Final Project - Bike Renting

Gouri Khan

January 26,2020

Contents

Chapter 1
Introduction

1. Problem Statement

The objective of this Case is to Predication of bike rental count on daily based on the environmental and seasonal settings. So that required bikes would be arranged and managed by the shops according to environmental and seasonal conditions and customer need.

The dataset “day.csv” was given with problem statement.This dataset contains daily count of rental bikes between years 2011 and 2012 in bike-share system with the corresponding weather and seasonal information. Bike sharing systems are a new way of traditional bike rentals. The data aggregated on hourly and daily basis and then added the corresponding weather and seasonal information that were extracted from http://www.freemeteo.com.

2. Data

Given dataset “**day.csv**”.
Our task is to build regression models which will predict the count of bike rented depending on various environmental and seasonal conditions.
Below is a sample of the data set that we are using to predict the count of bike rents:
In this dataset there are total 16 columns and 731 rows. In the course of the analysis we may generated more columns and drop some as per requirement.
df_day.shape
#(731, 16)

Table 1.1: day.csv Sample Data (Columns: 1-8)

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

Table 1.2: day.csv Sample Data (Columns: 9-16)

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

Below are the independent variables we will use to predict the counts of bike rent:

Table 1.3: column names using ***df_day.columns (in python)***

s.no	Variables
0	Instant
1	dteday
2	season
3	yr
4	mnth
5	holiday
6	weekday
7	workingday
8	weathersit
9	temp
10	atemp
11	hum
12	windspeed
13	Casual
14	registered
15	Cnt

2.1 Data description :

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

- instant: Record index
- dteday: Date
- season: Season (1:spring, 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
- weekday: Day of the week
- workingday: if day is neither weekend nor holiday is 1, otherwise is 0.
- weathersit:(extracted fromFreemeteo)
    - 1: Clear, Few clouds, Partly cloudy
    - 2: Mist and Cloudy, Mist and Broken clouds, Mist and Few clouds, Mist
    - 3: Light Snow, Light Rain and Thunderstorm and Scattered clouds, Light Rain and Scattered clouds
    - 4: Heavy Rain and Ice Pallets and Thunderstorm and Mist, Snow and 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
- registered: count of registered users
- cnt: count of total rental bikes including both casual and registered

Chapter 2
Methodology

1. Data Analysis

2.1.1 Univariate Analysis

In Figure 2.1.1.1 (temp), 2.1.1.2 (cnt), 2.1.1.3 (atemp), 2.1.1.4 (hum), 2.1.1.5 (windspeed),2.1.1.6 (casual), 2.1.1.7 (registered) we have the probability density functions for numeric variables present in the dataset including target variable cnt.

i. Target variable cnt is normally distributed
ii. Independent variables like 'temp','atemp', and 'regestered' data is distributed normally.
iii. Independent variable 'casual' data is skewed to the right so, there is presence of outliers.
iv. Other Independent variable 'hum' and 'windspeed' data is slightly skewed to the left , here data is already in normalised ,so outliers will be kept as humidity can be sometime different due to weather conditions.

Figure 2.1.1.2 Distribution of target variable (CNT) (python code in Appendix B)

Figure 2.1.1.1 (temp)

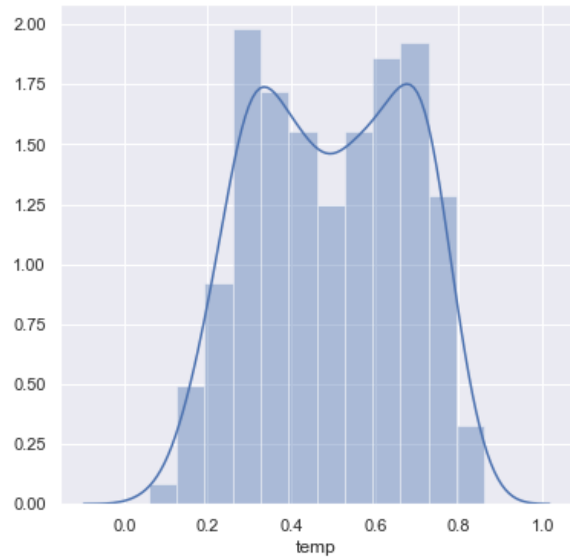


Figure 2.1.1.2 (cnt)

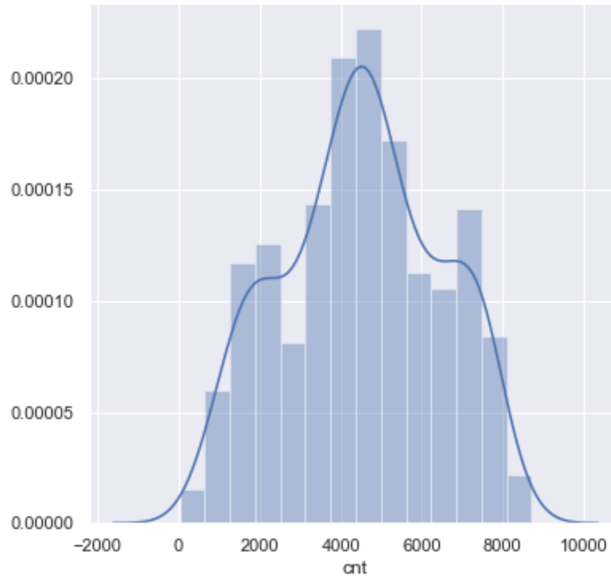


Figure 2.1.1.3 (atemp)

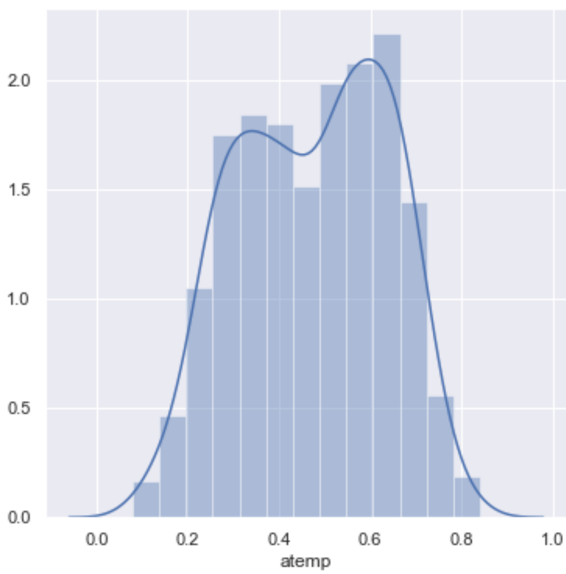


Figure 2.1.1.4 (hum)

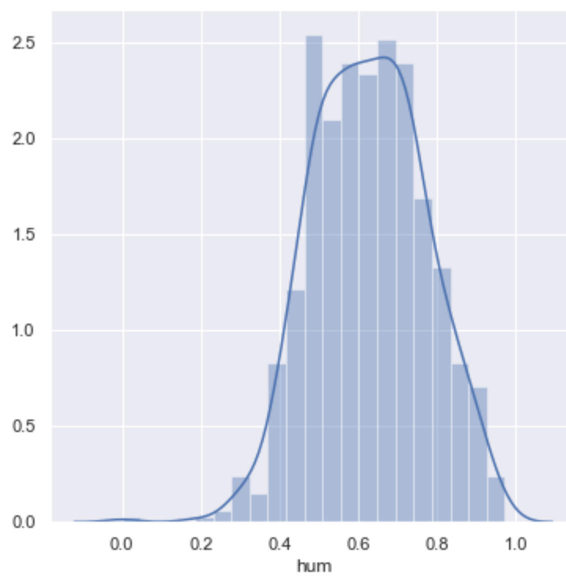
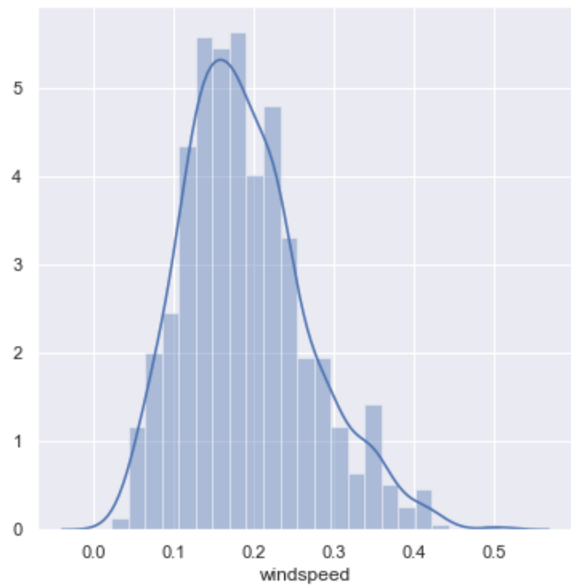


Figure 2.1.1.5 (windspeed)

Figure 2.1.1.6 (casual)



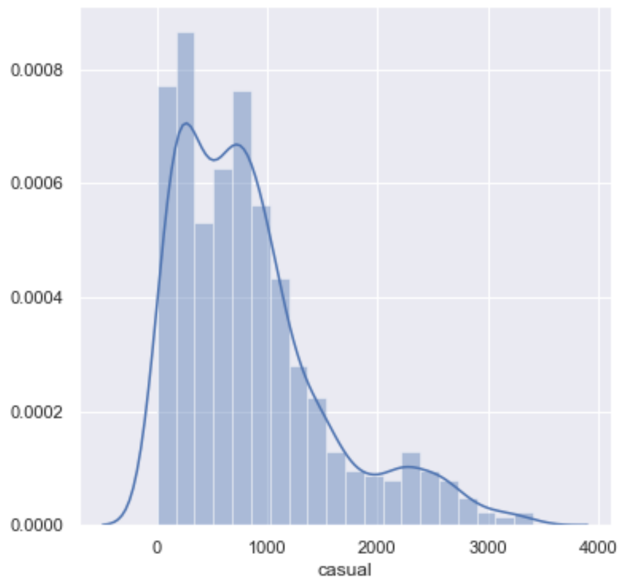
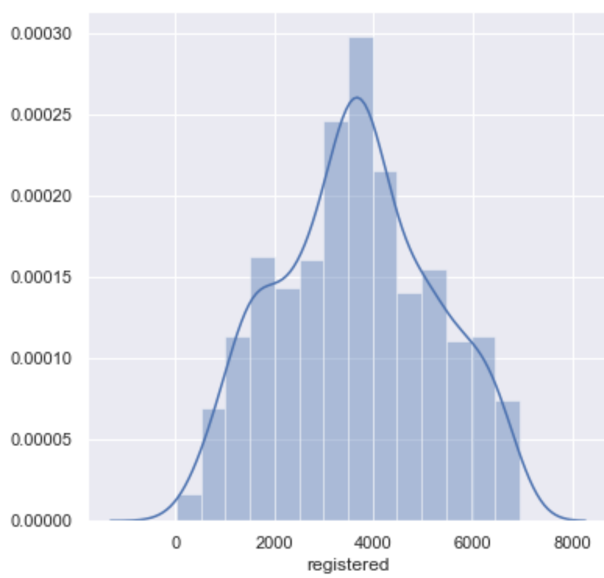


Figure 2.1.1.7 (registered)



2.1.2 Bivariate Analysis

Below we have plotted relationship of every non-numeric or categorical value with target cnt.

In Figure 2.1.2.1 (weekday), 2.1.2.2 (holiday), 2.1.2.3 (workingday), 2.1.2.4 (weathersit), 2.1.2.5 (yr), 2.1.2.6 (season), 2.1.2.7 (mnth) , 2.1.2.8 (dteday) we have the box ploy for non-numeric variables present in the dataset with respect to target variable cnt.

Figure 2.1.2.1 (weekday-cnt)

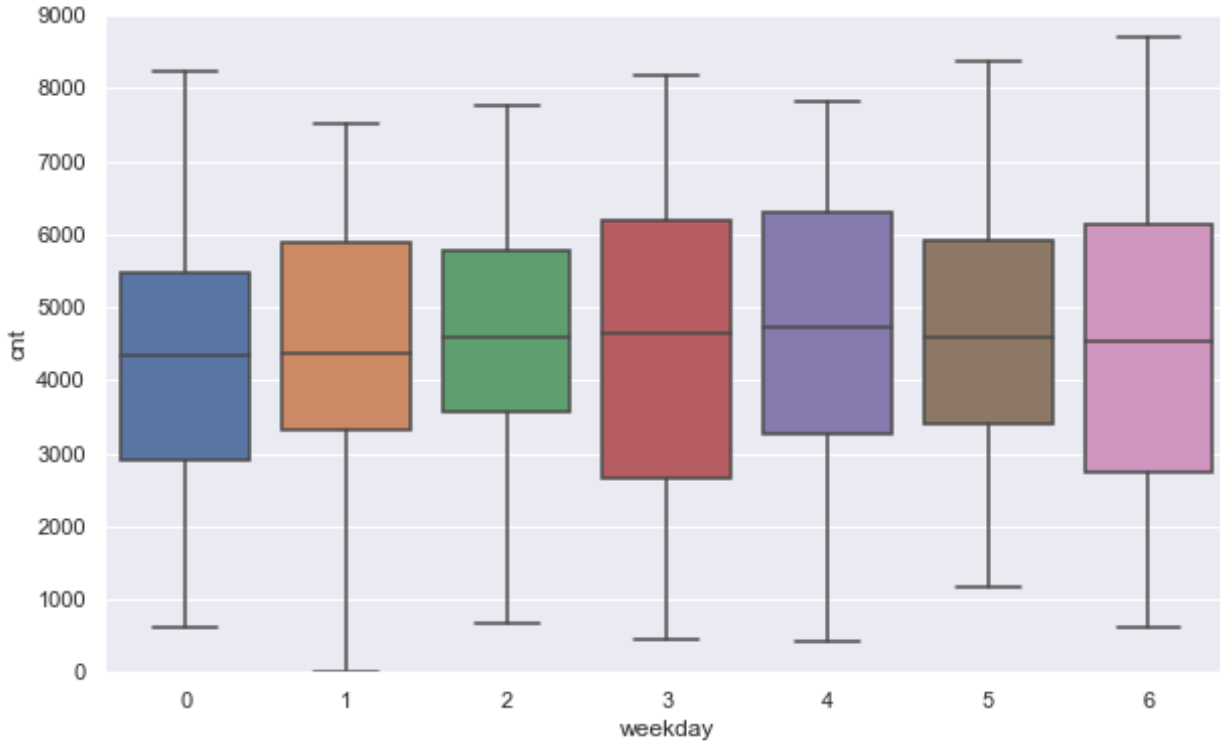


Figure 2.1.2.2 (holiday-cnt)

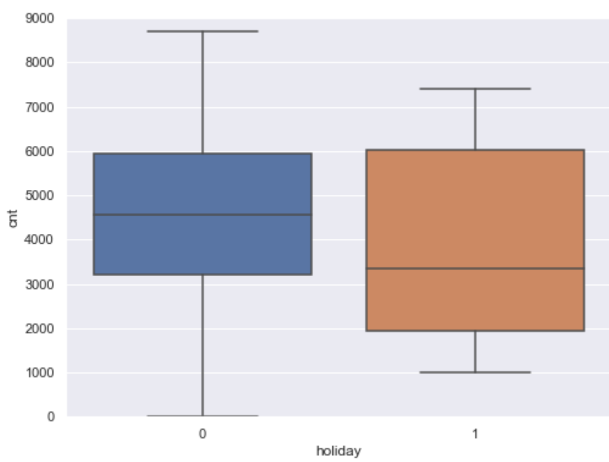


Figure 2.1.2.3 (workingday-cnt)

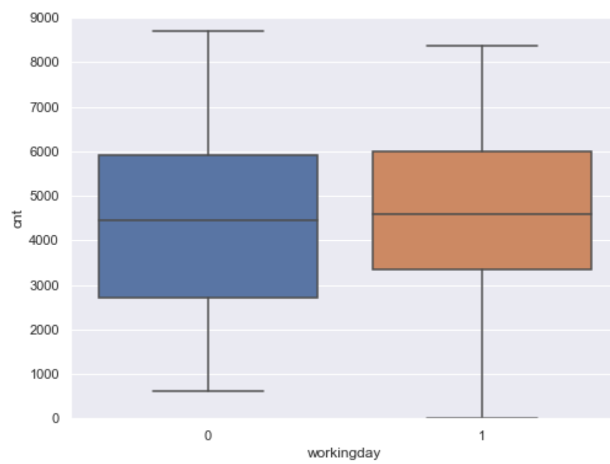


Figure 2.1.2.4 (weathersit-cnt)

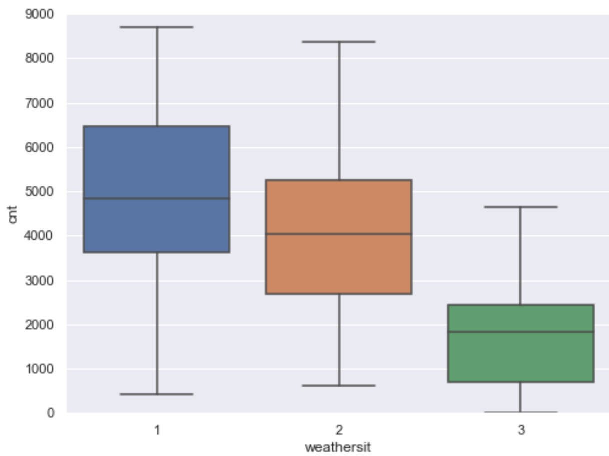


Figure 2.1.2.5 (yr-cnt)

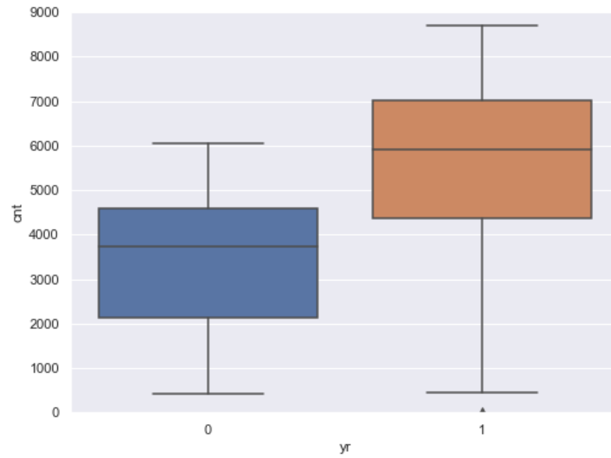


Figure 2.1.2.6 (season-cnt)

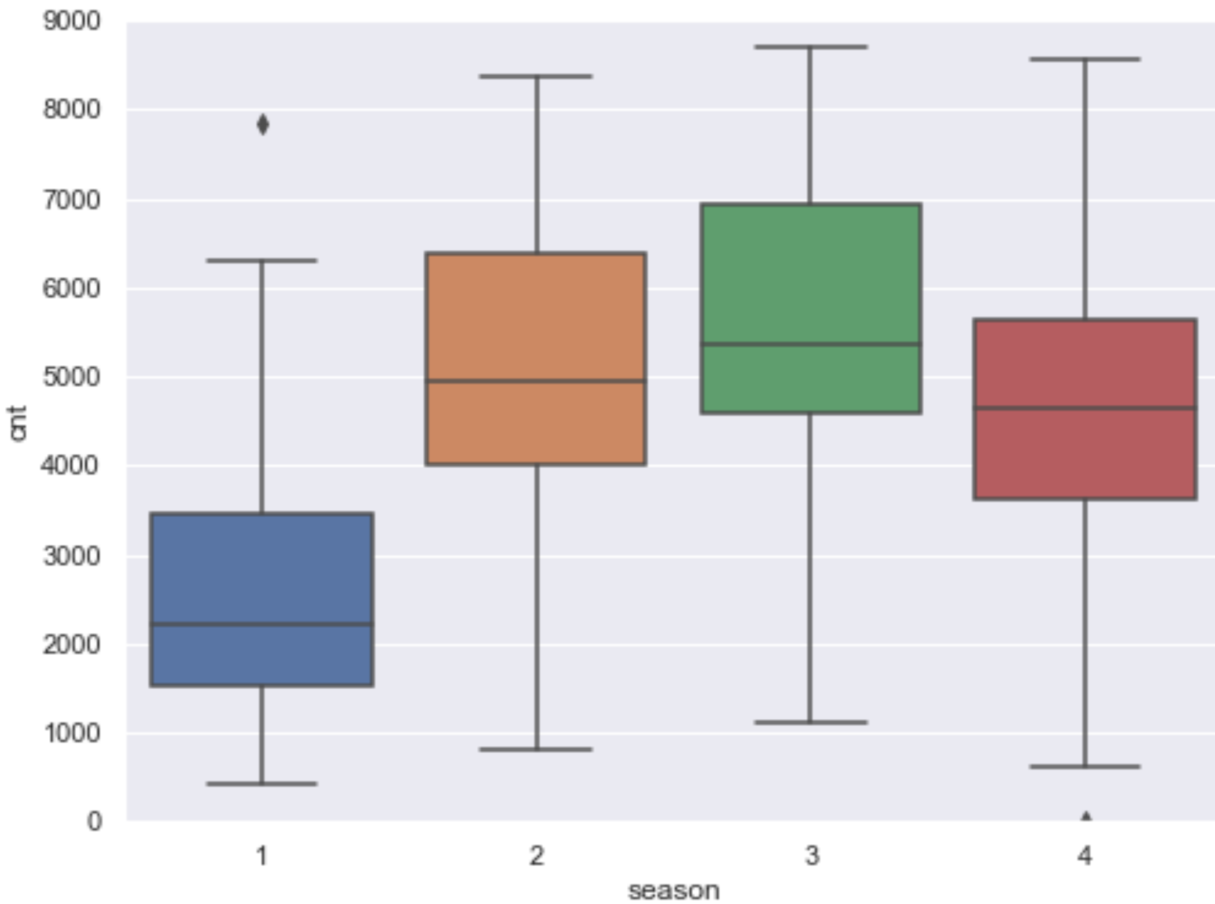


Figure 2.1.2.7 (mnth-cnt)

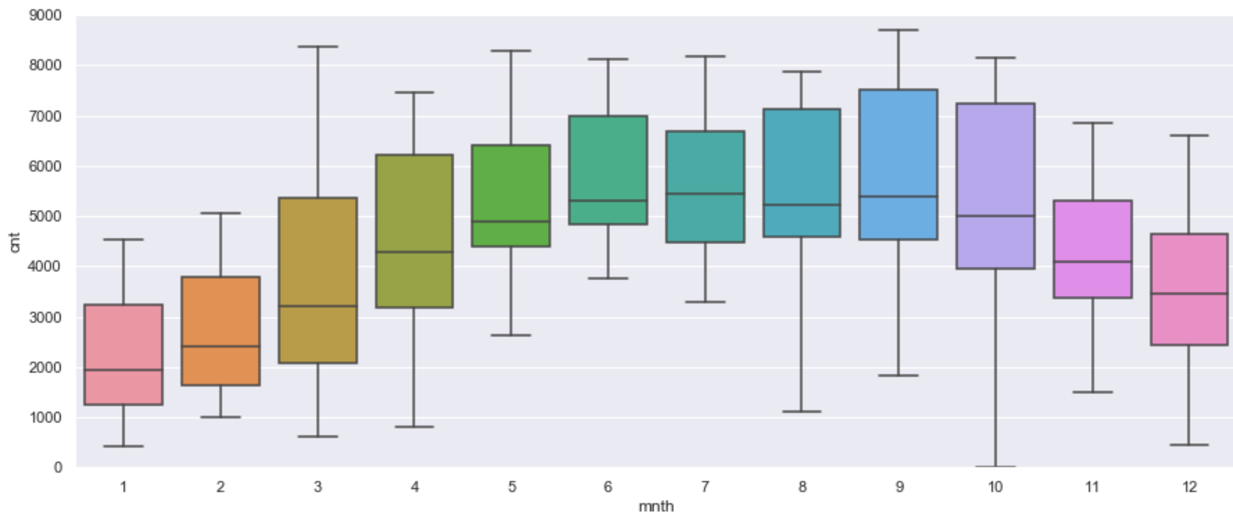
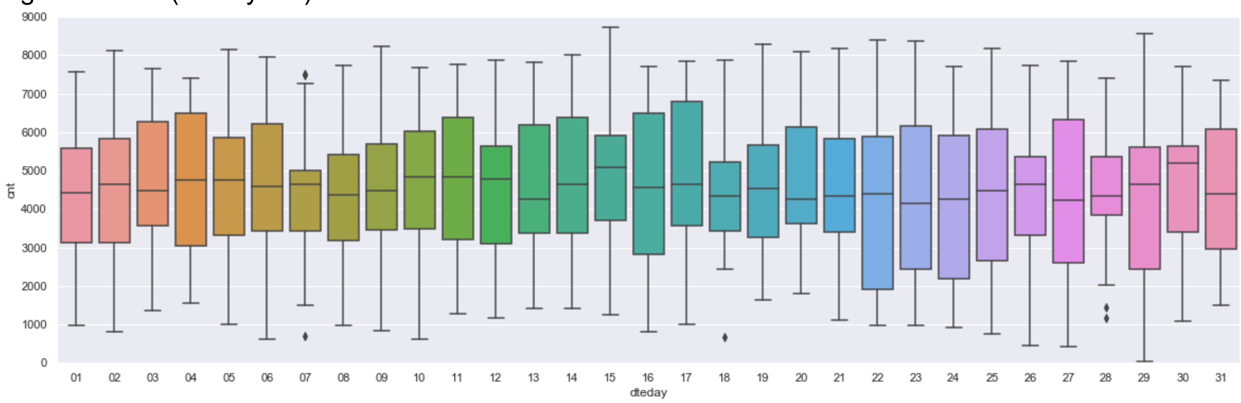


Figure 2.1.2.8 (dteday-cnt)



2.1.3 Multiple variable Analysis

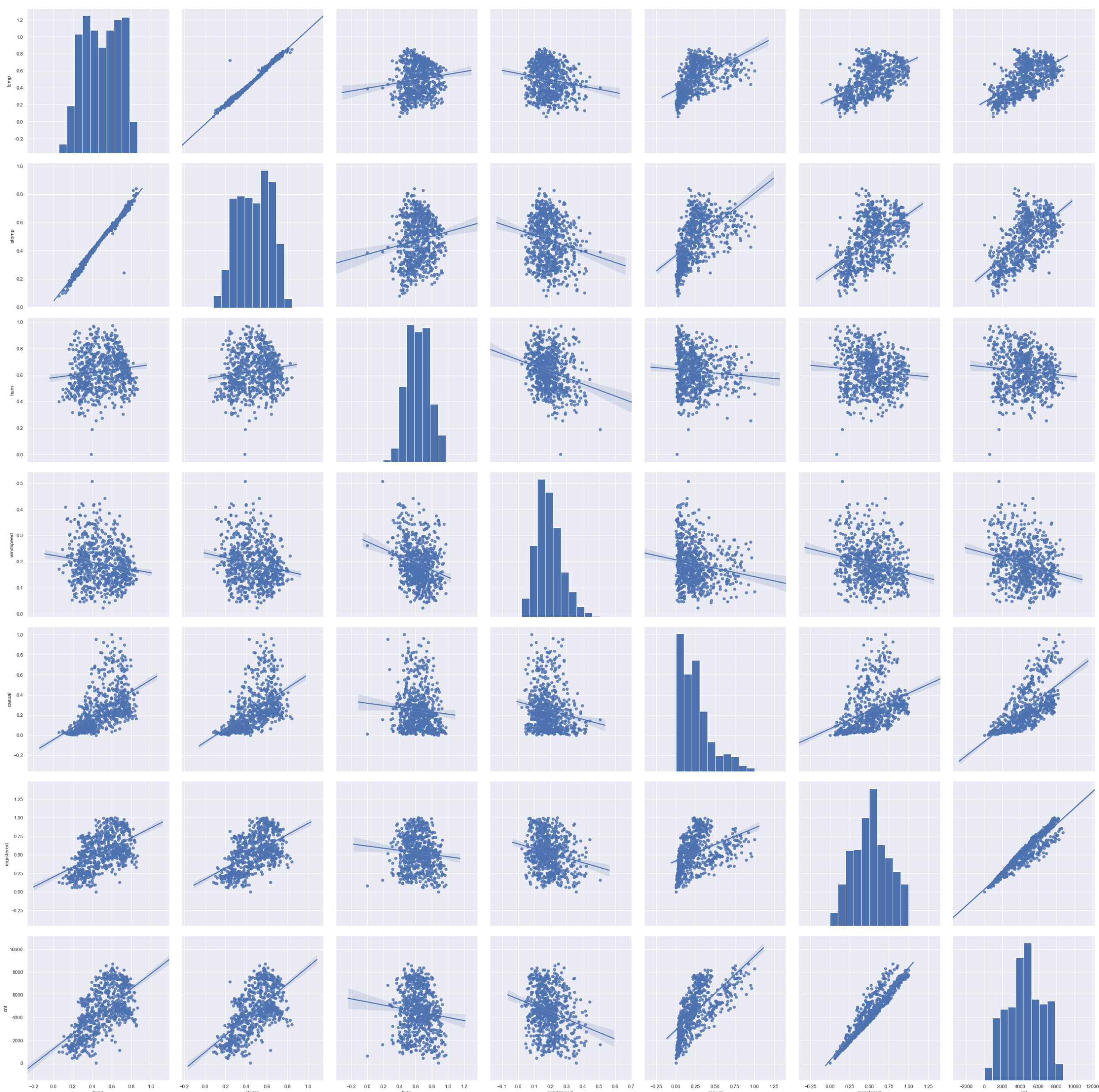
Pair plot provides templates for combining plots into a matrix for all numeric variables. Such a matrix of plots can be useful for quickly exploring the relationships between multiple columns of data in a data frame including a regression line.

Below figure shows relationship between independent variables and also with numeric target variable using pairplot-lot

i. Below plot showing relationship between variables ‘temp’ and ‘atemp’ are very strong.

ii. The relationship between ‘hum’ , ‘windspeed’ with target variable ‘cnt’ is less.

Figure 2.1.3.1 relationship between numeric variables (python code in Appendix B)



2. Pre Processing Techniques

Any predictive modelling requires a look at the data before starting modelings. However, in data mining looking at data refers so much more than just looking. Looking at data refers to exploring the data, cleaning the data as well and visualising the data through graphs and plots. This is called Exploratory Data Analysis(EDA).

2.2 **Exploratory Data Analysis (EDA) :**

To start this process we will first look at all the probability distributions of the variables. Most analysis like regression, require the data to be normally distributed. We can visualize that in a glance by looking at the probability distributions or probability density functions of the variable.

While exploring the data (EDA) we have :

- Converted season, mnth, yr, holiday, weekday, workingday, weathersit into categorical variables .
- **Feature Engineering** :Changed dteday variables's date value to day of date and converted to categorical variable having 31 levels as a month has 31 days.
- Deleted instant variable as it is nothing but an index.

2.2.1 Missing Value Analysis

Missing values in data is very common phenomenon in real life problems. Knowing how to handle missing values effectively is a required step to reduce bias and to produce powerful models. Missing value analysis is done to check is there any missing value present in given dataset. Missing values can be easily treated using various methods like mean, median method, KNN algorithm method to impute missing value.

- In R " *total_missing_value = data.frame(apply(df_day,2,function(x){sum(is.na(x))}))* " is the function used to check the sum of missing values.
- In python " *total_missing_value = df_day.isnull().sum()* " is used to detect sum of missing values present in data frame.

Below table is the output of above line of code for missing value :

2.2.1 missing values

s.no	Variables	missing values
1	dteday	0

2	season	0
3	yr	0
4	mnth	0
5	holiday	0
6	weekday	0
7	workingda y	0
8	weathersit	0
9	temp	0
10	atemp	0
11	hum	0
12	windspeed	0
13	casual	0
14	registered	0
15	Cnt	0

CONCLUSION : Dataset day.csv which imported as df_day has no missing value.

2.2.2 Outlier Analysis

The Other steps of Preprocessing Technique is Outliers analysis . In statistics, an outlier is an observation point that is distant from other observations. The above definition suggests that outlier is something which is separate/different from the crowd present in given dataset. Outlier analysis can only be done on continuous variable.We have used box plot to detect outliers in numerical variable.
Outliers in data can distort predictions and affect the accuracy, if we do not detect and handle them appropriately.

As we are earlier observed in fig 2.1.1.6 the data is skewed so, there is chance of outlier in independent variable ‘casual’ , one of the best method to detect outliers is Boxplot .

Below are the list of box plots: 2.2.2.1 (hum), 2.2.2.2 (windspeed), 2.2.2.3 (casual), 2.2.2.4 (registered).

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

Figure 2.2.2.1 Baxoplot for ‘hum’

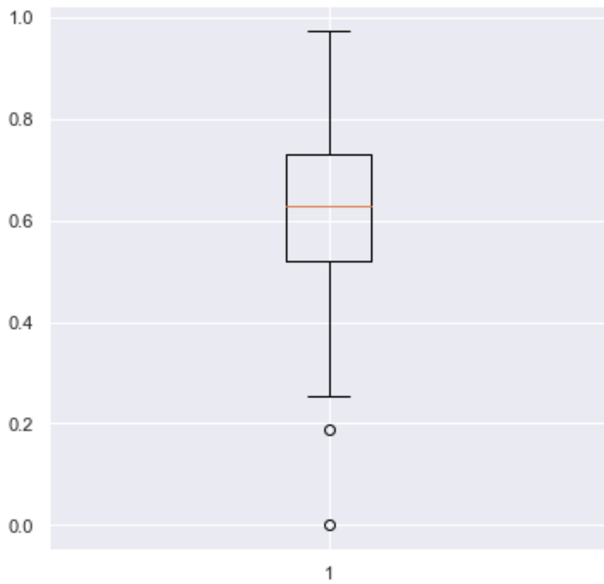


Figure 2.2.2.2 Baxoplot for ‘windspeed’

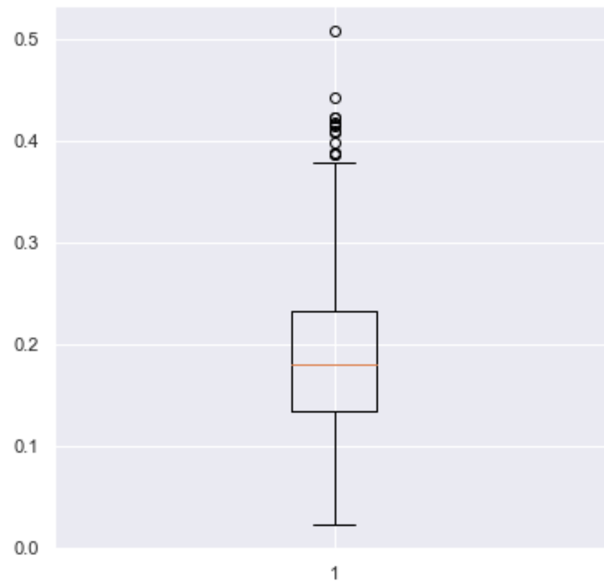


Figure 2.2.2.3 Baxoplot for ‘casual’

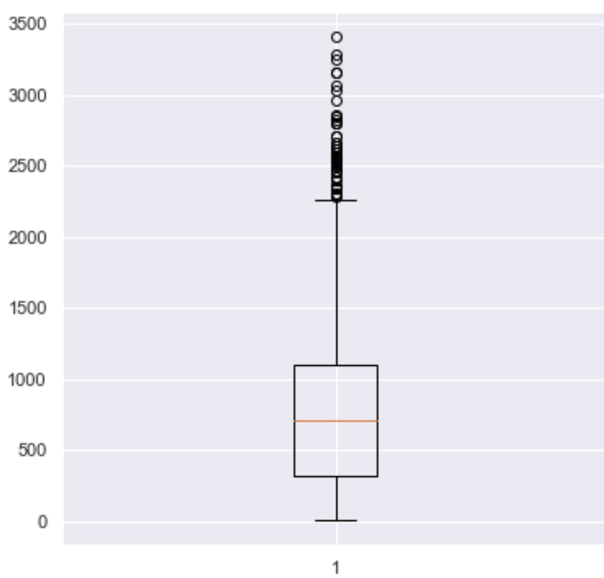
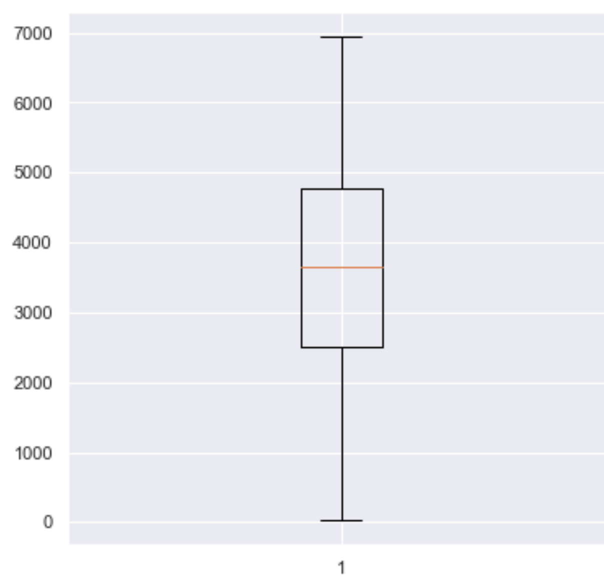


Figure 2.2.2.4 Baxoplot for ‘registered’



Since hum and weatherise is weather variables and gas very small no of outliers so, we are not going to treat the outliers.For casual we will normalise the variable in rage of 0 to 1 to get rid of outlier effect.

2.2.3 Features Selection

In Machine learning there is a simple rule – if you put garbage in, you will only get garbage to come out. By garbage here, it mean noise in data or bad unprocessed junk data.

This become more important when the number of features are very large. We need not use every feature at our disposal for creating an algorithm. We can assist our algorithm by feeding in only those features that are really important. This is very much possible that subsets giving better results than complete set of feature for the same algorithm or – “Sometimes, less is better!”.

We should consider the selection of feature for model based on below criteria

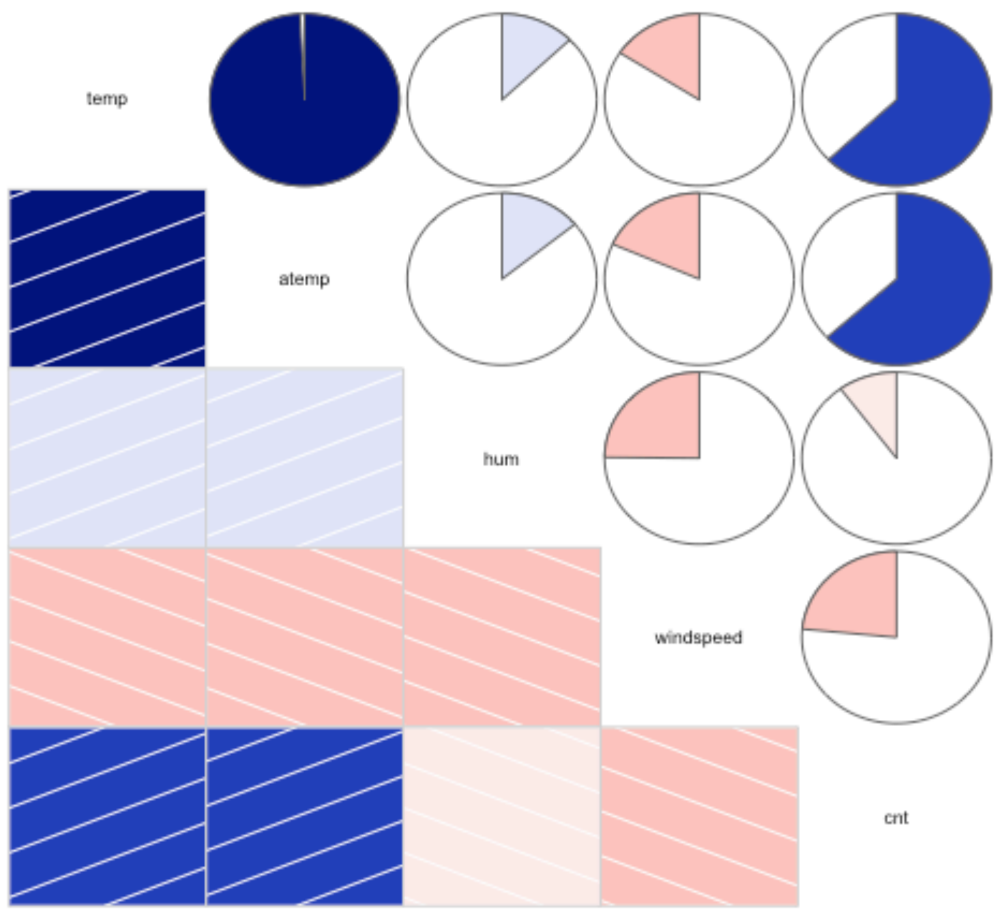
i. The relationship between two independent variable should be low and

ii. The relationship between Independent and Target variables should be high.

iii. Below fig 2.2.3.1 and 2.2.3.2 illustrates that relationship between all numeric variables using pearson method and heat map.

Figure 2.2.3.1 correlation plot of numeric variables (PYTHON code in Appendix B)

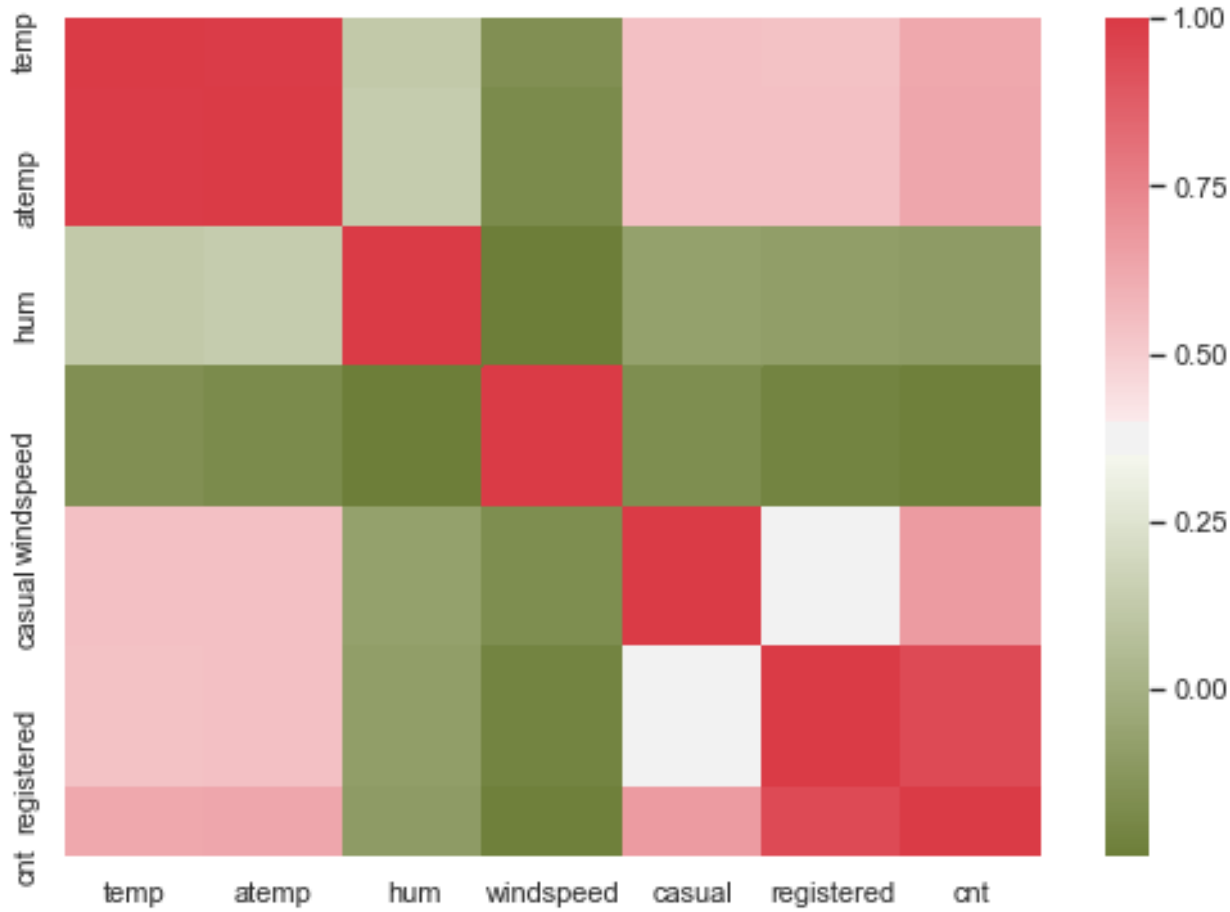
Correlation Plot



temp	atemp	hum	windspeed	casual	registered	cnt	
temp	1.0	0.992	0.127	-0.158	0.543	0.54	0.627
atemp	0.992	1.0	0.14	-0.184	0.544	0.544	0.631
hum	0.127	0.14	1.0	-0.248	-0.077	-0.0911	-0.101
windspeed	-0.158	-0.184	-0.248	1.0	-0.168	-0.217	-0.235
casual	0.543	0.544	-0.077	-0.168	1.0	0.395	0.673

registered	0.54	0.544	-0.0911	-0.217	0.395	1.0	0.946
cnt	0.627	0.631	-0.101	-0.235	0.673	0.946	1.0

Figure 2.2.3.2 Heat map of numeric variables (PYTHON code in Appendix B)



Dark colour indicates there is strong positive relationship and if darkness is decreasing indicates relation between variables are decreasing.Here dark red represents very high positive correlation and dark green represents very high negative correlation, almost white colour means there is no relation between variables.

Definition of Correlation: In statistics, correlation or dependence is any statistical relationship, whether causal or not, between two random variables or bivariate data. In the broadest sense correlation is any statistical association, though it commonly refers to the degree to which a pair of variables are linearly related.

2.2.4 Feature engineering

Above Fig 2.6 is showing there is strong relationship between independent variables 'temp' and 'atemp' so considering any one feature enough to predict the target "cnt" better.

And it is also showing there is almost no relationship between independent variable 'hum' and dependent variable 'cnt'. so, 'hum' is not so important to predict so we can discard this.

We have Created a new column named as "mean_temp_atemp" using the mean of temp and atemp.

Code in Python: *df_day["mean_temp_atemp"] = (df_day["temp"]+ df_day["atemp"])/2*

2.2.5 Dimensionality Reduction using Chi square test of independence

There are several methods to check the relation between categorical variable , but we used Chi square test of independence for categorical variables to get the importance of variables .

Figure 2.2.5.1 Variable Importance using chi square test

	dteday	season	yr	mnth	holiday	weekday	workingday	weathersit	temp	atemp	hum	windspeed	casual	registered	cnt
dteday	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
season	0	1	0	1	0	0	0	1	1	0	0	0	1	0	0
yr	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
mnth	0	1	0	1	0	0	0	1	1	0	0	0	0	0	0
holiday	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0
weekday	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0
workingday	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0
weathersit	0	1	0	1	0	0	0	1	1	0	1	0	0	0	0
temp	0	1	0	1	0	0	0	1	1	1	1	1	0	0	0
atemp	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0
hum	0	0	0	0	0	0	0	1	1	0	1	1	0	0	0
windspeed	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0
casual	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0
registered	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
cnt	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

The above figure shows that variable ‘dteday’ ‘holiday’ , ‘weekday ‘ and ‘mnth’ are less important in prediction of the target ‘cnt’.

So these variables need to remove before performing Modelling.

Dropping columns before final modelling are: [‘dteday', ‘temp', ‘atemp', ‘weekday', ‘hum', ‘season', 'holiday']

2.2.6 Features Scaling

In most cases, when we normalise data we eliminate the units of measurement for data, enabling to more easily compare data from different places. Some of the more common ways to normalise data include:

Transforming data using a z-score or t-score. This is usually called standardisation. In the vast majority of cases, Rescaling data to have values between 0 and 1. This is usually called feature scaling or normalising of variable. One possible formula to achieve this is.

Definition of Normalisation :- In statistics and applications of statistics, normalisation can have a range of meanings. In the simplest cases, normalisation of ratings means adjusting values measured on different scales to a notionally common scale, often prior to averaging.

$$x_{new} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

In our dataset numeric variables like ‘temp’ , ‘atemp’ ,’hum’ and ‘ windspeed’ are in normalised form so , we have to Normalise two variables ‘casual’ and ‘registered’.

After normalisation of ‘casual’ and ‘registered’ variables look like in table below where all values lies between 0 and 1.

Below Table contains values after Normalisation of ‘casual’ and ‘registered .

casual	registered
0.096538	0.091539
0.037852	0.093849
0.034624	0.174560

Chapter 3
Modelling

3.1 Model Selection

In previous stages of analysis (EDA) we found that few variables like ‘windspeed’ , ’casual, ’registered‘ are going to play key role in model development ,for model dependent first we need to check our target variable.

Target cnt is continuous variable , so we need to perform linear regression type predictive analysis.We will start our model building with Decision Tree regressor.

Before model selection we have divided the dataset into train and test part using random sampling. Where train contains 75% data of data set and test contains 25% data .

In this case we have to predict the count of bike renting according to environmental and seasonal condition. So the target variable here is a continuous variable. For continuous variable we can use various Regression models. Model having less error rate and more accuracy will be our final model.

Primarily we have chosen below Models :

1. Decision tree for regression target variable

2. Random Forest (with 200 trees)

3. Linear regression

3.1.1 Regression Model evaluation matrix

The main concept is called **residuals** or difference between our predictions Y_pred and actual outcomes Y_true.

We will using two methods to evaluate performance of model .

i. **MAPE** : (Mean Absolute Percent Error) measures the size of the error in percentage terms. It is calculated as the percentage of the average residual error.

$$MAPE = \frac{\overset{\text{Multiplying by 100\% converts to percentage}}{\boxed{100\%}}}{n} \Sigma \left| \frac{\overbrace{y-\hat{y}}^{\text{The residual}}}{\underset{\text{Each residual is scaled against the actual value}}{\boxed{y}}} \right|$$

ii. **RMSE :** (Root Mean Square Error) is a frequently used measure of the difference between predicted values by a model and the values actually observed from the model.

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

3.2 Decision Tree

Decision Tree algorithm can be used to construct a decision tree for regression by replacing Information Gain with Standard Deviation Reduction. A decision tree is built top-down from a root node and involves partitioning the data into subsets that contain instances with similar values (homogenous).
This algorithm has influenced a wide area of machine learning, covering both classification and regression. In decision tree analysis, a decision tree can be used to visually and explicitly represent decisions and decision making. As the name goes, it uses a tree-like model of decisions.

Creating Model

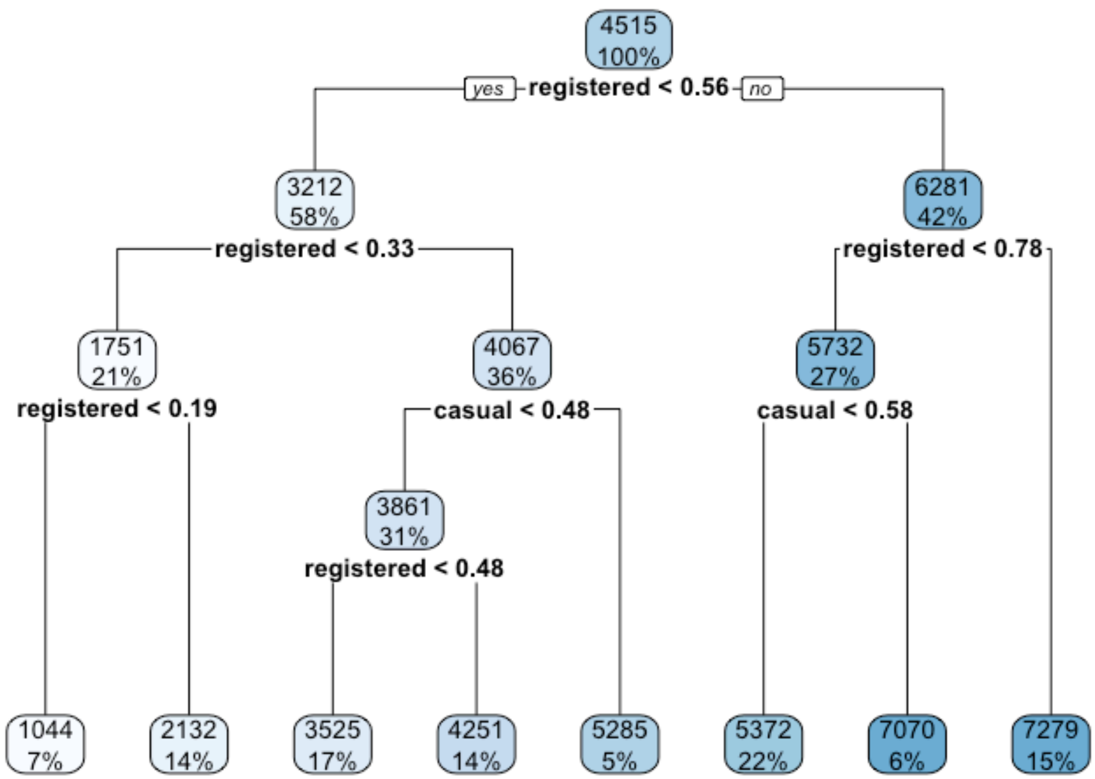
Code in R :

```
DT = rpart(cnt ~ ., data = train, method = "anova")
predictions_DT = predict(DT, target)

#Where target = target = subset(test, select= -c(cnt)) and
#train = train.index = createDataPartition(df_day$cnt, p = .75, list = FALSE)
#train = df_day[ train.index,]
```

Code in python :

```
max_depth = 9
min_samples_split =4
tree = DecisionTreeRegressor(max_depth =max_depth , min_samples_split
=min_samples_split, random_state = 1)
DT_model = tree.fit(X_train, Y_train)
predictions_DT = DT_model.predict(X_test)
```

Figure 3.2.1 Graphical Representation of Decision tree



Above figure is decision tree using two predictors variables to predict the model ,which is not very impressive ,but later will plot with multiple variables.two predictors are 'casual' and 'registered' .

Evaluation of Decision Tree Model

For our model :-

MAPE = 11.54524

RMSE = 538.3437

Accuracy = 88.45476

Model Accuracy is 88.45% it is quite good but RMSE is 538 which is very high so it's clearly stating that our Decision Tree Model is Overfitted and it working well for training data but won't predict good for new set of data. To overcome this overfit we have to tune the model using Random Forest.

3.3 Random Forest

Random forests regressor or random decision forests works on ensemble learning method, this is used for classification, regression and other tasks, that operate by constructing a multitude of decision trees at training time and as output we get the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct for decision trees's habit of overfitting to it's training set.

Random forest working principal shown in below :

i. Draws a bootstrap sample from training data

ii. For each sample grow a decision tree and at each node of the tree

   a. Ramdomly draws a subset of mtry variable and p total of features that are available

   b. Picks the best variable and best split from the subset of mtry variable

   c. Continues until the tree is fully grown.

Creating Model

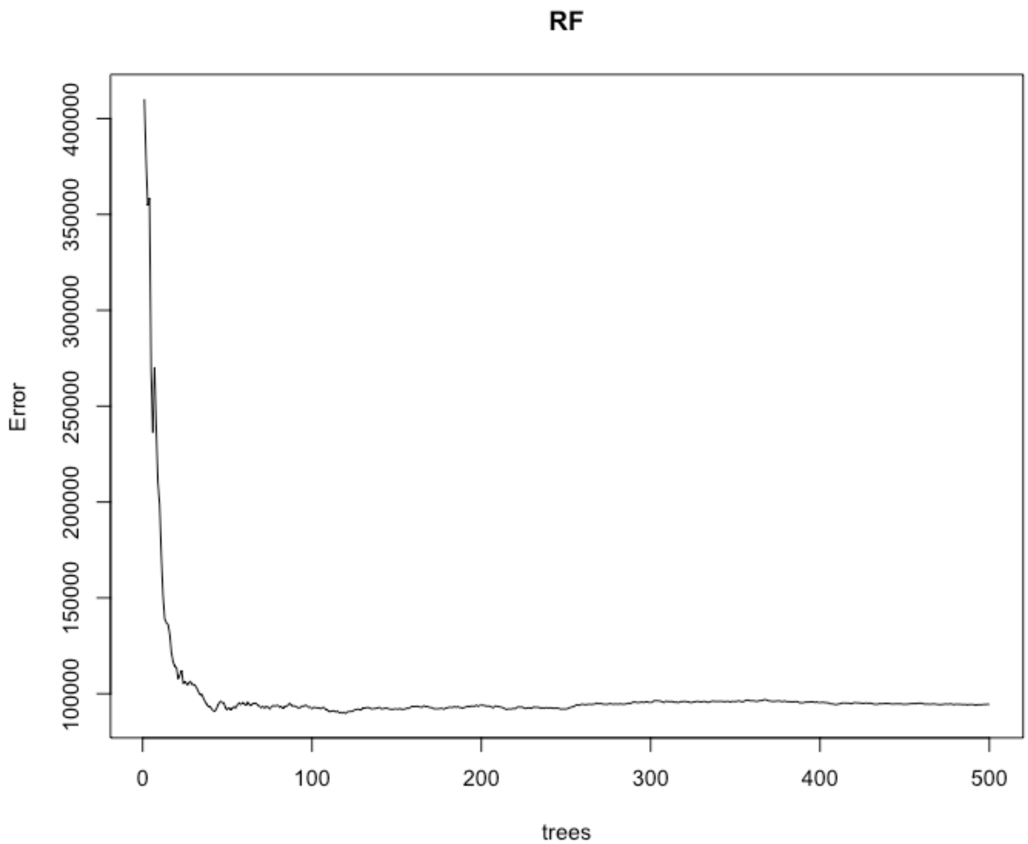
Code in R :

```
RF2=randomForest(cnt ~ . , data = train,mtry =7,ntree=130 ,nodesize =10 ,importance =TRUE)

predictions_RF2 = predict(RF2, target)
```

Code in python :

```
RF_model = RandomForestRegressor(n_estimators= 130,
random_state=100 ).fit(X_train,Y_train)

RF_predict= RF_model.predict(X_test)
```

Figure 3.3.1 shows RF model to show how increasing in no of trees gives high error avoidance.



Evaluation of Random forest Model

As we saw in previous section 3.2 Decision tree is overfitting and its accuracy MAPE and RMSE is also poor in order to improve the performance of the model developing model using Random Forest.

Let us check the error matrix for Random forest regressor model too:

For our model :-

MAPE = 1.902066

RMSE = 158.7459

Accuracy = 98.09793

We can see that now with 130 trees the Model Accuracy is 98% , which is very good and RMSE is decreased to 160 which is quite a low number .So it’s clearly stating that our Decision Tree Model is improved by adding more tree into ensemble method.

Code for Random Forest Implementation :

RF2=randomForest(cnt ~ . , data = train,mtry =7,ntree=130 ,nodesize =10 ,importance =TRUE)

Mtry : Number of variables to split at each node i.e. 7 and Nodesize : size of each node is 10 .

RF2 model is performing good because it utilises maximum no. of variables to predict target.

3.4 Linear Regression

Multiple linear regression is the most common form of linear regression analysis. As a predictive analysis, the multiple linear regression is used to explain the relationship between one continuous dependent variable and two or more independent variables. The independent variables can be continuous or categorical. Using Linear Regression we will predict the 'cnt ' values and compare with Random Forest model output.

VIF (Variance Inflation factor) : It quantifies the multicollinearity between the independent variables.We calculate VIF As Linear regression will work well if multicollinearity among the Independent variables are less.

Importance of Independent variables are presented below: in table figure 3.4.1

Variables	%IncMSE	IncNodePurity
yr	3.6268365	8496549.6
mnth	6.0850220	7663850.6
workingday	5.3135224	4866452.4
weathersit	0.5055354	406126.7
windspeed	1.9344373	1941412.3
casual	35.6986185	367889207.3
registered	68.3234390	1690469647.1

Variables	%IncMSE	IncNodePurity
mean_temp_atemp	5.3022815	10678276.4

Above figure showing variable registered is very highly important and windspeed is not that important. Variable importance in defending order is like :

"registered" > “casual" > “mnth" > “workingday" > "mean_temp_atemp" > "yr" > "windspeed" > "weathersit"

Figure 3.4.2 Multiple Linear Regression Model summary

	coef	std err	t	P>|t|	[0.025	0.975]
yr	-0.6034	0.486	-1.242	0.215	-1.558	0.351
mnth	0.2321	0.050	4.604	0.000	0.133	0.331
weekday	0.3489	0.083	4.215	0.000	0.186	0.511
weathersit	3.8768	0.280	13.838	0.000	3.326	4.427
windspeed	31.4641	1.821	17.275	0.000	27.886	35.042
casual	3409.861	1.079	3161.131	0.000	3407.742	3411.980
registered	6933.576	1.265	5483.208	0.000	6931.093	6936.061
mean_temp_atemp	6.6413	1.422	4.669	0.000	3.847	9.436

In above summary table for linear regression we can see the p value , standard error and regression coefficients .

Creating Model

Code in R :

lm_model = lm(cnt ~., data = train)
predictions_LR = predict(LR_model, target)

Code in python :

LR_model = sm.OLS(Y_train,X_train).fit()
predict_LR = LR_model.predict(X_test)

Evaluation of Linear regression Model

For our model :-

Residual standard error: 2.788e-12 on 531 degrees of freedom

Multiple R-squared: 1, Adjusted R-squared: 1

F-statistic: 1.421e+31 on 19 and 531 DF, p-value: < 2.2e-16

Here residual Standard error is quite less so the distance between predicted values y_pred and actaual values y_true are very less so this model is predicted almost accurate values, and Multiple R-Square value is 1 so, we can explain about 100 % of the data using our multiple linear regression model. This is very much impressive.

MAPE = 0.10486641678600257

RMSE = 3.7229888197677377

Accuracy = 99.89513 = ~100

We can see that the linear regression Model Accuracy is approximately 100% , which is very great and RMSE is also decreased a lot and value is 3.7, So it's clearly stating that our Linear Regression Model is the showstopper among three algorithms.

Final Model Selection :

As per our objective for this project is to predict counts for Bike Rental using three Models i.e Decision Tree, Random Forest and Linear Regression as MAPE and RMSE is less for the Linear regression Model compared to other two, so we can conclude based on error matrix :

Conclusion: - ***For the Bike Rental prediction using day.csv dataset, Linear Regression Model is best model to predict the count.***

Appendix A - R Code

```
#************************Load required libraries and packages***************************

rm(list=ls()) #remove everything from R, to clear RAM

setwd("/Users/gourikhan/Desktop/Gouri") #set the current working directory
getwd() #get current working directory
#Install required packages
x =
c("ggplot2","corrgram","caret","randomForest","C50","e1071","rpart","sampling","GoodmanKrusk
al","usdm")
lapply(x, require, character.only = TRUE)
rm(x)

#**************************** Install  Require libraries*****************************
library(GoodmanKruskal)
library(corrgram)
library(usdm)
library(rpart)
library(rpart.plot)
#******************************Loading Dataset*************************************
#load Bike rental data in R
df_day= read.csv("day.csv", header = T)

#******************************Exploratory Data Analysis***************************

summary(df_day) # Summarising  data
str(df_day) #structure of data
# Target variable is 'cnt',rest of the variables are independent variable(or predictors)
```

#It shows variables like 'mnth',holiday','weekday','weathersit','season' are catogical variables and we need to change to correct variable types encoding

#Numeric variables like 'temp','atem','hum','windspeed' are given in normalised form

```
df_day$season=as.factor(df_day$season)
df_day$mnth=as.factor(df_day$mnth)
df_day$yr=as.factor(df_day$yr)
df_day$holiday=as.factor(df_day$holiday)
df_day$weekday=as.factor(df_day$weekday)
df_day$workingday=as.factor(df_day$workingday)
df_day$weathersit=as.factor(df_day$weathersit)

df_day=subset(df_day,select = -c(instant))

d1=unique(df_day$dteday)
df=data.frame(d1)
df_day$dteday=format(as.Date(df$d1,format="%Y-%m-%d"), "%d")
df_day$dteday=as.factor(df_day$dteday)
rm(d1)
#***************************Missing Values Analysis**********************************
missing_val = data.frame(apply(df_day,2,function(x){sum(is.na(x))}))
missing_val #CONCLUSION : no missing  values are present in the data set

#**********************************Outlier Analysis*******************************************
# BoxPlots - Distribution and Outlier Check in  numerical variables
numeric_index = sapply(df_day,is.numeric) #selecting only numeric
numeric_data = df_day[,numeric_index]
cnames = colnames(numeric_data)

for (i in 1:length(cnames))
{
```

```r
  assign(paste0("gn",i), ggplot(aes_string(y = (cnames[i]), x = "cnt", group=1), data = subset(df_day))+
           stat_boxplot(geom = "errorbar", width = 0.5) +
           geom_boxplot(outlier.colour="red", fill = "blue" ,outlier.shape=18,
                        outlier.size=1, notch=FALSE) +
           theme(legend.position="bottom")+
           labs(y=cnames[i],x="cnt")+
           ggtitle(paste("Box plot of count for",cnames[i])))
}
#*****************************detect outliers using box plot************************************
gridExtra::grid.arrange(gn1,gn2,ncol=2)
gridExtra::grid.arrange(gn3,gn4,ncol=2)
gridExtra::grid.arrange(gn5,gn6,ncol=2)

#we can see there is very less outliers in hum and few in windspeed but this are weather variables and we thing this might be due to seasonal so will not remove any outliers.
#for casual there is a huge outliers that may be because of there is no normality in the variable
#so first we need to check normality for numerical variable the will decide removing outliers.

#********************************univariate data analysis*************************************
# function to create univariate distribution of numeric  variables
univariate_numeric <- function(num_x) {
  ggplot(df_day)+
    geom_histogram(aes(x=num_x,y=..density..),
                   fill= "grey")+
    geom_density(aes(x=num_x,y=..density..))}

# analyse the distribution of  target variable 'cnt'
univariate_numeric(df_day$cnt) # the above graph is showing 'cnt'  is normally distributed

# analyse the distribution of  independence variable 'windspeed'
univariate_numeric(df_day$casual) # the graph is showing 'casual'  is not normally distributed ,so we need to normalise this variable .
```

```
#************************feature  Scaling NORMALISATION*******************************
pnames = c("casual","registered")
for(i in pnames){
  df_day[,i] = (df_day[,i] - min(df_day[,i]))/
    (max(df_day[,i] - min(df_day[,i])))}

#***************************Bivariate data analysis*********************************
# Visualise categorical Variable 'mnth' with target variable 'cnt'
ggplot(df_day, aes(x=as.factor(mnth), y=cnt),fill="grey") +
  stat_summary(fun.y="mean", geom=“bar")

# Visualize categorical Variable 'weathersit' with target variable 'cnt'
ggplot(df_day, aes(x=as.factor(weathersit), y=cnt),fill="grey") +
  stat_summary(fun.y="mean", geom=“bar")

# Visualize categorical Variable 'weathersit'
ggplot(df_day) +
  geom_bar(aes(x=weathersit),fill="grey")
# count arise according to whether is Clear, Few clouds, Partly cloudy, Partly cloudy.

# Visualize categorical Variable 'mnth'
ggplot(df_day) +
  geom_bar(aes(x=mnth),fill="grey")
# it is showing  counts are not varying monthly

#************************bivariate relationship plot using correlation********************************
#check the relationship between all numeric variable using pair plot
ggpairs(df_day[,cnames])

# verify correleation between Numeric variables
corrgram(df_day[,cnames], order = F,
         upper.panel=panel.pie, text.panel=panel.txt, main = "Correlation Plot")
```

```
# that above plot stating that less relationship between cnt-hum and there is strong positive relationship between temp-cnt and  atemp-cnt,

#but temp and atemp are highly correlated. so we need to delete one variable to avoid multicollinearity or we can take mean value as a new variable and drop both the variables.


#********************visualise the relationship between categorical variable**********************


cat_values<- c("holiday", "mnth", "season","yr","weekday","workingday","weathersit","cnt")
subset_df_day<- subset(df_day, select = cat_values)
GKmatrix<- GKtauDataframe(subset_df_day)
plot(GKmatrix, corrColors = "blue")


#********************Feature Selection or Dimension Reduction***************************
colnames(df_day)
#feature engineering to make new variable out of mean of temp and atemp:
df_day$mean_temp_atemp = (df_day$temp + df_day$atemp)/2

df_day = subset(df_day,select=-c(dteday,temp,atemp,weekday,holiday,hum,season))


#***********************Dividing data into Test Train variables***********************
#Divide data into train and test using stratified sampling method
set.seed(1234)
train.index = createDataPartition(df_day$cnt, p = .75, list = FALSE)
train = df_day[ train.index,]
test  = df_day[-train.index,]

target = subset(test, select= -c(cnt))

#**********************************Model Development************************************

#*******************************develop Decision tree model*******************************
#rpart for regression
```

```
DT = rpart(cnt ~ ., data = train, method = "anova")
#Predict for new test cases
predictions_DT = predict(DT, target)
print(DT)

#  plotting decision tree
tree<- rpart(cnt~., data = train, method = 'anova')
rpart.plot(tree)
#******************************Evaluation Matrix***************************************
#calculate MAPE
MAPE = function(y_true, y_pred){100 *
    mean(abs((y_true - y_pred)/y_true))}

#Evaluate  Model using RMSE
RMSE <- function(y_test,y_predict) {
  difference = y_test - y_predict
  root_mean_square = sqrt(mean(difference^2))
  return(root_mean_square)}

#*******************************Evaluate  Decision tree**************************************
MAPE(test$cnt, predictions_DT)

RMSE(test$cnt, predictions_DT)
#********************************Random Forest regressor************************************
RF=randomForest(cnt ~ . , data = train)
RF
plot(RF)
predictions_RF = predict(RF, target)
#*****************************Evaluate Random Forest algorithm*****************************
MAPE(test$cnt, predictions_RF)

RMSE(test$cnt, predictions_RF)
```

```
#**************************Parameter Tuning for random forest*************************

RF2=randomForest(cnt ~ . , data = train,mtry =7,ntree=130 ,nodesize =10 ,importance =TRUE)
RF2
predictions_RF2 = predict(RF2, target) #Predict for new test cases
#Evaluate Random Forest algorithm after tuning :
MAPE(test$cnt, predictions_RF2)

RMSE(test$cnt, predictions_RF2)
#******************************check Variable  Importance********************************
varimp <- importance(RF2)
varimp
# sort variables as per importance
sort_var <- names(sort(varimp[,1],decreasing =T))
sort_var
varImpPlot(RF2,type = 2) # draw varimp plot

#***************************Develop  Linear Regression Model******************************
lm_model = lm(cnt ~., data = train) #run regression model
summary(lm_model) #Summary of the model

predictions_LR = predict(lm_model, target) # Predict  the Test data

#***************************Evaluate Linear Regression Model*******************************
MAPE(test$cnt, predictions_LR)

RMSE(test$cnt, predictions_LR)
```

 # Conclusion For this Dataset Linear Regression is Accuracy is '99.9'

Appendix B - Python Code

```
#!/usr/bin/env python
# coding: utf-8
# Final Project - Bike Renting
# Gouri Khan
# January 26,2020

# # Load required libraries

import os #To interact with local system directories
import pandas as pd # For data processing, CSV file import export (e.g. pd.read_csv)
import numpy as np  # for linear Algebra
from  matplotlib import pyplot
import matplotlib.pyplot as plt # For plotting and visualization
%matplotlib inline # or inline plots in jupyter notebook
import seaborn as sns # For plotting and visualization
sns.set(color_codes=True) # settings for seaborn plotting style
sns.set(rc={'figure.figsize':(6,6)}) # settings for seaborn plot size
from scipy.stats import uniform # import uniform distribution

os.chdir("/Users/gourikhan/Desktop/Gouri") ##set the current working directory
os.getcwd() ##set the current working directory

# # Loading Dataset

df_day = pd.read_csv("day.csv") ##load Bike rental data in PYTHON
df_day.head() #Print the top 5rows of the dataframe
df_day.shape #understanding shape of data #It contains (731 rows, 16 columns)
df_day.info() #data  consist of all non-null Integers , Float and Object(categorical) variables.
#df_day.dtypes
df_day.describe()
```

```
# iterating the columns to get column names:
df_day.columns

# # exploratory data analysis
import datetime
d1=df_day['dteday'].copy()
for i in range (0,d1.shape[0]):
    d1[i]=datetime.datetime.strptime(d1[i], '%Y-%m-%d').strftime('%d')
df_day['dteday']=d1

#Feature Engineering

#Converting respective variables to required data format :
df_day['dteday']=df_day['dteday'].astype('category')
df_day['season']= df_day['season'].astype('category')
df_day['mnth']=df_day['mnth'].astype('category')
df_day['yr'] = df_day['yr'].astype('category')
df_day['holiday'] = df_day['holiday'].astype('category')
df_day['workingday'] = df_day['workingday'].astype('category')
df_day['weekday']=df_day['weekday'].astype('category')
df_day['weathersit']=df_day['weathersit'].astype('category')

df_day = df_day.drop(['instant'], axis=1)
#we can drop instant column as this is only index values.

# # Missing value analysis

total_missing_value = df_day.isnull().sum()
total_missing_value
#CONCLUSION : There is no missing value in the dataframe.
#Also as per df_day.info() result all variavles are non-null so we can conclude there is no
missing value in dataset.
```

Outlier Analysis

```
plt.boxplot(df_day['temp']) #box plot of temp variable
plt.boxplot(df_day['atemp']) #box plot of atemp variable
plt.boxplot(df_day['hum']) #box plot of humidity variable
plt.boxplot(df_day['windspeed']) #box plot of windspeed variable
plt.boxplot(df_day['casual']) #box plot of casual variable
plt.boxplot(df_day['registered']) #box plot of registered variable
```

we can see there is very less outliers in hum and few in windspeed but this are weather variables and we thing this might be due to seasonal condition so will not remove any outliers.

for casual there is a huge outliers that may be because of there is no normality in the variable

so first we need to check normality for numerical variable the will decide removing outliers.

Univariant analysis for numerical variables

```
# Target variable  analysis
sns.distplot(df_day['cnt']) #Check whether target variable is normal or not
print("Skewness of target variable: %f" % df_day['cnt'].skew())
print("Kurtosis of target variable: %f" % df_day[‘cnt'].kurt())
df_day['cnt'].describe() #descriptive statistics summary
#Skewness is very low,so target variable is normally distributed



sns.distplot(df_day['casual']) #Check whether  variable 'casual'is normal or not
print("Skewness of casual: %f" % df_day['casual'].skew())
print("Kurtosis of casual: %f" % df_day['casual'].kurt())
df_day['casual'].describe()
```

feature Scaling : Normality Check

```
cnames = ['casual','registered']
```

```
for i in cnames :
    df_day[i] = (df_day[i] - min(df_day[i]))/(max(df_day[i]) - min(df_day[i]))



# # Bivariant analysis for numerical variables


#box plot of 'Weekdays' with target 'cnt'
data = pd.concat([df_day['cnt'], df_day["weekday"]], axis=1)
f, ax = plt.subplots(figsize=(10, 6)) #Set the width and hieght of the plot
fig = sns.boxplot(x="weekday", y="cnt", data=data)
fig.axis(ymin=0, ymax=9000);


#CONCLUSION: below Boxplot is saying that for all the weekdays median in between 4000-5000


#box plot of 'holiday' with target 'cnt'
data = pd.concat([df_day['cnt'], df_day['holiday']], axis=1)
f, ax = plt.subplots(figsize=(8, 6))
fig = sns.boxplot(x='holiday', y="cnt", data=data)
fig.axis(ymin=0, ymax=9000);
df_day['holiday'].value_counts()


#CONCLUSION: below Boxplot is saying that median high on non-holidays but on holidays range is huge


#box plot of'workingday' with target 'cnt'
data = pd.concat([df_day['cnt'], df_day['workingday']], axis=1)
f, ax = plt.subplots(figsize=(8, 6))
fig = sns.boxplot(x='workingday', y="cnt", data=data)
fig.axis(ymin=0, ymax=9000);
df_day['workingday'].value_counts()


#CONCLUSION: below Boxplot is saying that median is same almost when compare to workingday
```

#box plot of 'weathersit' with target 'cnt'

data = pd.concat([df_day['cnt'], df_day['weathersit']], axis=1)

f, ax = plt.subplots(figsize=(8, 6))

fig = sns.boxplot(x='weathersit', y="cnt", data=data)

fig.axis(ymin=0, ymax=9000);

#CONCLUSION: below Boxplot is saying that as the weather is good moderate and bad bike renting is varies accordingly

#box plot of 'season' with target 'cnt'

data = pd.concat([df_day['cnt'], df_day['season']], axis=1)

f, ax = plt.subplots(figsize=(8, 6))

fig = sns.boxplot(x='season', y="cnt", data=data)

fig.axis(ymin=0, ymax=9000);

#CONCLUSION: below Boxplot is saying that median is hign on summer and fall season and lowest on spring

#box plot of 'year' with target 'cnt'

data = pd.concat([df_day['cnt'], df_day['yr']], axis=1)

f, ax = plt.subplots(figsize=(8, 6))

fig = sns.boxplot(x='yr', y="cnt", data=data)

fig.axis(ymin=0, ymax=9000);

df_day['yr'].value_counts()

#CONCLUSION: below Boxplot is saying that median higher in 2012, but total count is same for both years

#box plot of 'month' with target 'cnt'

data = pd.concat([df_day['cnt'], df_day['mnth']], axis=1)

f, ax = plt.subplots(figsize=(15, 6))

fig = sns.boxplot(x='mnth', y="cnt", data=data)

fig.axis(ymin=0, ymax=9000);

#CONCLUSION: below Boxplot is saying that median is varying in every month but not the counts

```
#box plot of 'dteday' with target 'cnt'
data = pd.concat([df_day['cnt'], df_day['dteday']], axis=1)
f, ax = plt.subplots(figsize=(20, 6))
fig = sns.boxplot(x='dteday', y="cnt", data=data)
fig.axis(ymin=0, ymax=9000);
```

```
#Feature selection on the basis of various features like correlation, multicollinearity.
cnames = df_day.columns
df_corr = df_day.loc[:,cnames]
f, ax = plt.subplots(figsize=(9, 6))
#Generate correlation matrix
corr = df_corr.corr()
#Plot heatmap using seaborn library
sns.heatmap(corr, mask=np.zeros_like(corr, dtype=np.bool), cmap=sns.diverging_palette(100, 10, as_cmap=True), square=True, ax=ax)
```

```
#correlation matrix among all numeric variables and analyse what are the important variables using pearson corelation
df_corr.corr(method='pearson').style.format("{:.3}").background_gradient(cmap=plt.get_cmap('coolwarm'), axis=1)
```

```
# check relationship of all numeric variables with each other with pair plots
numerical_columns = ["temp","atemp","hum","windspeed","casual","registered","cnt"]
sns.set()
sns.pairplot(df_corr[numerical_columns], height = 5, kind="reg")
plt.show();
```

```
# CONCLUSION :As per above scatter plots and Correlation graph there is strong relation between Independent variable

# 'temp' and 'atemp' so we can discard one of them as they are carrying same information or can take average to a new

# column and discard both.

# variable hum is very less correlated with target cnt, so we can discard this too.


# # Chi Square Test of Independence for Categorical variables

import scipy

from scipy import stats #import chi2_contigency #  for Chi square Test

from scipy.stats import chi2

from scipy.stats import chi2_contingency


cat_names = df_day.columns

len(cat_names)

index=0;

no_of_col = len(cat_names)

relationship_matrix = [[0] * no_of_col for column in range(no_of_col)]

jIndex=0;

for j in cat_names:

    iIndex=0;

    strng=''

    for i in cat_names:

        alpha = 0.05 #Significance Level 5%

        chi2, p, dof ,ex = chi2_contingency(pd.crosstab(df_day[j], df_day[i]))

        critical_value=stats.chi2.ppf(q=1-alpha,df=dof)

        if chi2>=critical_value and p<=alpha:

            #string 1 means there is relation between variables

            strng = "1"

        else:

            #string 0 means there is no relation between variables

            strng = "0"

        Q = [chi2, critical_value, p, alpha , dof]
```

```
        relationship_matrix[iIndex][jIndex] = strng
        iIndex=iIndex+1
    jIndex=jIndex+1


data = pd.DataFrame(relationship_matrix)
data = data.rename(columns={0:"dteday",1:"season",2:"yr",3:"mnth",4:"holiday",5:"weekday",6:"workingday",7:"weathersit",8:"temp",9:"atemp",10:"hum",11:"windspeed",12:"casual",13:"registered",14:"cnt"})
data = data.rename(index={0:"dteday",1:"season",2:"yr",3:"mnth",4:"holiday",5:"weekday",6:"workingday",7:"weathersit",8:"temp",9:"atemp",10:"hum",11:"windspeed",12:"casual",13:"registered",14:"cnt"})
data


# There is a relationship between variable : season mnth
# There is a relationship between variable : season weathersit
# There is a relationship between variable : season temp
# There is a relationship between variable : season casual
# There is a relationship between variable : mnth weathersit
# There is a relationship between variable : mnth temp
# There is a relationship between variable : temp weathersit
# There is a relationship between variable : hum weathersit
# There is a relationship between variable : atemp temp
# There is a relationship between variable : hum temp
# There is a relationship between variable : hum windspeed
# There is a relationship between variable : windspeed temp
# There is a relationship between variable : holiday weekday
# There is a relationship between variable : holiday workingday
# There is a relationship between variable : weekday workingday
# so we can
# Remove weekday, holiday because this is co-related with workingday, removing season because this is correlated with weathersit ,mnth,
# dteday variable as this variable has no impact on output.
```

```
## Feature Engineering
#Create a new column of the mean of temp and atemp.
df_day["mean_temp_atemp"] = (df_day["temp"]+ df_day["atemp"])/2

df_day = df_day.drop(['dteday','temp','atemp','weekday','hum','season','holiday'], axis =1)

# # Splitting Test and train data using sklearn train_test_split
X = df_day.loc[:, df_day.columns != 'cnt']
Y = df_day['cnt']

from sklearn.model_selection import train_test_split ,cross_val_score
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.25, random_state=25)

# # Decision Tree Regressor
#Importing Decision Tree Regressor from sklearn.tree
from sklearn.tree import DecisionTreeRegressor
from sklearn import metrics

#Calculate MAPE
def MAPE(y_true, y_pred):
    mape = np.mean(np.abs((y_true - y_pred) / y_true))*100
    return mape

#Calculate RMSE
def RMSE(y_test,y_predict):
    mse = np.mean((y_test-y_predict)**2)
    rmse=np.sqrt(mse)
    return rmse


max_depth = 9
min_samples_split =4
```

```
tree = DecisionTreeRegressor(max_depth =max_depth , min_samples_split
=min_samples_split, random_state = 1)

DT_model = tree.fit(X_train, Y_train)

print(DT_model)


predictions_DT = DT_model.predict(X_test)

MAPE(Y_test,predictions_DT)

RMSE(Y_test,predictions_DT)

metrics.mean_absolute_error(Y_test, predictions_DT)

metrics.mean_squared_error(Y_test, predictions_DT)


# # Decision tree plot using pydotplus and graphviz:

from sklearn.externals.six import StringIO

from IPython.display import Image

from sklearn.tree import export_graphviz

import pydotplus

import graphviz


dot_data = StringIO()

export_graphviz(tree, out_file=dot_data,

                filled=True, rounded=True,

                special_characters=True)

graph = pydotplus.graph_from_dot_data(dot_data.getvalue())

Image(graph.create_png())


# # Random Forest

# Import random forest regressor from sklearn.ensemble

from sklearn.ensemble import RandomForestRegressor


#Random forest model building

RF_model = RandomForestRegressor(n_estimators= 130,
random_state=100).fit(X_train,Y_train)

print(RF_model)
```

```
# Predict the model using predict funtion
RF_predict= RF_model.predict(X_test)

#Evaluate Random forest using  MAPE and RMSE
MAPE(Y_test,RF_predict)
RMSE(Y_test,RF_predict)
#Mean Absolute Error (MAE)
metrics.mean_absolute_error(Y_test, RF_predict)
#Mean Squared Error (MSE)
metrics.mean_squared_error(Y_test, RF_predict)

# # Linear Regression
#import required library for linear regreesion
import statsmodels.api as sm

cnames = df_day.columns
df_day[cnames] = df_day[cnames].apply(pd.to_numeric, errors='coerce', axis=1)

X = df_day.loc[:, df_day.columns != 'cnt']
Y = df_day[‘cnt']
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.25, random_state=25)

#develop Linear Regression model using sm.ols
LR_model = sm.OLS(Y_train,X_train).fit()
LR_model
#predict target using LR model
predict_LR = LR_model.predict(X_test)
# Print the statistics
LR_model.summary()
#Predict the model using  RMSE and MAPE
print(RMSE(Y_test,predict_LR))
```

#evaluate model using

print(MAPE(Y_test,predict_LR))

it is showing that Linear Regression model is best suitable for the dataset.

 # Conclusion : Linear regression is the best model for the dataset.

** **End of Document**************************************