Applied Data Analytics II – Course Project
Date completed: 05/02/2020
Research question: Bike Sharing Demand
Document name: Project

I. Research statement

1.1 Project Background

Bike sharing systems are a means that enable the users to rent a bike from one specific location and return the bikes in another specific location. The users are able to register as a member, rent the bikes, return the bikes via the internet by themselves. Currently, there are over 500 bike-sharing programs around the world, such as Citi-bike and so on.

1.2 Problem statement

By using only information available prior to the rental period, we will predict the non-registered user rentals count of bikes, the registered user rentals count of bikes and the total rentals count of bikes during each hour covered by the test set.

The analysis is of great significance since it aims at helping the enterprises to reduce the costs by putting the appropriate numbers of bikes into the market.

What is more, Bike sharing systems function as a sensor network, which can be used for studying mobility in a city.

Applied Data Analytics II – Course Project
Date completed: 05/02/2020
Research question: Bike Sharing Demand
Document name: Project

II. Dataset (description and link)

2.1 Dataset links and descriptions

The dataset can be publicly obtained from Kaggle (Links: https://www.kaggle.com/c/bike-sharing-demand/overview/description), which contains bike hourly rental figures spanning two years (2011 and 2012), with variables such as season, holiday, working day, weather included in the dataset. The train dataset, which is composed of the first 19 days of each month in both 2011 and 2012, contains more than 10,800 rows of data, while the test dataset, consisted of the rest days from the twentieth day to the end day of the month, contains approximately 6,500 rows of data.

By combining historical usage patterns with weather data and so on, I will forecast bike rental demand in the Capital Bikeshare program in Washington, D.C.

2.2 Data fields

The data fields are concluded as below to help to comprehend the meaning of the data.

2.2.1 Independent variables

Variables	Descriptions
datatime	Hourly late + timestamp
season	1 = spring; 2 = summer; 3 = fall; 4 = winter
holiday	Whether the day is considered a holiday

workingday	Whether the day is neither a weekend nor holiday
weather	1: Clear, Few clouds, Partly cloudy, Partly cloudy 2: Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist 3: Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds 4: Heavy Rain + Ice Pallets + Thunderstorm + Mist, Snow + Fog
temp	Temperature in Celsius
atemp	"feel like" temperature in Celsius
humidity	Relative humidity
windspeed	Wind speed

Table 1 – Data fields explanation of the dataset – Independent variables

2.2.2 Dependent variables

casual	Number of non-registered user rentals initialed
registered	Number of registered user rentals initialed
count	Number of total rentals

Table 2 – Data fields explanation of the dataset –Dependent variables

Applied Data Analytics II – Course Project
Date completed: 05/02/2020
Research question: Bike Sharing Demand
Document name: Project

III. Applied Data Analytics

3.1 Data exploration, analysis and applicable visualizations

3.1.1 Overall visualization

After initially comprehend the structure of the dataset (refer to original Jupyter notebook), and I need to predict hourly rentals count, I checked the relationship between casual count, registered count, total count and other features generally.

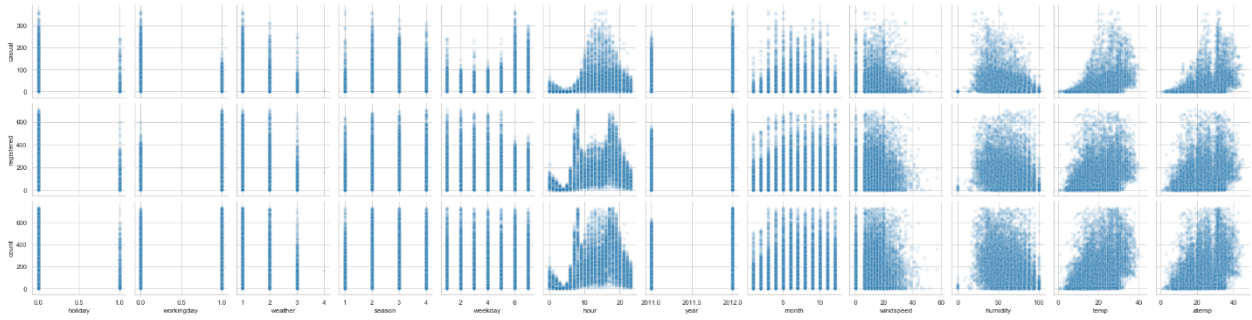

Chart 1

From the chart shown above, I found out that:

a. Registered users usually used the bikes in working days, while casual users tended to use the bikes in holidays.

b. The first quarter of the year witnessed the lowest rental numbers.

c. With the level of the weather being higher (we know that the higher the weather level is, the more terrible the weather represents), the rental numbers went down.

d. Hour had an obvious impact on rental numbers, with registered users reaching the highest around 7 to 9 am and 4 to 6 pm. However, casual users displayed normal distribution.

e. With the wind speed increasing, the rental numbers decreased.

f. Temperature and humidity factors affect casual users than registered users.

By creating the correlation matrix and heat map, the correlation between each feature and the rental counts from the strongest to the weakest were ranked (Negatively correlative relationship is consider to be stronger than positively correlative relationship) : Hour > Temperature (including "feel like temperature) > Humidity > Year > Month > Season > Weather > Wind speed > Weekday > working day> Holiday.

```
count         1.000000
registered    0.966209
casual        0.704764
hour          0.405437
temp          0.385954
atemp         0.381967
year          0.234959
month         0.164673
season        0.159801
windspeed     0.106074
weekday       0.022602
holiday       0.002978
workingday   -0.020764
weather      -0.127519
humidity     -0.317028
Name: count, dtype: float64
```

Chart 2

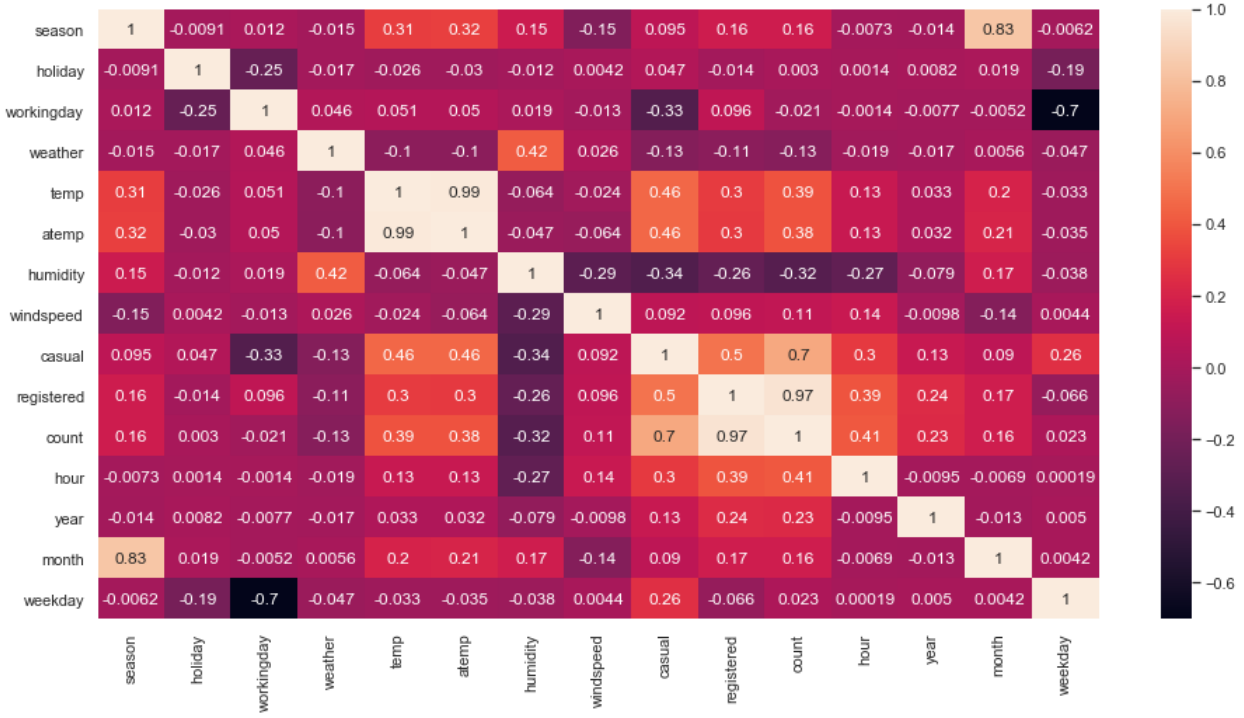

Chart 3

My inferences from the above heatmap:

a. Temp and atemp are highly related as expected.

b. Humidity is inversely related to count as expected due to the fact that people may be unwilling to travel on bike when the weather is humid, and it is likely to lead to terrible weather when it is extremely humid in the air.

c. Working day and number of casual user rentals are inversely related.

d. Holiday and count are inversely related, because most of the registered users rented the bikes in working days.

e. Temp (or atemp) has a huge effect on the count.

f. Weather and count are highly inversely related because the weather parameter implied that the weather was getting worse when it increased, which resulted in people rejecting to get out.

3.1.2 Individual exploration and analysis

3.1.2.1 The impact of hour on rental counts

Based on my previous analysis, hour factor had the most crucial impact on rental counts.

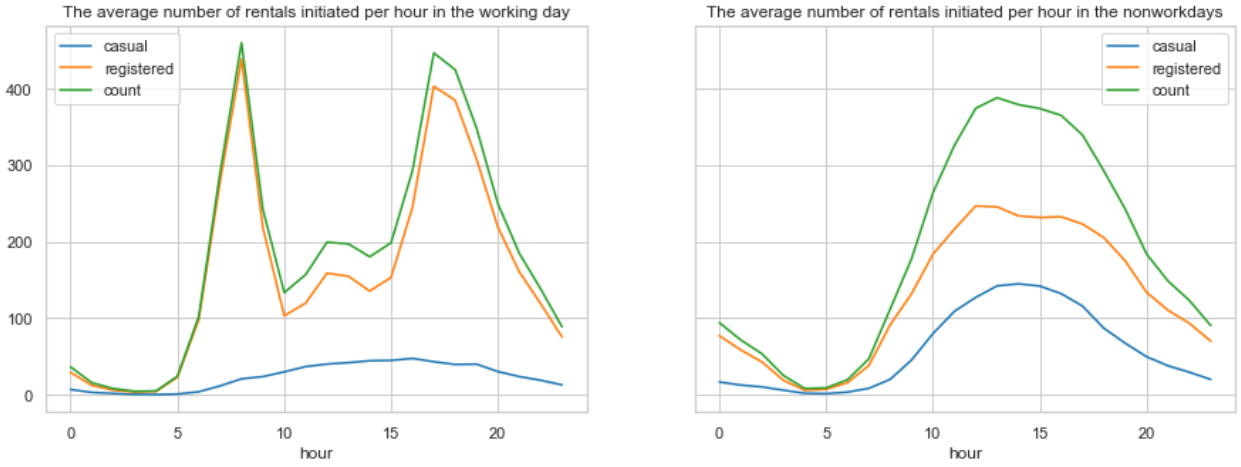

Chart 4

a. Based on the shown graphs, the registered users reached the peak to use the bikes from 7 to 9 am and from 4 to 6 pm in the working days. There also existed a slight increase from 11 am to 1 pm. I assumed that the registered users need to ride the bike to work (or to the subway or bus station) in the morning and to home (or to the subway or bus station) in the afternoon in the working days. What is more, some people might go out for lunch and use the bikes from 11 am to 1 pm.

b. Comparing with registered users, casual users showed a more stable change of number of rentals in working days, which arrived at the peak around 4 pm.

c. In comparison with distribution in the working days, the graph in the non-working days displayed normal distribution, with the peak being around 2 pm and the bottom being around 5 am.

d. Overall, the number of registered rentals tremendously surpassed the number of casual rentals.

3.1.2.2 The impact of temperature on rental counts

Firstly, I checked the change trend of average temperature per day in two years.

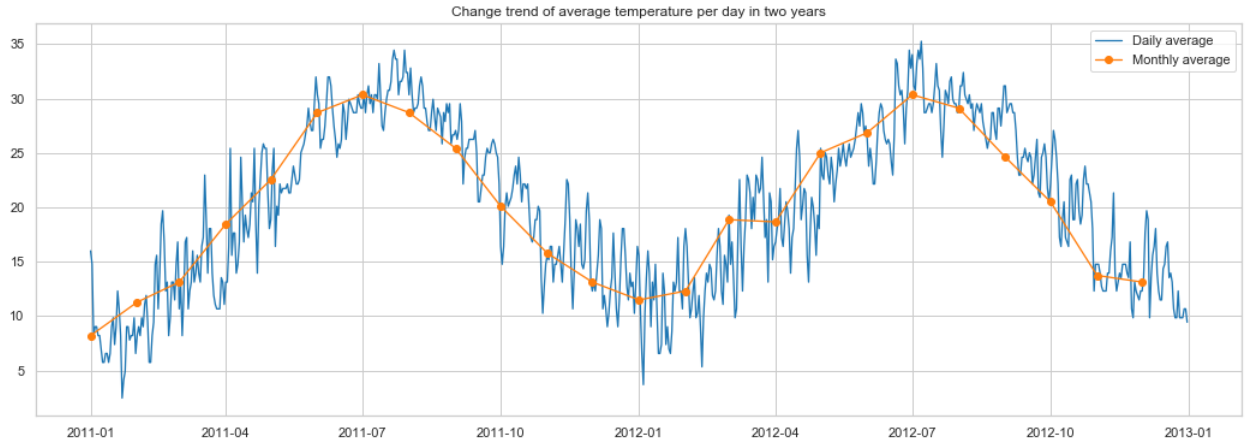

Chart 5

It is not difficult to find out that the highest temperature was in July while the lowest was in January, which was reasonable. I also checked the average number of rentals initiated per hour changes with the temperature.

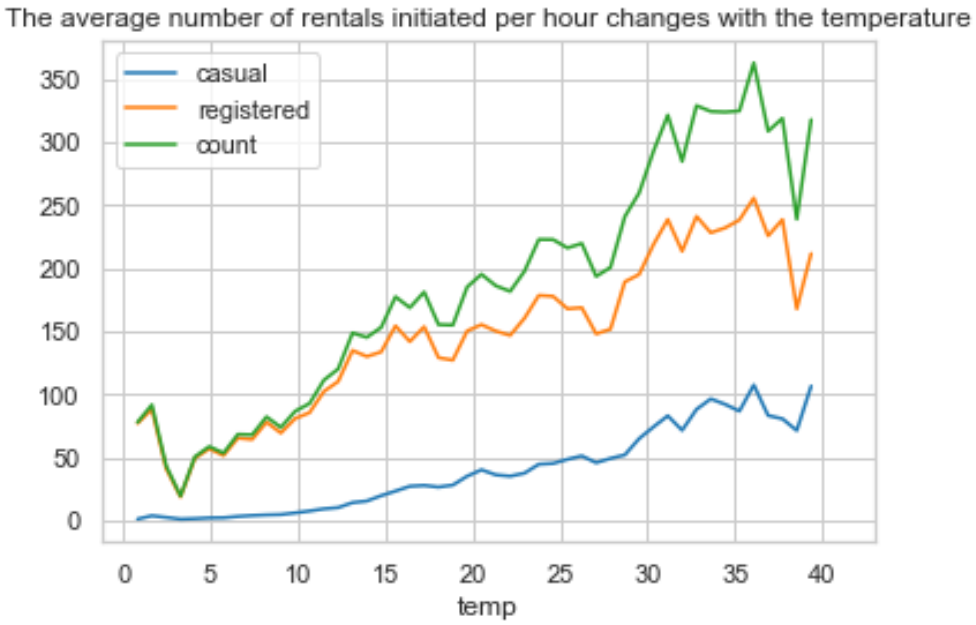

Chart 6

With the temperature going up, the number of rentals witnessed a slightly fluctuated increase. However, the number of rentals began to decrease after the temperature became higher than 35℃.

When temperature was around 4℃, the number of rentals reached the lowest level.

3.1.2.3 The impact of humidity on rental counts

I checked change trend of average humidity per day in two years in the following chart.

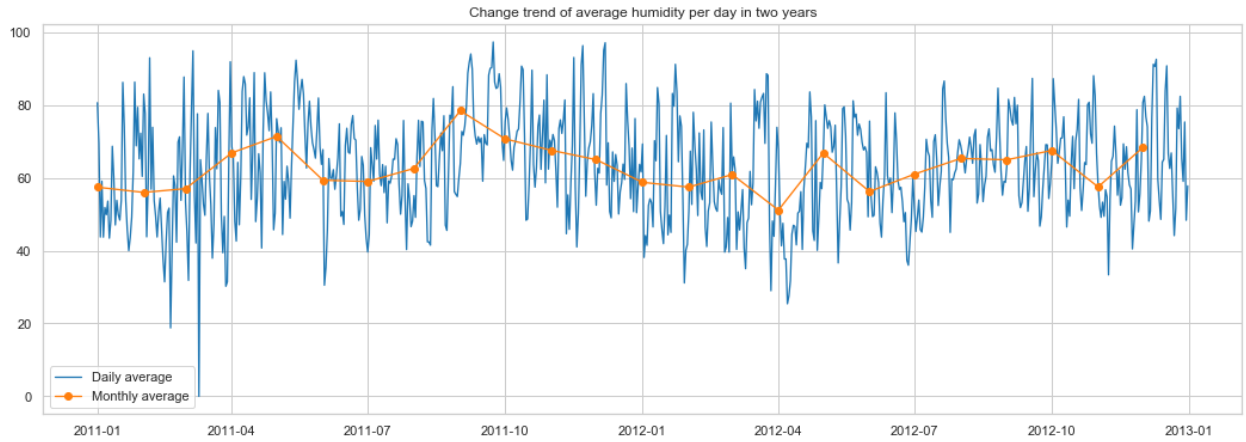

Chart 7

I checked average number of rentals initiated per hour in different humidity in the following chart.

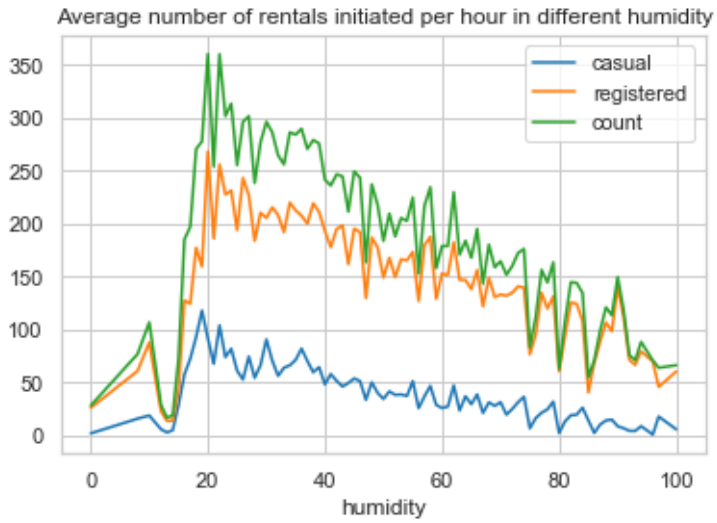

Chart 8

By observing the above chart, average number of rentals per hour increased swiftly from humidity of 0 to humidity of 20, yet it gently decreased when the humidity overweighed 20.

3.1.2.4 The impact of year, hour on rental counts

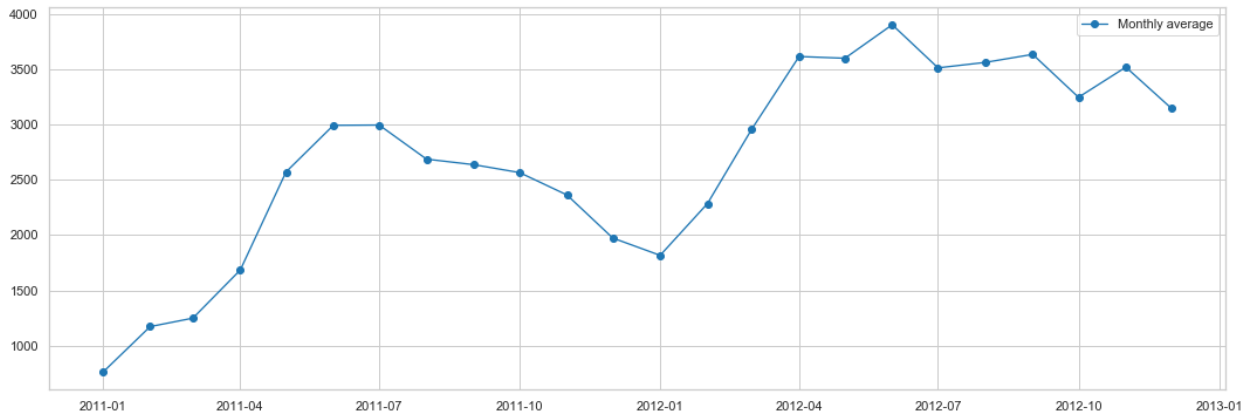

Chart 9

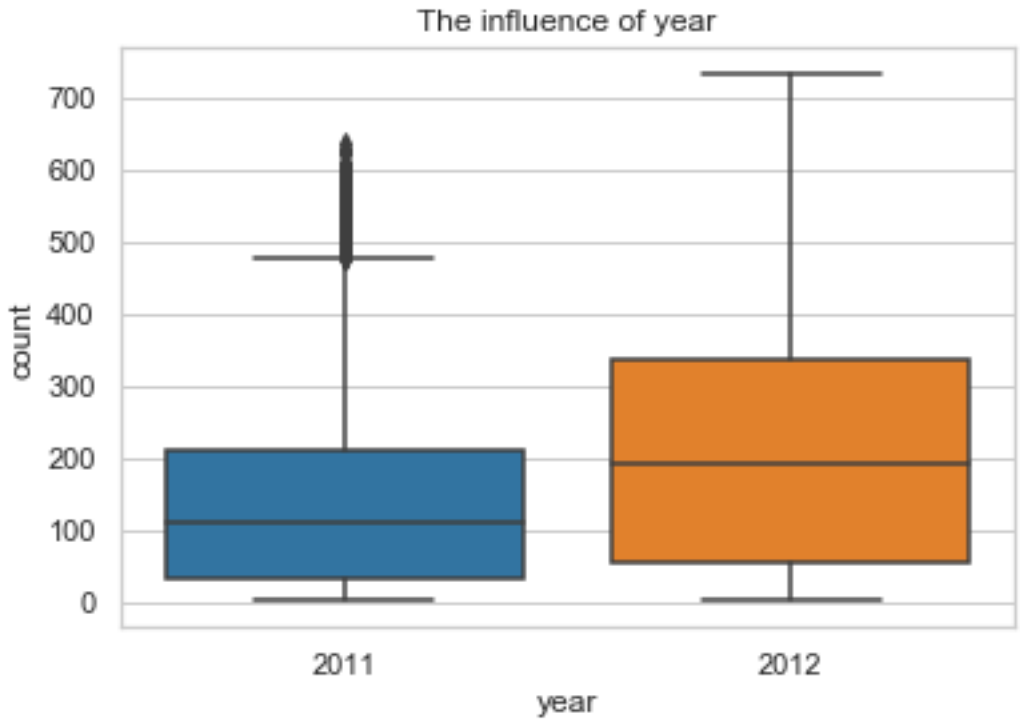

Chart 10

From the charts above, I noticed that:

a. The total number of rentals increased in 2012 compared to 2011, which indicated that an increasing number of people tended to use the sharing bikes as their transportation method.

b. The number of rentals increased sharply from January to June in 2011 and 2012.

3.1.2.5 The impact of season on rental counts

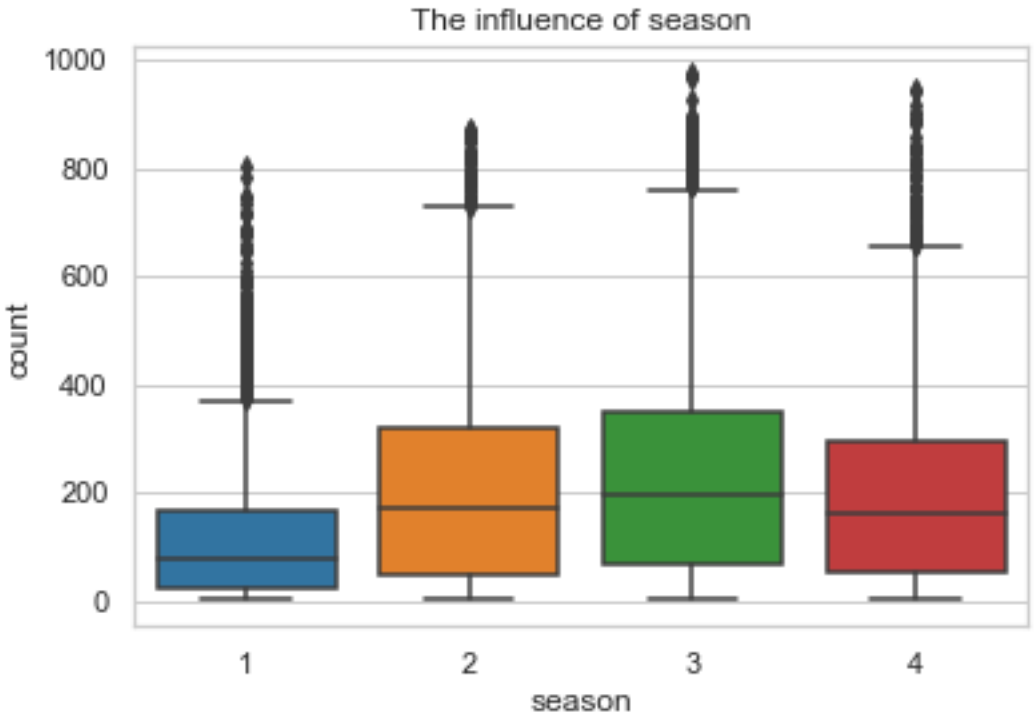

Chart 11

The box graph of season influence indicated the same tendency as the line chart of month influence. The following conclusions can be determined by the box graph.

a. The number of rentals arrived at the lowest level in the first quarter of the year, since the weather was the coldest and users were unwilling to ride bikes at that time.

b. With the weather being warmer and warmer, the number of rentals increased and reached to the highest level in the third quarter of the year, because fall is neither too hot nor too cold for riding bikes.

c. The number of rentals went down in the fourth quarter for the reason that the weather became colder again.

3.1.2.6 The impact of season on rental counts

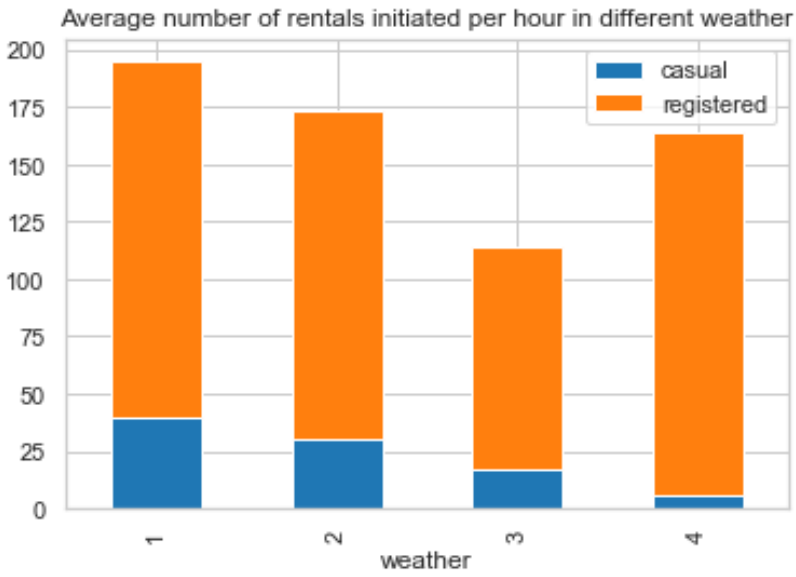

Chart 12

I assumed that with the weather level being higher, the average number of rentals initiated per hour should be lower, however, due to chart 11, there existed strange rise in weather level 4. By checking the detail (refer to code), I noticed that there was an outlier.

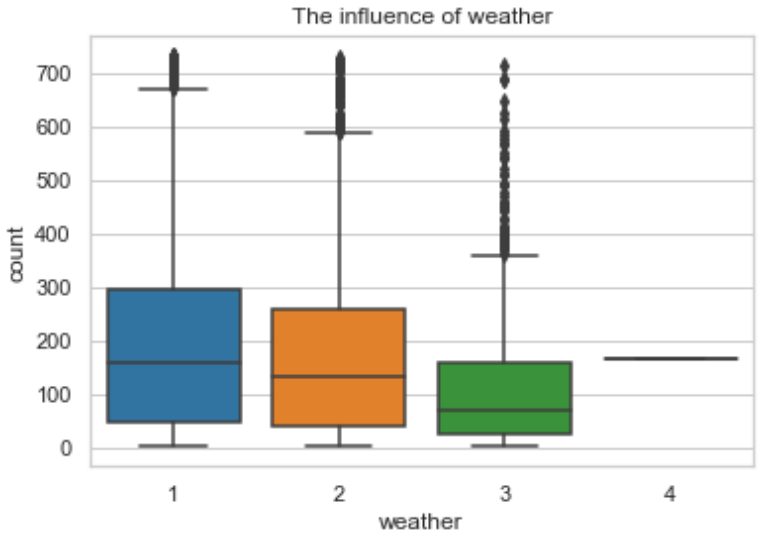

Chart 13

3.1.2.7 The impact of wind speed on rental counts

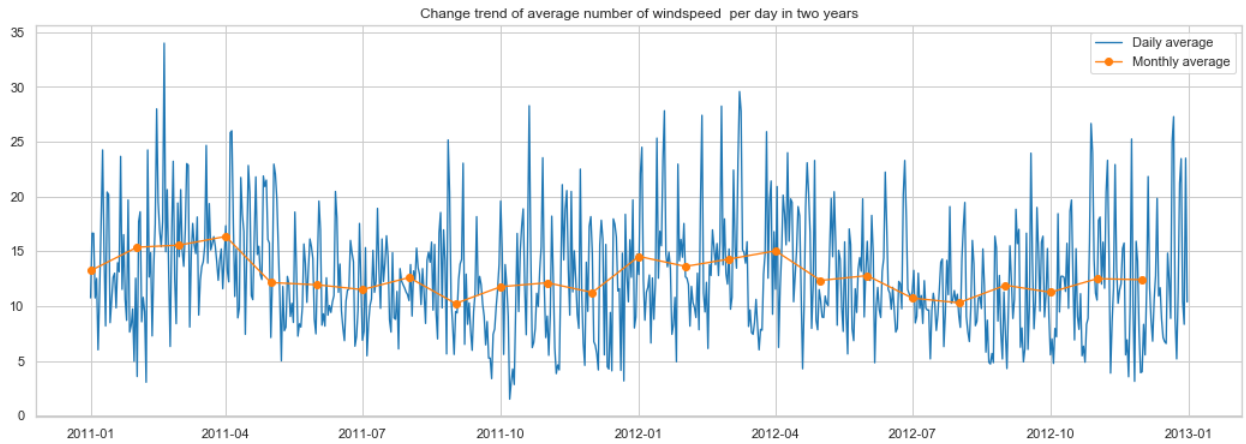

Chart 14

Considering that if the wind speed was extremely huge, it would have affected the chart and have resulted in abnormal situation, I adapted max number in this case.

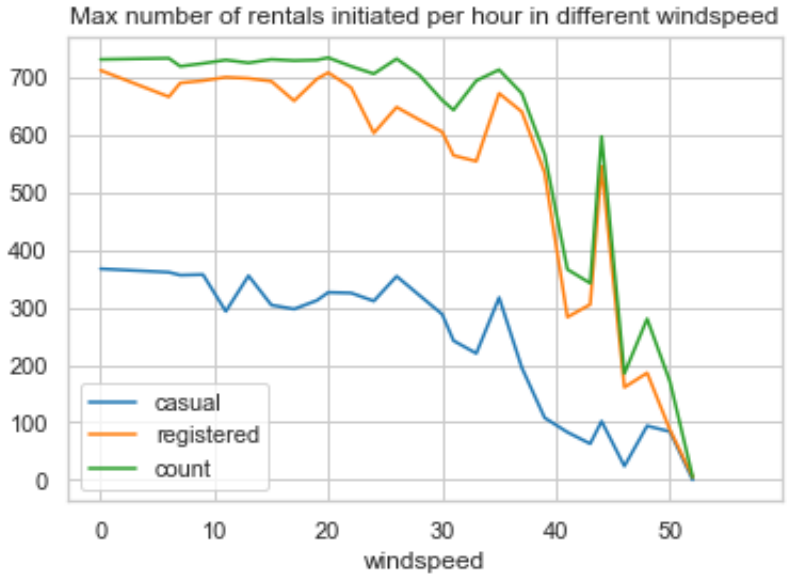

Chart 15

Based on chart 15, max number of rentals went down when wind speed increased. However, it suddenly went up rapidly around wind speed was 45, it could be outliers.

By checking the data, I noticed an outlier (refer to code).

3.1.2.8 The impact of working day on rental counts

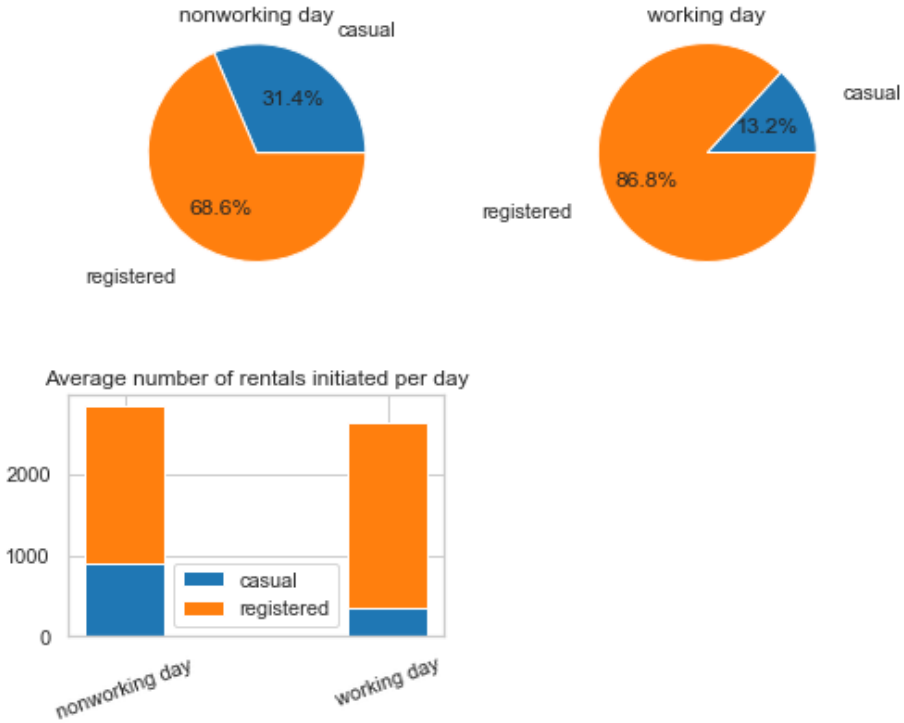

Chart 16

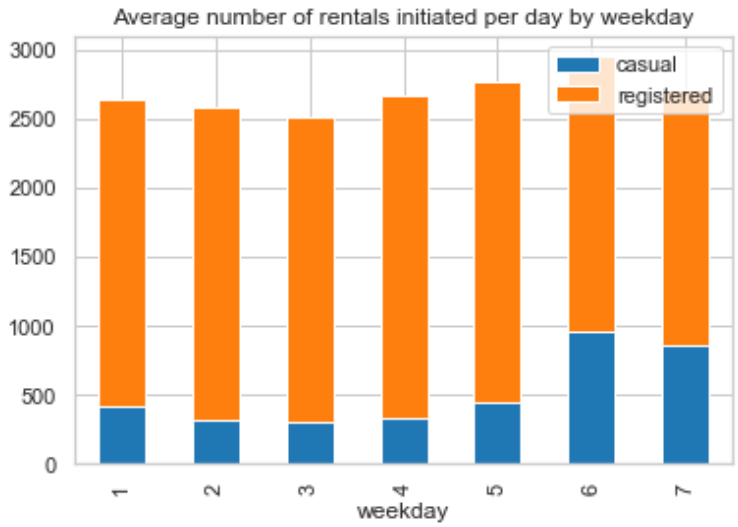

Chart 17

By comparing the above charts, I concluded that:

a. In the working days, registered users were more likely to use the bikes.

b. Even though the number of rentals by registered users surpassed crucially than the number of rentals by casual users, the number of rentals by casual users went up promptly in the non-working days.

3.1.2.9 The impact of holiday on rental counts

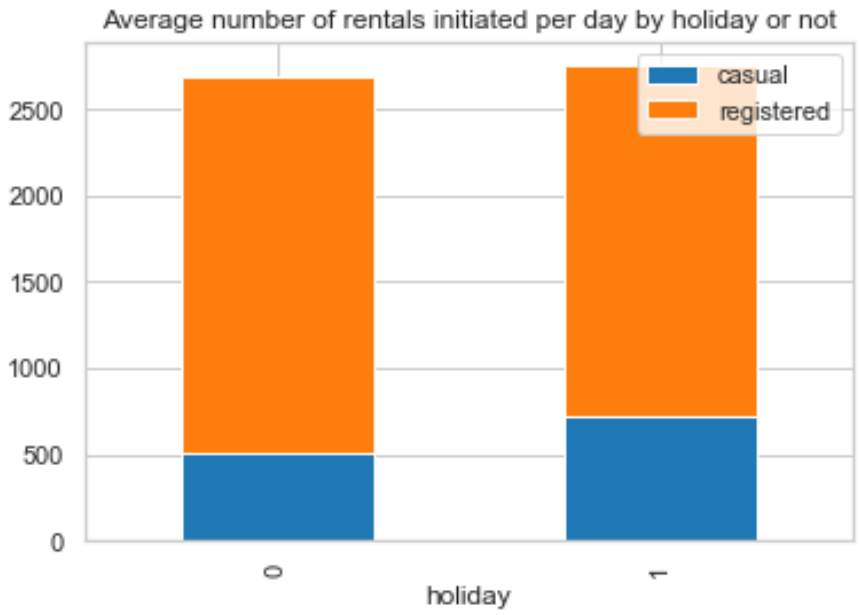

Chart 18

By counting the holidays in 2011 (10 days, refer to code) and in 2012 (11 days, refer to code), the proportion of average number of rentals initiated per day by holiday or not was shown as chart 18.

3.2 Data preparation

3.2.1 Check missing values

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10886 entries, 0 to 10885
Data columns (total 12 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   datetime    10886 non-null  object
 1   season      10886 non-null  int64
 2   holiday     10886 non-null  int64
 3   workingday  10886 non-null  int64
 4   weather     10886 non-null  int64
 5   temp        10886 non-null  float64
 6   atemp       10886 non-null  float64
 7   humidity    10886 non-null  int64
 8   windspeed   10886 non-null  float64
 9   casual      10886 non-null  int64
 10  registered  10886 non-null  int64
 11  count       10886 non-null  int64
dtypes: float64(3), int64(8), object(1)
memory usage: 1020.7+ KB
```

Chart 19 – Training data information

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6493 entries, 0 to 6492
Data columns (total 9 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   datetime    6493 non-null   object
 1   season      6493 non-null   int64
 2   holiday     6493 non-null   int64
 3   workingday  6493 non-null   int64
 4   weather     6493 non-null   int64
 5   temp        6493 non-null   float64
 6   atemp       6493 non-null   float64
 7   humidity    6493 non-null   int64
 8   windspeed   6493 non-null   float64
dtypes: float64(3), int64(5), object(1)
memory usage: 456.7+ KB
```

Chart 20 – Testing data information

The dataset does not contain missing values, however, it does not indicate that it does not contain outliers. As a matter of fact, I have noticed outliers in the previous analysis.

3.2.2 Check outliers

After using “describe” function, I noticed that the standard deviation of “count” was extremely large, which might indicate that the figures were likely to vary from each other hugely. As a consequence, I checked its distribution.

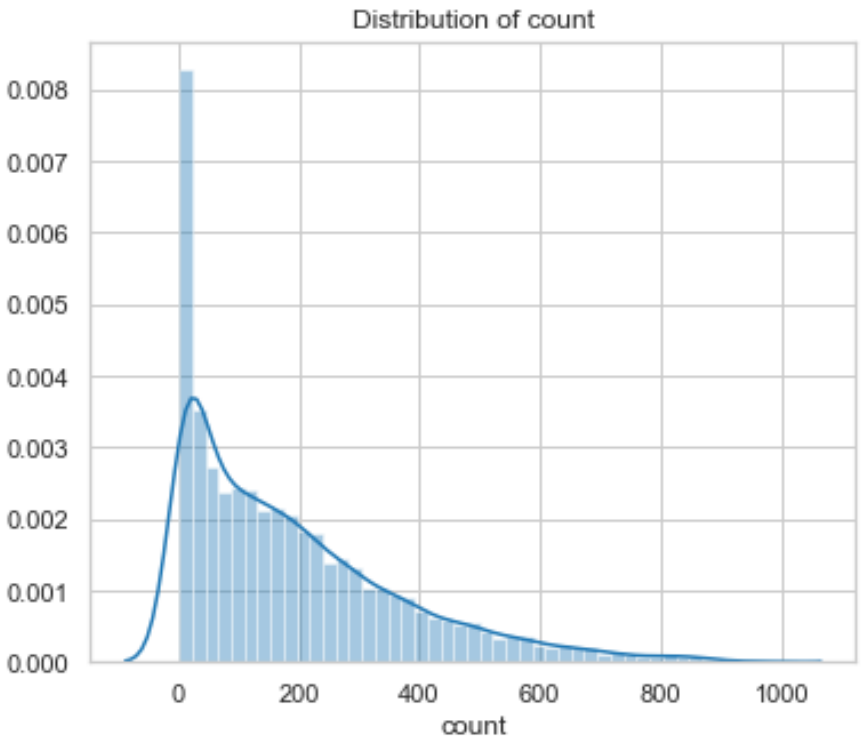

Chart 21

The distribution of “count” was skewed severely and tailed to the right. In order to deal with the tail, I excluded the figures out of three standard deviations.

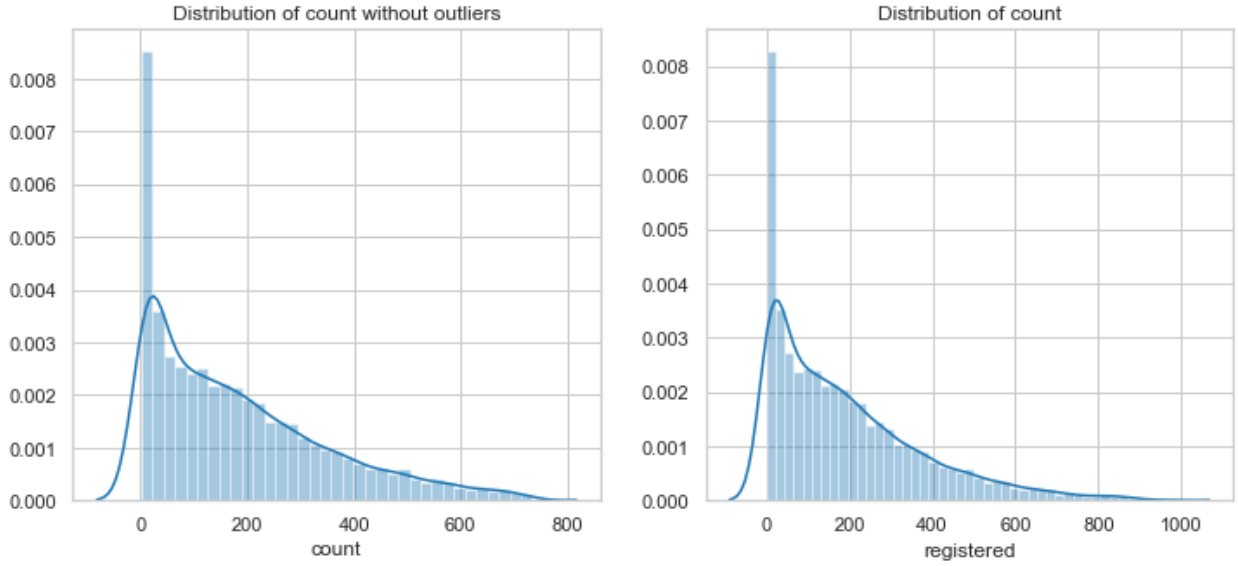

Chart 22

However, it cannot solve the problem. In order to avoid overfitting issue, I performed data transformation by applying log function and checked the new distribution diagram.

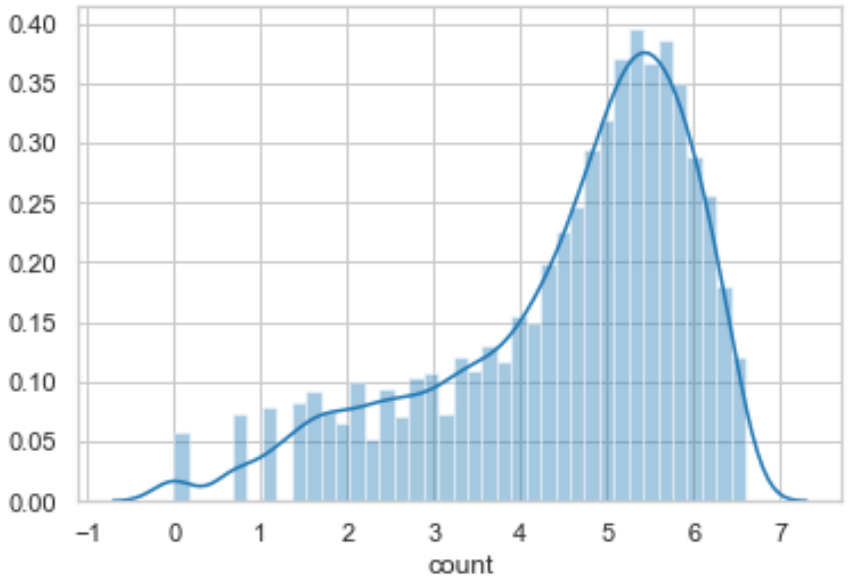

Chart 23

After applying log transformation towards the label (count), the data distributed more evenly, and the differences between the figures became smaller, which would be of great significance to train the models.

3.2.3 Remove a random 10% from the original dataset

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	count
0	2011-01-01 00:00:00	1	0	0	1	9.84	14.395	81	0.0000	3.0	13.0	16.0
1	2011-01-01 01:00:00	1	0	0	1	9.02	13.635	80	0.0000	8.0	32.0	40.0
2	2011-01-01 02:00:00	1	0	0	1	9.02	13.635	80	0.0000	5.0	27.0	32.0
3	2011-01-01 03:00:00	1	0	0	1	9.84	14.395	75	0.0000	3.0	10.0	13.0
4	2011-01-01 04:00:00	1	0	0	1	9.84	14.395	75	0.0000	0.0	1.0	1.0
...	...	...	...	...	...	...	...	...	...	...	...	...
17227	2012-12-31 19:00:00	1	0	1	2	10.66	12.880	60	11.0014	NaN	NaN	NaN
17228	2012-12-31 20:00:00	1	0	1	2	10.66	12.880	60	11.0014	NaN	NaN	NaN
17229	2012-12-31 21:00:00	1	0	1	1	10.66	12.880	60	11.0014	NaN	NaN	NaN
17230	2012-12-31 22:00:00	1	0	1	1	10.66	13.635	56	8.9981	NaN	NaN	NaN
17231	2012-12-31 23:00:00	1	0	1	1	10.66	13.635	65	8.9981	NaN	NaN	NaN

17232 rows × 12 columns

Chart 24

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	count
10055	2012-11-10 10:00:00	4	0	0	1	16.40	20.455	58	0.0000	86.0	264.0	350.0
12455	2011-07-26 12:00:00	3	0	1	1	34.44	36.365	28	0.0000	NaN	NaN	NaN
12521	2011-07-29 06:00:00	3	0	1	2	29.52	35.605	84	0.0000	NaN	NaN	NaN
2406	2011-06-08 04:00:00	2	0	1	1	25.42	28.790	83	0.0000	0.0	6.0	6.0
6823	2012-04-02 19:00:00	2	0	1	1	20.50	24.240	27	16.9979	66.0	428.0	494.0
...	...	...	...	...	...	...	...	...	...	...	...	...
10	2011-01-01 10:00:00	1	0	0	1	15.58	19.695	76	16.9979	12.0	24.0	36.0
14277	2012-02-21 05:00:00	1	0	1	1	8.20	12.880	69	0.0000	NaN	NaN	NaN
8406	2012-07-13 20:00:00	3	0	1	1	30.34	33.335	48	8.9981	95.0	294.0	389.0
5009	2011-12-02 19:00:00	4	0	1	1	13.94	15.910	53	16.9979	13.0	252.0	265.0
11281	2011-03-23 12:00:00	2	0	1	2	13.94	15.910	87	15.0013	NaN	NaN	NaN

15509 rows × 12 columns

Chart 25

In order to validate the data, I removed 10% of the data from the original dataset. As is shown in chart 24 and chart 25, after removing the outliers, there are 17,232 rows in the dataset. I selected 90% of the data (15,509 rows) randomly and formed a new dataset to perform feature engineering.

3.2.4 Time-format data processing and transformation

Since the data type of “datetime” was object, I split “datetime” into “date”, “hour”, “ year”, “month”, “weekday”.

3.2.5 feature selections

Based on the observations and analysis in the previous sections, I determined to involve in hour, temp, atemp, humidity, year, month, season, weather, wind speed, weekday, working day, holiday as independent variables to train the models.

3.2.6 Handle categorical data

Many machine learning algorithms cannot operate on label data directly. They require all input variables and output variables to be numeric, which means that categorical data must be converted to a numerical form.

In this dataset, “season”, “holiday”, “working day”, “weather”, “weekday”, “month”, “year”, “hour” are categorical features. I applied one-hot encoding to remove the integer encoded variable and added a new binary variable for each unique integer value.

3.2.7 Evaluation

I used Root Mean Squared Logarithmic Error (RMSLE) to evaluate the prediction result. Root Mean Square Logarithmic is the ratio (the log) between the actual values in the data and predicted values in the model. I select RMSLE instead of RMSE because in this case, under-prediction, which is likely to result in being lack of putting enough bikes into the market, is worse than an over-prediction. RMSLE can be calculated as the following formula.

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

3.3 Linear Regression

Linear Regression fits a linear model with coefficients $w = (w_1, ..., w_p)$ to minimize the residual sum of squares between the observed targets in the dataset, and the targets predicted by the linear approximation.

The RMSLE value for linear regression is 0.5556.

```
from sklearn.linear_model import LinearRegression,Ridge,Lasso
from sklearn.model_selection import GridSearchCV
from sklearn import metrics
import warnings
pd.options.mode.chained_assignment = None
warnings.filterwarnings("ignore", category=DeprecationWarning)

lModel = LinearRegression()

yLabelsLog = np.log1p(yLabels)
lModel.fit(X = dataTrain,y = yLabelsLog)

preds = lModel.predict(X= dataTrain)
print ("RMSLE Value For Linear Regression: ",rmsle(np.exp(yLabelsLog),np.exp(preds),False))
```

```
RMSLE Value For Linear Regression:  0.5556190747919039
```

Chart 26

3.4 Ridge Regression

Ridge regressor solves a regression model where the loss function is the linear least squares function and regularization is given by the l2-norm. Also known as Ridge Regression or Tikhonov regularization. This estimator has built-in support for multi-variate regression (i.e., when y is a 2d-array of shape (n_samples, n_targets)).

In ridge regression, the cost function is altered by adding a penalty equivalent to square of the magnitude of the coefficients. Therefore, ridge regression shrinks the coefficients and it helps to reduce the model complexity and multi-collinearity.

The RMSLE value for ridge regression is 0.5556.

```
RidgeModel = Ridge()

yLabelsLog = np.log1p(yLabels)
RidgeModel.fit(dataTrain,yLabelsLog)

preds = RidgeModel.predict(X= dataTrain)
print ("RMSLE Value For Ridge Regression: ",rmsle(np.exp(yLabelsLog),np.exp(preds),False))
```

```
RMSLE Value For Ridge Regression:  0.5555464793231847
```

Chart 27

3.5 Lasso Regression

Comparing with ridge regression, Lasso regression not only helps in reducing over fitting but it can help us in feature selection.

The RMSLE value for lasso regression is 1.0230.

```
LassoModel = Lasso()

yLabelsLog = np.log1p(yLabels)
LassoModel.fit(dataTrain,yLabelsLog)

preds = LassoModel.predict(X= dataTrain)
print ("RMSLE Value For Lasso Regression: ",rmsle(np.exp(yLabelsLog),np.exp(preds),False))
```

```
RMSLE Value For Lasso Regression:  1.0229624453514283
```

Chart 28

3.6 Random Forest

A random forest is a meta estimator that fits a number of classifying decision trees on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting.

Random Forest is widely used and always has higher accuracy.

The RMSLE value for random forest is 0.1042.

```
from sklearn.ensemble import RandomForestRegressor
rfModel = RandomForestRegressor(n_estimators=200,random_state=42)
yLabelsLog = np.log1p(yLabels)
rfModel.fit(dataTrain,yLabelsLog)
preds = rfModel.predict(X= dataTrain)
print ("RMSLE Value For Random Forest: ",rmsle(np.exp(yLabelsLog),np.exp(preds),False))
```

```
RMSLE Value For Random Forest:  0.10426971851401805
```

Chart 29

3.7 Gradient Boost

Gradient boosting regressors are a type of inductively generated tree ensemble model. At each step, a new tree is trained against the negative gradient of the loss function, which is analogous to (or identical to, in the case of least-squares error) the residual error.

The RMSLE value for gradient boosting regressors is 0.4105.

```
from sklearn.ensemble import GradientBoostingRegressor
gbm = GradientBoostingRegressor(n_estimators=100,alpha=0.01);
yLabelsLog = np.log1p(yLabels)
gbm.fit(dataTrain,yLabelsLog)
preds = gbm.predict(X= dataTrain)
print ("RMSLE Value For Gradient Boost: ",rmsle(np.exp(yLabelsLog),np.exp(preds),False))
```

```
RMSLE Value For Gradient Boost:  0.41054422370624716
```

Chart 30

3.8 Bagging Regression

The RMSLE value for Bagging Regressor is 0.1058, which indicates that bagging regressor can predict the y-value in a relative higher accuracy. The RMSLE is just a little higher than random regressor model.

```
from sklearn.ensemble import BaggingRegressor
bgr = BaggingRegressor(n_estimators=100);
yLabelsLog = np.log1p(yLabels)
bgr.fit(dataTrain,yLabelsLog)
preds = bgr.predict(X= dataTrain)
print ("RMSLE Value For BaggingRegressor: ",rmsle(np.exp(yLabelsLog),np.exp(preds),False))
```

```
RMSLE Value For BaggingRegressor:  0.10583439053851548
```

Chart 31

3.9 AdaBoost Regression

The RMSLE value for AdaBoost Regressor is 0.6553.

```
from sklearn.ensemble import AdaBoostRegressor
adr = AdaBoostRegressor(n_estimators=100);
yLabelsLog = np.log1p(yLabels)
adr.fit(dataTrain,yLabelsLog)
preds = adr.predict(X= dataTrain)
print ("RMSLE Value For AdaBoostRegressor: ",rmsle(np.exp(yLabelsLog),np.exp(preds),False))
```

```
RMSLE Value For AdaBoostRegressor:  0.6552913709364451
```

Chart 32

3.10 KNeighbors Regression

The RMSLE value for AdaBoost Regressor is 0.5676.

```
from sklearn.neighbors import KNeighborsRegressor

KNR = KNeighborsRegressor();
yLabelsLog = np.log1p(yLabels)
KNR.fit(dataTrain,yLabelsLog)
preds = KNR.predict(X= dataTrain)
print ("RMSLE Value For KNeighborsRegressor: ",rmsle(np.exp(yLabelsLog),np.exp(preds),False))
```

```
RMSLE Value For KNeighborsRegressor:  0.5676339799365299
```

Chart 33

3.11 Random Forest validation

Since random forest regression model has the lowest RMSLE among the above eight models, I used it to predict the data.

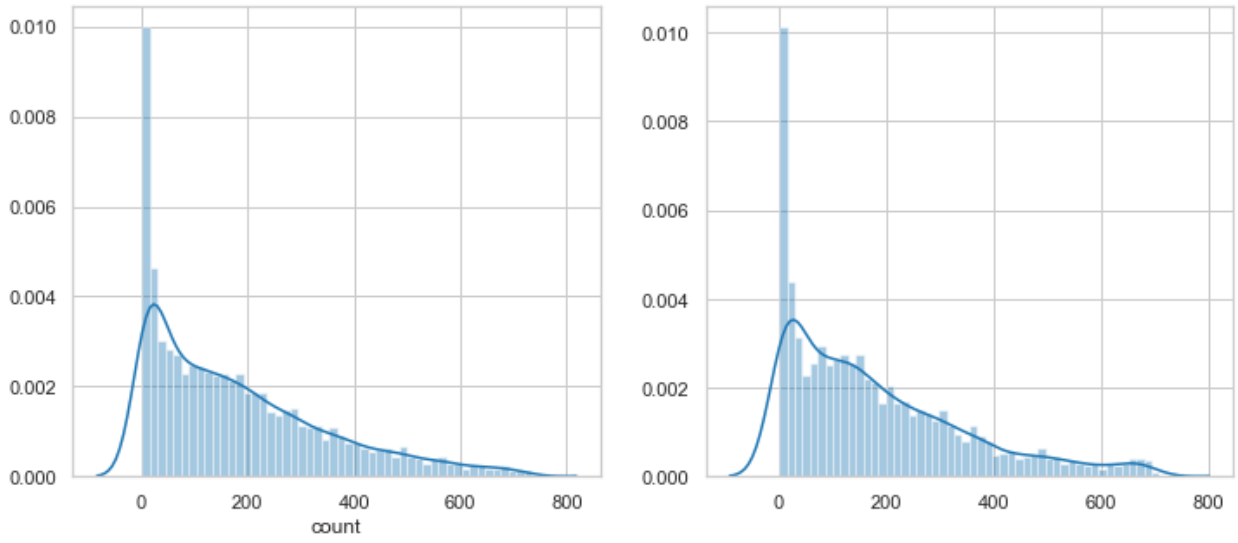

Chart 34

I compared the distribution diagram of training and test data, to a certain extent, they are quite identity to each other, which implies that random forest regression model is able to predict greatly and also avoid over-fitting issues.

IV. Conclusions

4.1 Conclusions

a. Random forest performs the best among the eight models I have applied. The RMSLE is not large and it basically predicts the dependent variables well.

b. Hours play the most significant part in bike usages. Temperature, season factors also have an enormous impact on the bike usages. If the enterprises can consider these factors to increase use ratio of sharing bikes, they will lower the costs and increase their profits.

c. Based on the two years' data, casual users are far fewer than registered users. If the companies can attract the casual users to use their services, they are able to increase demand.

4.2 Improvements

a. Data cleaning plays an indispensable role in machine learning, which may even affect the model results. This dataset is almost completed and need not to fill in missing values, which is marvelous. However, I merely handle y label outliers due to the time. If I can deal with the outliers of the wind speed or weather, the model may perform better.

b. Even though I analyzed the relationship pf each variable and the dependent variable, I took all the factors into consideration while building the models. However, some of the factors may not be strongly correlated to the dependent variables. What is more, some of the features can be combined together to train the model such as "temp" and "atemp", yet I did not perform this function.

c. Dimensionality reduction or MinMaxScaler is largely possible to improve the model. I did not perform these algorithms while I performed data preprocessing.

d. I used four ensemble algorithms in my notebook, I intended to use xgBoost, lightGBM and MLP Regressor but did not have enough time, these algorithms may perform better than random forest.

V. Appendix

5.1 Reference

[1] https://www.kaggle.com/fatmakursun/bike-sharing-feature-engineering

[2] https://www.datacamp.com/community/tutorials/categorical-data

[3] https://towardsdatascience.com/metrics-and-python-850b60710e0c

[4] https://stackoverflow.com/questions/46202223/root-mean-log-squared-error-issue-with-scitkit-learn-ensemble-gradientboostingre

[5] https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Ridge.html

[6] https://docs.scipy.org/doc/numpy/reference/generated/numpy.nan_to_num.html

[7] https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.notnull.html

[8] https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.get_dummies.html

[9] https://miamioh.instructure.com/courses/38817/pages/data-cleaning

[10] https://www.geeksforgeeks.org/python-pandas-dataframe-count/

[11] https://machinelearningmastery.com/why-one-hot-encode-data-in-machine-learning/

[12] https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OneHotEncoder.html

[13] https://www.w3resource.com/pandas/notnull.php

[14] https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Ridge.html

[15] https://towardsdatascience.com/ridge-and-lasso-regression-a-complete-guide-with-python-scikit-learn-e20e34bcbf0b

[16] https://en.wikipedia.org/wiki/Random_forest