edWiser

Bike Rental Prediction

Subodh Pradhan

October 2019

Table of Contents

Chapter -1

1. Introduction

A **bike rental** business rents out bicycles for short periods of time, usually for a few hours. Most rentals are provided by bike shops as a sideline to their main businesses of sales and service, but some shops specialize in rentals.

Bicycle rental shops primarily serve people who do not have access to a vehicle, typically travelers and particularly tourists. Specialized bicycle rental shops therefore typically operate at beaches, parks, or other locations those tourists frequent. In this case, the fees are set to encourage renting the bikes for a few hours at a time, rarely more than a day.

Other bike rental shops rent by the day or week as well as by the hour, and these provide an excellent opportunity for those who would like to avoid shipping their own bikes, but would like to do a multi-day bike tour of a particular area.

1.1 Problem statement

The Bike Rental Data contains the daily count of rental bikes between the year 2011 and 2012 with corresponding weather and seasonal information. Our objective of this Case is to predict the bike rental count on daily based on the environmental and seasonal settings to automate the system.

Our task is to build Regression model which will give the daily count of rental bikes based on weather and season.

1.2 Data

Table 1.1: Bike Rental Sample Data (Columns: 1-7)

instant	dteday	Season	yr	mnth	holiday	weekday
1	1/1/2011	1	0	1	0	6
2	1/2/2011	1	0	1	0	0
3	1/3/2011	1	0	1	0	1
4	1/4/2011	1	0	1	0	2
5	1/5/2011	1	0	1	0	3

weathersit	temp	atemp	hum	windspeed	casual	registered	cnt
2	0.344167	0.363625	0.805833	0.160446	331	654	985
2	0.363478	0.353739	0.696087	0.248539	131	670	801
1	0.196364	0.189405	0.437273	0.248309	120	1229	1349
1	0.2	0.212122	0.590435	0.160296	108	1454	1562
1	0.226957	0.22927	0.436957	0.1869	82	1518	1600

Table 1.2: Bike Rental Sample Data (Columns: 8-14)

The variables in the dataset are instant, dteday, season, yr, mnth, holiday, weekday, weathersit, temp, atemp, hum, windspeed, casual, registered and cnt.

The details of data attributes in the dataset are as follows –

instant: Record index
dteday: Date
season: Season (1:springer, 2:summer, 3:fall, 4:winter)
yr: Year (0: 2011, 1:2012)
mnth: Month (1 to 12)
hr: Hour (0 to 23)
holiday: weather day is holiday or not (extracted fromHoliday Schedule)
weekday: Day of the week
workingday: If day is neither weekend nor holiday is 1, otherwise is 0.
weathersit: (extracted from Freemeteo)
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: Normalized temperature in Celsius. The values are derived via (t-t_min)/(t_max-t_min),
t_min=-8, t_max=+39 (only in hourly scale)
atemp: Normalized feeling temperature in Celsius. The values are derived via (t-t_min)/(t_maxt_
min), t_min=-16, t_max=+50 (only in hourly scale)
hum: Normalized humidity. The values are divided to 100 (max)
windspeed: Normalized wind speed. The values are divided to 67 (max)
casual: count of casual users

Chapter-2

Methodology

2.1 Data pre processing

Any predictive modeling requires that we look at the data before we start modeling. However, in data mining terms *looking at data* refers to so much more than just looking. Looking at data refers to exploring the data, cleaning the data as well as visualizing the data through graphs and plots. This is often called as **Exploratory Data Analysis**. To start this process we will first try and look at all the distributions of the Numeric variables. Most analysis like regression, require the data to be normally distributed.

2.1.1 Exploratory Data Analysis

In the dataset the there are 731 observations and 15 variables. Out of them, cnt, registered and casual (cnt is the total counts of bike rental which is the sum of registered and casual) are the target features.
All the categorical features are in integer format and the numerical features are in normalized form.
So in Exploratory data Analysis,

- We have converted the categorical features (mnth, yr, season, weathersit, holiday, weekday) into factor numeric format.
- Extracted the day number of the month from dteday and converted to categorical variable having 31 levels as a month has maximum 31 days.
- Dropped the instant variable which is nothing but only a index number or serial number.
- Dropped the registered and the casual variable because since we have to predict the total count of the bike rental **cnt**(total count) which is the sum of registered and the casual variable.
- In the attributes 'holiday' and 'workingday', we can say that holiday is a subset of workingday because if it is a holiday then it is not workingday and vice-versa. so we have dropped the holiday variable from the dataset.

After dropping these variables, the rest variables are ***dteday, season, yr, mnth, weekday, workingday, weathersit, temp, atemp, hum, windspeed, cnt.***

Dteday, season, yr, mnth, weekday, workingday and weathersit are the categorical variables and temp, atemp, hum, windspeed are the numerical features and cnt is our numerical target feature.

2.1.2 Univariate Analysis

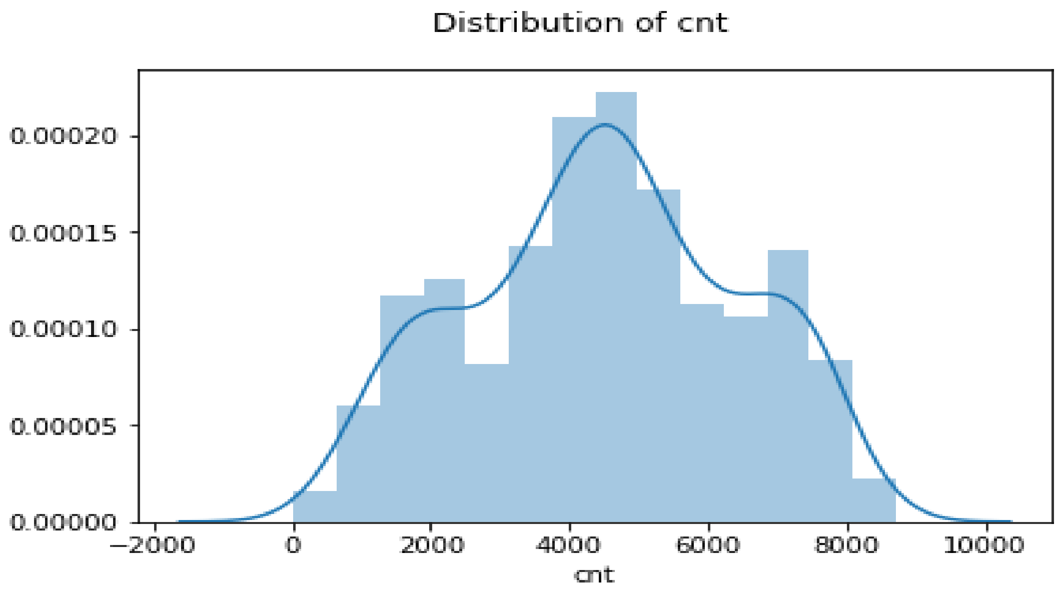
In univariate analysis we will check the behavior and distribution of numerical variables.

First we will check the summary of the target variable cnt.

```
count      731.000000
mean      4504.348837
std       1937.211452
min         22.000000
25%       3152.000000
50%       4548.000000
75%       5956.000000
max       8714.000000
Name: cnt, dtype: float64
```

Below are the distribution graphs of the numerical variables.

Fig 2.1 distribution of the target variable 'cnt'



```
skewness = -0.04735278011900183
kurtosis = -0.8119223847191548
```

The skewness of the cnt is -0.047 and the kurtosis is -0.811 which is closed to 0 so we can say that the variable cnt is fairly symmetrical or normally distributed.
The formula of skewness is

Skewness = 3(mean-median)/standard deviation

In python we can directly calculate the skewness and kurtosis by the functions ***skew()*** *for skewness and* ***kurt()*** *for kurtosis* which are present in **scipy.stats** library.
And in R we can calculate the skewness and kurtosis by using the library **(e1071).**

Skewness*:*

It is the degree of distortion *from the symmetrical bell curve or the normal distribution. It measures the lack of symmetry in data distribution.*
It differentiates extreme values in one versus the other tail. A symmetrical distribution will have a skewness of 0.

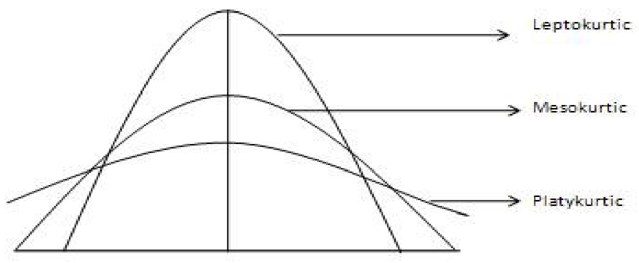
- If the skewness is between -0.5 and 0.5, the data are fairly symmetrical.

- If the skewness is between -1 and -0.5(negatively skewed) or between 0.5 and 1(positively skewed), the data are moderately skewed.
- If the skewness is less than -1(negatively skewed) or greater than 1(positively skewed), the data are highly skewed.

Kurtosis:

Kurtosis is all about the tails of the distribution — not the peakedness or flatness. It is used to describe the extreme values in one versus the other tail. It is actually the measure of outliers present in the distribution.

- **High kurtosis** in a data set is an indicator that data has heavy tails or outliers. If there is a high kurtosis, then, we need to investigate why do we have so many outliers. It indicates a lot of things, maybe wrong data entry or other things. Investigate!
- **Low kurtosis** in a data set is an indicator that data has light tails or lack of outliers. If we get low kurtosis(too good to be true), then also we need to investigate and trim the dataset of unwanted results.

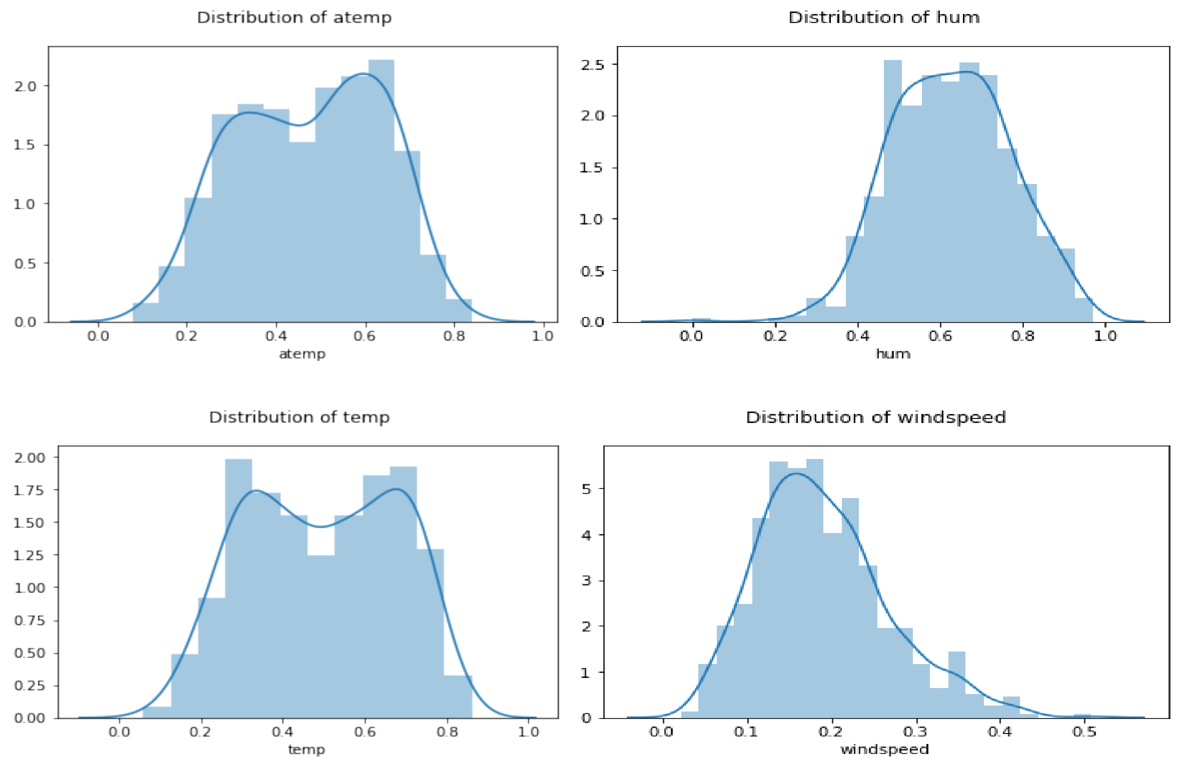


Mesokurtic: This distribution has kurtosis statistic similar to that of the normal distribution. It means that the extreme values of the distribution are similar to that of a normal distribution characteristic. This definition is used so that the standard normal distribution has a *kurtosis of three.*
Leptokurtic (*Kurtosis > 3*): Distribution is longer, tails are fatter. Peak is higher and sharper than Mesokurtic, which means that data are heavy-tailed or profusion of outliers.
Outliers stretch the horizontal axis of the histogram graph, which makes the bulk of the data appear in a narrow (“skinny”) vertical range, thereby giving the “skinniness” of a leptokurtic

distribution.
Platykurtic: (*Kurtosis* < *3*): Distribution is shorter; tails are thinner than the normal distribution. The peak is lower and broader than Mesokurtic, which means that data are light-tailed or lack of outliers.
The reason for this is because the extreme values are less than that of the normal distribution

- ***Distribution of the numerical independent variables :***

Fig 2.2 Distribution of independent variables



The skewness and the kurtosis of the numeic independent variable are given in the table below.

Variable name	Skewness	Kurtosis
temp	-0.054520964760408255	-1.1188641545735662
atemp	-0.13108804205446004	-0.9851305305195086
hum	-0.0697834339909521	-0.06453013469388669
windspeed	0.6773454211095377	0.4109222677315345

From the above plots and skewness we can see that the numerical independent variables temp and atemp are normally distributed and the hum is negatively or left skewed and the windspeed is positively or right skewed.

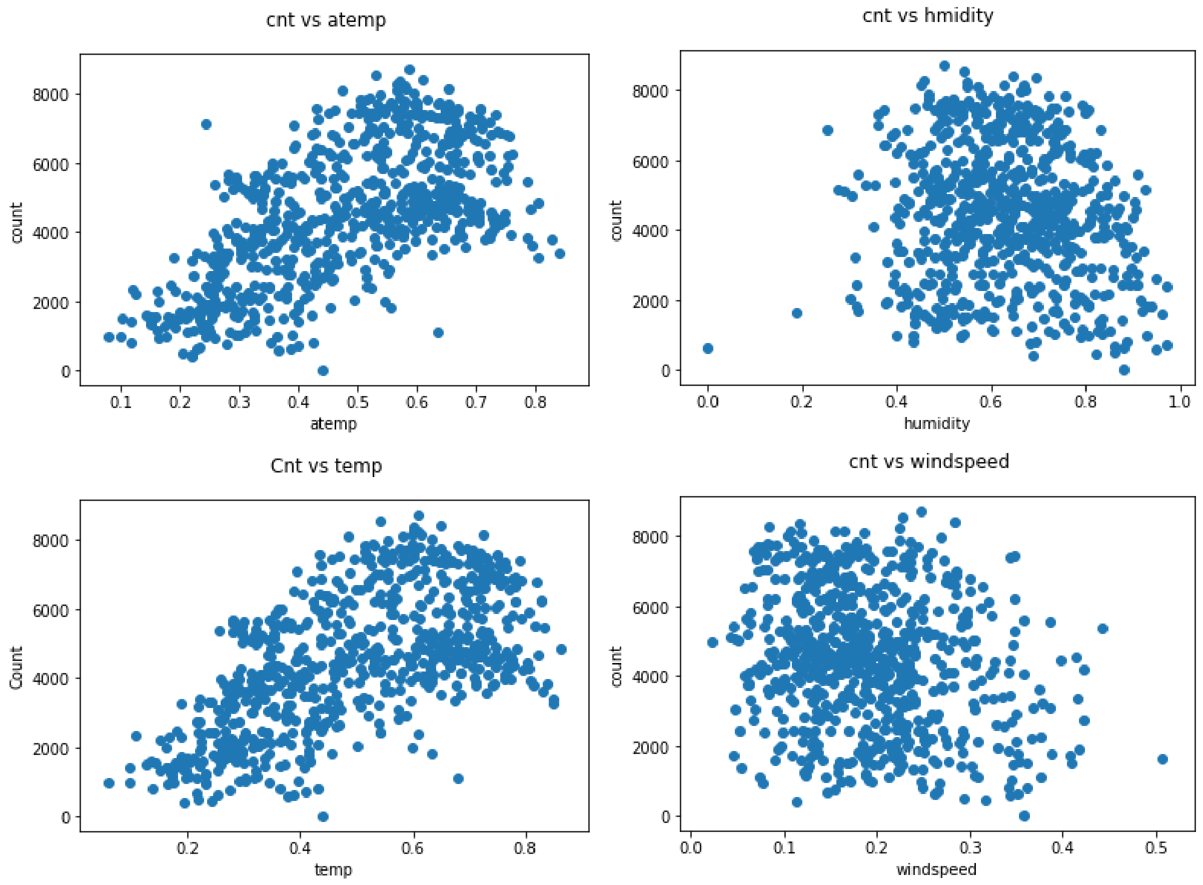
It seems to b some outliers present in the hum and windspeed but all the numeric independent variables are already in normalized form. We will detect and remove the outiers if any in the outlier analysis step.

2.1.3 Bivariate Analysis

In Bivariate analysis we will determine the relationship between the target variable cnt and the independent variables.

First we will check the relation between the numeric independent variables (temp, atemp, hum and windspeed) and the target variable cnt.

Fig 2.3 relation between cnt and the numerical independent features



The graph 2.3 shows that when the temp and atemp are between 0.1 – 0.8 more number of bikes are rented. When the humidity is between 0.4 – 0.9 there are more rental of bikes and when the windsped is between 0.1 - 0.4 the counts of rental is high but when the humidity is less than 0.4 and windspeed is greater than 0.4   the bike rental counts are very less which tells that some outliers may present in the variables hum and windspeed.

Now we will check the relationship between the categorical variables and the target variable cnt using boxplot.

Boxplot : *boxplot is a method for graphically depicting groups of numerical data through their quartiles. Box plots may also have lines extending vertically from the boxes (whiskers) indicating variability outside the upper and lower quartiles*

1. Relation between the categorical variable season and the target variable cnt.

Fig 2.4 relation between season and cnt

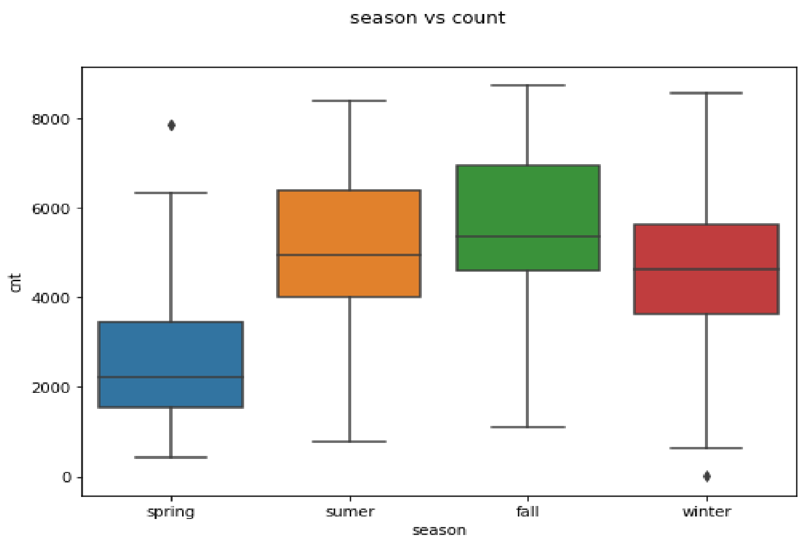


Fig 2.4 shows that the as compare to summer, fall and winter , the count of bike rental is very less in spring and the demand of bike rental is high in fall season.

2. Relation between the categorical variable workingday/holiday and the target variable cnt.

Fig 2.5 relation between workingday/holiday and cnt

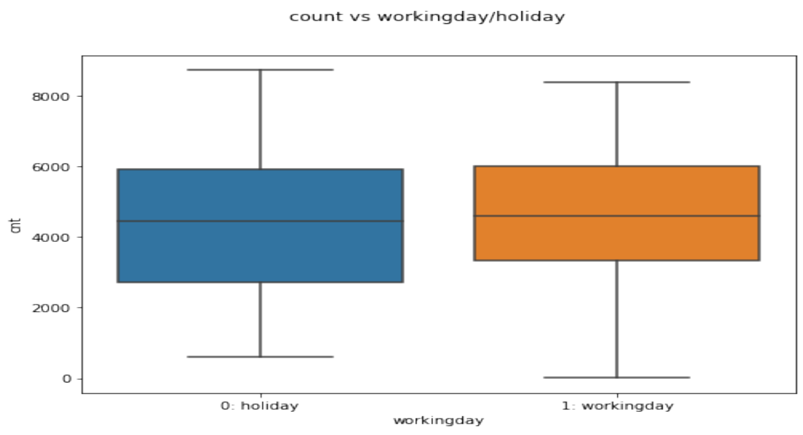
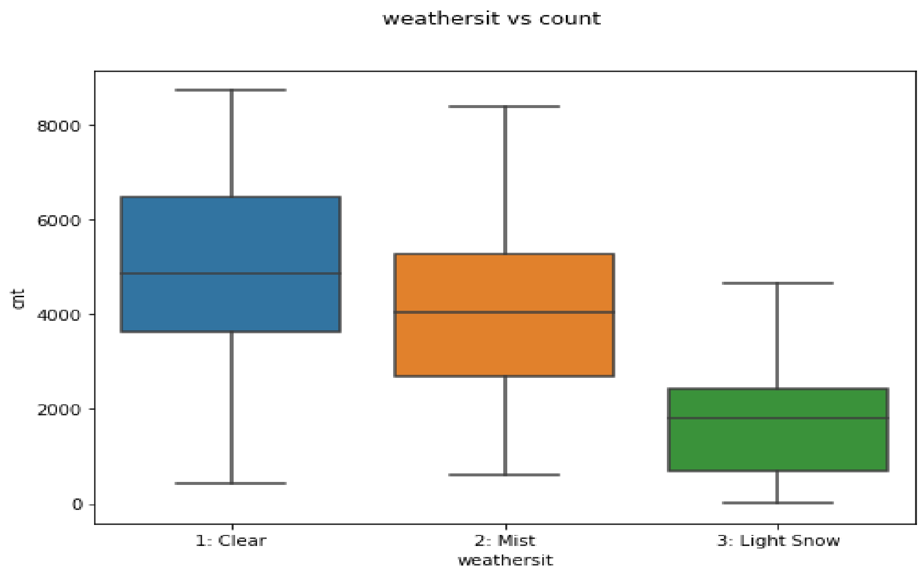


Fig 2.5 shows that the counts of bike rental is quite high on working days as compare to holidays.

3. Relation between the categorical variable weathersit and the target variable cnt.

Fig 2.6 relation between weathersit and cnt

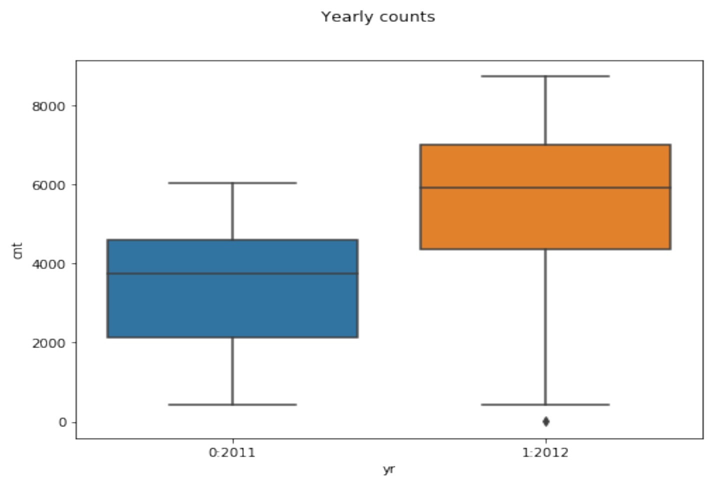


From the fig 2.6 we can see that if the weather is clear then the count of bike rental is very high and peoples do not preferring to use bikes when the weather is rainy.

When the weather is light snow then fewer people are using bikes and when weather is clear and mist then the demand of bike is high.

4. Relation between the categorical variable year and the target variable cnt.

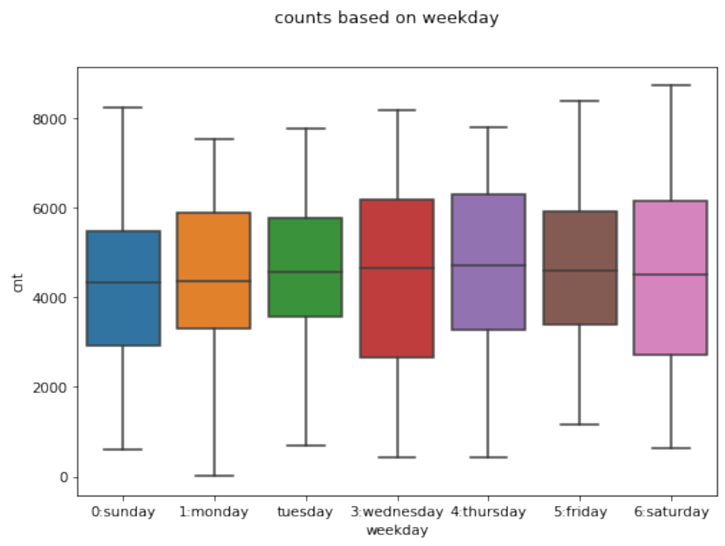
Fig 2.7 counts of bikes based on year



Here in fig 2.7 if we compare the counts of bikes rented between the year 2011 and 2012, there are more bikes were rented in the year of 2012. That means gradually peoples are preferring bikes for their transportation.

5. Relation between the categorical variable weekday and the target variable cnt.

Fig 2.8 counts of bikes based on weekdays



In fig 2.8 which shows that the counts of bike rental based on weekdays, we can see that there s not much difference in the counts based on weekdays but in Sunday the counts of bike rental is quite less than rest days.

6. Relation between the categorical variable mnth and the target variable cnt.

Fig 2.9 counts of bikes based on months

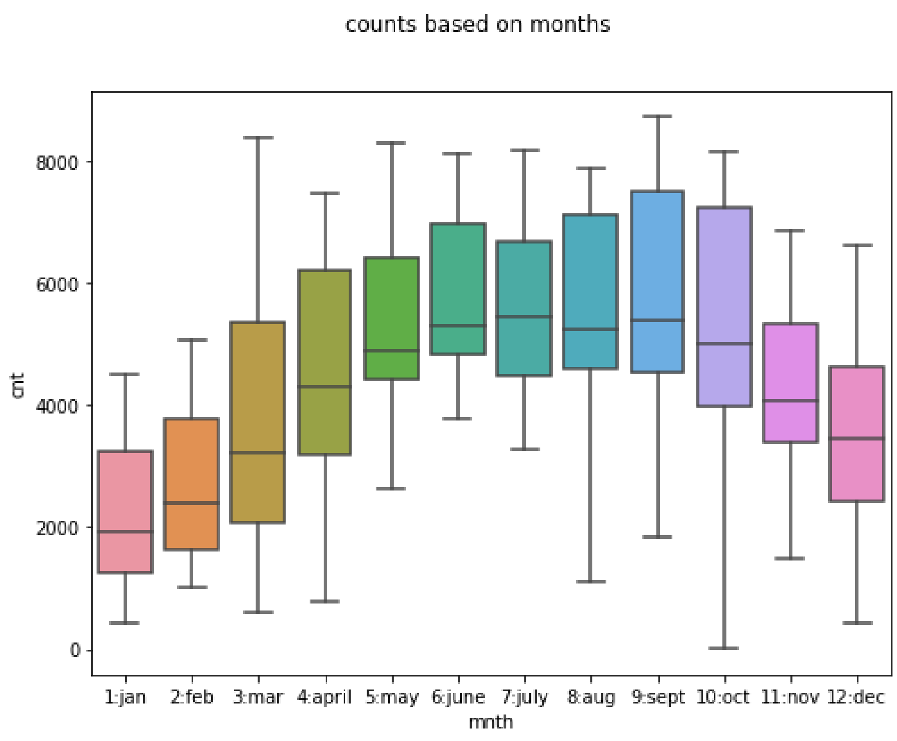


Fig 2.9 shows the counts of bike rental based on months. This plot shows that the demand of bike is very high in July and September and the demand of bikes rental in June, August and October are nearly same.

If we see the demand of bike rental from January to July, it is gradually increasing and from September to December it is gradually decreasing. As compare to all months, there less demand of rental bikes in January.

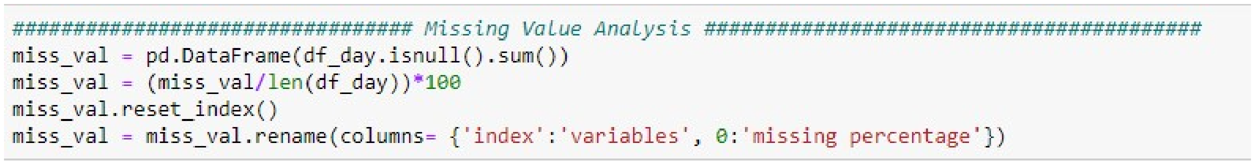
2.1.4 Missing value analysis

Missing values occur when no data value is stored for the variable in an observation. Missing data are a common occurrence and can have a significant effect on the conclusions that can be drawn from the data.

Missing value analysis is done to check whether any missing value presents in the given dataset or not. Missing values can be treated using various method like mean, median, mode, KNN imputation and prediction method to impute missing values.

In python ***isnull().sum()*** function and in R **sum**(***is.na()***) function is used to check the sum of missing values in a dataset.

Python output of missing value output is given in the below table.



```
################################### Missing Value Analysis ###########################################
miss_val = pd.DataFrame(df_day.isnull().sum())
miss_val = (miss_val/len(df_day))*100
miss_val.reset_index()
miss_val = miss_val.rename(columns= {'index':'variables', 0:'missing percentage'})
```

variables	Missing percentage
dteday	0.0
season	0.0
yr	0.0
mnth	0.0
weekday	0.0
workingday	0.0
weathersit	0.0
temp	0.0
atemp	0.0
hum	0.0
windspeed	0.0
cnt	0.0

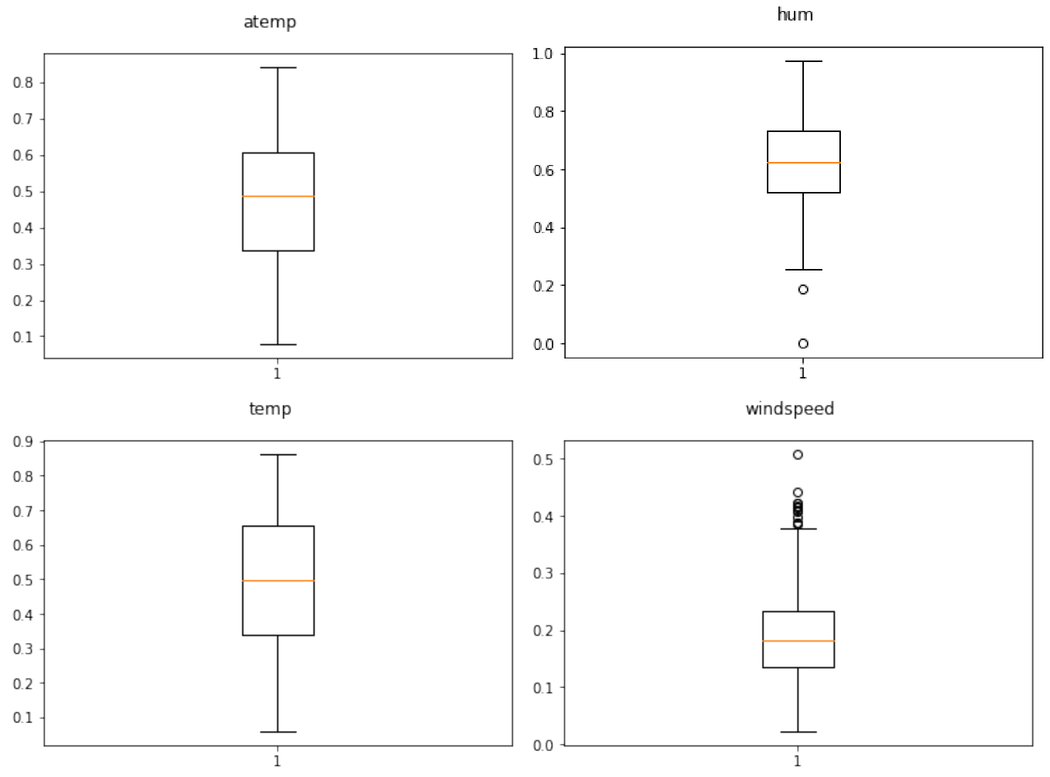
There are no missing values in any variables of the given dataset. So we need not to impute or delete any value.

2.1.5 Outlier Analysis

An outlier is an observation which is inconsistent with rest of the dataset. It is an observation which lies an abnormal distance from the other values in the dataset or any value is falling away from the actual bunch is count as an outlier. So we will perform outlier analysis to handle all inconsistent observations present in the given dataset.

Outliers can be present only on continuous variables and not in categorical variable so we will check outliers only for the continuous variables i.e temp, atemp, windspeed, hum and cnt by using box plot.

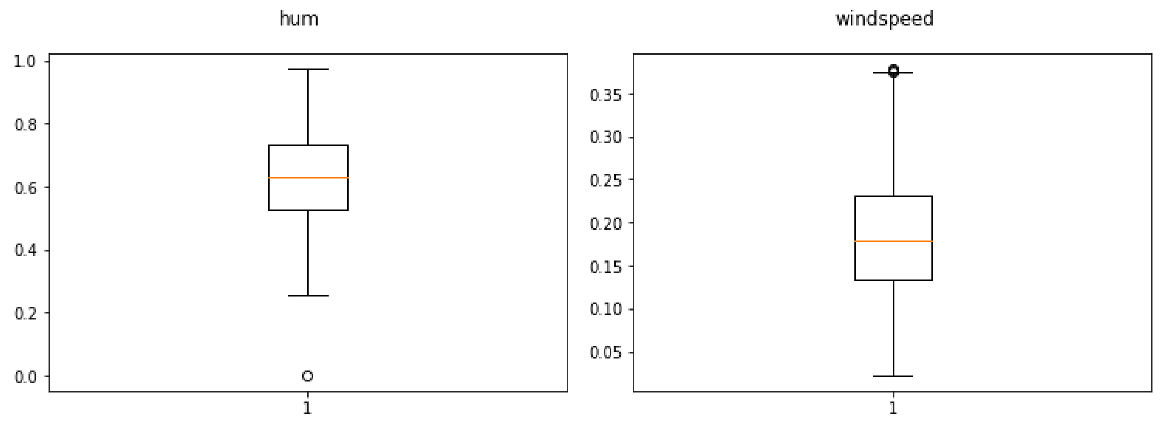
Fig 2.10 Box plots of numerical variables



From the above fig 2.10 we can see that there are no outliers in temp and atemp. It seems to be some negative outliers present in hum and some positive outliers in windspeed. As windspeed defines the speed of wind on a particular day and humidity defines the humidity of a day so we can remove these outliers because both variable defines about environmental condition.

After removing outliers from numerical variables, box plots of hum and windspeed is given below.

Fig 2.11 Box plots of hum and windspeed



Above fig 2.11 shows that the boxplots of hum and windspeed after removing outliers.

2.1.6 Feature selection

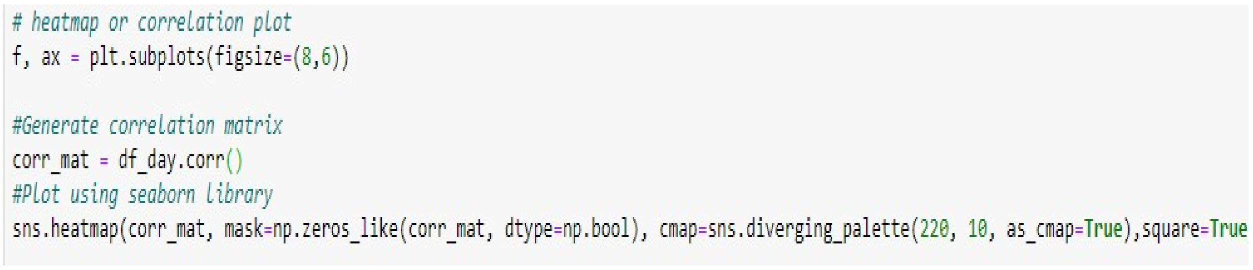
Selecting a subset of relevant and meaningful features from the dataset which contributes much information to develop a model is called feature selection. The agenda of feature selection is to identify and remove a needed or irrelevant attribute from the data that do not contribute much information.

There are different methods to reduce the dimensions of data or based on which we can select or drop variables.

1. Correlation analysis (For numerical continuous variable)
2. Chi-square test of independence (For categorical variable)

Since our target variable is a continuous variable so we will go for only correlation analysis.

Python code for Correletion plot

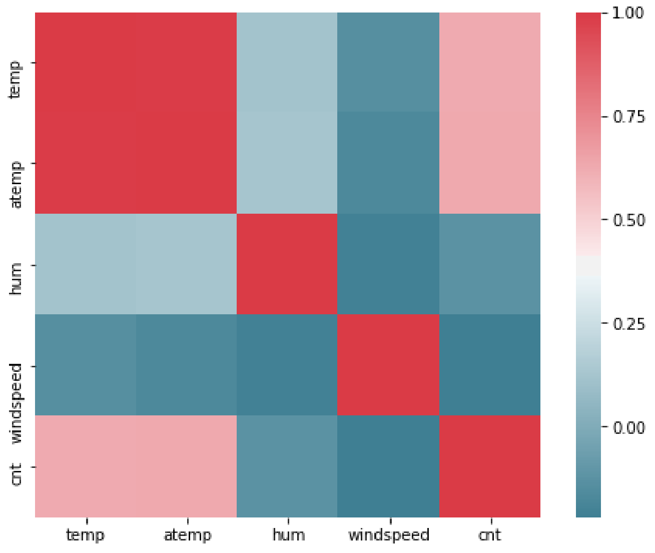


```
# heatmap or correlation plot
f, ax = plt.subplots(figsize=(8,6))

#Generate correlation matrix
corr_mat = df_day.corr()
#Plot using seaborn library
sns.heatmap(corr_mat, mask=np.zeros_like(corr_mat, dtype=np.bool), cmap=sns.diverging_palette(220, 10, as_cmap=True),square=True
```

Below graph shows the correlation plot of all numeric variable present in the dataset.

Fig 2.12 Correlation plot for the numerical variables



In the above plot red color represents highly positively correlated variables and dark grey represents the highly negatively correlated variable. So we can see that there is high correlation between variables temp and temp, which means both the variables temp and atemp providing nearly equal information so we can drop one of them from the dataset. So I have removed atemp variable from the dataset.

2.1.7 Feature scaling

Feature scaling is the method to limit the range of variable so they can be compared on the ground and it is perform only on the variable.

To scale the features there are two methods,

1. Normalization
2. Standardization (zee score)

1.Normalization- Normalization is the process of reducing unwanted variation either within or between variable(i.e it converts all the data points of each variable between 0 to 1).

2.Standardization- It transforms the data set to have a zero mean and unique variance.

Since all the numerical variables are already given in normalized form, so we need not to perform feature selection.

Chapter -3

Modeling

3.1 Model selection

Our problem statement is to predict the counts of bike rental per day according to environmental and seasonal condition and here our target variable is a continuous variable. For continuous variable we can go for different Regression models. And the model having less error rate and high accuracy will be the best model for the given dataset.

Using following 3 models we will predict the counts.

- Decision tree Regressor(C50)
- Random Forest
- Linear Regression

Before modeling we have divided the data into train and test data using Simple Random Sampling (SRS). Where train data contains 75% of the dataset and test data contains 25% data of the dataset having 11 variables where 11th variable is the target variable.

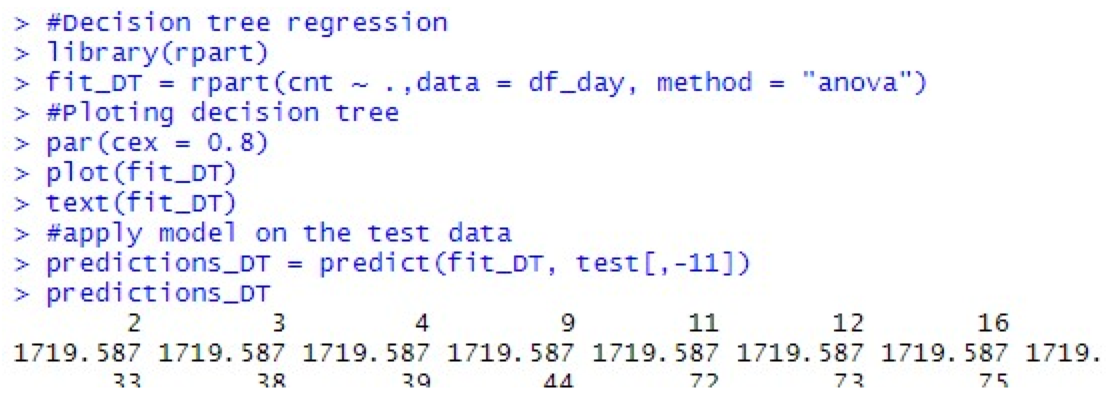
3.1.1 Decision tree

Decision tree is a predictive model based on a branching series of Boolean tests. General motive of using Decision tree is to create a training model which can be used to predict the class or a value of target variable by learning distinct rules insert from the past data. It's an algorithm that uses the tree like graph or a model of decisions.

Under the family of decision tree there are different types of Decision tree algorithms are available, like C50, C45, CART, CHECK etc. Here we have used C50 algorithm which works on information gain.

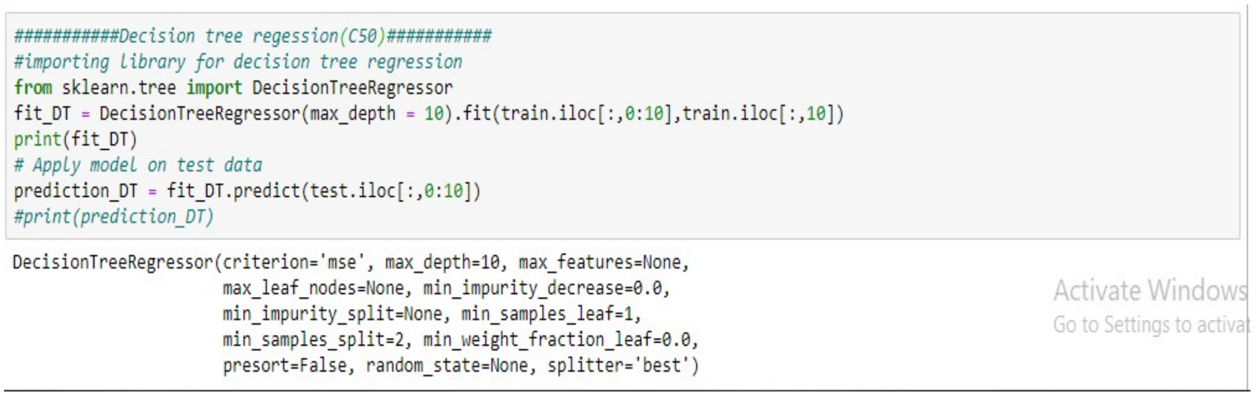
Information gain: it is the statistical approach which helps to select the node and develop a tree like structure to make any conclusion or decision on the data. It represent the expected amount of information that would be needed (i.e out of all the independent variable as a parent node who is contributing much information of all).

Creating model in R



```
> #Decision tree regression
> library(rpart)
> fit_DT = rpart(cnt ~ .,data = df_day, method = "anova")
> #Ploting decision tree
> par(cex = 0.8)
> plot(fit_DT)
> text(fit_DT)
> #apply model on the test data
> predictions_DT = predict(fit_DT, test[,-11])
> predictions_DT
       2        3        4        9       11       12       16
1719.587 1719.587 1719.587 1719.587 1719.587 1719.587 1719.587 1719.
      33       38       39       44       72       73       75
```

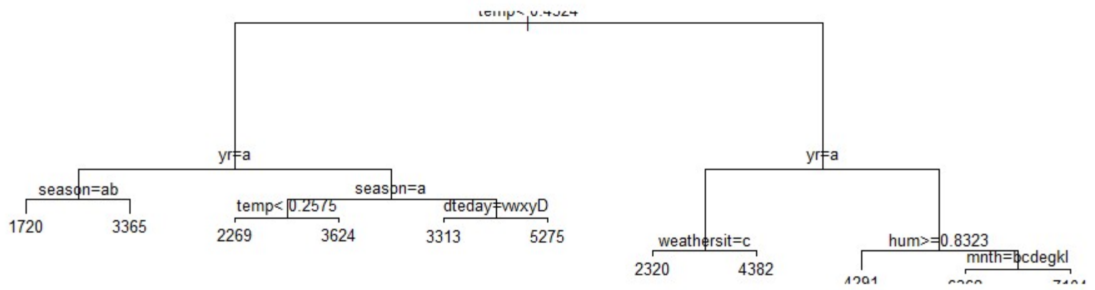
Creating model in python



```
###########Decision tree regession(C50)###########
#importing library for decision tree regression
from sklearn.tree import DecisionTreeRegressor
fit_DT = DecisionTreeRegressor(max_depth = 10).fit(train.iloc[:,0:10],train.iloc[:,10])
print(fit_DT)
# Apply model on test data
prediction_DT = fit_DT.predict(test.iloc[:,0:10])
#print(prediction_DT)
```

```
DecisionTreeRegressor(criterion='mse', max_depth=10, max_features=None,
                      max_leaf_nodes=None, min_impurity_decrease=0.0,
                      min_impurity_split=None, min_samples_leaf=1,
                      min_samples_split=2, min_weight_fraction_leaf=0.0,
                      presort=False, random_state=None, splitter='best')
```

Fig 3.1 Graphical representation of Decision Tree

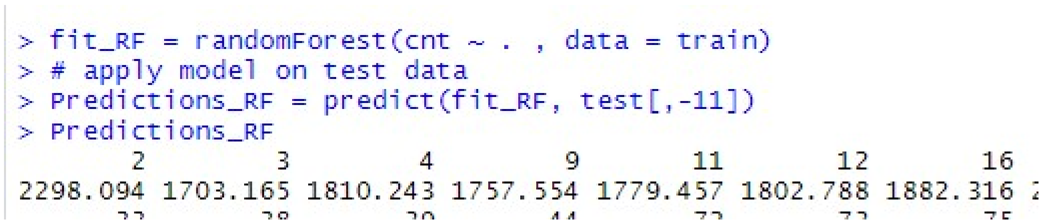


Above figure 3.1 shows the graphical representation of Decision tree for the given data set.

3.1.2 Random Forest

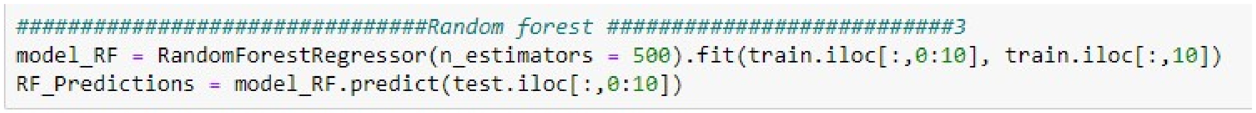
Random forest is an ensemble that consists of many decision trees. To reduce the error and improve the accuracy we will combine the multiple decision trees to prepare a strong classifier or regressor.

Code in R



```
> fit_RF = randomForest(cnt ~ . , data = train)
> # apply model on test data
> Predictions_RF = predict(fit_RF, test[,-11])
> Predictions_RF
       2        3        4        9       11       12       16
2298.094 1703.165 1810.243 1757.554 1779.457 1802.788 1882.316
```

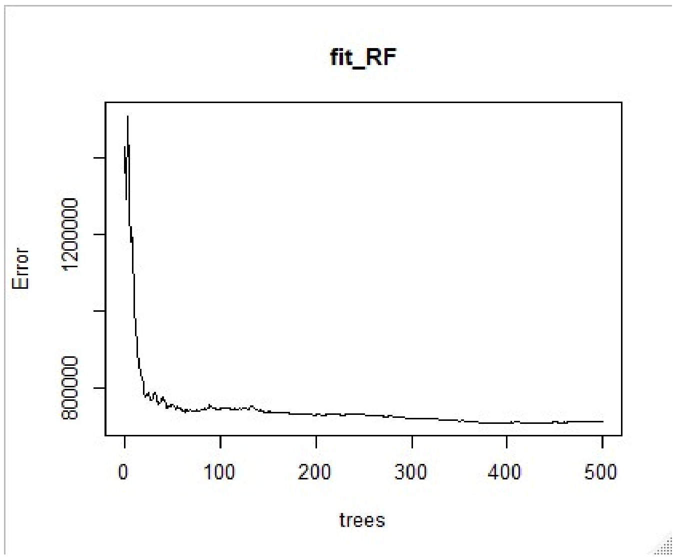
Code in python



```
##################################Random forest ###########################3
model_RF = RandomForestRegressor(n_estimators = 500).fit(train.iloc[:,0:10], train.iloc[:,10])
RF_Predictions = model_RF.predict(test.iloc[:,0:10])
```

In this model we are using 500 trees to predict the target variable.

Fig 3.2 Error rate based tree



Above figure 3.2 represents the curve of error rate as the number of trees increases. After 500 trees the error rate reaches to be constant.

3.1.4 Multiple Linear Regression

Multiple linear regression (MLR is a statistical technique that uses several explanatory variables to predict the outcome of a response variable. The goal of multiple linear regression (MLR) is to model the linear relationship between the explanatory (independent) variables and response (dependent) variable.
In essence, multiple regression is the extension of ordinary least-squares (OLS) regression that involves more than one explanatory variable.

The equation of multiple regression is

$Y_i=\beta_0+\beta_1x_{i1}+\beta_2x_{i2}+...+\beta_px_{ip}+\epsilon$

Y_i =dependent variable

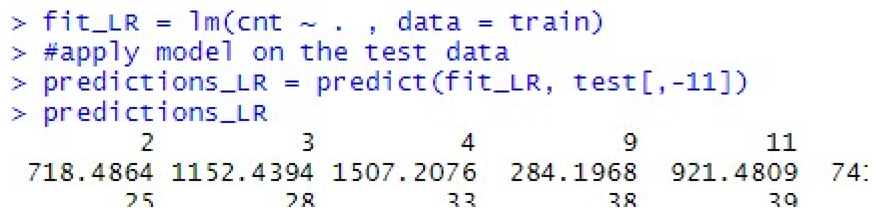
x_i= independent variables

β_0=y-intercept (constant term)

βp = slope coefficients for each explanatory variable

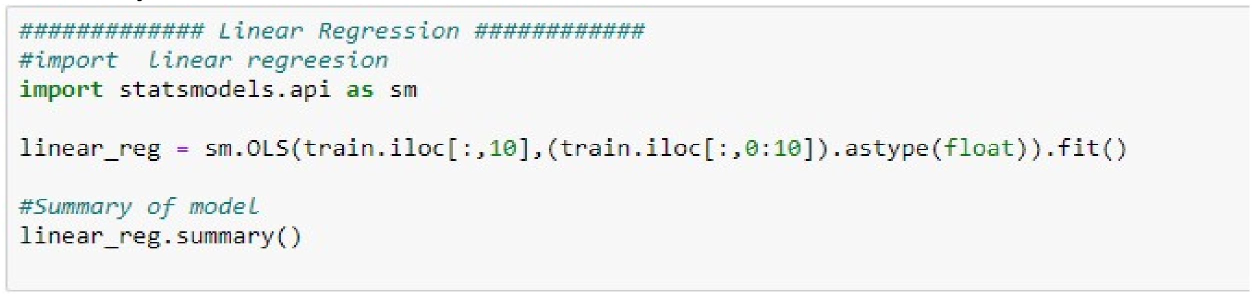
ϵ=the model's error term (also known as the residuals)

Code in R



```
> fit_LR = lm(cnt ~ . , data = train)
> #apply model on the test data
> predictions_LR = predict(fit_LR, test[,-11])
> predictions_LR
        2         3         4         9        11
 718.4864 1152.4394 1507.2076  284.1968  921.4809  74
       25        28        33        38        39
```

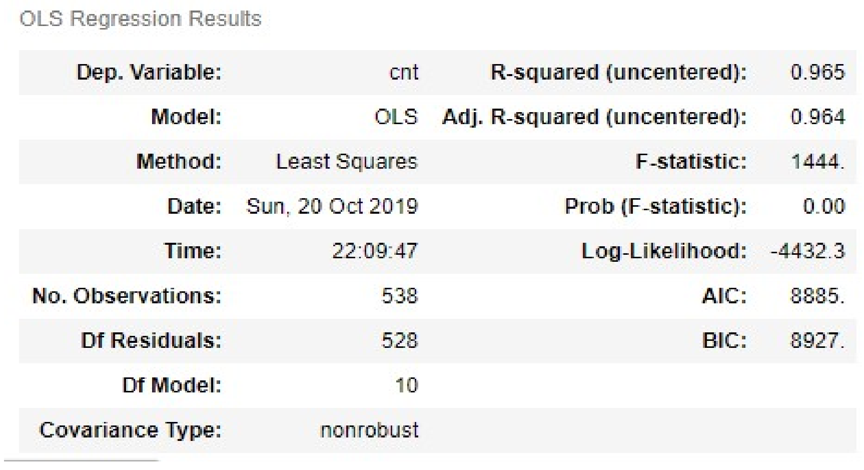
Code in Python



```
############# Linear Regression ############
#import  linear regreesion
import statsmodels.api as sm

linear_reg = sm.OLS(train.iloc[:,10],(train.iloc[:,0:10]).astype(float)).fit()

#Summary of model
linear_reg.summary()
```



OLS Regression Results

Dep. Variable:	cnt	R-squared (uncentered):	0.965
Model:	OLS	Adj. R-squared (uncentered):	0.964
Method:	Least Squares	F-statistic:	1444.
Date:	Sun, 20 Oct 2019	Prob (F-statistic):	0.00
Time:	22:09:47	Log-Likelihood:	-4432.3
No. Observations:	538	AIC:	8885.
Df Residuals:	528	BIC:	8927.
Df Model:	10		
Covariance Type:	nonrobust		

Here R- squared = 0.965 and adjusted R-squared = 0.964 which are closed to 1 which indicates that the outcome can be predicted without error from the independent variables.

Chapter -4

Conclusion

4.1 Model Evaluation

We have developed few models to predict the target variable cnt. Now we have to choose the suitable model which is providing best result for the dataset based on predictive performance.

Predictive performance can be measured by comparing predicted values of the models with the actual values of the target variable and calculating some average error measures.

Using following error metrics we will calculate the error rate and then we will select the appropriate model.

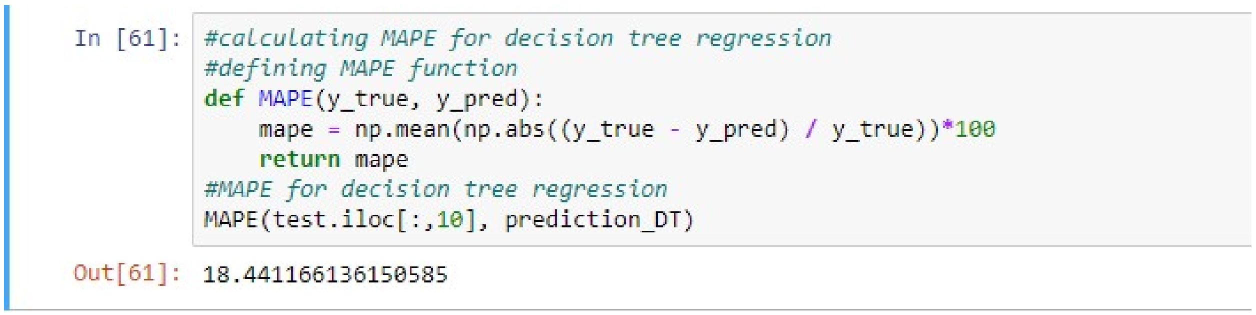
4.1.1 MAPE

It stand for Mean Absolute Percentage Error. It is one of the error measures used to calculate the predictive performance of the model. It measures accuracy as a percentage of error.

Formula of mape is

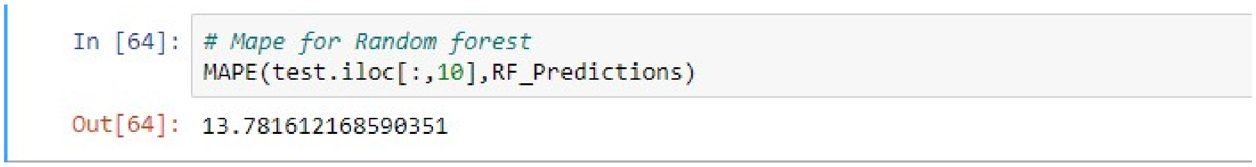
$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{actual\ value - predicted\ value}{actual\ value}\right| * 100$$

Python code to calculate MAPE



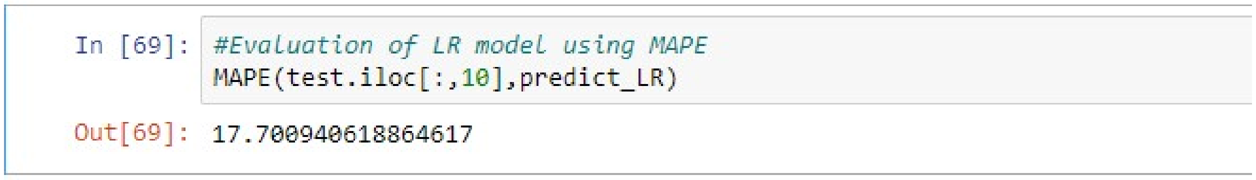
```
In [61]: #calculating MAPE for decision tree regression
         #defining MAPE function
         def MAPE(y_true, y_pred):
             mape = np.mean(np.abs((y_true - y_pred) / y_true))*100
             return mape
         #MAPE for decision tree regression
         MAPE(test.iloc[:,10], prediction_DT)

Out[61]: 18.441166136150585
```



```
In [64]: # Mape for Random forest
         MAPE(test.iloc[:,10],RF_Predictions)

Out[64]: 13.781612168590351
```



```
In [69]: #Evaluation of LR model using MAPE
         MAPE(test.iloc[:,10],predict_LR)

Out[69]: 17.700940618864617
```

Here we can see that mape of Random forest is less as compared to Decision tree and Linear regression.

Where MAPE of

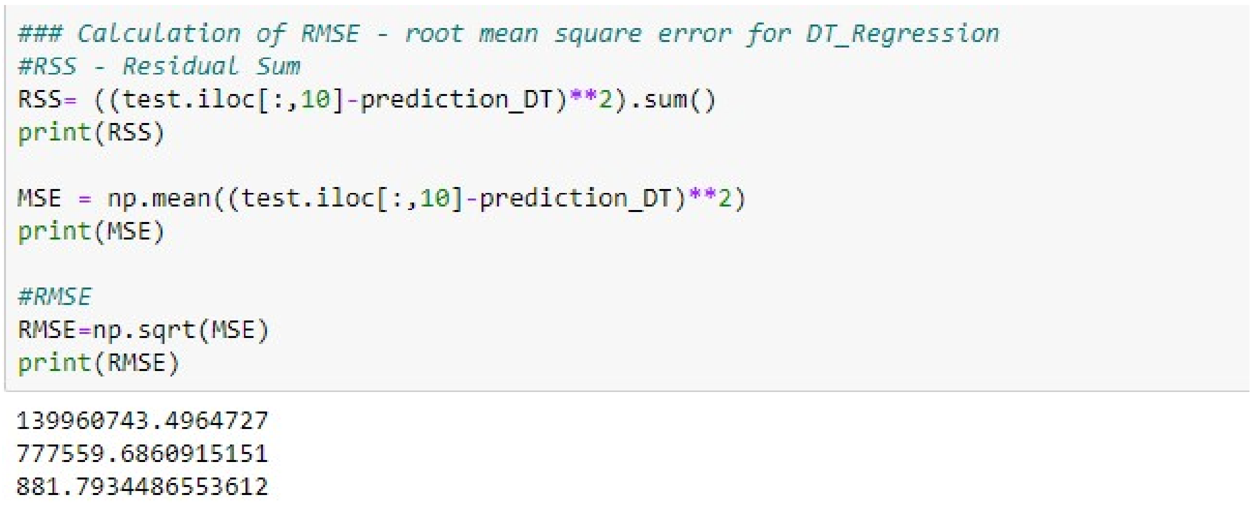
Decision Tree (C50) = 18.441166136150585
Random Forest = 13.781612168590351
Linear Regression = 17.700940618864617

4.1.2 RMSE

Root Mean Square Error is a frequently used measure of the difference between values predicted by a model and the values actually observed from the environment that is being modeled. RMSE square the errors, finds their average and take their square root.

Formula of RMSE is

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(X_{obs,i} - X_{model,i})^2}{n}}$$

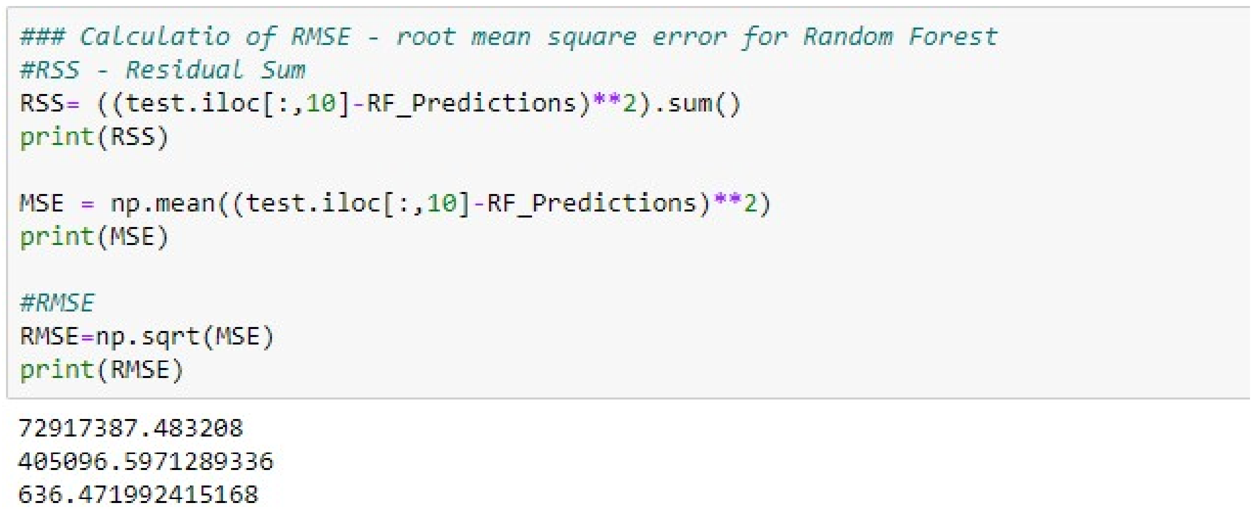


```
### Calculation of RMSE - root mean square error for DT_Regression
#RSS - Residual Sum
RSS= ((test.iloc[:,10]-prediction_DT)**2).sum()
print(RSS)

MSE = np.mean((test.iloc[:,10]-prediction_DT)**2)
print(MSE)

#RMSE
RMSE=np.sqrt(MSE)
print(RMSE)
```

```
139960743.4964727
777559.6860915151
881.7934486553612
```

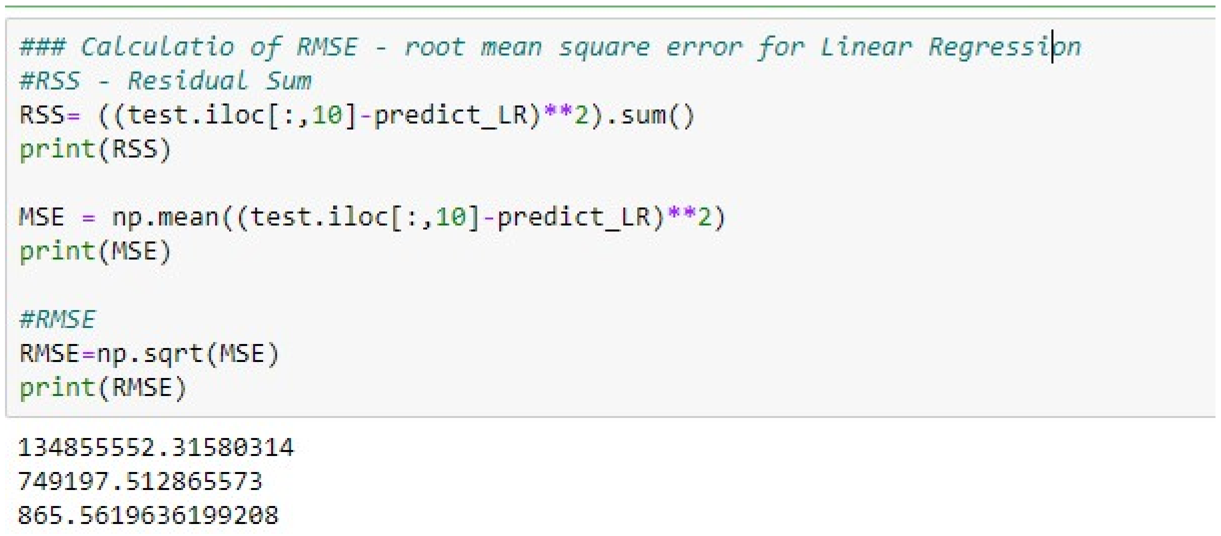


```
### Calculatio of RMSE - root mean square error for Random Forest
#RSS - Residual Sum
RSS= ((test.iloc[:,10]-RF_Predictions)**2).sum()
print(RSS)

MSE = np.mean((test.iloc[:,10]-RF_Predictions)**2)
print(MSE)

#RMSE
RMSE=np.sqrt(MSE)
print(RMSE)
```

```
72917387.483208
405096.5971289336
636.471992415168
```



```
### Calculatio of RMSE - root mean square error for Linear Regression
#RSS - Residual Sum
RSS= ((test.iloc[:,10]-predict_LR)**2).sum()
print(RSS)

MSE = np.mean((test.iloc[:,10]-predict_LR)**2)
print(MSE)

#RMSE
RMSE=np.sqrt(MSE)
print(RMSE)
```

```
134855552.31580314
749197.512865573
865.5619636199208
```

Here RMSE (Root Mean Square Error) of Random forest is also less as compared to Decision tree regressor(C50) and Linear Regression model.

Where the RMSE of

Decision Tree = 881.7934486553612
Random Forest =636.471992415168
Linear Regression = 865.5619636199208

4.2 Model selection

After calculating the error matrices I both R and Python, we saw that the mean absolute percentage error (MAPE) and Residual mean square error (RMSE) of Random Forest is less as compared to the MAPE sand RMSEs of Decision Tree Regressor (C50) and Multiple Linear Regression.

That means for this dataset, Random Forest predicted the counts of bike rental with high accuracy and low error rate. So Random Forest is the Best model for this Data set.