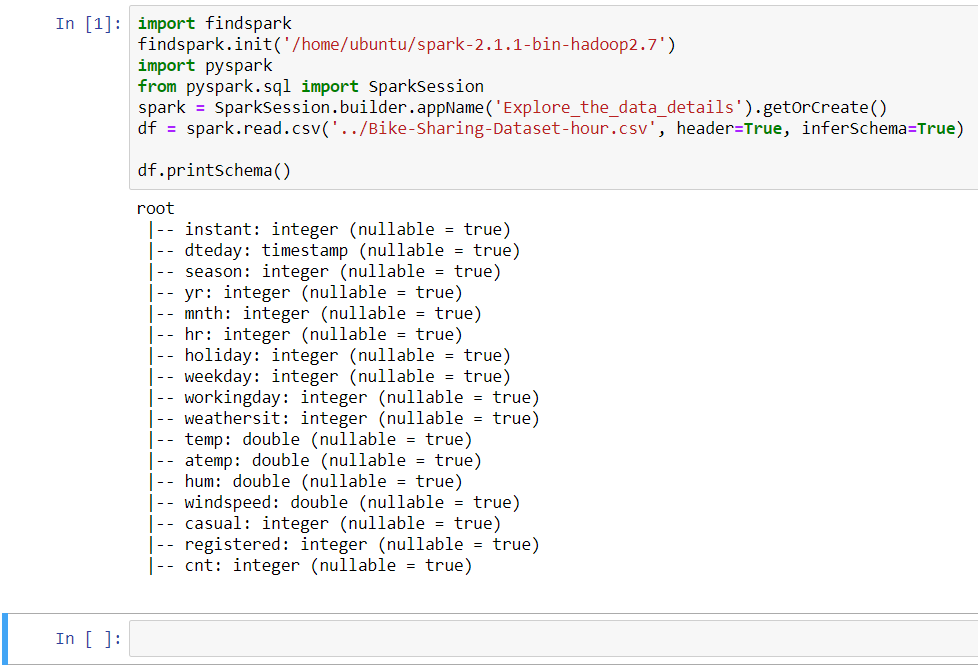
Run the following scripts (Figure 5), and I got total 17379 records which is exactly same as the raw CSV data file. Hence, I can confirm that the CSV file in repository is successfully and correctly placed in GitHub.



*Figure 5: Retrieve and Display Whole Dataset*

We take a look at the *Figure 5* – *scroll the dataset from top to down*. It is noticed that the casual, registered and cnt numbers are getting bigger when the hr lines during 8am to 19pm, and the cnt is the total of casual plus registered. Also, workingday falls on 1 (- working days). If it is not a working day but it is holiday, the counts drop sharply. This is interesting fact that the bike sharing is potentially suit for those people are running in rush hours. However, it needs to have further assessment to confirm this.

In order to understand the data more, I then get the type of the dataset (*Figure 6*). Most of them are type of integers except dteday that is timestamp, and temp, atemp, hum and windspeed are doubles. Values are allowed to be nullable.



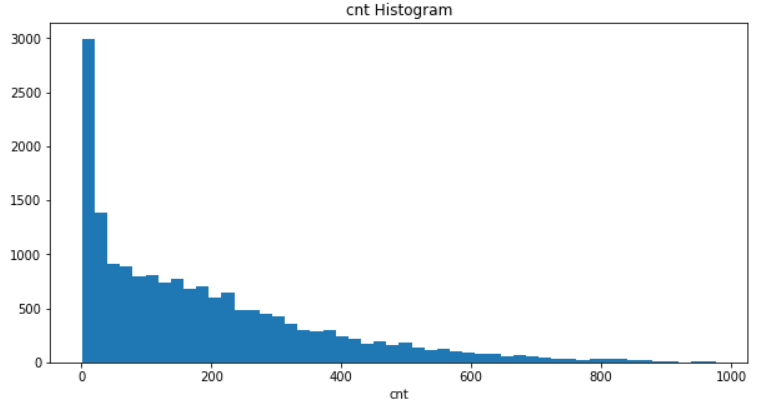
*Figure 6: Check Data Type*

I then summarize the dataset as follows (*Figure 7*):

|  |  |
| --- | --- |
| **Fields** | **Explanation** |
| instant | Record index |
| dteday | Date range from 1st Jan 2011 to 31st Dec 2012 |
| season | 1, 2, 3 and 4 that represent spring, summer, fall and winter respectively |
| yr | 0 is 2011, 1 is 2011 |
| mnth | 1 to 12, where 1 is Jan, 2 is Feb, 3 is Mar, etc. |
| hr | 0 to 23 in 24-hour format, where 0 is midnight, 1 is 1 am, 2 is 2 am, etc. |
| holiday | 1 and 0, which represent that 1 is holiday and 0 is not holiday |
| weekday | It starts with 0 that represents Sunday, 1 that represents Monday, 2 that represents Tuesday, etc. |
| workingday | It has 1 that is working day and 0 that is not working day |
| weathersit | 1 is clear and sunny, 2 is mist and cloudy, 3 is light snow and light rain, 4 is heavy rain, ice pallets, thunderstorm and severe weather |
| temp | Normalized temperature in Celsius. The values are divided to 41 (max) |
| atemp | Normalized feeling temperature in Celsius. The values are divided to 50 (max) |
| hum | Normalized humidity. The values are divided to 100 (max) |
| windspeed | Normalized wind speed. The values are divided to 67 (max) |
| casual | The count of casual users |
| registered | The count of registered users |
| cnt | The count of total rental bikes including both casual and registered |

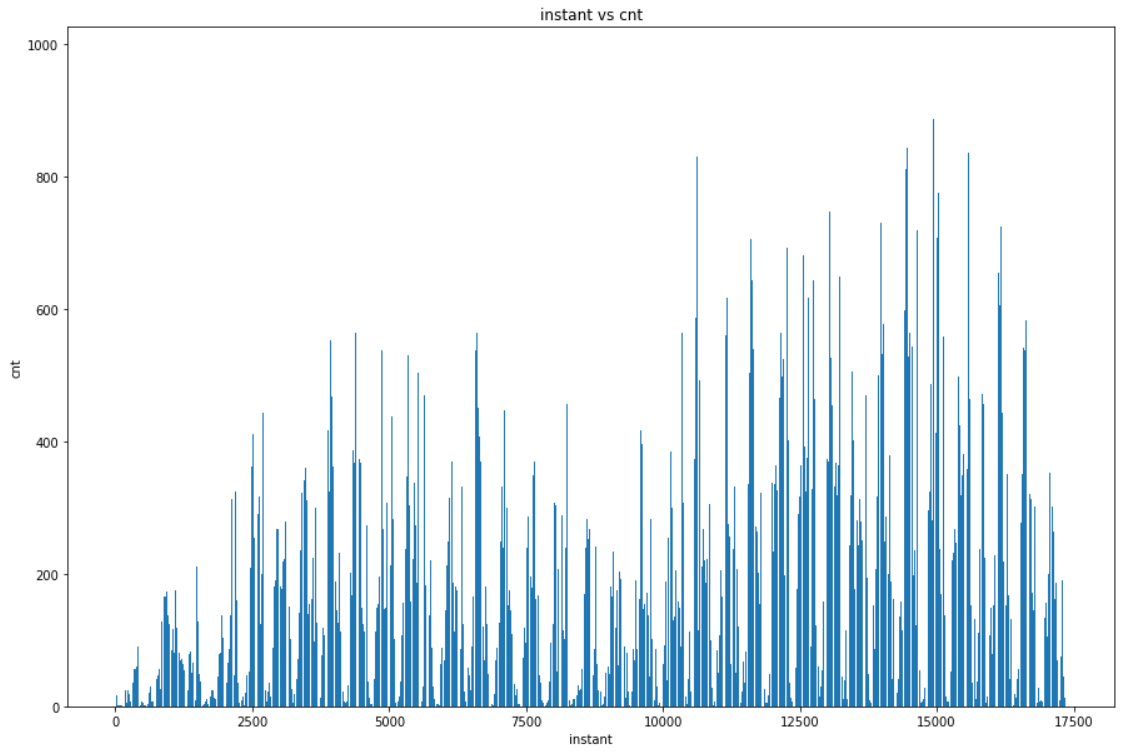
*Figure 7: The Data Fields Explanation*

I code the following for cnt and build the histogram as shown below (*Figure 8*). It is not symmetric one but right skewed.



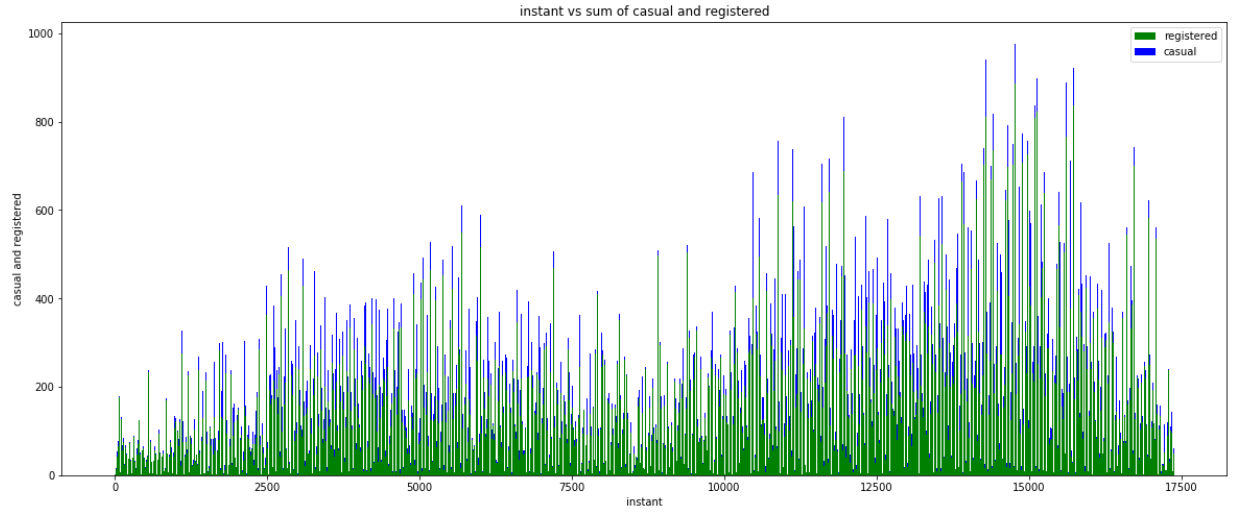
*Figure 8: cnt Skewed Right Histogram*

Plot the instant against cnt, I got results below (*Figure 9*). It indicates that the cnt is increasing from left to right. So, I could expect the growth of bike sharing is stably going up with a seasonal effect, because instant also represents the time frame of 2011 to 2012. However, there are a number of concaves. I couldn’t tell from here, I will analyze deeper with different aspects of columns to check the reason behind.



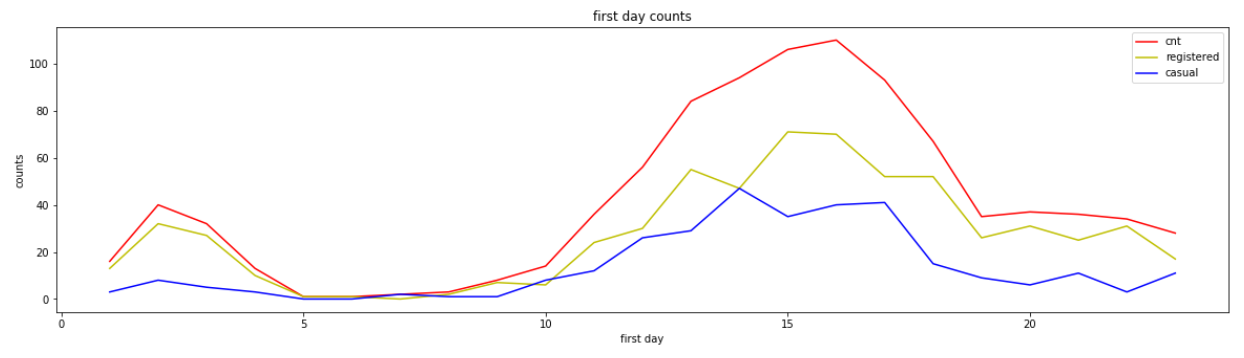
*Figure 9: The Bar Chart of instant vs cnt*

As known, the cnt is the sum of casual and registered. Hence, I put casual and registered counts as the total cnt counts across the entire duration (*Figure 10*). From the graph, it illustrates that the main portions towards the total cnt is always the registered in green color. Casual in blue color occupies small amount and it also tells that this group of users is not certain.



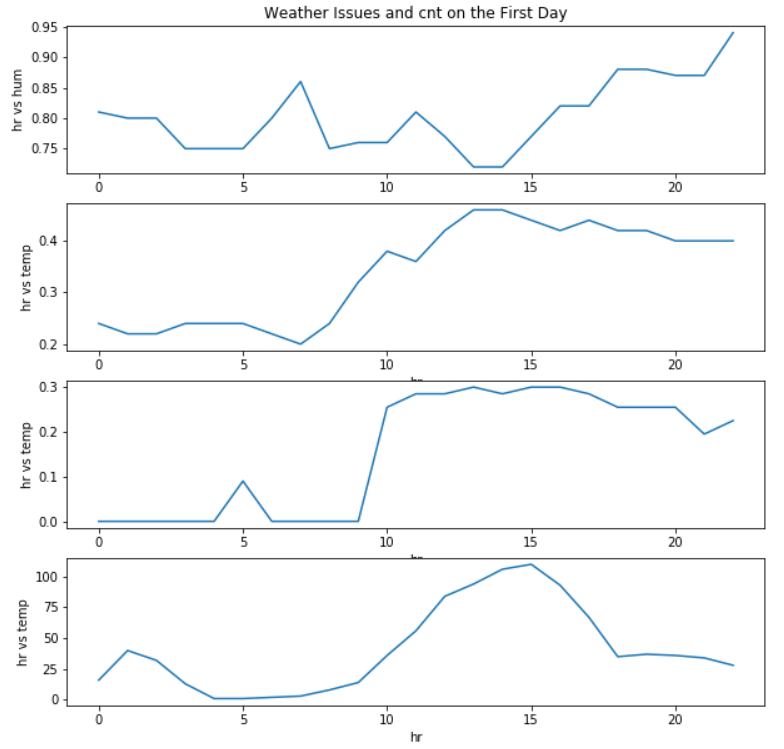
*Figure 10: Counts of casual and registered against cnt*

I then plot the very first day counts as follows (*Figure 11*). It indicates that the casual is still one of the uncertainties towards the total counts. The registered is closely aligning with cnt as we know it is the main portion to cnt.



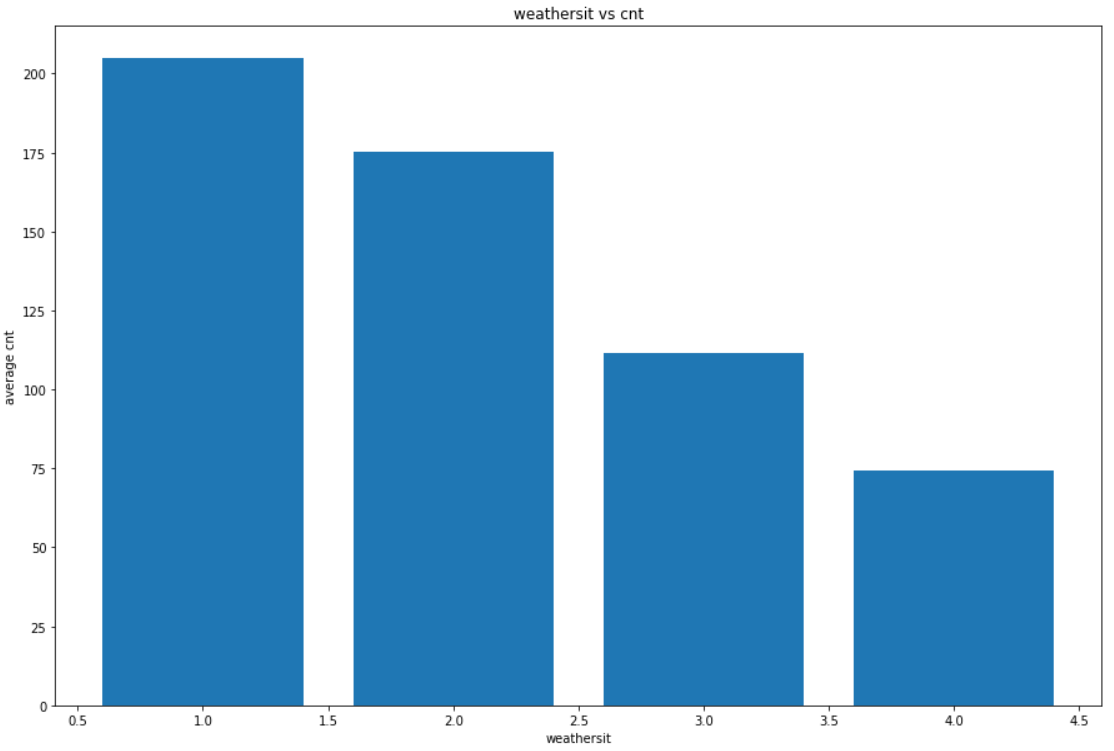
*Figure 11: First Day Counts of casual, registered and cnt*

I try the weather issues and plot them together to compare with the cnt below (*Figure 12*). The interesting thing here is that when temp and windspeed go up around the afternoon hours, the total cnt goes up. However, the cnt goes down after 16pm, temp and windspeed still remain about the same level. For hum, I couldn’t see more details from here, because it doesn’t change much.



*Figure 12: First Day Weather Issues against cnt*

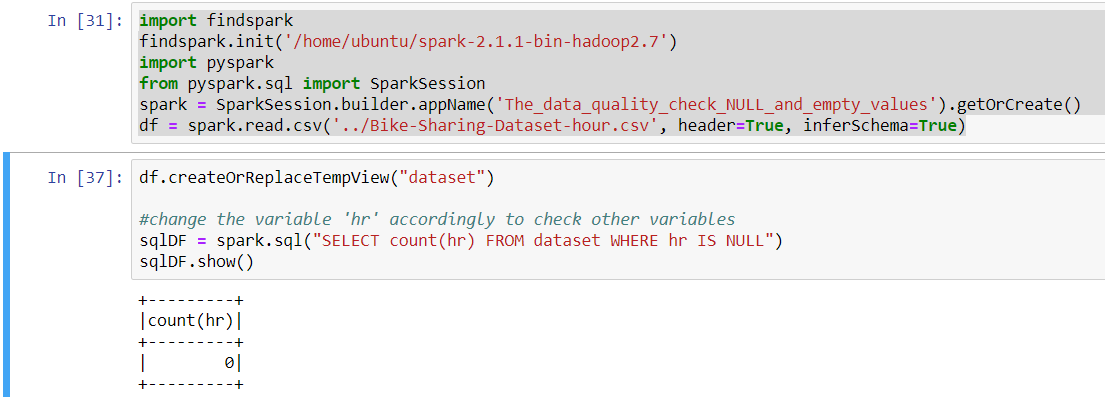
The weathersit seems to be involved with the cnt, because the more the cnt is, the less the weathersit is (*Figure 13*). This is reasonable as we know that weathersit represents the weather condition. In a very good day, there will be more people using the bike. Otherwise, there will be less in cnt.



*Figure 13: weathersit against cnt*

2.4 The data quality

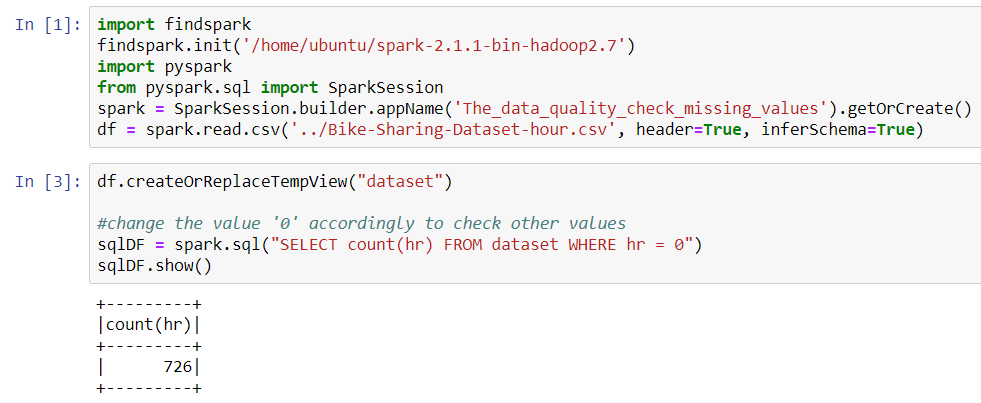
In data quality step, I am going to check if there is any null or empty values of my dataset. I use hr as an example to show how I am going to code to count the nullor empty values (*Figure 14*) – *please change the variable accordingly to check others.* As a result, 0 return which means there is no nullor empty values.

**

*Figure 14: Count Column with Null or Empty Values*

However, I realize that 17379 is not the total amount of hours in two years (2011-2012). 2011 is a normal year, whereas 2012 is a leap year. So, that will be total 731 (365 plus 366) days and 17544 hours instead of 17379 hours.

Now, I am going to count the hour of 0, 1, 2 till 23 one by one and check if it really misses some hours in between. Execute the script, and I have following results in hr of 0 (*Figure 15*). I can see the total count is 726, which indicates that there is missing values of hour 0. Because the total days should be 731. Moreover, I check the rest of the hours from 1 to 23 and find out there are missing values as well.



*Figure 15: Check Missing Values in hr*

Finally, I use different Python libraries to query the entire raw data from top to down again. It seems to be intact. In addition, all data are numeric without any strings or special characters or unknown formats – *I will perform the validation to confirm if all data are numeric in data clean step*. One of the columns should be in date-time format which is the dteday. Other than that, data quality is good for further processes.

**3. Data Preparation:**

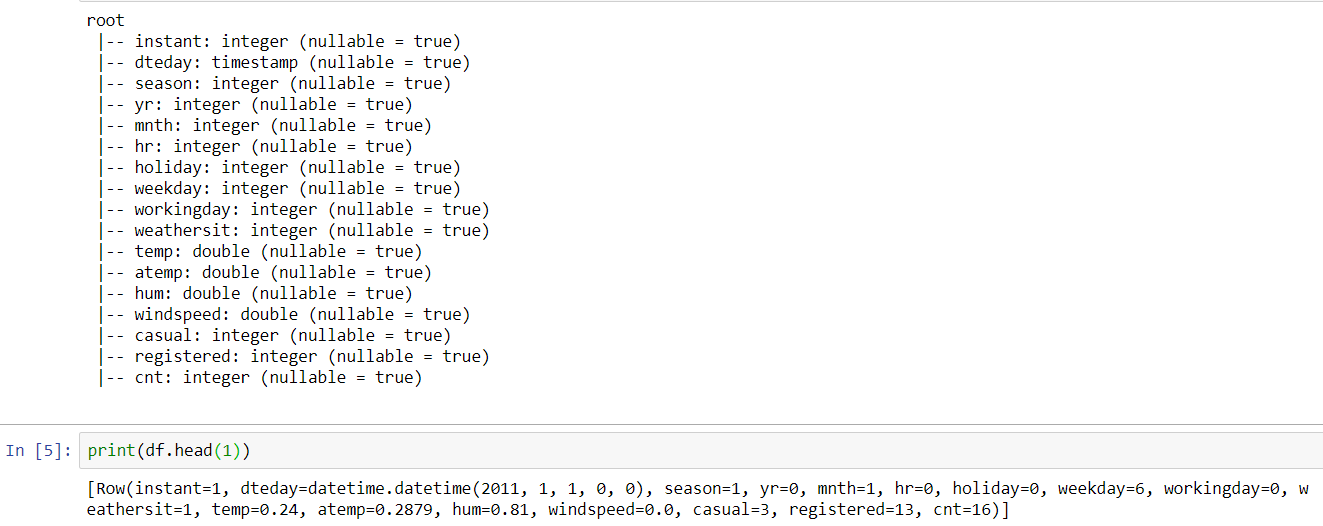
3.1 Select the data

I then continue to use a series of tools to complete the task. Those tools include GitHub, AWS (EC2/AMI), Jupyter, PySpark and Spark. Prior to this iteration, I have already installed them and in the previous steps I have been using some of them.

My steps here are to clean, construct, integrate and format the data. After finish the steps, then I will continue to pull out the processed new data to do the statistics check. Visualize the data in Jupyter workspace again, which will give a brighter view for me to analyze the full complete data clearer.

At the end, I will use Jupyter workspace and further the data mining processes. Now, I am going to use Jupyter to complete the data preparation steps. The data selected for this study is the CSV raw data file.

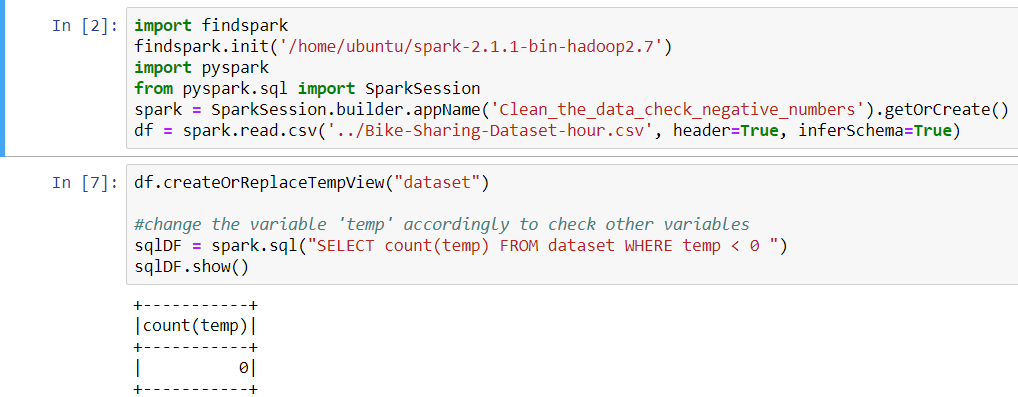
As I have imported the data into GitHub in step 2.3 and synchronized it to my AWS virtual machine, I will then check the type of the data if their format has been established correctly during the importing and synchronizing processes.I may have done this step earlier, however, this time I will retrieve more details toconfirm and show all the data types and their specific amount of rows (*Figure 16*). I can see that the data types are not changed even though I’ve done tests on the CSV, which is confirmed that the data processing won’t change the original file. The types are still integer for most of the fields, timestamp for dteday, and double for temp, atemp, hum and windspeed. They are allowed null values inputting. I randomly print one of the rows and check, which are all good.



*Figure 16: Inspect the Data Types*

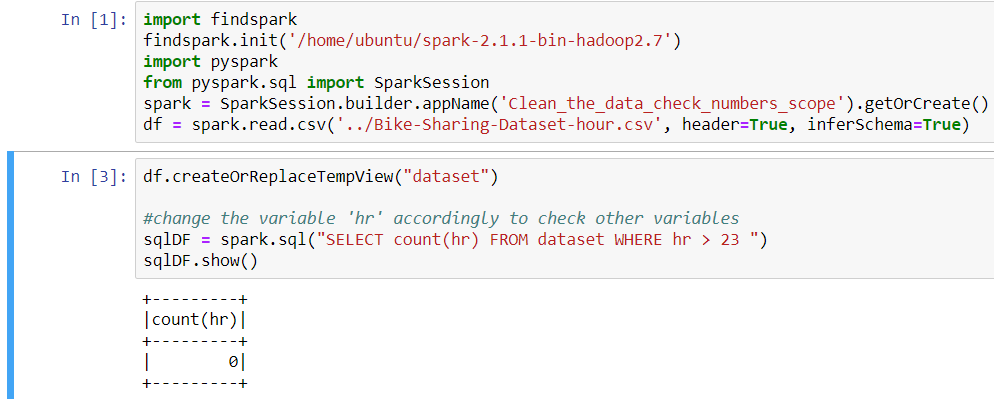
3.2 Clean the data

In this step, I am going to check the variables and clean all the empty or null value if there is any – *I’ve checked on 2.4 and found there is not any empty or null value*. In addition, I will check if the values and numbers of the fields are correct. For example, check if any column has negative numbers (*Figure 17*). Return 0, which means there is no negative numbers encountered.



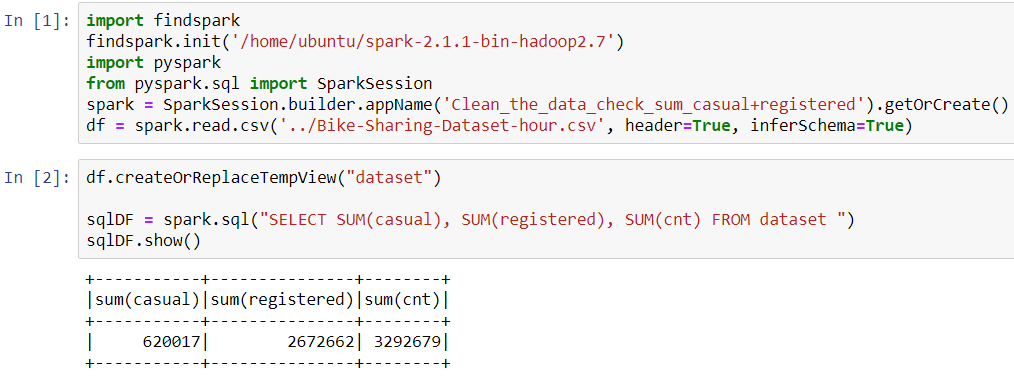
*Figure 17: Check Negative Numbers*

Then I will check if hr column has more than 23 – *hour starts at 0 and ends at 23*, if season has more than 4, and if mnth has more than 12 (*Figure 18*). Return 0, which means the hr has no more than 23 hours and result is correct.



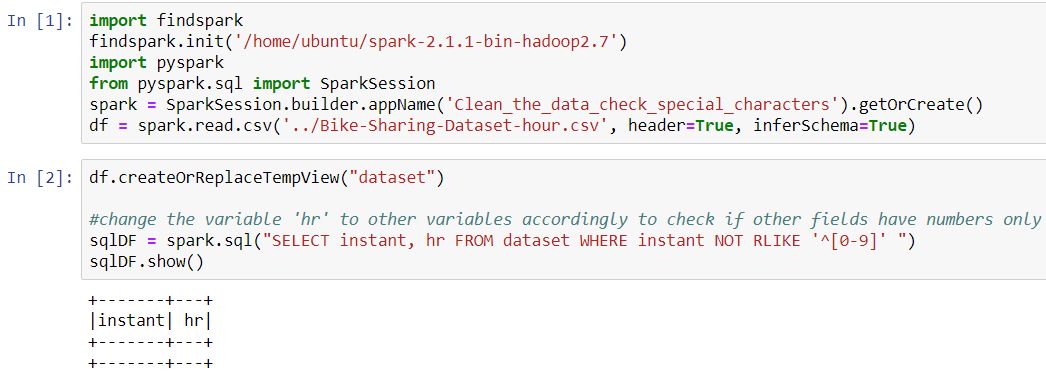
*Figure 18: Check Scope of Numbers.*

In addition, sum the total casual 620017 and registered 2672662 numbers, which should be equal to the summation of the total cnt numbers 3292679 (*Figure 19*)*.*



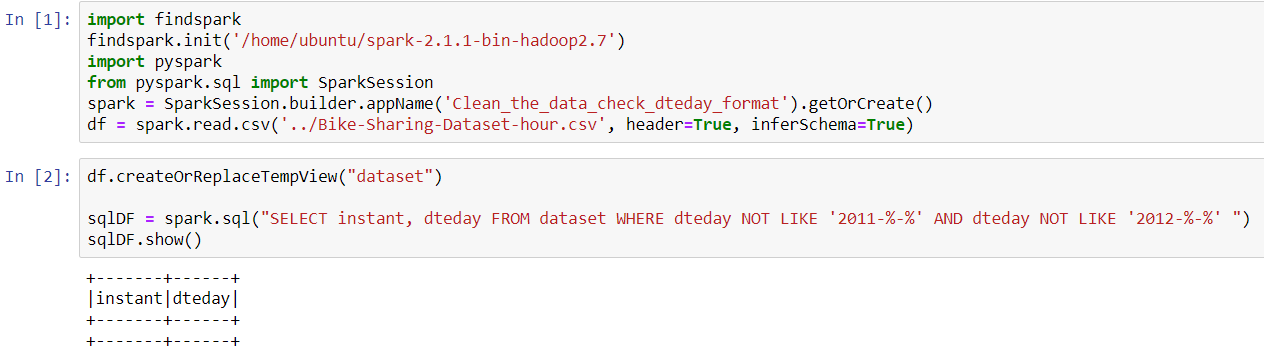
*Figure 19: Summation of casual, registered and cnt*

Secondly, I will query out all the NULL values or characters values that are not supposed to be existed in my data set. I use hr as an example as follows (*Figure 20*). As a result, there isn’t any invalid data. Continuously, I try other fields and find out there isn’t any invalid data either.



*Figure 20: Validate Values of hr Field*

Last but not least, the date format is a little different from other fields. So, I change my script to validate the dteday fields below (*Figure 21*). I try to get dteday values not with the format of ‘2011-%-%’ and ‘2012-%-%’. If there is return results, it means the value is wrong. However, there is no return results and dteday field and values are eligible.



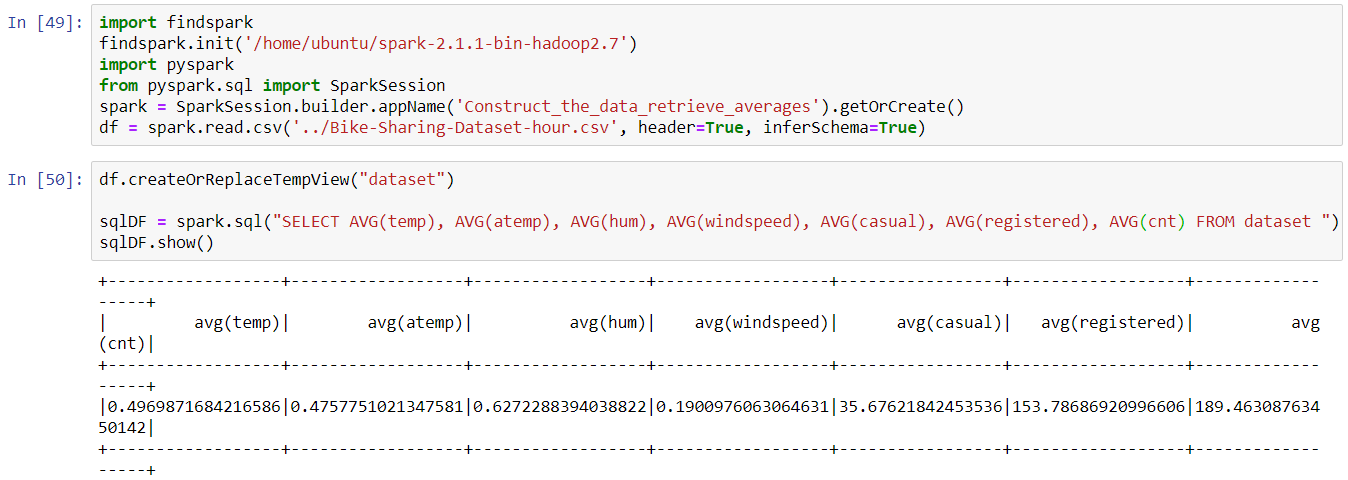
*Figure 21: Validate Values of dteday Field*

3.3 Construct the data

As in previous step 2, I have mentioned there are 165 hours missing because total hours are 17544 instead of 17379. So, I will add back those 165 (17544 – 17379 = 165) hours into my two years data set.

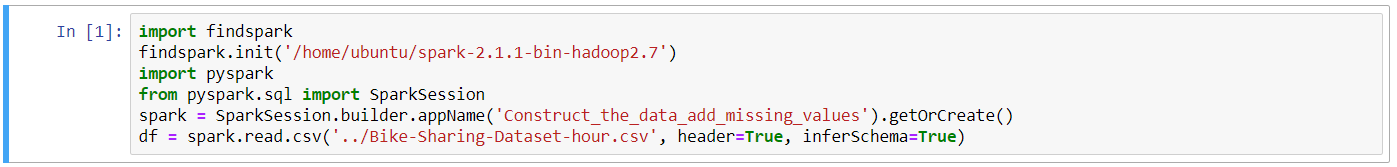
First of all, I divide this adding missing data process into monthly basis, which mean that there are 24 months in total and 24 adding processes. So, I will demonstrate how to add missing data in the first month that is January 2011 here. Then the adding missing data for rest of months, I will just put all the scripts I create under Jupyter folder Construct\_the\_data*.* Hence, it should be able to add all missing data by simulating the first month to create other 23 months missing data, because I don’t want to occupy too much space in this report by repeating the same processes.

To begin with, I use scripts to get all the average values of fields in temp, atemp, hum, windspeed, casual, registered and cnt (*Figure 22*). I will use these average values by inserting them back to the missing values later.



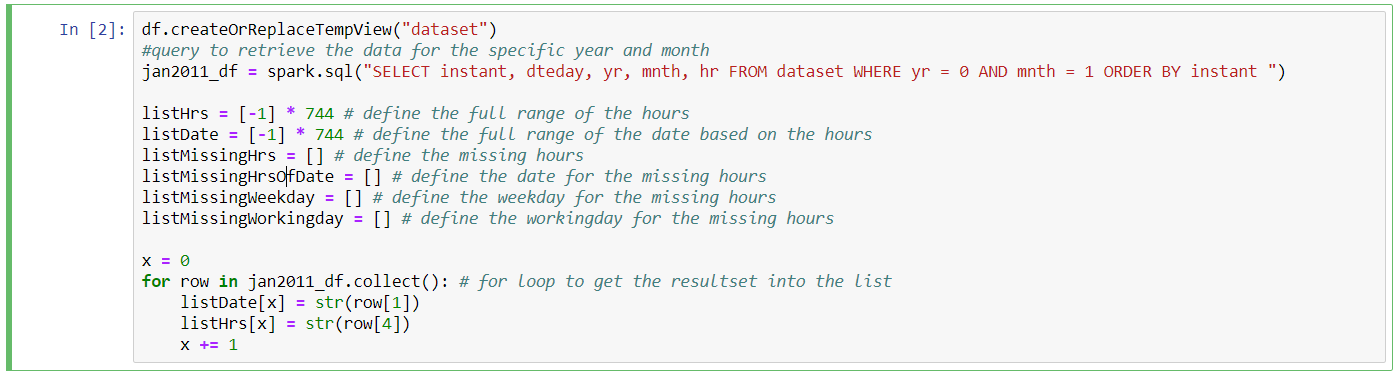
*Figure 22: Average Values of Several Fields*

Then in Jupyter, I import spark and pyspark libraries to connect to the CSV file (*Figure 23*).



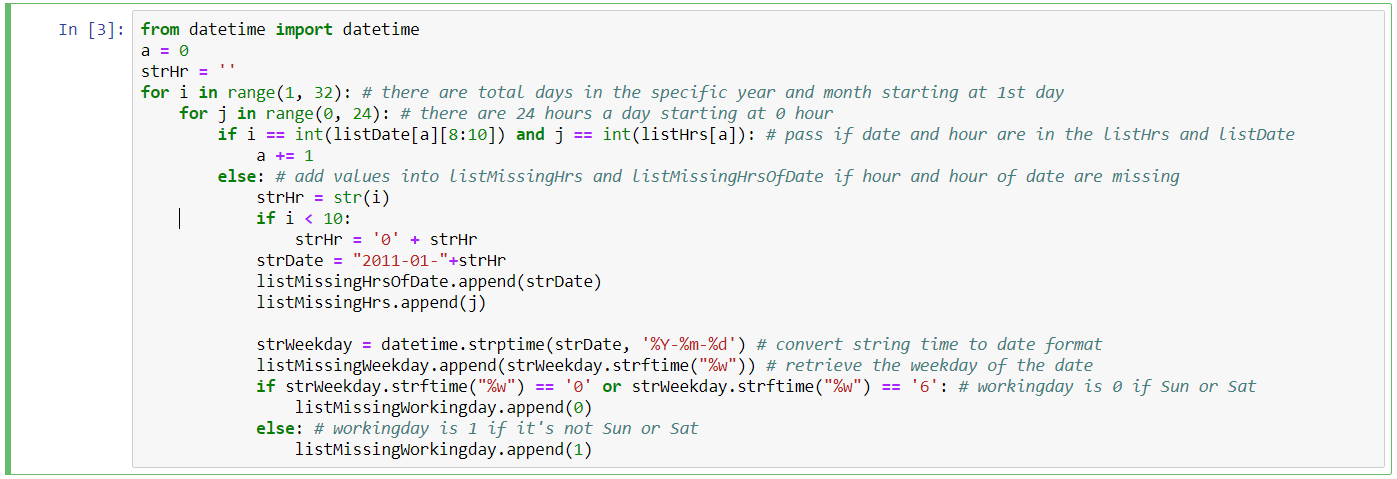
*Figure 23: Connect CSV File*

Next, I make a query to retrieve the results, define all the variables and put the results into the variables (*Figure 24*).



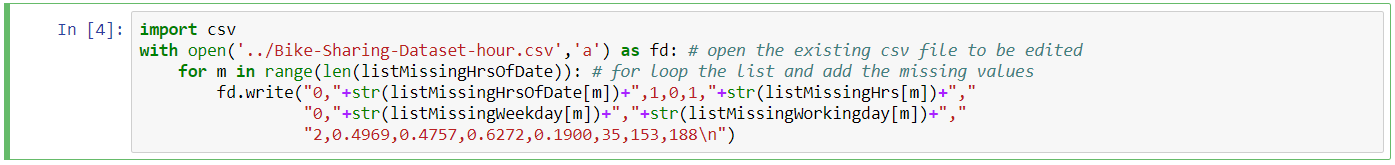
*Figure 24: Query Results and Assign*

After assign the results into variables, I will use the for-loop to produce the missing values. Also, during producing the missing values, I am differentiating the weekday and workingday values by using Python library of datetime, because datetime is especially handling all kinds of datetime issues and formats (*Figure 25*).



*Figure 25: Store Missing Values and Handle Datetime Values*

Furthermore, after producing the missing values, I then insert these missing values back into the CSV file (*Figure 26*). To note that, I put instant as 0, and I will re-order the sequence for all data later. So, this 0 is just to do the filter easier later because the existing ones have from 1 to 17379.

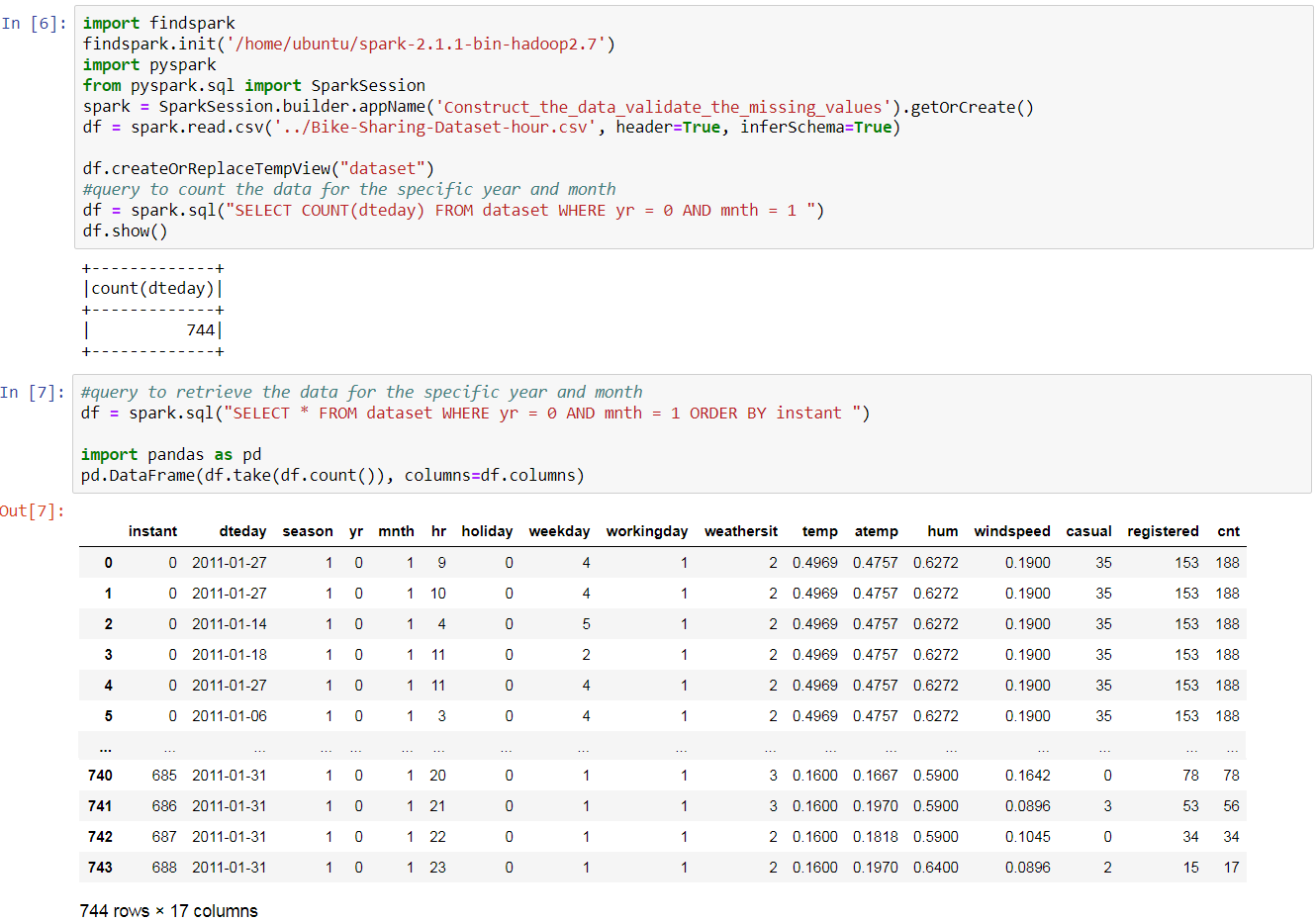


*Figure 26: Insert Missing Values in CSV File*

Also, I put 1 for season as January is in first quarter. We know that yr is 0, mnth is 1. Holiday is 0 that means not a holiday, because there are only 165 missing values that won’t really seriously affect the entire data even though the day is actual holiday. Weathersit is just taken the mean value that is 2.

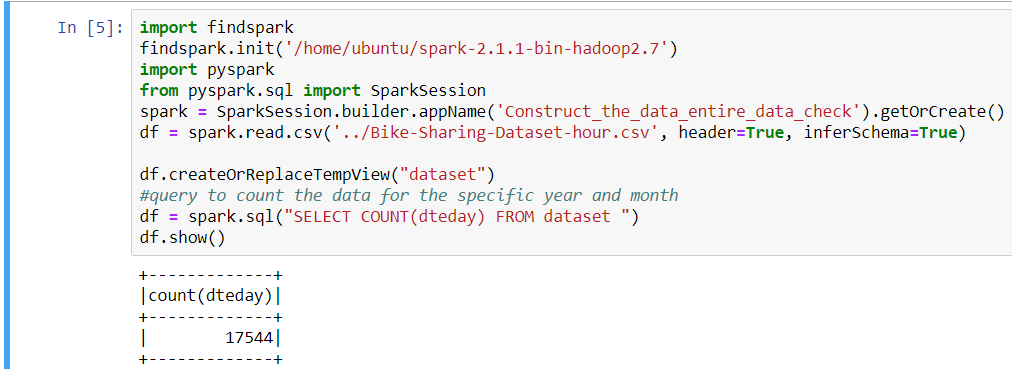
For temp, atemp, hum windspeed, casual, registered and cnt, I have already got the averages earlier in this section, and I take only 4 decimal places of the values instead of all decimal places. Run the entire script, the missing values will be appended at the end of the CSV file.

Last but not least, I do a full query check to confirm the missing values if they have been added. First to check the total counts then secondly query the entire January 2011 data (*Figure 27*). As we can see, the total results are 744 which mean that there are 744 hours in January 2011. Hence, the insertion is correctly done.



*Figure 27: Validate the Missing Values*

I then continue to insert the missing values for the rest of years and months, and will skip the writing for those repeated steps. As I mentioned, please find those scripts to generate the rest missing values under folder Construct\_the\_data. Finally, I do the entire data check (*Figure 28*). The total number is 17544 which represents all hours from 2011 to 2012 correctly.



*Figure 28: The Complete Data Set*

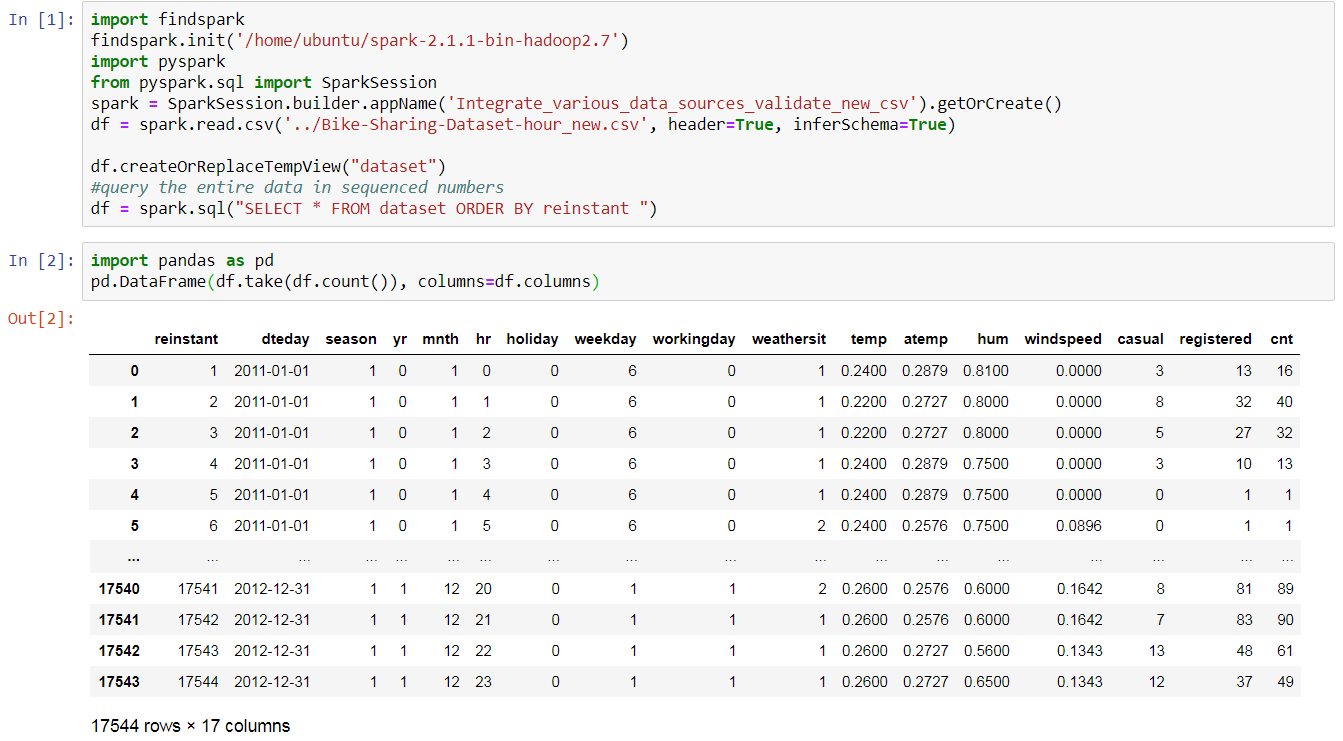
3.4 Integrate various data sources

Now, I am going to re-order the sequence for all the data. To start with, I need to create a new column that is called reinstant (*Figure 29*). Then I put all existing data into a new CSV file with new column reinstant – *new CSV file is Bike-Sharing-Dataset-hour\_new.csv*. It may be realized that x is the index counter increasing =+1 each time for the new column reinstant, so that it could have the correct sequence for all the data.



*Figure 29: New CSV with New Column reinstant*

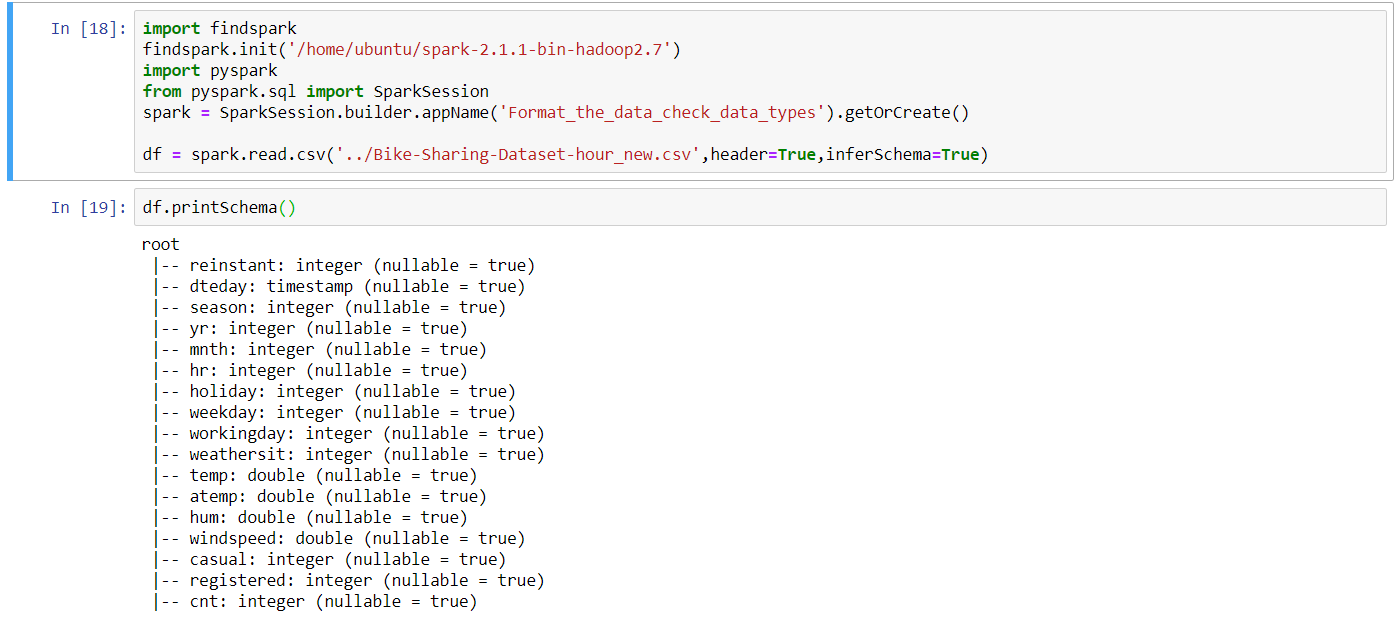
Now, we can see the reinstant column has exact 17544 data and it is in sequence correctly (*Figure 30*). We can now start using column reinstant instead of column instant.



*Figure 30: Validate the Entire Data Set*

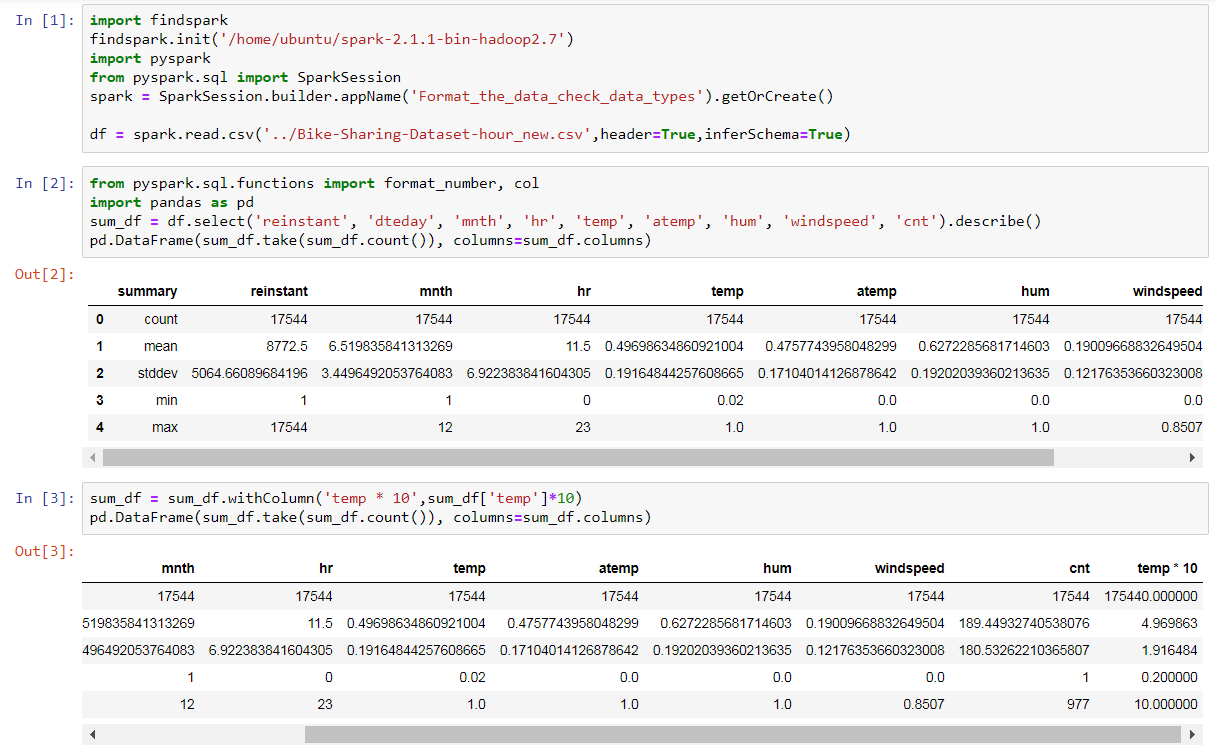
3.5 Format the data

In this step, I am going to check the data types of the new CSV file. If there is any type of the field that is not correct, I will format it. Use the following scripts to retrieve all field types (*Figure 31*). We can see that fields are mostly integers. Four of them, temp, atemp, hum and windspeed are doubles. Dteday is timestamp. Also, all of them allow nullable inputs.



*Figure 31: Retrieve All Data Types*

In order to check more details of the data and confirm the formats are correct, I am going to do some combination checks and assessments, such as get the mean and standard deviation, filter the numbers, search the date and time, because integers could be calculated by using “+, -, \*, /” and filter method, whereas string couldn’t. If the field’s format is not correct, there should be an error thrown. The following checks are proved that all data formats are correct (*Figure 32*). Therefore, there is no further data format required, and I conclude that all data types are correct. *Please refer to ‘Format the data - check more data format.ipynb’ for more detail checks.*

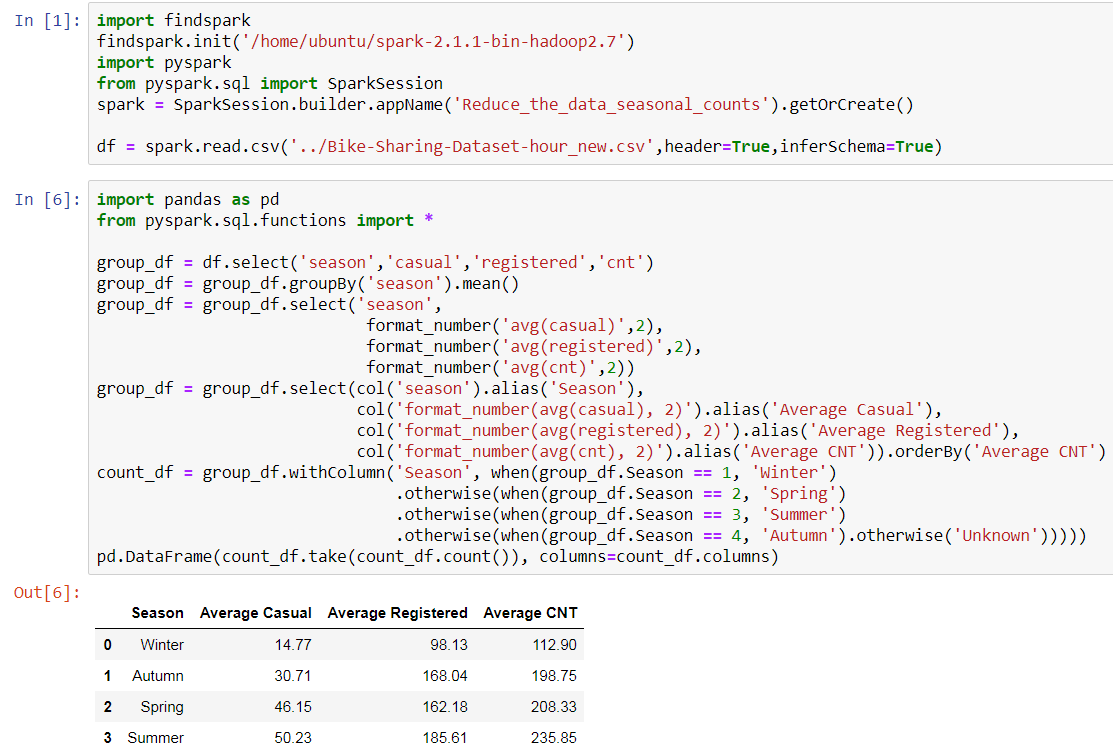


*Figure 32: Combination Checks for Data Format*

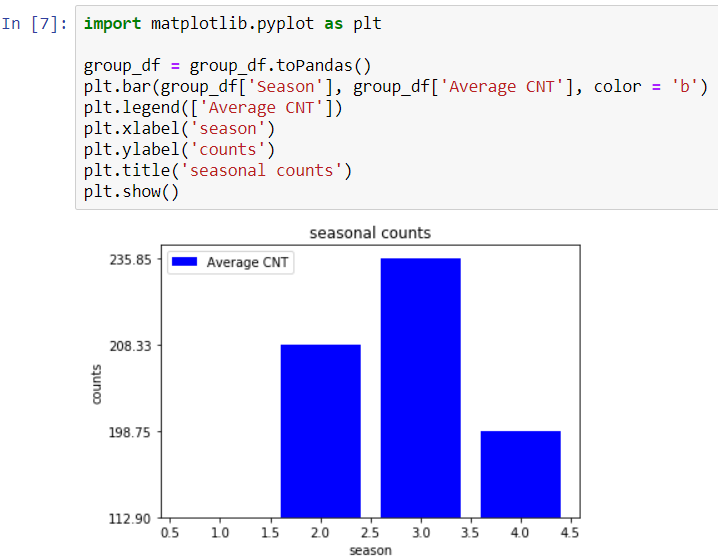
**4. Data Transformation:**

4.1 Reduce the data

In this step, let’s discuss if some data need to be removed or retained. I now group the season against the casual, registered and cnt (*Figure 33*) and produce the bar chart for seasonal counts (*Figure 34*). It indicates that more people use the bike sharing services in spring and summer than in autumn and winter.

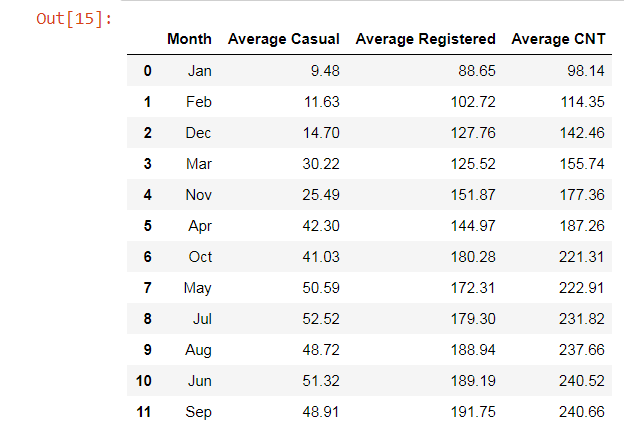


*Figure 33: season against casual, registered and cnt*

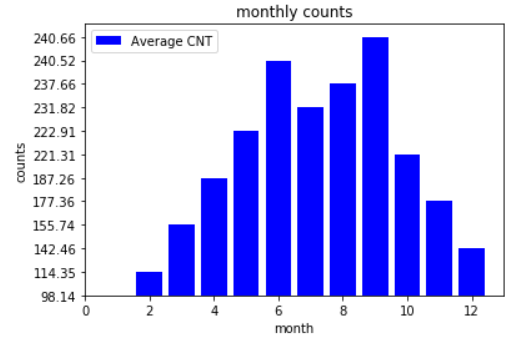
**

*Figure 34: Bar Chart of Seasonal Counts*

Again, I try to group the mnth against the casual, registered and cnt, which has similar counts during the seasonal periods (*Figure 35*) and similar patterns from the graph (*Figure 36*). For example, Dec to Mar have the small amounts, which is in cold weather (winter and autumn); whereas around Jun to Sep have the most counts, which is in warm weather (spring and summer). Compare between the seasonal and monthly counts, I find that I can get more details from monthly counts. Hence, I may just use mnth data rather than season, because using mnth data has more specified details as 12 months versus to 4 seasons.

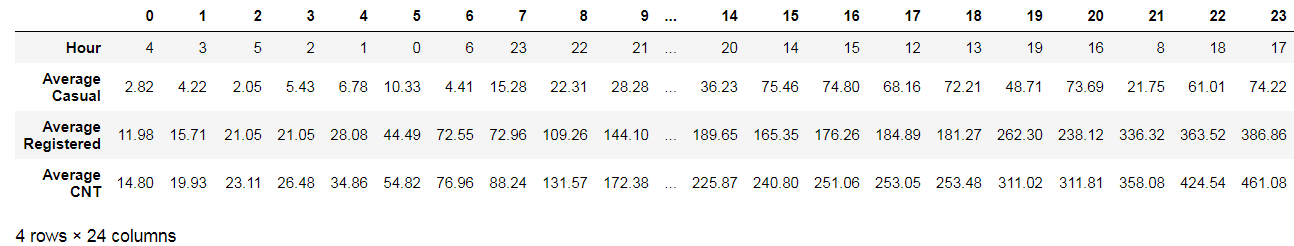
**

*Figure 35: mnth against casual, registered and cnt*

**

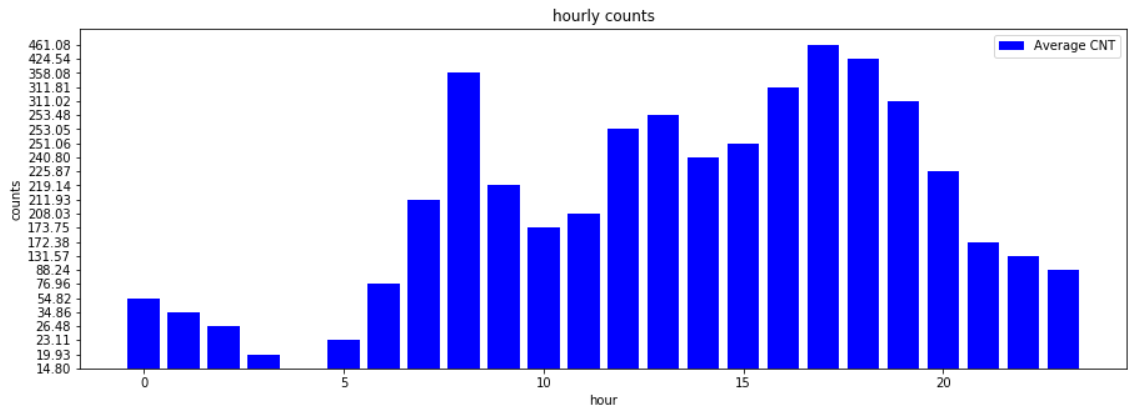
*Figure 36: Bar Chart of Monthly Counts*

In addition, I get the average of casual, registered and cnt to versus hr (*Figure 37*). I could see much more details now as it illustrates that during the peak hours, bike sharing services are occupied much more than the off-peak hours. There two peak hours in a day, the afternoon peak hours are more significant than the morning peak hours.



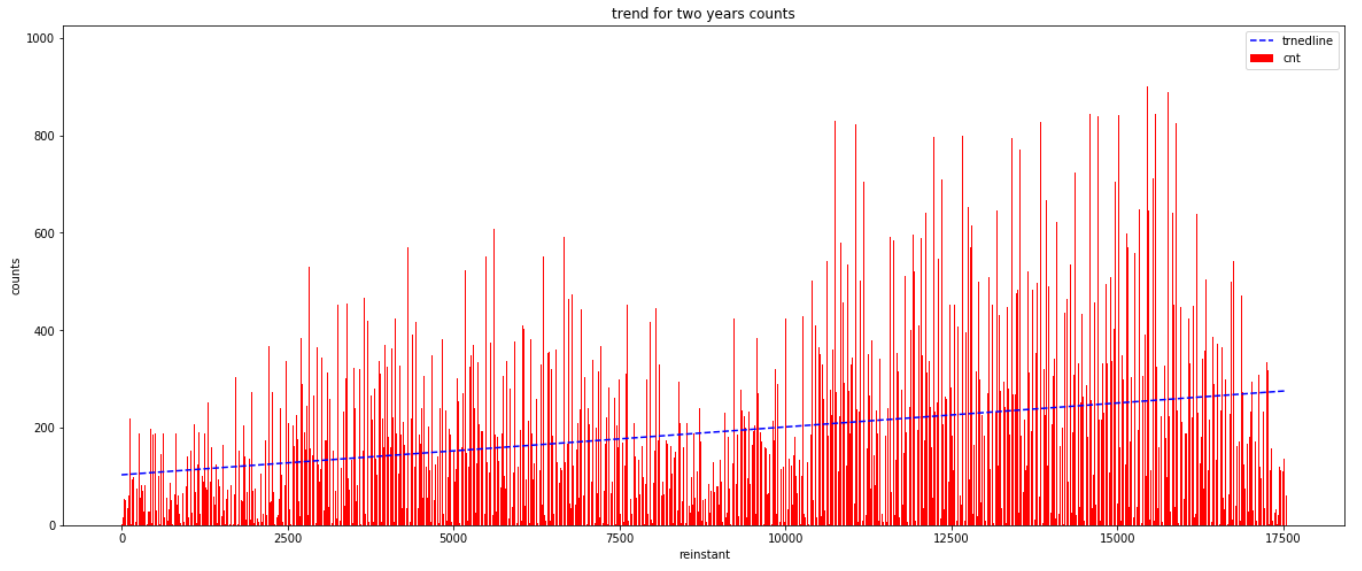
*Figure 37: hr against casual, registered and cnt*

Produce the hourly counts graph as follows (*Figure 38*). It’s interesting that even in a daily basis, it has the similar patterns as shown in seasonal and monthly graphs. Most importantly, using the hourly counts, which I could get even more details and vivid vision of the patterns.



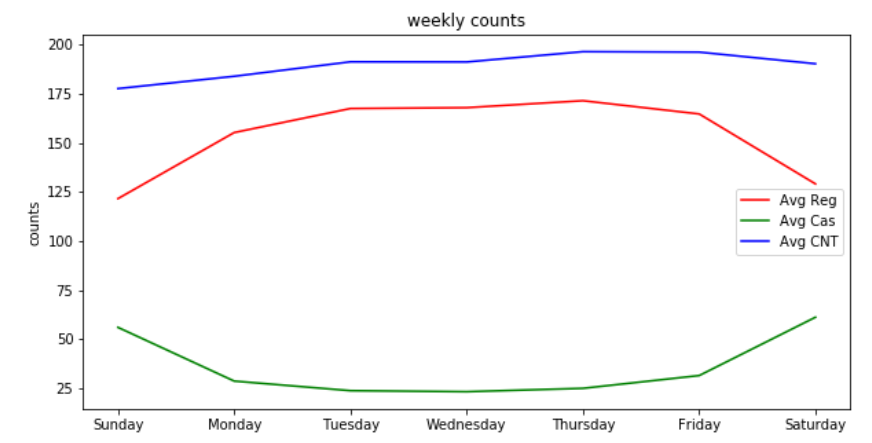
*Figure 38: Bar Chart of Hourly Counts*

Also, as known, cnt is the summation of casual and registered. So, I plot the cnt only to versus dteday – *I use reinstant instead of dteday here because reinstant represents dteday as in time series and easier for calculation* (*Figure 39*). I can see much clearer details from this graph. The graph interprets that more people use the services during May to Oct than other months. The counts have the lowest around the Dec and Jan. It is with the highest counts in the middle of the year around May to Oct. Then it starts decreasing and ends at Dec. It seems to be a usual pattern for bike sharing business, because this pattern applies to two consecutive years. The important thing is predictable that the trend is going up. It may be expected it will get more users in the following years after 2012.



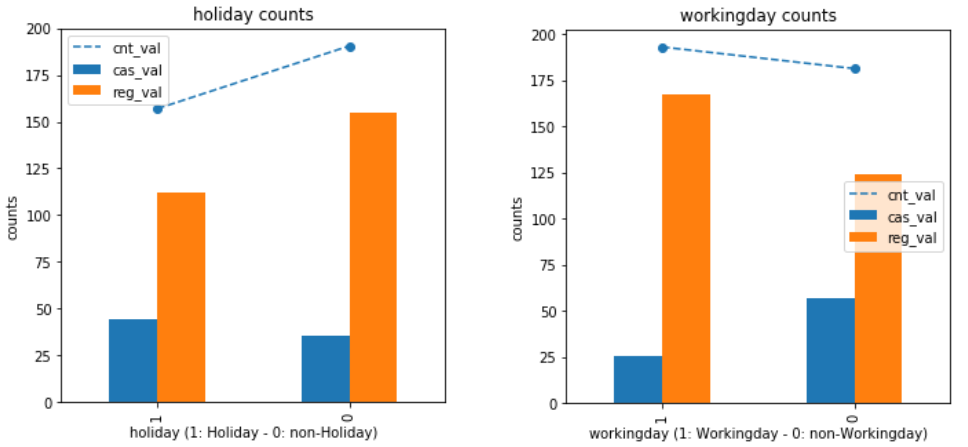
*Figure 39: The Trend Line and cnt V.S. reinstant*

Furthermore, I try to plot the weekday towards casual, registered and cnt (*Figure 40*). Throughout Monday to Sunday, there isn’t many difference on the total cnt. However, on Sunday and Saturday, registered users are declined as comparing to weekdays; whereas, casual users have more on weekends rather than weekdays. The reason behind is probably because weekends’ public transports are limited rather than on weekdays. Registered users are mostly using bike for the commute on weekdays. On the other hand, casual users are using bike for casual purpose on weekends. Therefore, summing up registered and casual users, which balance off the weekends and weekdays users as in total cnt.



*Figure 40: casual, registered and cnt V.S. weekday*

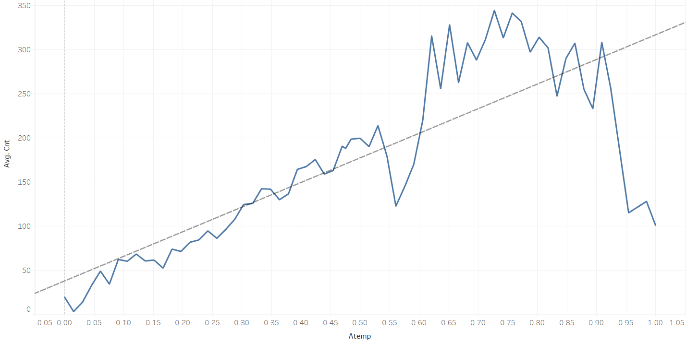
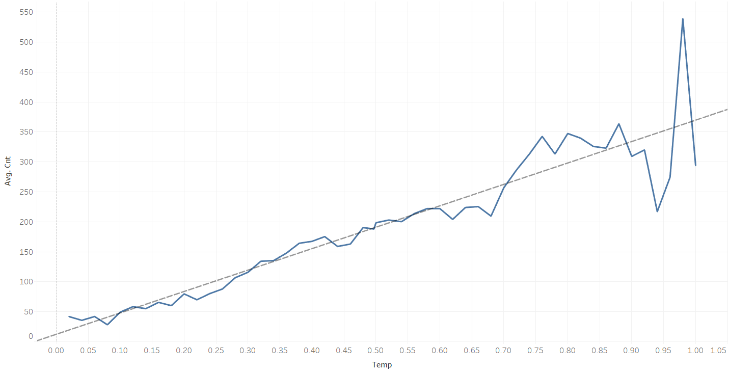
Also, I try to put holiday on the left and workingday on the right against casual, registered and cnt separately (*Figure 41*). Interestingly, it seems that non-holiday has just a little bit more than holiday on average; whereas, workingday is more than non-workingday. The gap between workingday and non-workingday is smaller than holiday and non-holiday.



*Figure 41: casual, registered and cnt V.S. workingday and holiday*

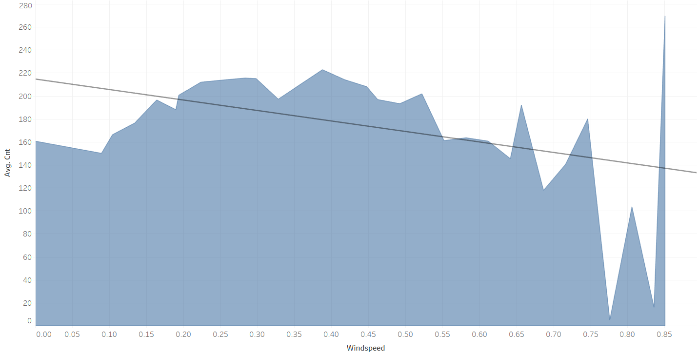
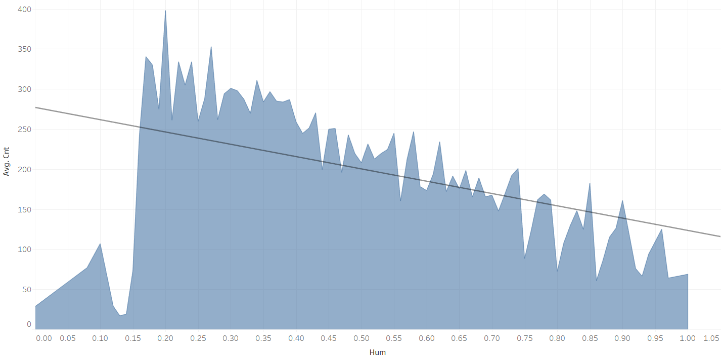
For the time series data, I will not utilize season field as it has already reflected in the mnth field. To note that, yr is included in dteday. So, yr will not be used either. Also, we know that the total cnt is summed by casual and registered. I will then utilize the cnt only, and leave casual and registered. Although registered and casual have differences on weekdays and weekends, for the business perspective, it will be more reliable on the total cnt. Hence, cnt will be sufficient for the further processes.

Now, let’s look at the environmental factors. Firstly, I plot the temp and atemp against cnt (*Figure 41*). I am thinking of there should be more people using bike in warm temperature but not in the both extremes. However, in the temp graph on the left, there are sudden increase and decrease closed between 0.95 and 1; whereas in atemp, between 0.6 to 0.9, the volumes remain high, which interpret that it’s in high demand for bike.



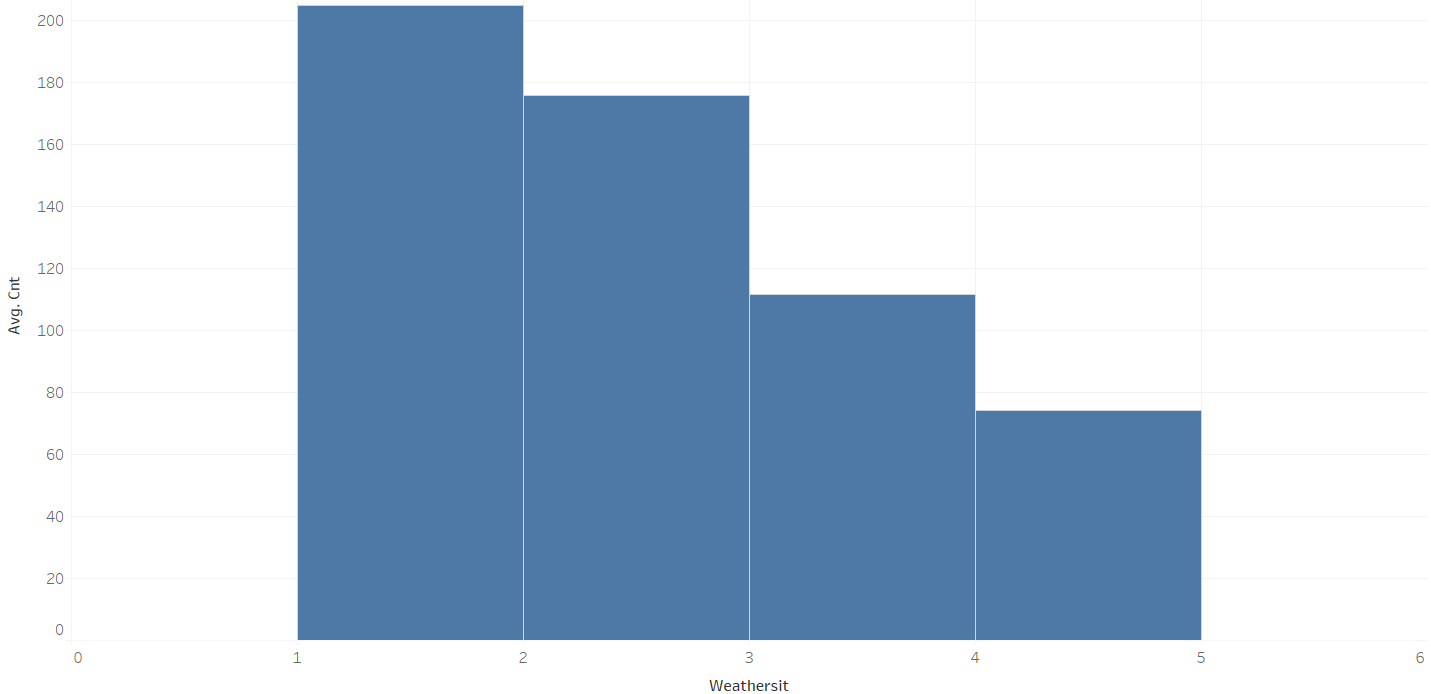
*Figure 41: cnt V.S. temp and atemp*

Then, I plot the hum and windspeed against the cnt (*Figure 42*). As we can see, start from 0.15, the hum has most of users and declines afterward; whereas in windspeed, it has stable user rate, but has huge concaves around 0.8, which don’t make sense to me. Therefore, it might be associated with some other events and factors in those unusual areas.



*Figure 42: cnt V.S. hum and windspeed*

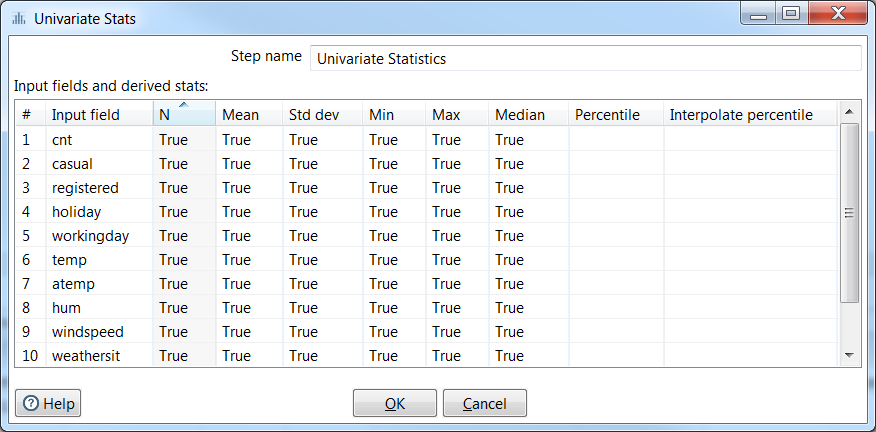
Admittedly, when I plot the weathersit against the cnt (*Figure 43*), the counts are less while weathersit is bigger. This is because extremely bad weather prevents people from using the bike. Hence, weathersit is one of the important fields to take into account. For the environmental series data, weathersit should be definitely taken into consideration, and I will also put temp, atemp, hum and windspeed for further assessments. For other fields, such as instant (has been removed), casual, registered, mnth, season and yr, I will exclude them for the further assessments.



*Figure 43: cnt V.S. weathersit*

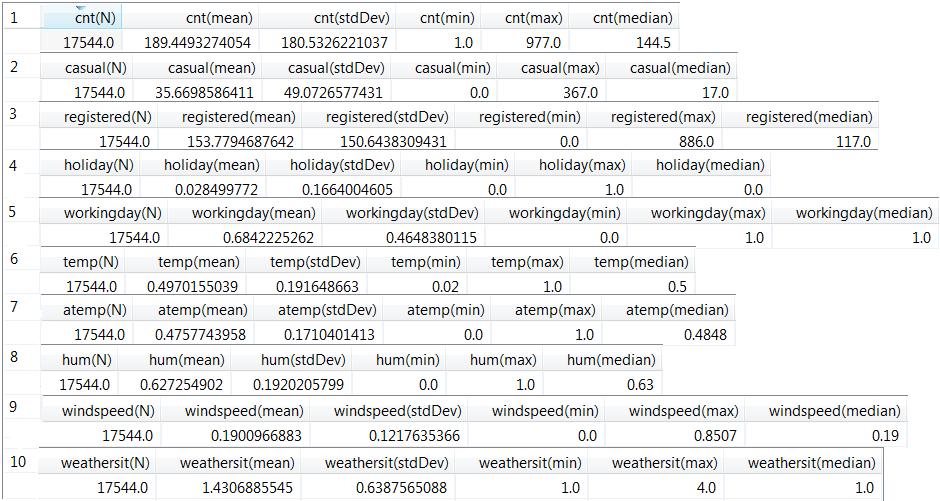
4.2 Project the data

In this step, first of all I will use Kettle/Spoon to retrieve the statistic information for all the fields. Start with a new workspace called *Project\_data* and connect the *Table input* with MySQL which is similar to the step 2.4. Instead, I use *Univariate Statistics* icon this time and try to get all the statistic information. After connect both of them, then run the workspace. Edit *Univariate Statistics* and I am able to get all the input fields. Select those fields I want to see the statistic information and put *True* respectively (*Figure 44*). Click *OK* to complete.



*Figure 44: Configure Univariate Statistics*

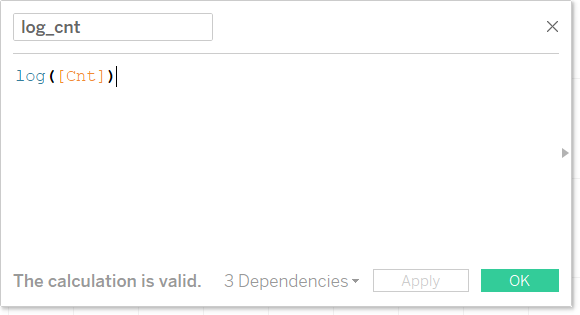
Next, I click the *Preview data* button on *Univariate Statistics* and I got the following results (*Figure 45*) – *I purposely make it display vertically instead of horizontally for a better view*. Retrieve the statistic information can be done on MySQL, however, use Kettle/Spoon will be easier without writing long scripts.



*Figure 45: Statistic Information of Selected Fields*

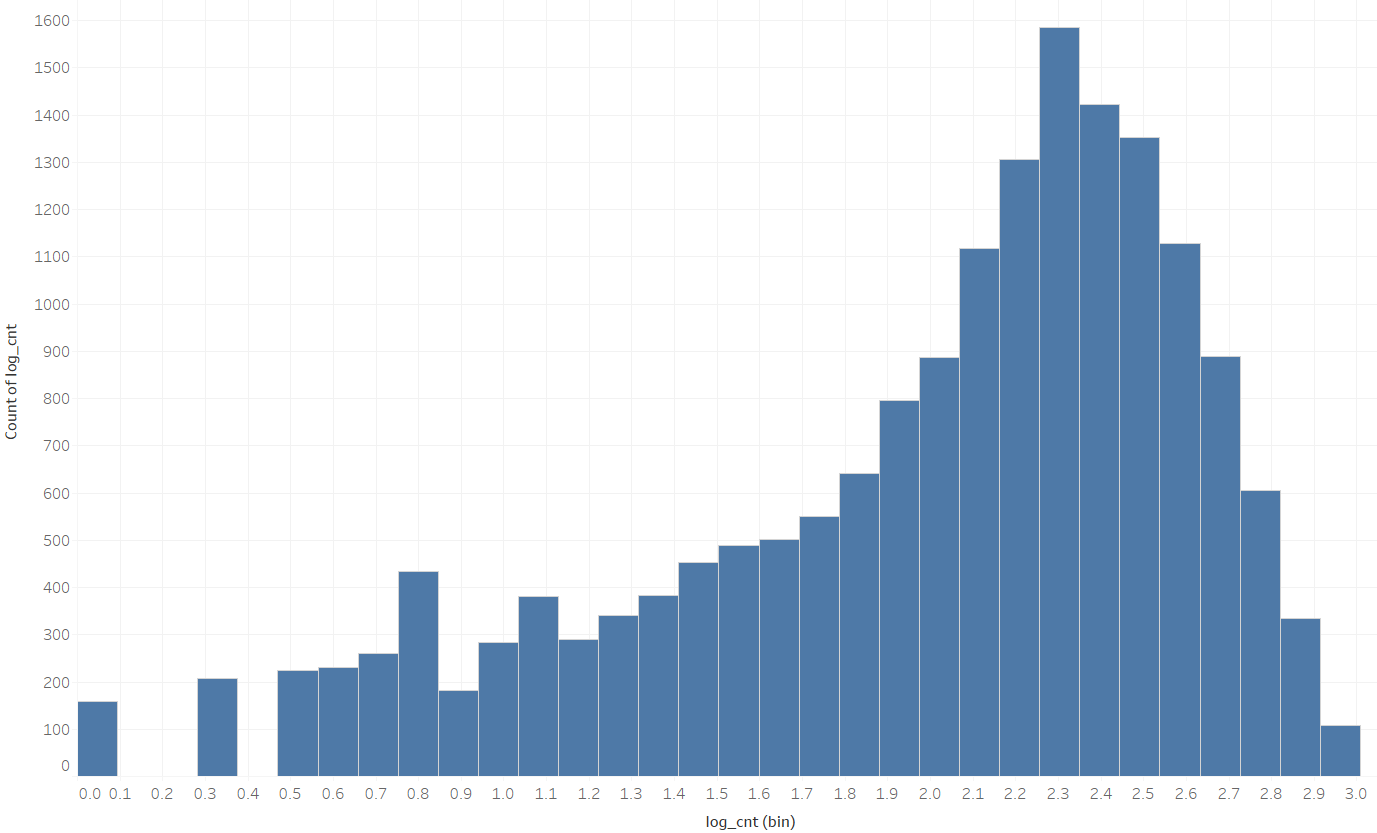
Since I have the hourly data set, I would not need to transform the dteday filed and extract it into hour, day of week and month again. This is the reason why in the very beginning of the data selection process I choose to use hourly data set rather than the daily data set. In addition, I have done quite a bit work on section 3, where I use Python and MySQL to extract, include, format and transform the data into better organized and readable dataset. It is an advantage that *Tableau* can transform the dteday field into different scales, such as monthly, quarterly, weekly, daily. Hence, *Tableau* can do much more sophisticated tasks and I will use it to get the logarithmic statistic information and graph.

Start a new workspace with *Tableau* and connect to MySQL again. Generate the histogram for cnt – it can be followed the step in 2.3. Right click on cnt and select C*reate Calculated Fields*. Then rename it to *log\_cnt* and input the scripts as follows (*Figure 46*). Click *OK* to complete and a new measure variable appears in the list.



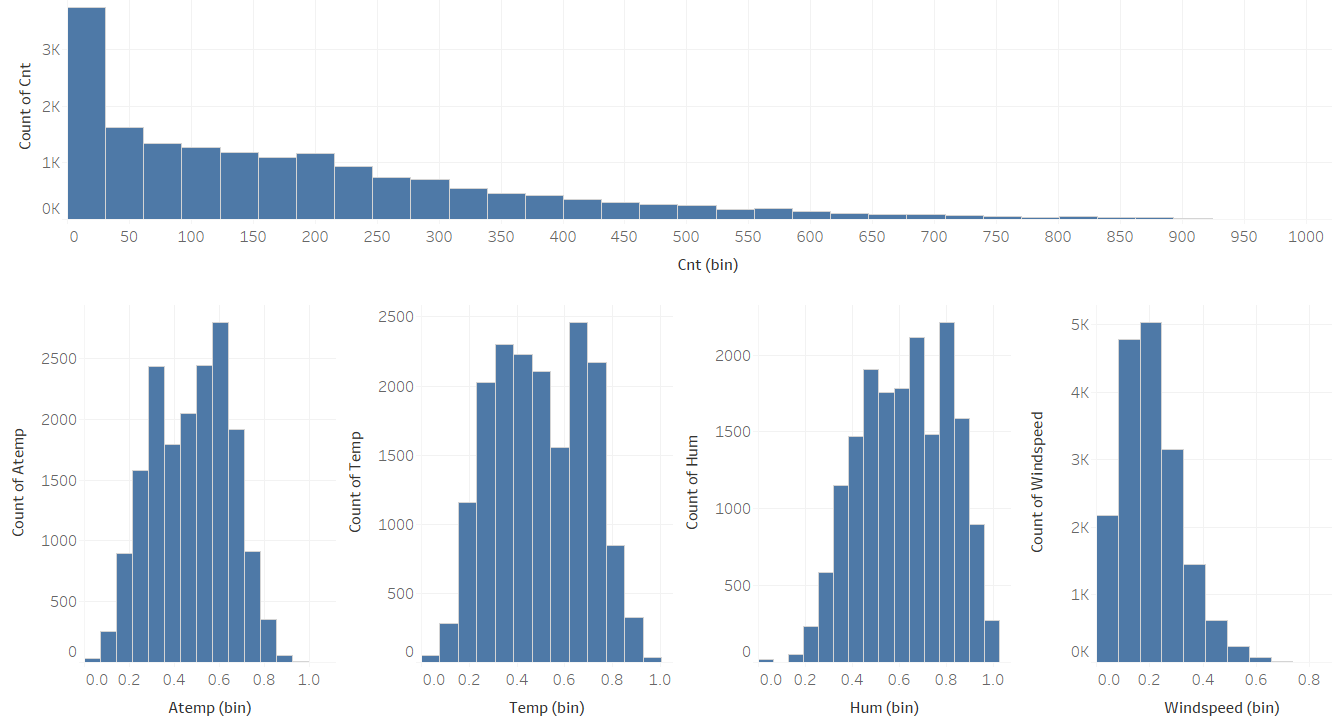
*Figure 46: Script of Logarithm for cnt*

Create a new *Sheet*, and double click the new created variable *log\_cnt*. Then choose histogram again. Now the logarithm of cnt appears (*Figure 47*).

**

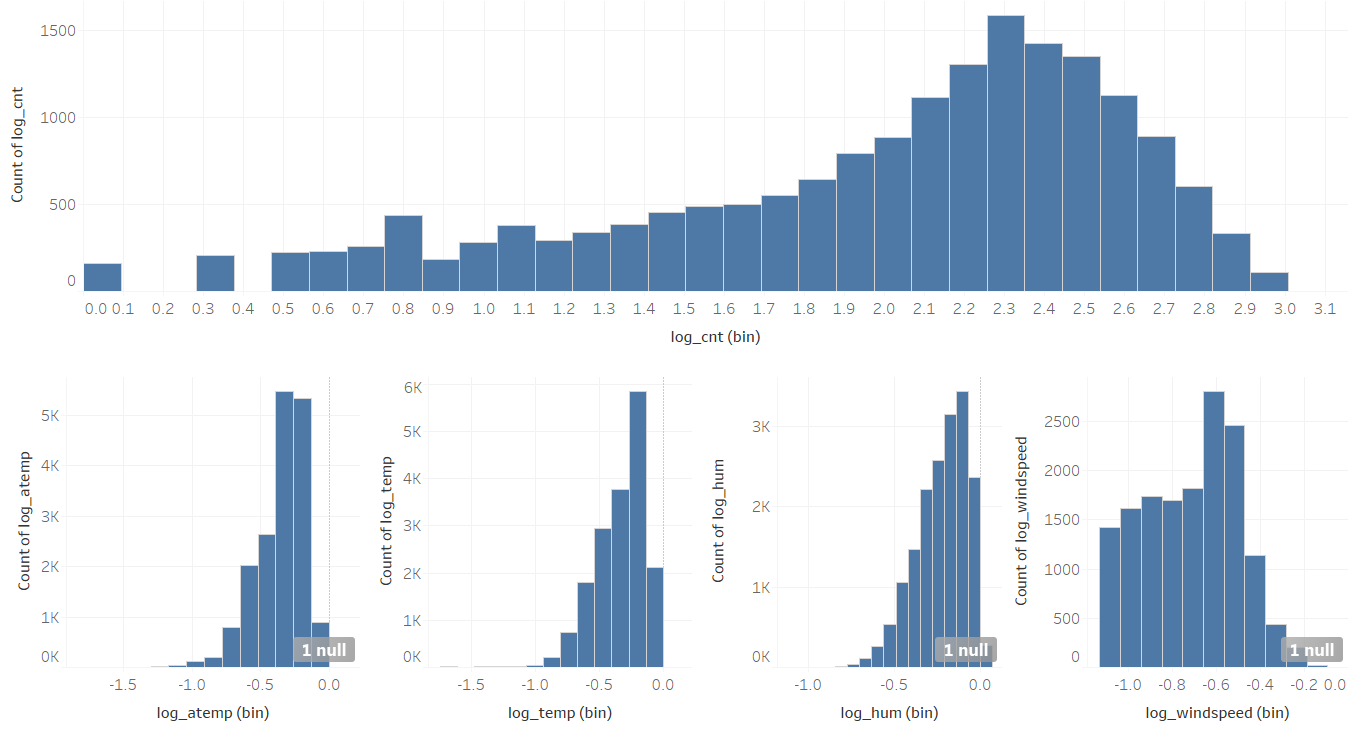
*Figure 47: Logarithm of cnt*

Finally, I am repeating the steps above to create the logarithms for other fields. There are two *Dashboards* created. One is for showing the histograms for all selected fields (*Figure 48*). We can see the cnt is skewed right. Atemp, temp and hum have bimodal and multimodal. Windspeed has unimodal though but seems to be not quite symmetric.



*Figure 48: Histograms for All Selected Fields in Dashboard*

The other one is for showing the logarithms (*Figure 49*). The cnt is getting symmetric unimodal but still got a long tail on the left. The rest seems to be unimodal but with skewed left.



*Figure 49: Logarithms for All Selected Fields in Dashboard*

**5. Data-mining method(s) selection:**

5.1 Match and discuss the objectives of data mining to data mining methods

As mentioned in the objectives of data mining earlier in this study, I am going to see the bike sharing total counts’ differences between weekday and weekend usage. Based on the weekday and weekend usage, I may see how the weekday and weekend factors affect the total count.

Borgnat et al. (2011) argued that weekdays show usage peaks in the morning, afternoon and late afternoon, whilst usage is concentrated in the afternoon on weekends. A statistical model for the prediction of the number on daily and hourly basis is better for the analysis.

Vogel et al. (2011) asserted that the recent research on bike sharing models either focus on mining of bike sharing data or building decision models without including the real world bike sharing behavior, such as ignoring the hourly fluctuations.

hr factor would be shown more details in this study rather than the data in daily basis. This is what I have always mentioned that the reason why I chose the data in hourly basis. I could probably see the differences between peak and off-peak usage during the day. Also, I will see the trend on holiday and workingday, because it implies the usage of bike sharing system on workingday and non-holiday.

As bike sharing grows as an important mode in urban transportation systems, shared bikes could find a significant niche in the transportation system. However, more studies with real-world data are required (Ji et al., 2014), for instance, the environmental issues which affect how people use the bike.

Environmental factors are also playing important parts on bike sharing systems as mentioned by the authors above. I will then include the environmental fields into the data mining processes, such as temp, atemp, windspeed, weathersit.

5.2 Select the appropriate data-mining method(s) based on discussion

Hearty and Gibney (2008) argued that supervised data mining methodology is used to model an output variable based on one or multiple input variables, and these models can be used to predict or forecast future cases.

Chen et al. (2016) analyzed that the supervised learning is used in dynamic link prediction method based on a model learnt from the variation of properties.

Therefore, based on the above discussion, I will then use supervised method on the data of weekday and weekend, peak usages, workingday and holiday, and all the environmental fields.

Weka data mining will be the tool for this study, and it helps me to understand the data. It also learns from the variation of properties, such as hr, weekdays, workingday. Finally, based on the Weka learning results, I could predict the total counts much better in the coming years.

**6. Data-mining algorithm(s) selection:**

6.1 Conduct exploratory analysis and discuss

In this step, I will try to find the algorithms that suit to my data set. One of the studies employed spatial multiple linear regression analysis to examine the impact of built environmental variables on bike sharing trip demand (Zhang et al., 2017). They found that users prefer to choose a bike-friendly environment and more accessible to location where the bikes are easy to get. Moreover, a large turnover generated nearby a residential community during weekends and off-peak hours of weekdays, and lower turnover and demand generated nearby a park during morning and evening peaks of weekdays.

According to above researchers’ saying, the case they studied is quite similar to mine. So, I am going to use the same algorithms that are *Linear Regression to* apply into my data set.

6.2 Select data-mining algorithms based on discussion

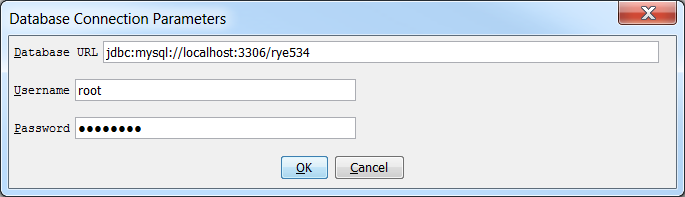
There are a number of supervised algorithms for data mining. To specific, linear regression that is supervised algorithm will be used in this study as discussed.

I will use *Linear Regression* in Weka workspace. At the end, the results will be shown each field against the cnt with the respective statistics, and also the summary of the statistics for all fields as the whole model and algorithm. Based on the statistics, I could tell how well or bad the models and algorithms are. Evaluation may be performed to those models and get much better understanding of them. Finally, I may get the best models for this study.

6.3 Build/Select appropriate model(s) and choose relevant parameter(s)

In Weka, I will use supervised regression models for the data set. Before to build the model, I need to connect the dataset to Weka workspace (*in order to make it work, mysql-connector-java-5.1.14.jar library has to be included into Weka build path. Please refer to online resource for more details*).

Start Weka application, select *Explorer* and then click *Open DB…* in *Weka Explorer*. In the new prompt, click *User*, and another prompt pops up. Type in the following information (*Figure 50*) - *This information is the same information as to login to MySQL.*



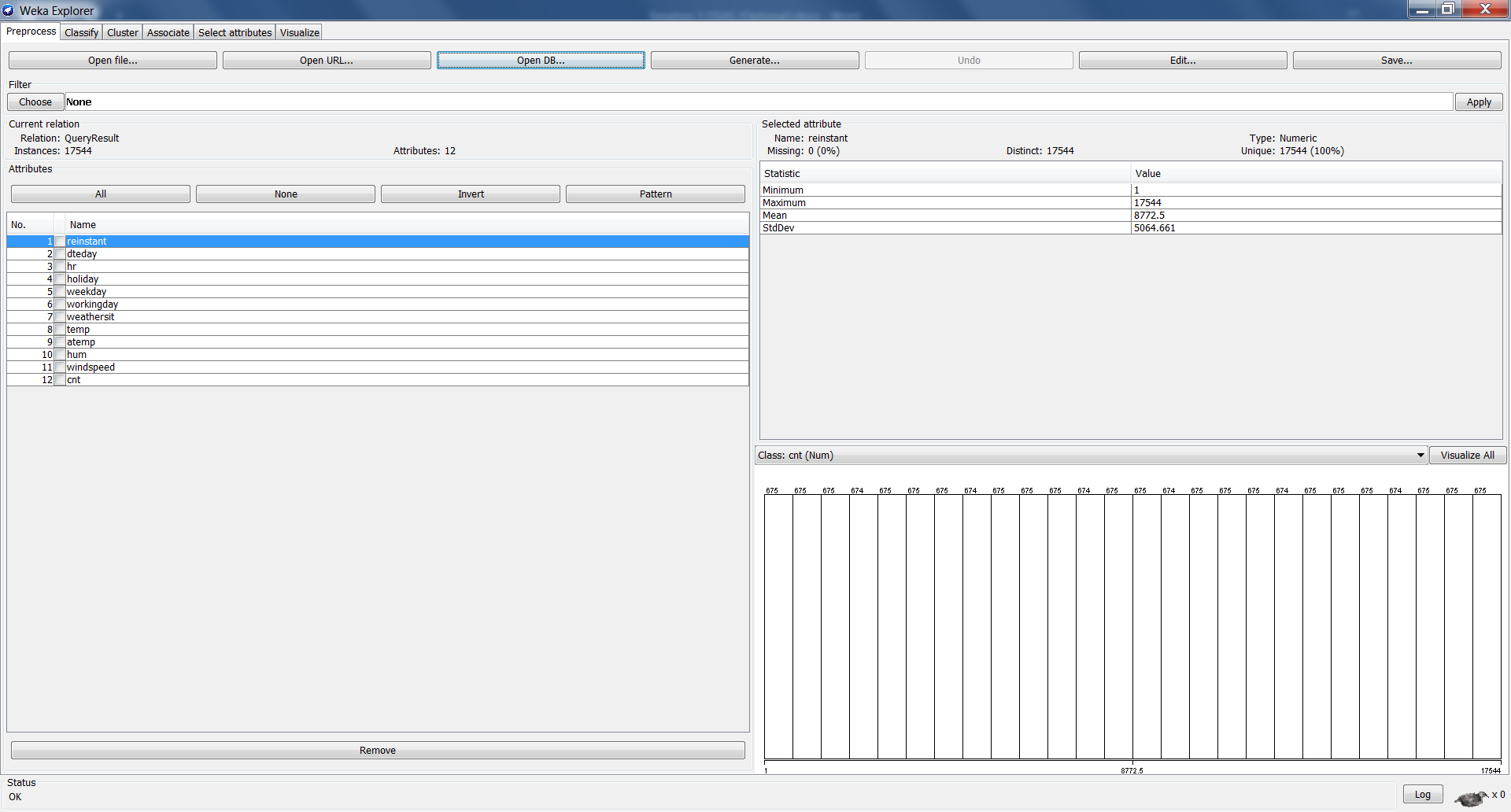
*Figure 50: Settings for MySQL Connection in Weka*

Click *OK* to complete the settings, and now click *Connect*. If connection is correct, it should be expected to have the messages below (*Figure 51*). Then, in the *Query* text area, write the query to retrieve the entire dataset and this will be passed into the Weka application *Cache* and used by the Weka workspace later. As discussed, I will query those 12 fields needed for further processes but not the whole dataset. Click *OK* to complete and continue.



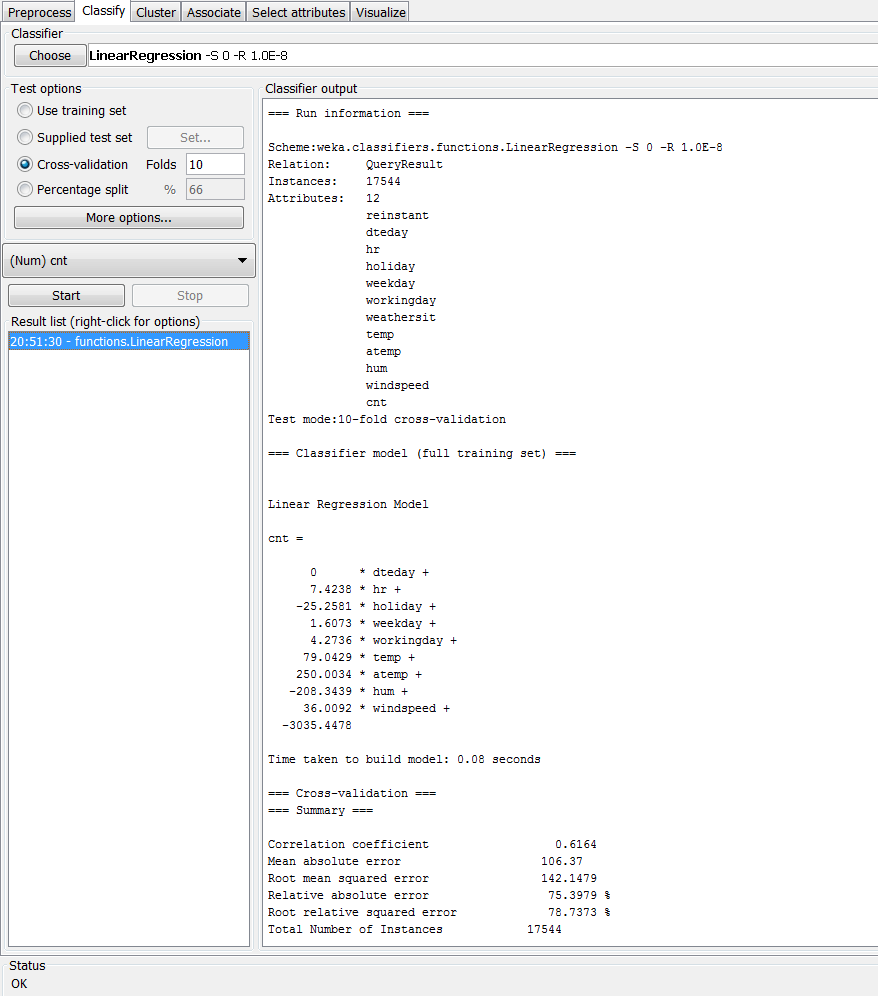
*Figure 51: Connection Established*

Now, I can build the model directly by using the latest dataset without exporting the dataset into any other formats then importing it back to Weka. In the workspace, I got the whole interface below showing all selected fields (*Figure 52*). At the right hand side, I can also see the statistics when I choose the fields – *I can ignore these statistics because I’ve done that with other tools in previous steps*. Basically, I can view and edit the dataset in the *Preprocess* interface, which has the similar features to other software above I’ve used.



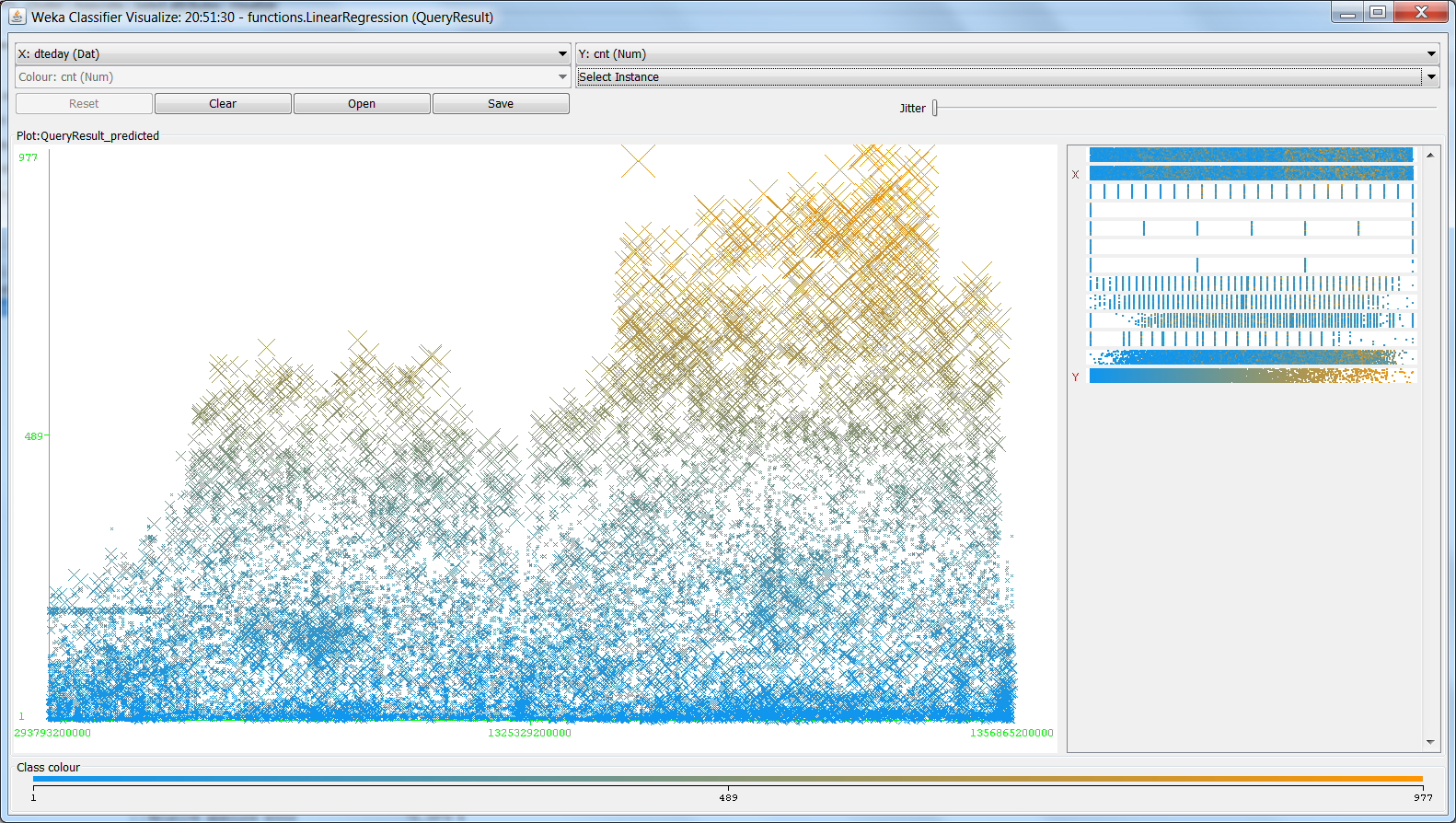
*Figure 52: Weka Workspace Preprocess Stage*

The dataset is ready, so I am trying to build the *Linear Regression* model. Click *Classify* in the menu bar, choose *Classifier* and under functions, select *Linear Regression*. In the attribute, remember to select (Num) cnt as the parameter so that the other attributes will be built against to cnt. Click *Start* button and at the right hand side, I got the output below (Figure 53). Just take a quick glance at *Correlation coefficient*, it is 0.6164, which isn’t a very good result. However, I will improve the results later.



*Figure 53: First Build Linear Regress*

Right click on the *Result list,* select *Visualize classifier errors and* I can get a better visualization on the output (*Figure 54*). Go back the *Preprocess* interface, and save the workspace – *the workspace can be saved only the dataset readable for Weka, but the model built is unable to be saved. Please find all the Weka workspaces under Weka folder*.



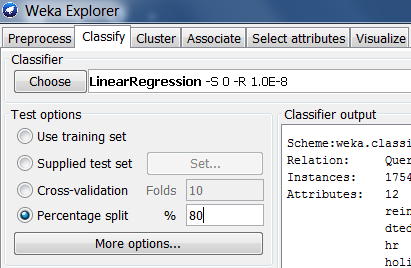
*Figure 54: Visualization on the First Build*

**7. Data Mining:**

7.1 Create and justify test designs

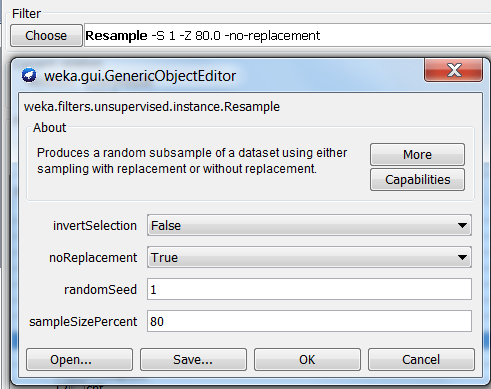
In this step, I will split the entire data into two parts that are training and testing data sets. The ratio of them is 80% (training):20% (testing). The total records are 17544, so there are about 14000 records for training and 3500 records for testing, which should be sufficient for the models in this study. Rajer-Kanduc, Zupan, and Majcen (2003) illustrated that the data set should be divided into the training and testing data sets in order to apply most of the standard chemometric modelling methods. The training sets and the modelling method are paramount important, whereas the testing sets are inevitable for evaluation of model’s characteristics.

There are many different ways to split the data. The easy way is in the *Classify* panel, I can just click *Percentage split %* and enter *80 under Test options* (*Figure 53*). This means that the test mode will split 80% data for the training, and remainings are for the testing.



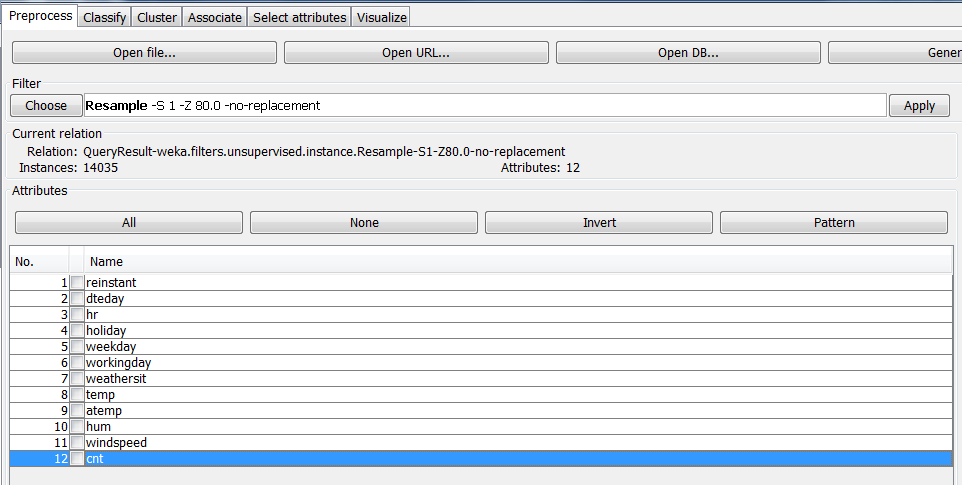
*Figure 55: Quick Split Dataset Percentage*

However, it can only test the training data in this way. Hence, I will use other method to split the datasets. To slipt the data, select *Resample* from the *Filter* in *Preprocess* panel. Then click on the selected *Filter*, a new prompt shows and type in the followings (*Figure 56*) – *80% for training data, so type in 80 in sampleSizePercent, randomSeed means random dataset from the entire dataset, noReplacement means not to change the dataset if dataset is Null or empty, invertSelection will be shown later.* Click *OK* to complete.



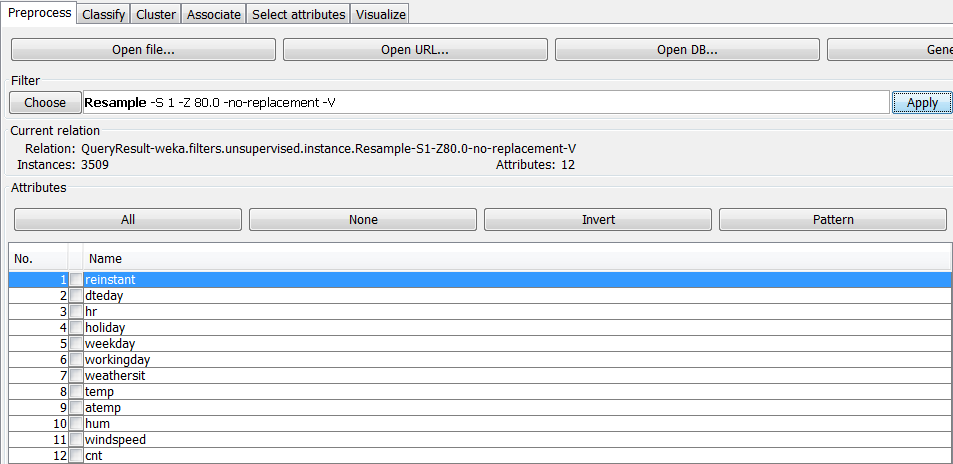
*Figure 56: Select Training Dataset*

Back on the main panel, click *Apply* and I can see the entire dataset changed to 14035 (Figure 57). Click *Save* the training dataset for the models analysis later.



*Figure 57: Apply the Training Dataset*

Prior to get the testing dataset, I need to click *Undo* button to have full dataset 17544 back on, because I am getting testing dataset from the full dataset but not from 14035 training dataset. Then click on the *Filter* again, I select *True* in the *invertSelection* dropdown – *this means to select the testing dataset.* Click *OK* and *Apply*, I got 3509 for the testing dataset (*Figure 58*). Also, I save the testing dataset for the models analysis later.

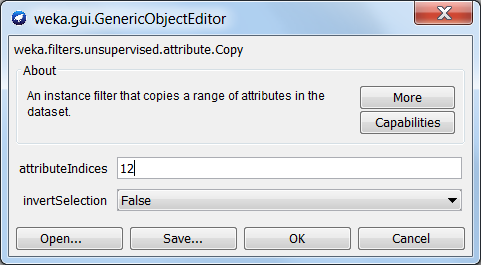


*Figure 58: Apply the Testing Dataset*

7.2 Conduct data mining – classify, regress, cluster, etc. (models must execute)

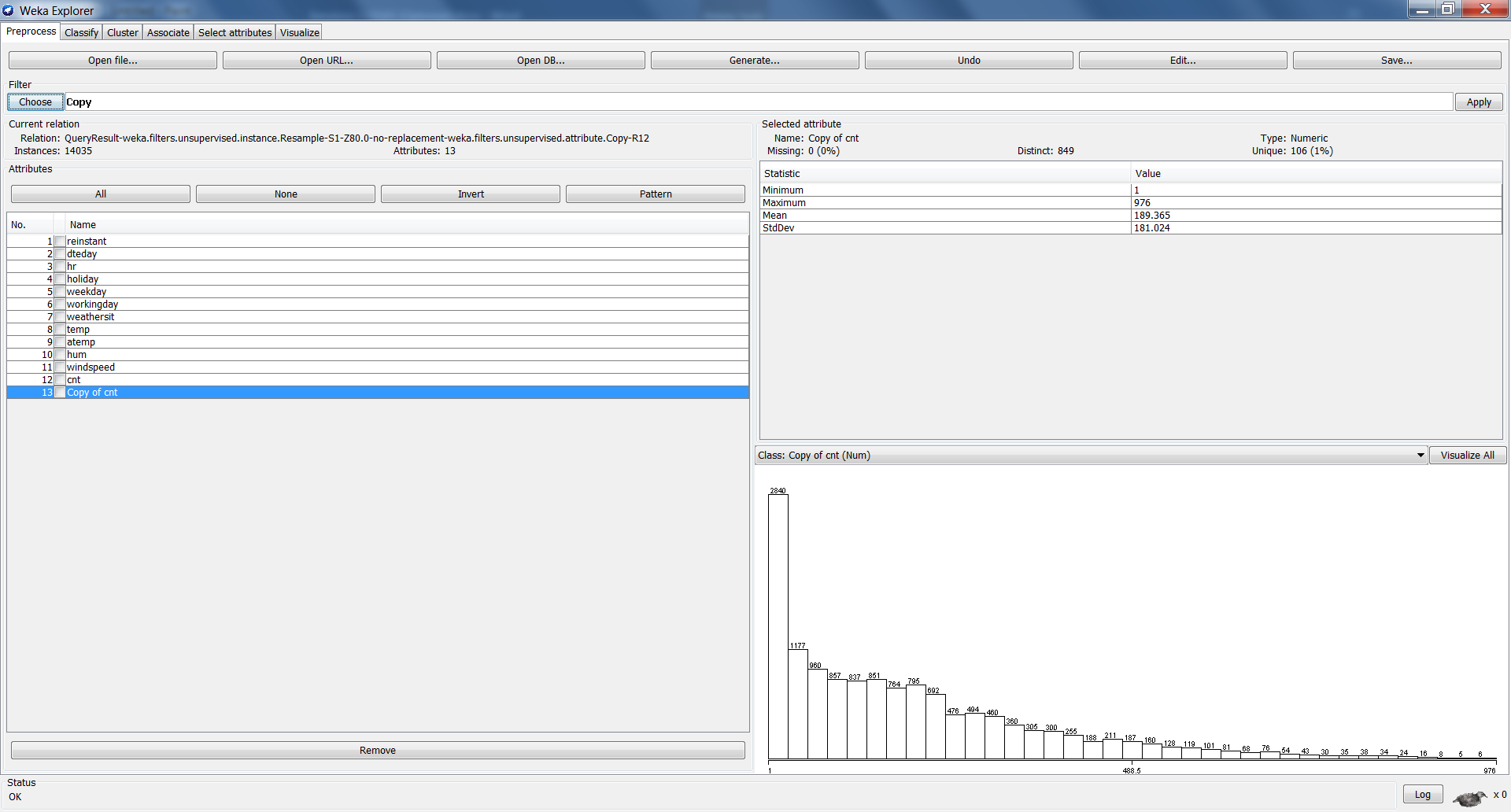
I am going to conduct data mining in this step. Either I use *Filter* to get the training dataset or import the saved training dataset in the workspace. Prior to this step, I’ve built the model with entire dataset and got *Correlation coefficient* 0.6164. It’s believed that the training dataset will has similar results. Execute the *Linear Regression* with the training dataset, and I got *Correlation coefficient* 0.6134 which is even worse than the previous one.

Hence, I am going to improve it by using lag. Select *Copy* from *Filter,* click *Copy* and key in value 12 in *attributeIndices* (*Figure 59*)*,* which mean that the index 12 cnt will be copied.



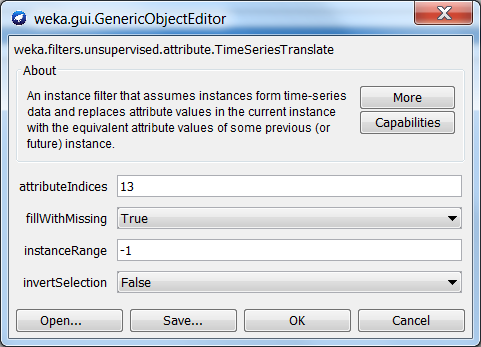
*Figure 59: Copy Configuration*

Click *OK* to continue and click *Apply* to copy cnt. I will get a new field that is *13 Copy of cnt* below (*Figure 60*).



*Figure 60: New Field Copied*

Actually, the new field *Copy of cnt* is same to cnt. Furthermore, choose *TimeSeriesTranslate* from the *Filter.* Edit it and enter 13 in *attributeIndices,* put -1 in *instanceRange* (*Figure 61*), which indicate to have lag 1. Click *OK* to complete.



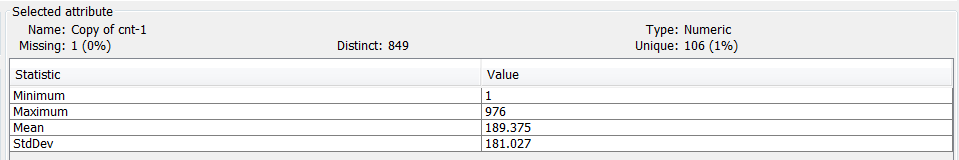
*Figure 61: Lag Configuration*

At the graphic area, select *Class: cnt (Num) as shown Figure 62*. This indicates that applying lag to the new field against the cnt field. Then click *Apply* to complete.



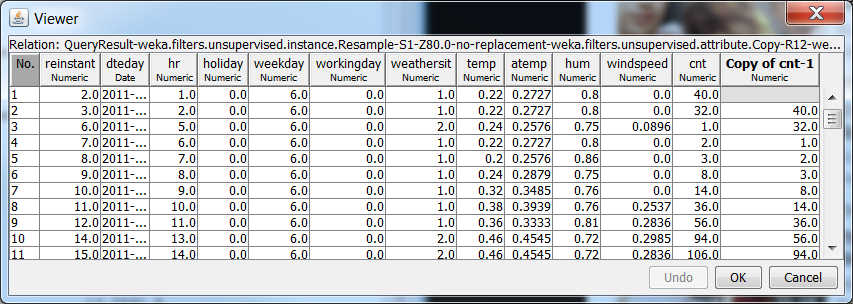
*Figure 62: Select Class: cnt (Num)*

The statistics will have slightly different details between cnt and Copy of cnt (*Figure 63*)



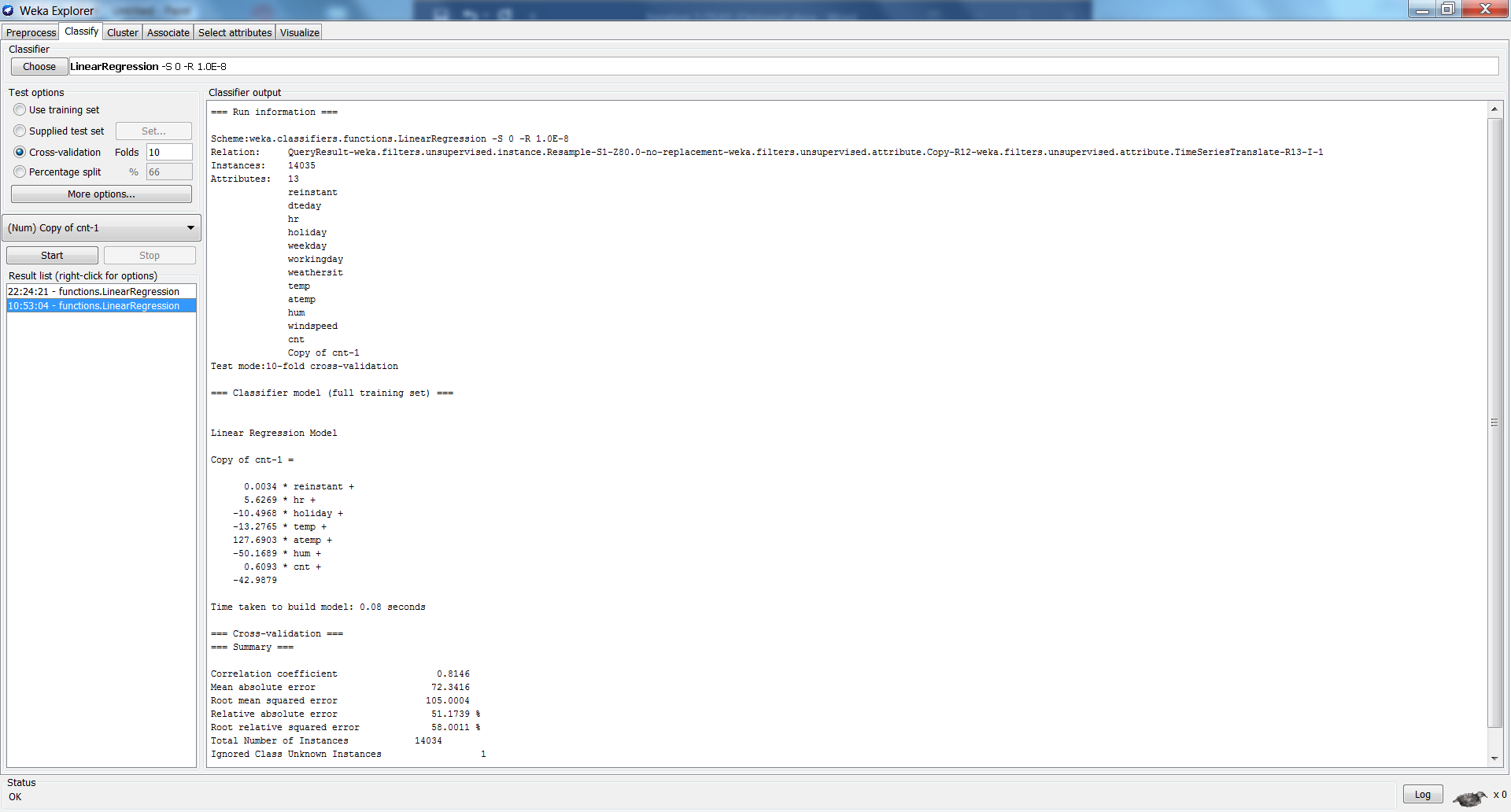
*Figure 63: New Statistics for Copy of cnt*

Click the *Edit* button, I can see one empty value locates at the first row of *Copy of cnt-1* (*Figure 64*), which indicate that the Lag 1 feature is successful.



*Figure 64: New Column with Lag 1*

Go to *Classify* panel and choose *Linear Regression Classifier,* leave the *Test options* alone, because the current dataset is training dataset so I don’t need to change every default settings. However, I got the new field Copy of cnt and will apply the model against to the new filed, so I have to select the measure field as *(Num) Copy of cnt-1.* Click *Start* to execute the model (*Figure 65*). Excellent, the model can be executed and I will save the information accordingly under Weka folder – *I will re-produce the same model but with all testing dataset that is also saved*.

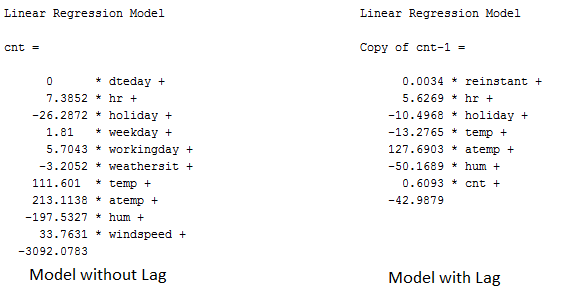


*Figure 65: Executable Linear Regression Model*

7.3 Search for patterns

As we can see, the *Correlation coefficient* now is 0.8146 which is much better than the previous one is 0.6164. The model has a greater improvement. Also, the *Mean absolute error* is 72.3416, *Root mean squared error* is 105.0004, *Relative absolute error* is 51.1739% and *Root relative squared error* is 58.0011%, which are much less and better model now. However, when I use the testing dataset, I got a better model without *lag 1* rather than with *lag 1*.

As in the *Linear Regression* model statistics, I can see the patterns for the fields are getting closer to zero (*Figure 66*). Also, use the lag feature from Weka, it will remove some variables accordingly to fit the model, such as weekday, weathersit, windspeed.



*Figure 66: Linear Regression Training Models Statistics with and without Lag*

**8. Interpretation:**

8.1 Study and discuss the mined patterns

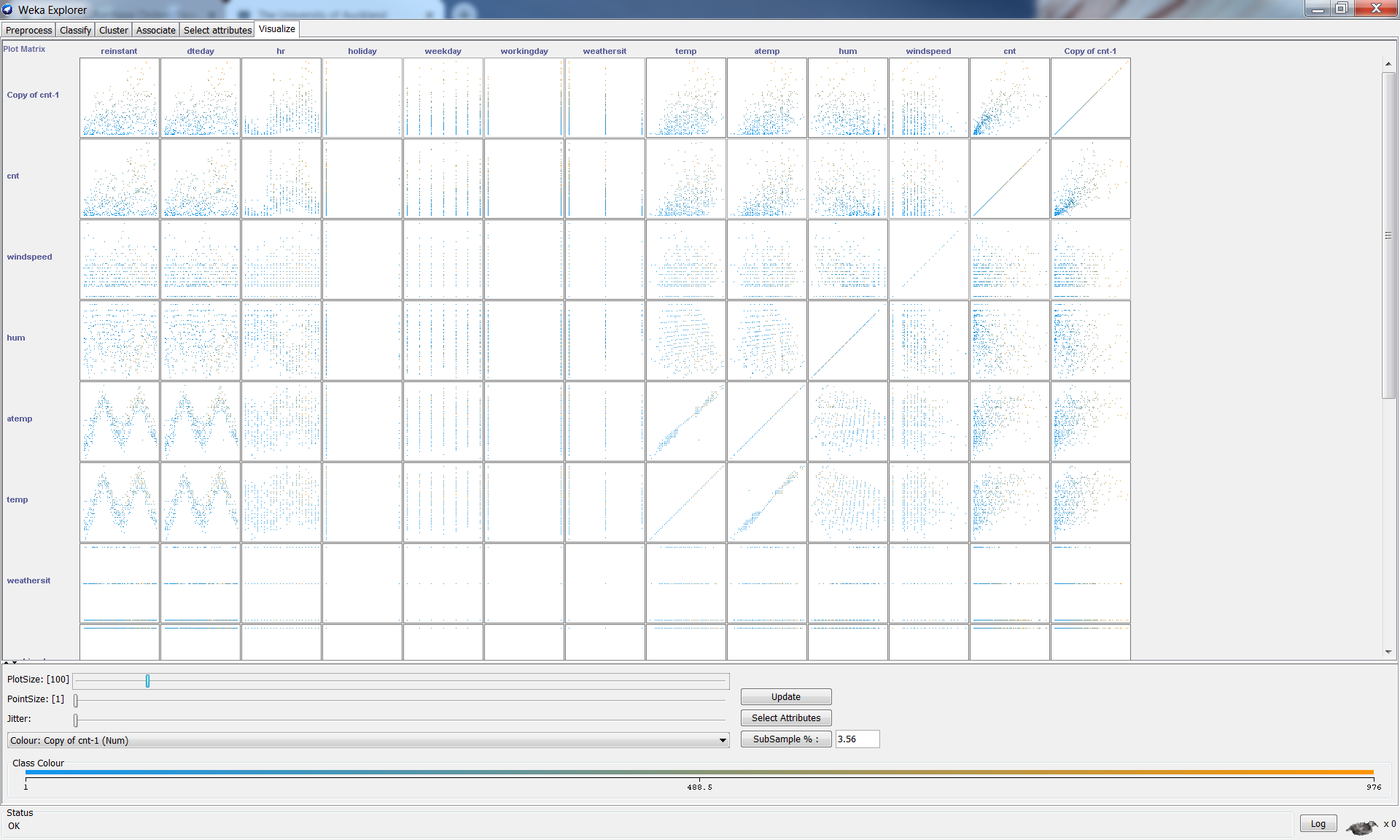
What we can see from the above patterns are getting improved and better as they are close to zero from 0.8146. This is so-far the best model I got, which is the Lag 1 *Linear Regression.* The *Correlation coefficient* numbers indicate that it has positive correlation and for every positive increase in one variable, there is a positive increase of a fixed proportion in the other. The numbers are closer to 1, the greater relations they have.

The next important numbers I am looking at are the *Mean absolute error* that is 72.3416. It suggests that the absolute value of the different between the forecasted value and the actual value. It tells us how big of an error we can expect from the forecast on average.

Last but not least, *Root mean squared error* produces residuals that are the measure of how far from the regression line data points are. It means a measure of how spread out these the data are. On the other hand, it indicates how concentrated the data is around the line of best fit. 105.0004 is just the average numbers for my model.

8.2 Visualize the data, results, models, and patterns.

Click on the *Visualize* menu bar, I can generate the following graphs for variables (*Figure 67*).



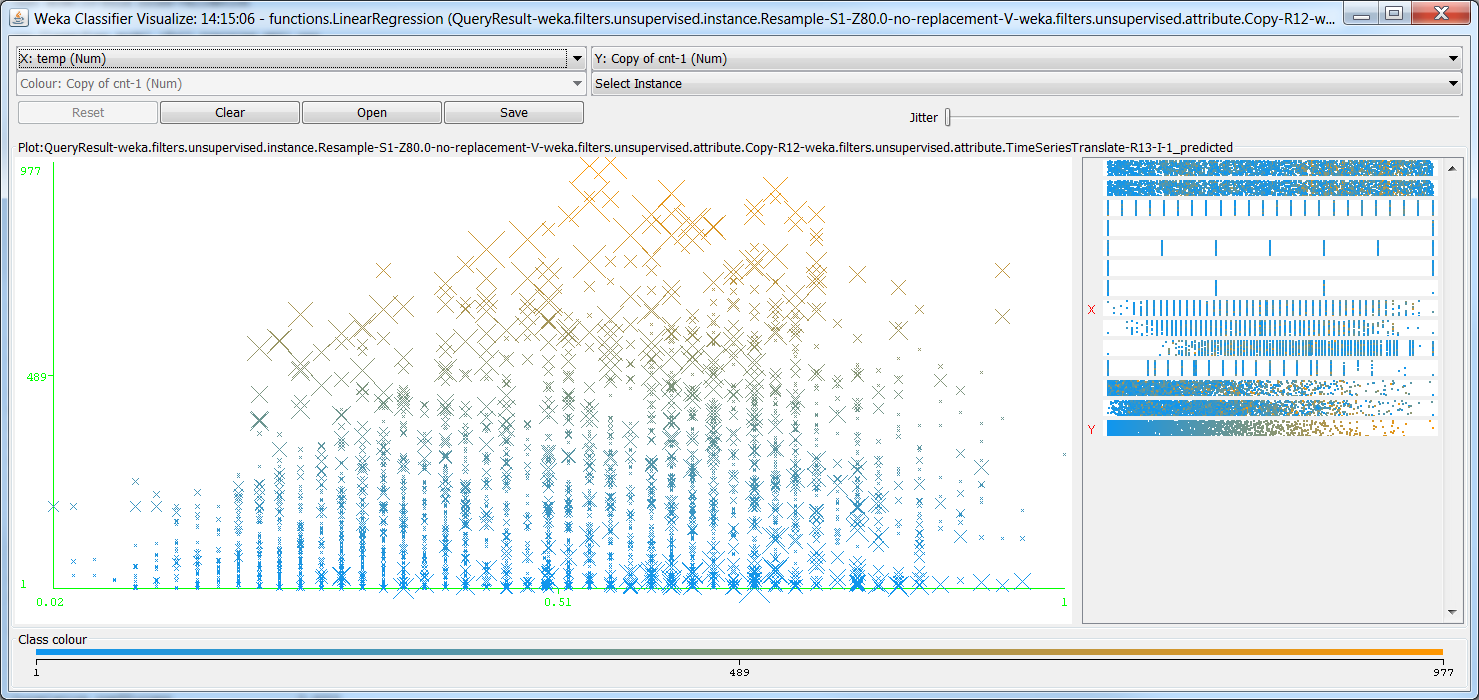
*Figure 67: Visualize All Variables in Summary*

Also, I could click on each individual cell to get a zoom-in and nicer view, such as *Copy of cnt-1* vs. *temp* (*Figure 68*)*.* However, it isn’t taking any reflections from the *Linear Regression* into account.



*Figure 68: Visualize an Individual Graph from Summary*

In order to reflect the model results, right click on the *Result list* under *Classify* panel. Select *Visualize classifier errors* and I got *Copy of cnt-1* vs. *temp* (*Figure 69*). As you can see, this graph and the previous one are different.

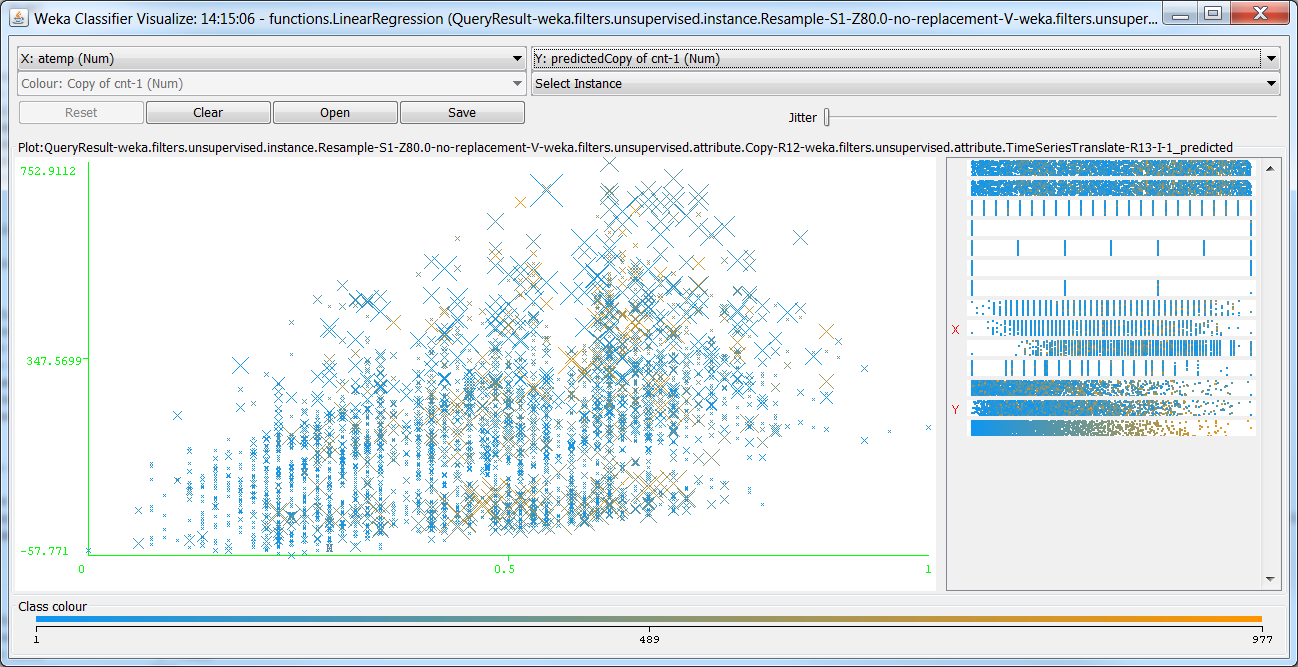


*Figure 69: Visualize Classifier Errors on Copy of cnt-1 vs temp*

8.3 Interpret the results, models, and patterns

As analyzed, I can find that the bike sharing services have seasonal effects. At the beginning of this study, I have already mentioned that season one and four have less counts than season two and three, which interpret that people probably don’t want to use the bike in the cold weather rather than the warm weather. Where in a daily basis, the morning peak and afternoon peak hours are the two important duration for bike sharing.

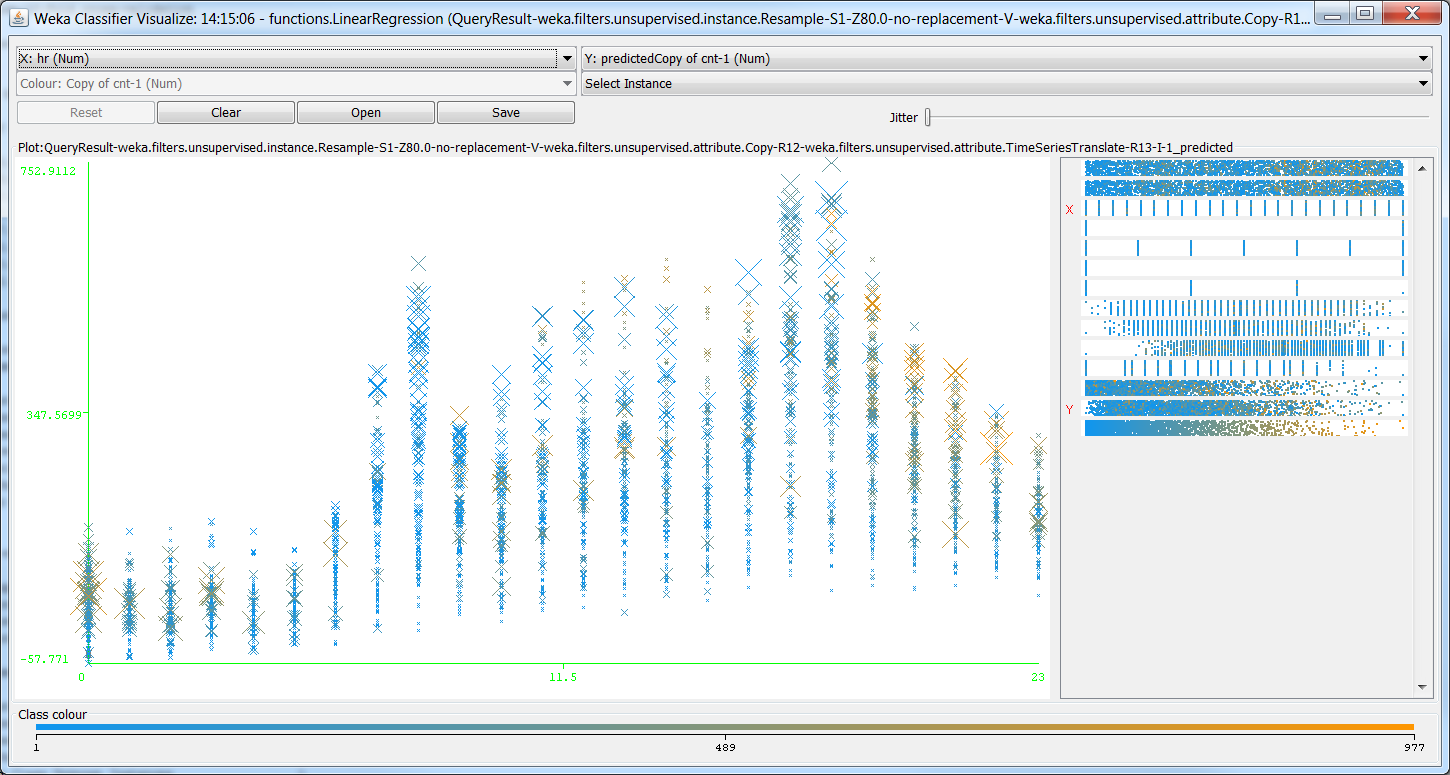
Besides, the temp, atemp, weathersit, windspeed are also affects the bike sharing services to some extent, especially the atemp. Let’s visualize the atemp vs. the predicted *Copy of cnt-1* (*Figure 70*).



*Figure 70: Visualize Classifier Errors on Predicted Copy of cnt-1 vs. atemp*

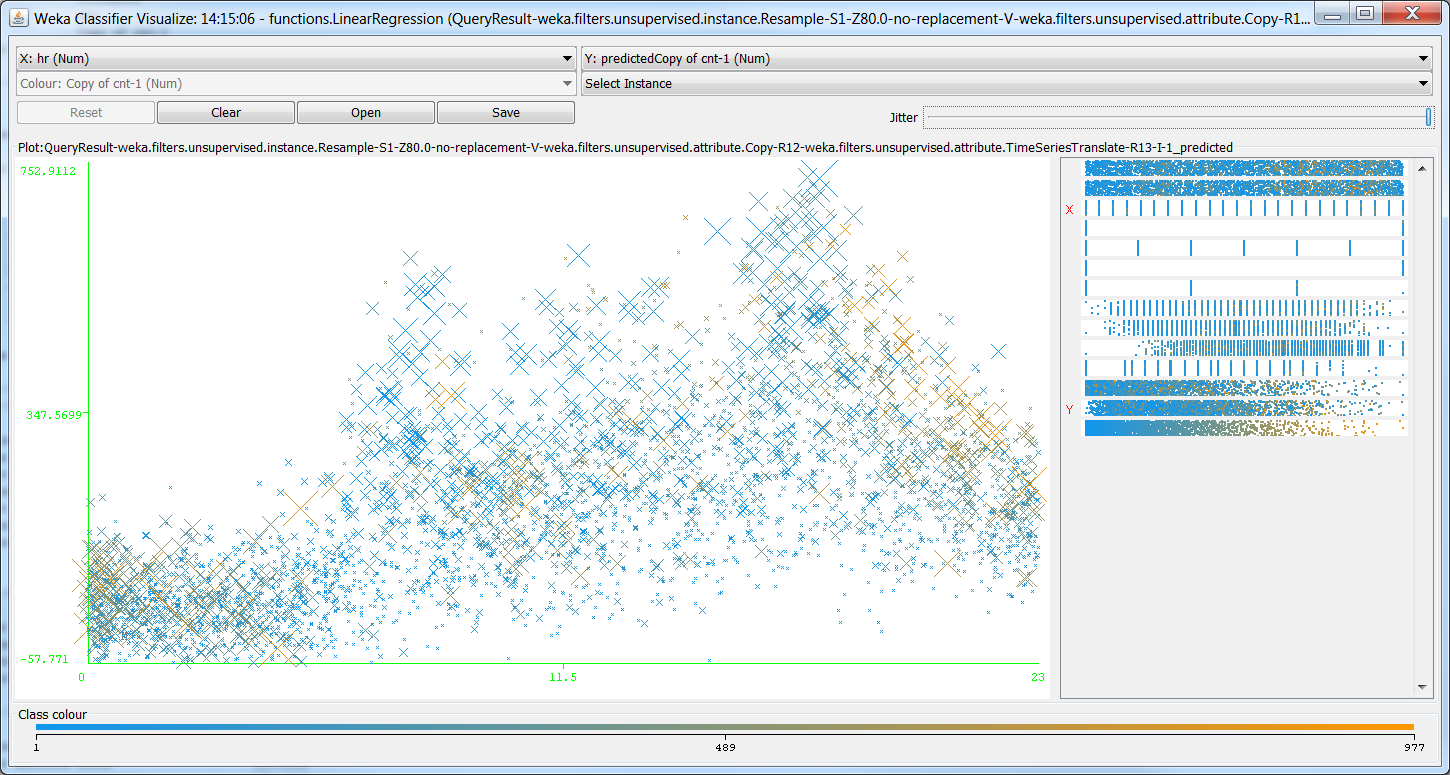
We can see the graph is in upward trend then going downward after around 18pm, with many scatter points on the high side. The bigger the cross is, the bigger the error is. If I got the original one (*atemp vs cnt*) comparing this one, I can easily spot out where those errors lcoate.

Get the hr vs. the predicted *Copy of cnt-1* (Figure 71), I can still see a curve that interprets the morning and afternoon peak hours. However, as most of the crosses on the high side show how big the errors are, basically the low side is the reflection that indicates the true numbers of bike users in a day.



*Figure 71: Visualize Classifier Errors on Predicted Copy of cnt-1 vs. hr*

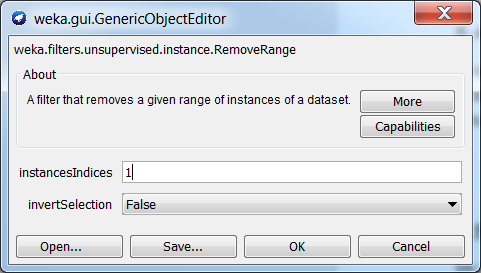
Drag the *Jitter* bar from left to right, I have another view (*Figure 72*). Now I have better view that shows more crowded crosses at the beginning, in the middle and right corner. This illustrates that more errors are lying on this three areas. I then try other variables and got similar results.



*Figure 72: Visualize Classifier Errors on Predicted Copy of cnt-1 vs. hr Jitter Edited*

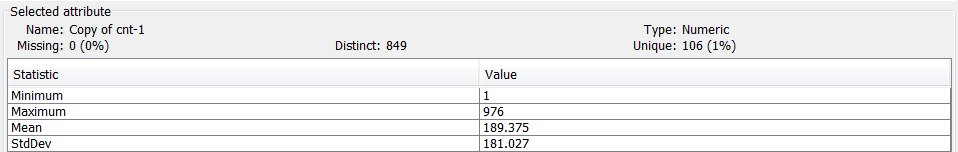
8.4 Assess and evaluate results, models, and patterns.

We know that the lag 1 model has 1 empty value in the beginning. So I am going to make it up and try to evaluate the results if they are getting better. In the *Filter,* choose *RemoveRange*, and put 1 (*Figure 73*), because I have 1 empty value. Click *OK* to continue and *Apply*.



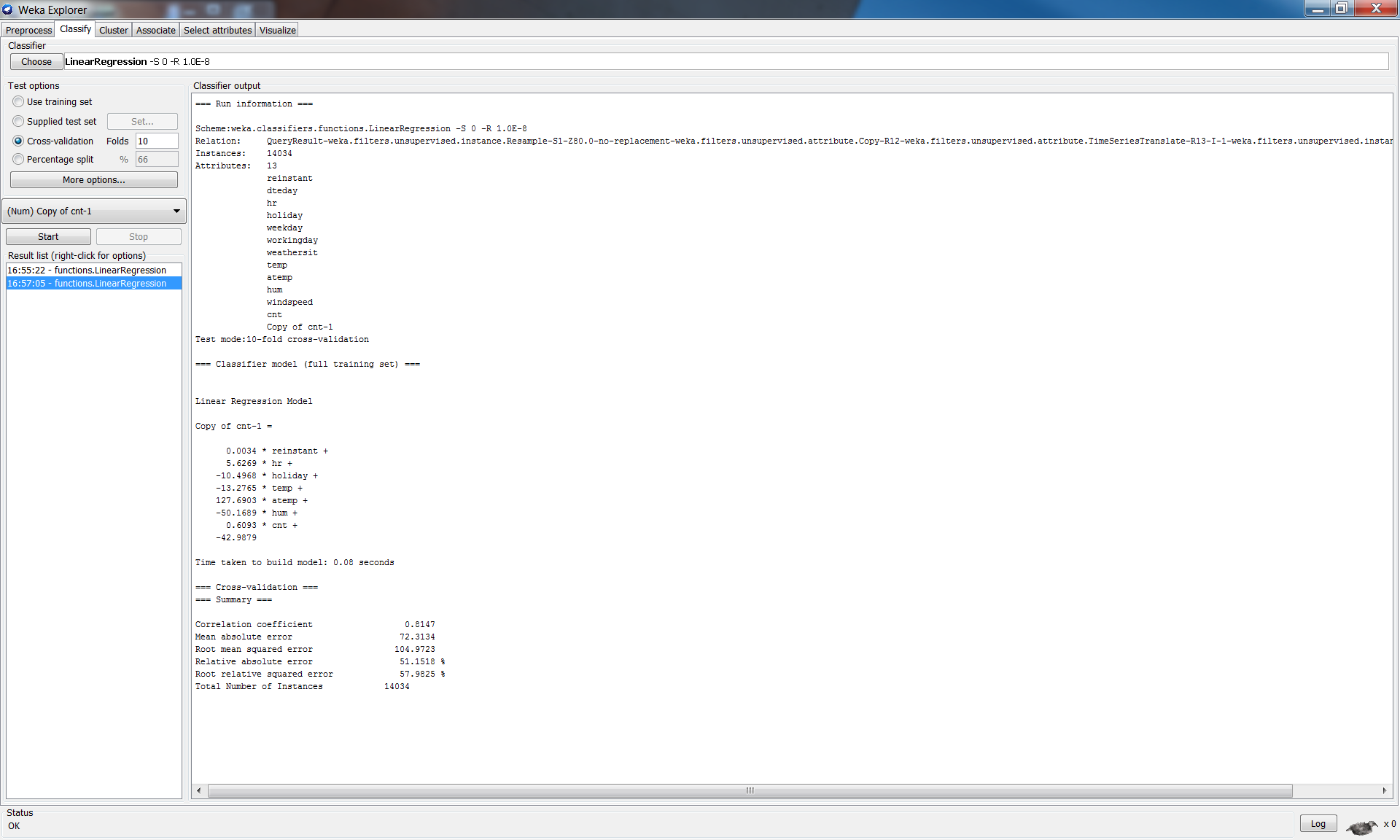
*Figure 73: RemoveRange Configuration*

The statistics are changed to *Missing: 0 (0%)* as comparing to previous variable *Copy of cnt-1* (*Figure 74*).



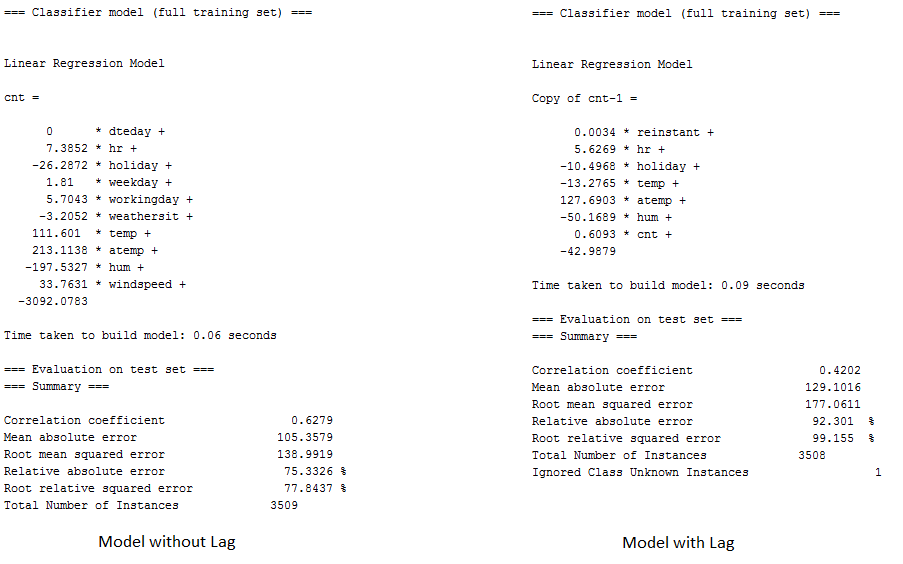
*Figure 74: Missing Lag 1 is gone*

In the *Classify* panel, click *Start* button to get the model with minor changes and evaluate it (*Figure 75*). However, the data isn’t changed much as we can see from the statistics comparing the one above. Hence, for the training dataset, the best model is the *Linear Regression* with Lag 1.



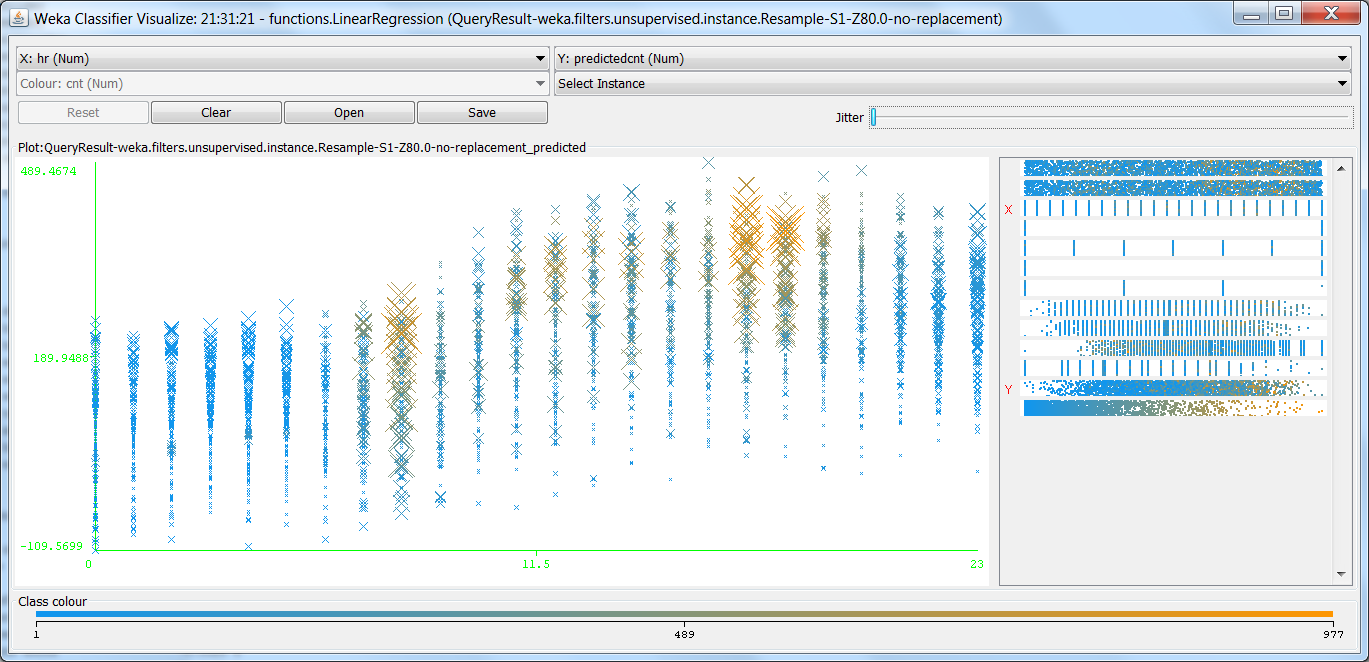
*Figure 75: Evaluate the Linear Regression Model with Lag 1*

Now I am going to try the training model with the *Linear Regression* for lag 0 and lag 1, and evaluate the model on the testing dataset. I got both results as follows (*Figure 76*). As we can see, for the testing dataset, it is better to use lag 0 rather than lag 1, because it got *Correlation coefficient* 0.6279 with lag 0 compare to 0.4202 with lag 1.



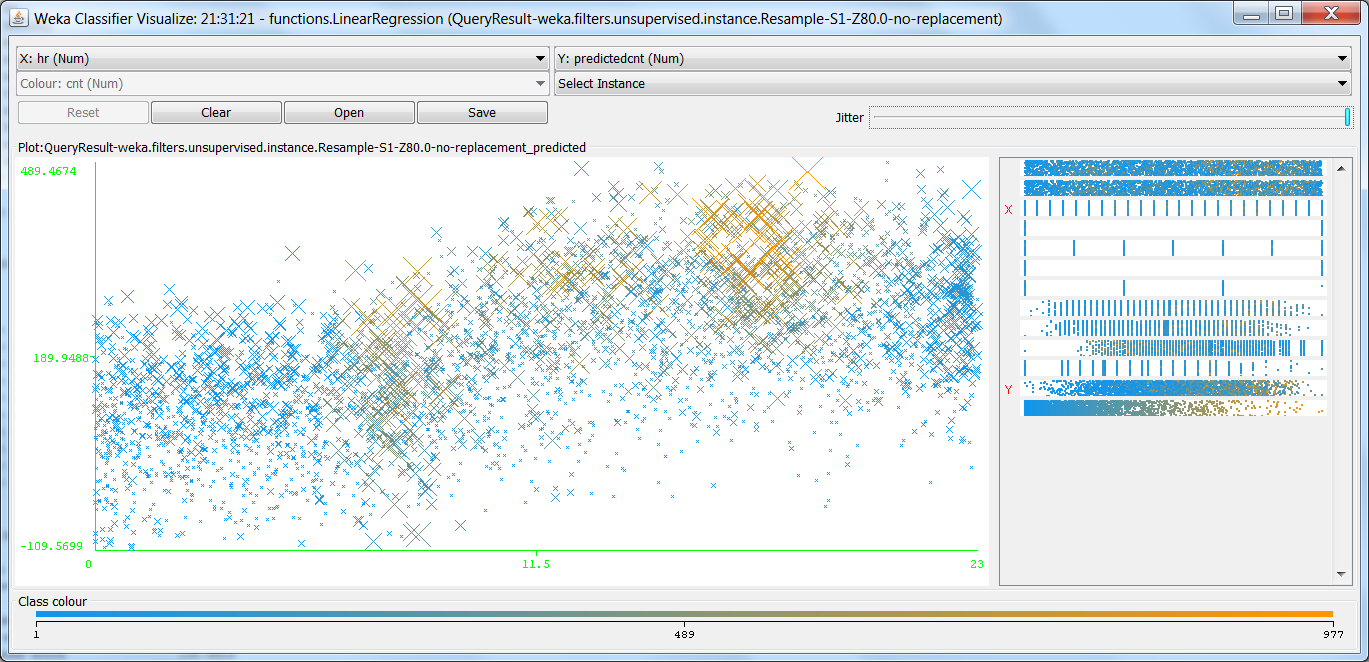
*Figure 76: Evaluate on the Training Model with Testing Dataset*

Search for *Visualize classifier errors,* I got predicted cnt vs hr, which is similar to the training data set above (*Figure 77*).



*Figure 77: Visualize Classifier Errors on Predicted cnt vs. hr*

Drag the *Jitter* from left to right, however, I can see there are two differences about in 8am-10am and 5pm-7pm (*Figure 78*). This might indicate that these two areas have errors. In other words, it’s volatile.



*Figure 78: Visualize Classifier Errors on Predicted cnt vs. hr Jitter Edited*

8.5 Iterate prior steps (1 – 7) as required

In this study, I use model of *Linear Regression* to analyze the bike sharing dataset. It is basically the time series dataset, that’s why I believe the *Linear Regression* is one of the best models to use.

I first build the model for the training dataset, and improve it with lag 1. As results, the *Linear Regression* with lag 1 for training model is the best model. However, use the best model for those testing dataset with and without lag. It is found that the best model is used without lag, which is best suit to testing dataset.

Iterate steps 1-7 by using *Simple Linear Regression* this time, it is suggested that *Simple Linear Regression* is not as good as *Linear Regression* for training dataset with and without lag, as well as testing dataset with and without lag – *I also complete the Simple Linear Regression in Weka, and all the produced works are saved under Weka folder*. Therefore, I conclude that *Linear Regression* model is the best model for this study.

**9. Action:**

9.1 Discuss how you would apply the knowledge and deploy the implementation

The selected fields I use above for the models are incorporated the *Linear Regression* model. All these fields are important, but only few can be ignored. Therefore, I will pay a significant attention to those important ones. For example, in peak hours, weekday and workinday, increasing and putting more bikes in the city area in case of bike shortage, which will be added huge benefit for the business. For casual usages, it is better to allocate the bikes near the neighbor and communities, and the public event area, such as parks, zoos, stadiums, and large playgrounds. This could possibly help people get awareness of the bike sharing services.

Additionally, prices award for registered users rather than casual users, which may encourage more casual users to register the services. Eventually, people will stick on the services and utilize the services even more.

9.2 Discuss how you would monitor the implementation

Occasionally, sending survey and having interview with the users at the docking stations. It is possible that monitoring people’s activities by installing CCTV at the docking station, which could capture how and when people use the services.

Alternatively, send survey via the internet systems, in particular the mobile app. Enforce people use the services through the mobile app, which means that people must use mobile to book and pay for the systems in order to use the bike sharing services.

Also, plant smart card technology to the bike sharing systems, which generate station-based data or trip-level data and facilitates studies of the practical use of bike sharing systems, as well as uncovers the spatial and temporal patterns of cycle trips (Zhang et al., 2016). This technology may discover the aspects of bike sharing services, such as distance of usage.

9.3 Discuss how you would maintain the implementation

Collect the data in periodical manner, then the data can be deployed to the model again. From there, we can see if the model is still fit. Adjustment needs to be made accordingly to the model and the bike sharing services. For example, as discussed, in winter people may use the services less because of the cold weather and perhaps people dress more clothes that make the cycling activities much more difficult. Hence, bike could be amended to have more space and easier to ride. Reduce cost to motivate people, as well as provide gloves to have cold protection, etc.

9.4 How could you enhance the solution in the future?

To enhance the solution, I would like to have more fields to be added to the model. For example, in the beginning of this study, I have already mentioned that the distance is one of the important fields to decide whether people want to use the bike sharing services, because it is unlikely that people cycle for a long distance.

Moreover, docking station is another important field. For instance, if docking station is far away from the destination, people would not use the services. However, the nowadays bike sharing services can be parked anywhere. This implies a concern if the location is allowed to be parked and occupied by the bikes.

Finally, how easy and safe to use the bike sharing services, which include the physical usages, such as is the bike easy to be used, does the bike have any potential hazard that would hurt people. Also, it includes the system usages, such as is it easy to book and grab a bike, how to make payment. All this information needs to be put together so that I could make a thorough consideration and produce a much better model and solution.

**Bibliography:**

Borgnat, P., Abry, P., Flandrin, p., Robardet, C., Rouquier, J., & Fleury, E. (2011). Shared bicycles in a city: a signal processing and data analysis perspective, *Advances in Complex Systems, 14*(3), 415-438. Retrieved from doi:10.1142/S0219525911002950

Chen, K., Chen, Y., Li, Y., & Han, J. (2016). A supervised link prediction method for dynamic networks, *Journal of Intelligent & Fuzzy Systems, 31*, 291-299. Retrieved from doi:10.3233/IFS-162141

Fanaee-T, H., & Gama, J. (2013). Event labeling combining ensemble detectors and background knowledge, *Progress in Artificial Intelligence*, 1-15. Springer Berlin Heidelberg. Retrieved from doi:10.1007/s13748-013-0040-3

Hearty, A., & Gibney, M. (2008). Analysis of meal patterns with the use of supervised data mining techniques – artificial neural networks and decision trees, *The American Journal of Clinical Nutrition, 88*(6), 1632-1642. Retrieved from doi:doi.org/10.3945/ajcn.2008.26619

Ji, S., Cherry, C., Han, L., & Jordan, D. (2014). Electric bike sharing: simulation of user demand and system availability, *Journal of Cleaner Production, 85,* 250-257. Retrieved from doi:doi.org/10.1016/j.jclepro.2013.09.024

Matrai, T., & Toth, J. (2016). Comparative assessment of public bike sharing systems, *Transportation Research Procedia 14,* 2344-2351. Retrieved from doi: 10.1016/j.trpro.2016.05.261

O’Brien, O., Cheshire, J., & Batty, M. (2014). Mining bicycle sharing data for generating insights into sustainable transport systems, *Journal of Transport Geography 34,* 262-273. Retrieved from doi:doi.org/10.1016/j.jtrangeo.2013.06.007

Papantonopoulos, G., Takahashi, K., Bountis, T., & Loos, B. (2014). Artificial neural networks for the diagnosis of aggressive periodontitis trained by immunologic parameters, *PLoS ONE, 9*(3), 1-8. Retrieved from doi:doi.org/10.1371/journal.pone.0089757

Rajer-Kandoc, K., Zupan, J., & Majcen, N. (2003). Separation of data on the training and test set for modelling: a case study for modelling of five colour properties of a white pigment, *Chemometrics and Intelligent Laboratory Systems, 65*(2), 221-229. Retrieved from doi:doi.org/10.1016/S0169-7439(02)00110-7

Ricci, M. (2015). Bike sharing: A review of evidence on impacts and processes of implementation and operation, *Research in Transportation Business & Management* *15*, 28-38. Retrieved from doi:doi.org/10.1016/j.rtbm.2015.03.003

Vogel, P., Greiser, T., & Mattfeld, D. (2011). Understanding bike-sharing systems using data mining: exploring activity patterns, *Procedia Social and Behavioral Sciences, 20*, 514-523. Retrieved from doi:10.1016/j.sbspro.2011.08.058

Zhang, Y., Thomas, T., Brussel, M., & van Maarseveen, M. (2016). Expanding bicycle-sharing systems: lessons learnt from an analysis of usage, *PLoS ONE, 11*(12), 1-25. Retrieved from doi:doi.org/10.1371/journal.pone.0168604

Zhang, Y., Thomas, T., Brussel, M., & van Maarseveen, M. (2017). Exploring the impact of built environment factors on the use of public bikes at bike stations: case study in Zhongshan, China, *Journal of Transport Geography, 58,* 59-70. Retrieved from doi:doi.org/10.1016/j.jtrangeo.2016.11.014