Bike Renting Analysis and Prediction

Sourabh Shivendra Pathak

10th oct 2019

CONTENTS

Introduction

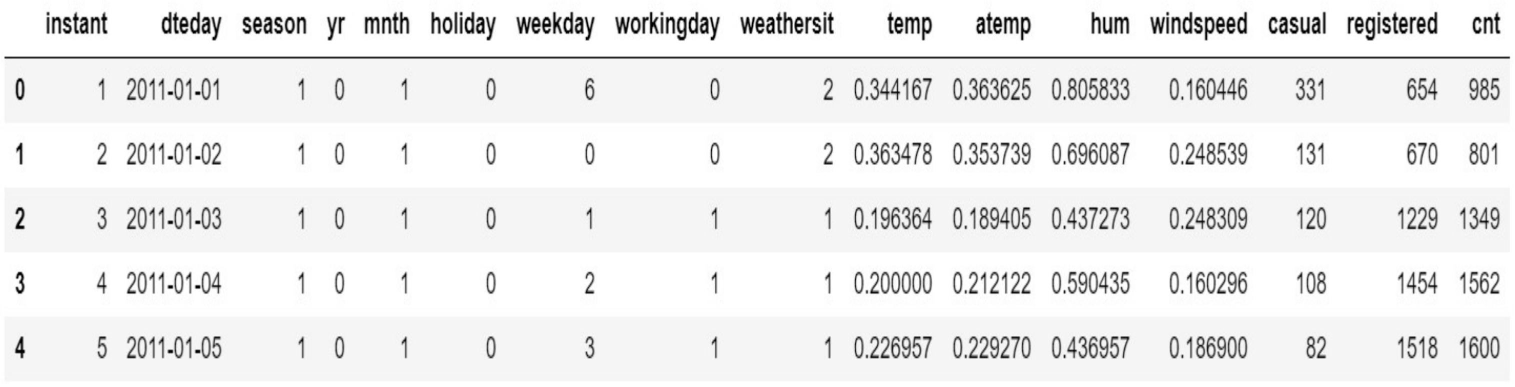
1.1 Problem Statement

A bicycle-sharing system, public bicycle scheme, or bike-share system is a service in which bicycles are made available for shared use to individuals on a short-term basis for a price or free.

The objective of this Case is to Predication of bike rental count on daily based on the environmental and seasonal settings

1.2 Data

Our goal is to develop a regression model which will give the daily count of rental bikes based on weather and season based on following dataset.



	instant	dteday	season	yr	mnth	holiday	weekday	workingday	weathersit	temp	atemp	hum	windspeed	casual	registered	cnt
0	1	2011-01-01	1	0	1	0	6	0	2	0.344167	0.363625	0.805833	0.160446	331	654	985
1	2	2011-01-02	1	0	1	0	0	0	2	0.363478	0.353739	0.696087	0.248539	131	670	801
2	3	2011-01-03	1	0	1	0	1	1	1	0.196364	0.189405	0.437273	0.248309	120	1229	1349
3	4	2011-01-04	1	0	1	0	2	1	1	0.200000	0.212122	0.590435	0.160296	108	1454	1562
4	5	2011-01-05	1	0	1	0	3	1	1	0.226957	0.229270	0.436957	0.186900	82	1518	1600

There are 16 variables in the dataset

S.no	Variables	Definition	S.no	Variables	Definition
1	instant	Record index for entry in dataset	9	weathersit	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
2	dteday	date of the data	10	temp	Normalized temperature in Celsius.
3	season	Season (1: springer, 2:summer, 3:fall, 4:winter)	11	atemp	Normalized feeling temperature in Celsius.
4	yr	Year (0: 2011, 1:2012)	12	hum	Normalized humidity
5	mnth	Month (1 to 12)	13	windspeed	Normalized wind speed.
6	holiday	weather day is holiday or not (extracted from Holiday Schedule)	14	casual	count of casual users
7	weekday	Day of the week	15	registered	count of registered users
8	workingday	: If day is neither weekend nor holiday is 1, otherwise is 0.	16	cnt	count of total rental bikes including both casual and registered

Methodology

2.1 Data Preprocessing

Data preprocessing is a data mining technique which is used to transform the raw data in a useful and efficient format.

Steps Involved in Data Preprocessing:
1. Data Cleaning:
The data can have many irrelevant and missing parts. To handle this part, data cleaning is done. It involves handling of missing data, noisy data etc.

2.DataTransformation:
This step is taken to transform the data in appropriate forms suitable for mining process. This involves following ways:

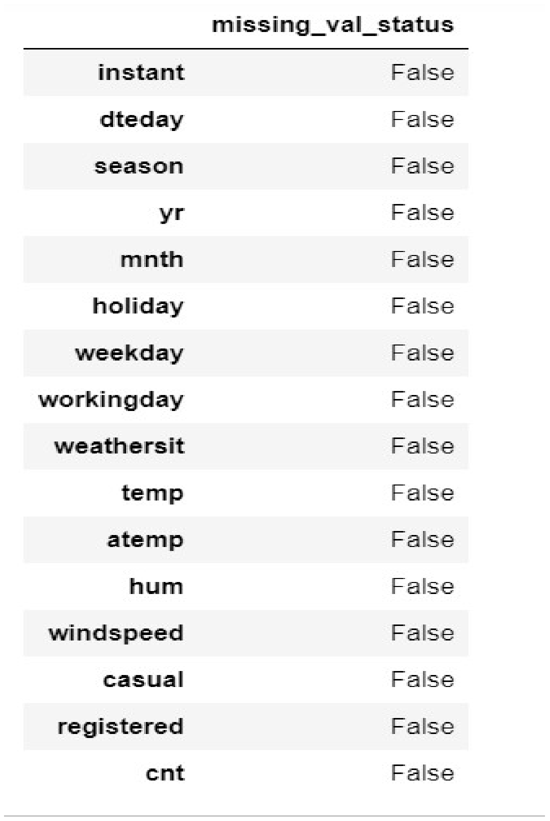
3.DataReduction:
Since data mining is a technique that is used to handle huge amount of data. While working with huge volume of data, analysis became harder in such cases. In order to get rid of this, we use data reduction technique. It aims to increase the storage efficiency and reduce data storage and analysis costs.

Let’s discuss one by one the data preprocessing technique that has been used in this project

2.1.1 Missing value analysis

This situation arises when some data is missing in the dataset. It can be handled in various ways.
Some of them are:

1. **Ignore the entry:**
   This approach is suitable only when the dataset we have is quite large and multiple values are missing within a tuple.
2. **Fill the Missing values:**
   There are various ways to do this task. You can choose to fill the missing values manually, by attribute mean or the most probable value.

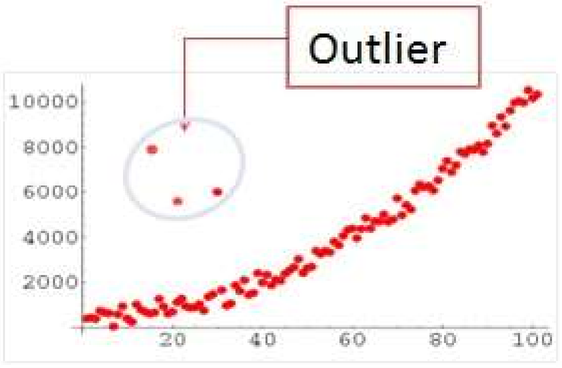


	missing_val_status
instant	False
dteday	False
season	False
yr	False
mnth	False
holiday	False
weekday	False
workingday	False
weathersit	False
temp	False
atemp	False
hum	False
windspeed	False
casual	False
registered	False
cnt	False

There is no missing value in the dataset so we don't have to go for any missing data imputation

2.1.2 Outlier analysis

An outlier is an element of a data set that distinctly stands out from the rest of the data. In other words, outliers are those data points that lie outside the overall pattern of distribution as shown in figure below.

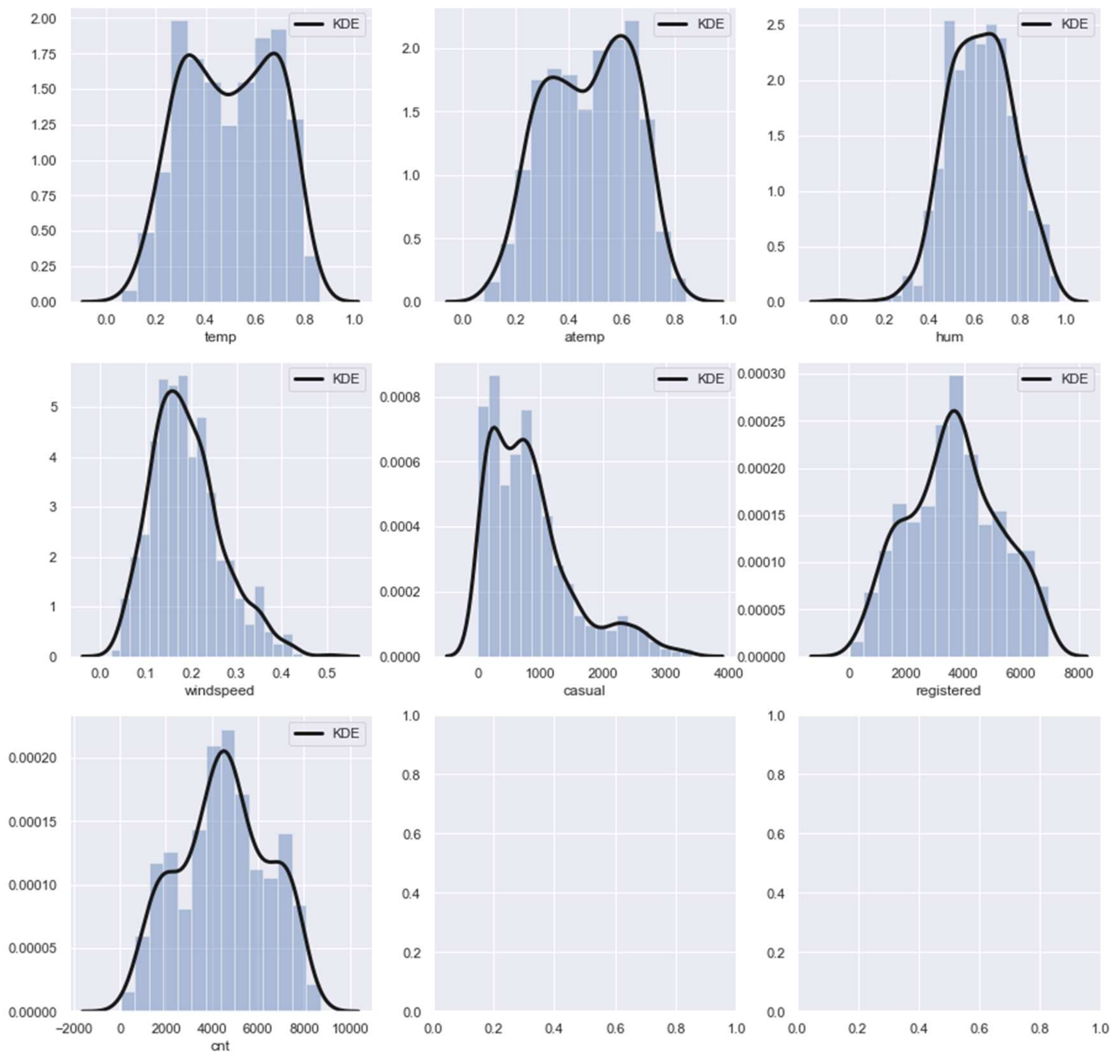


The easiest way to detect outliers is to create a graph. Plots such as Box plots, Scatterplots and Histograms can help to detect outliers. Alternatively, we can use mean and standard deviation to list out the outliers. It can be handled in various ways.

Some of them are:

1. **Ignore the entry:**
   This approach is suitable only when the dataset we have is quite large and outliers are in some data
2. **Delete the outlier and Fill the Missing values:**
   There are various ways to do this task. You can choose to fill the missing values manually, by attribute mean or the most probable value.

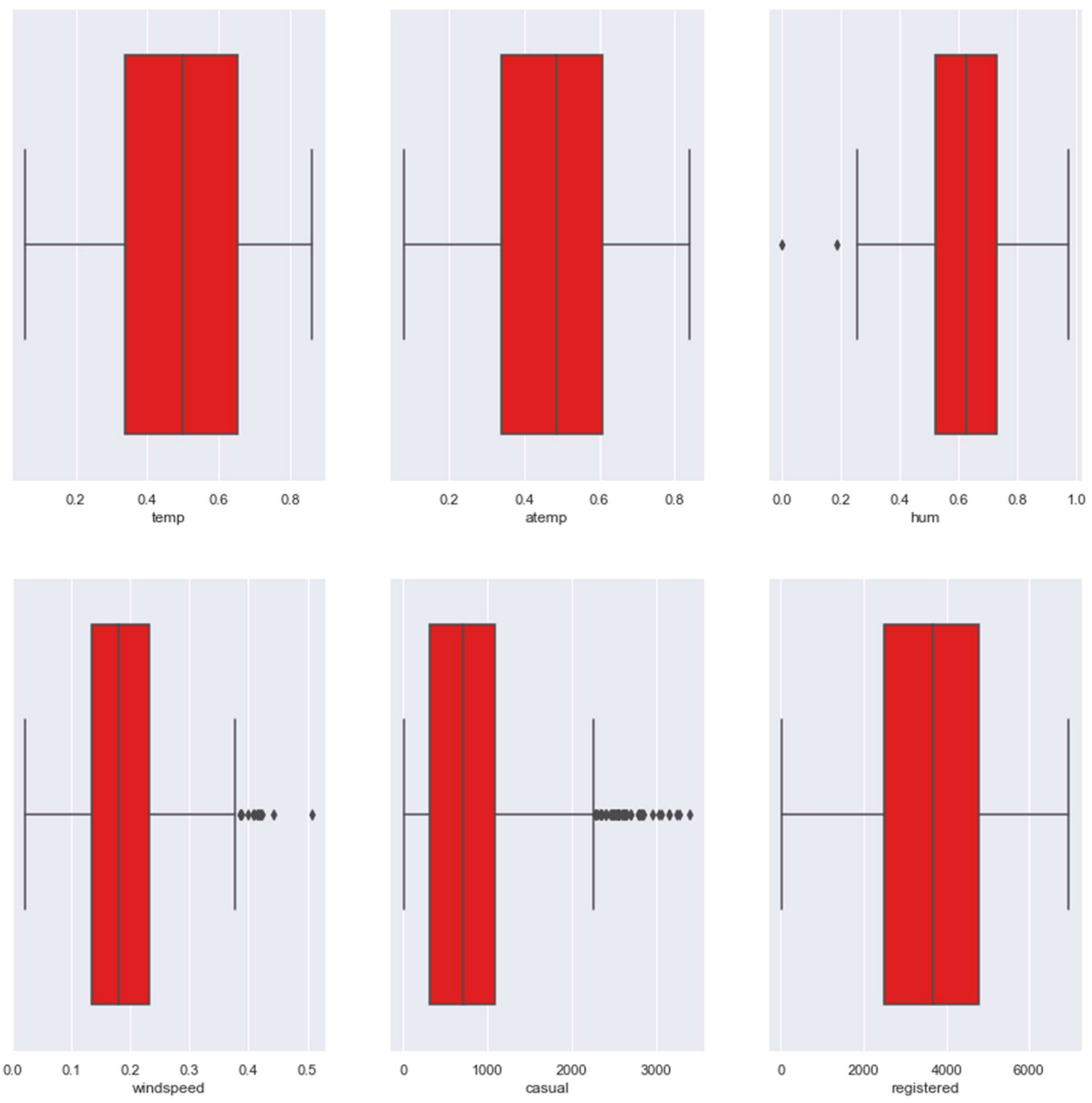
   Below is the distribution graph for all the variable in the given bike dataset



Observations from graphs

1- temp is almost normally distributed
2- atemp is same as temp there might be corelation between temp and a temp
3- hum is almost normally distributed with some otliers in the left side
4- windspeed is slighty right skewed with some outliers in the right side
5- casual is highly right skewed implying that signifacnt amount of outliers are there in casual
6- registered is again almost normally distributed
7- cnt is slightly left skewed but not so significant

Till now the data is almost normally distributed with some outliers in windspeed and casual



Observations from graphs

1-there are only two outliers in the hum column

2-there are few outliers in windspeed

3- there are many outliers in casual column

outliers in casual: 44
outliers in windspeed: 13
outliers in hum: 1
count of data entry lost: 58
data loss percentage after removing outliers: 7.93%
the data loss is less than 10% so we are going for removal of row instead of imputation

2.1.3 Feature Selection

Feature Selection is the process where you automatically or manually select those features which contribute most to your prediction variable or output in which you are interested in. Having irrelevant features in your data can decrease the accuracy of the models and make your model learn based on irrelevant features.

Here are some benefit of feature selection

· **Reduces Overfitting**: Less redundant data means less opportunity to make decisions based on noise.

· **Improves Accuracy**: Less misleading data means modeling accuracy improves.

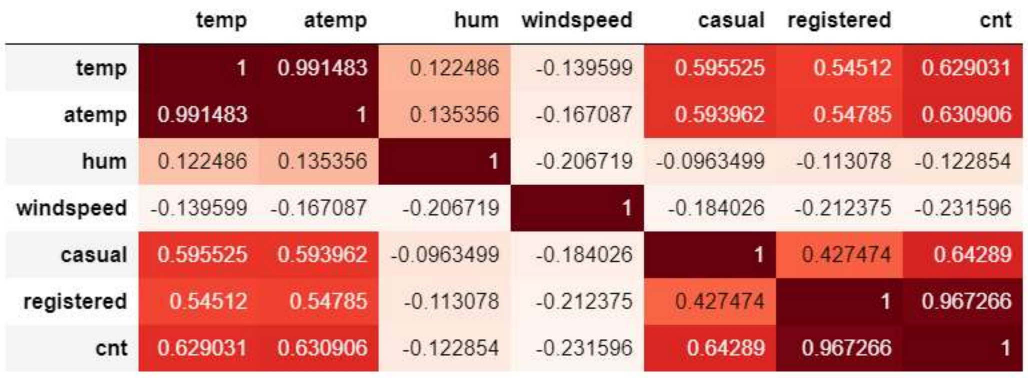
· **Reduces Training Time**: fewer data points reduce algorithm complexity and algorithms train faster.

The numerical variable should have following property

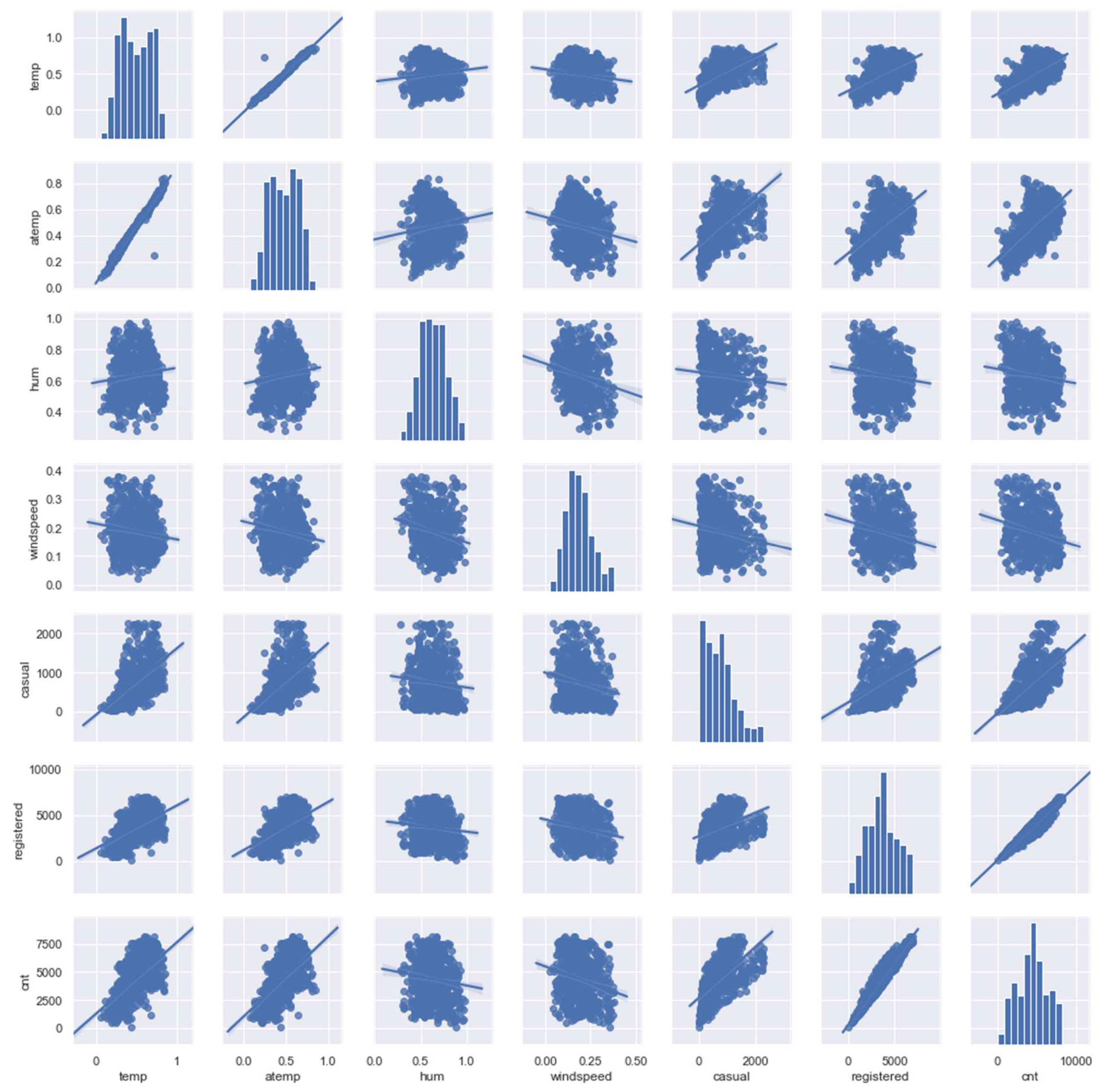
1-high correlation with dependent variables

2-low correlation with other independent variables

Let's check correlation between variable using correlation matrix and pairplot



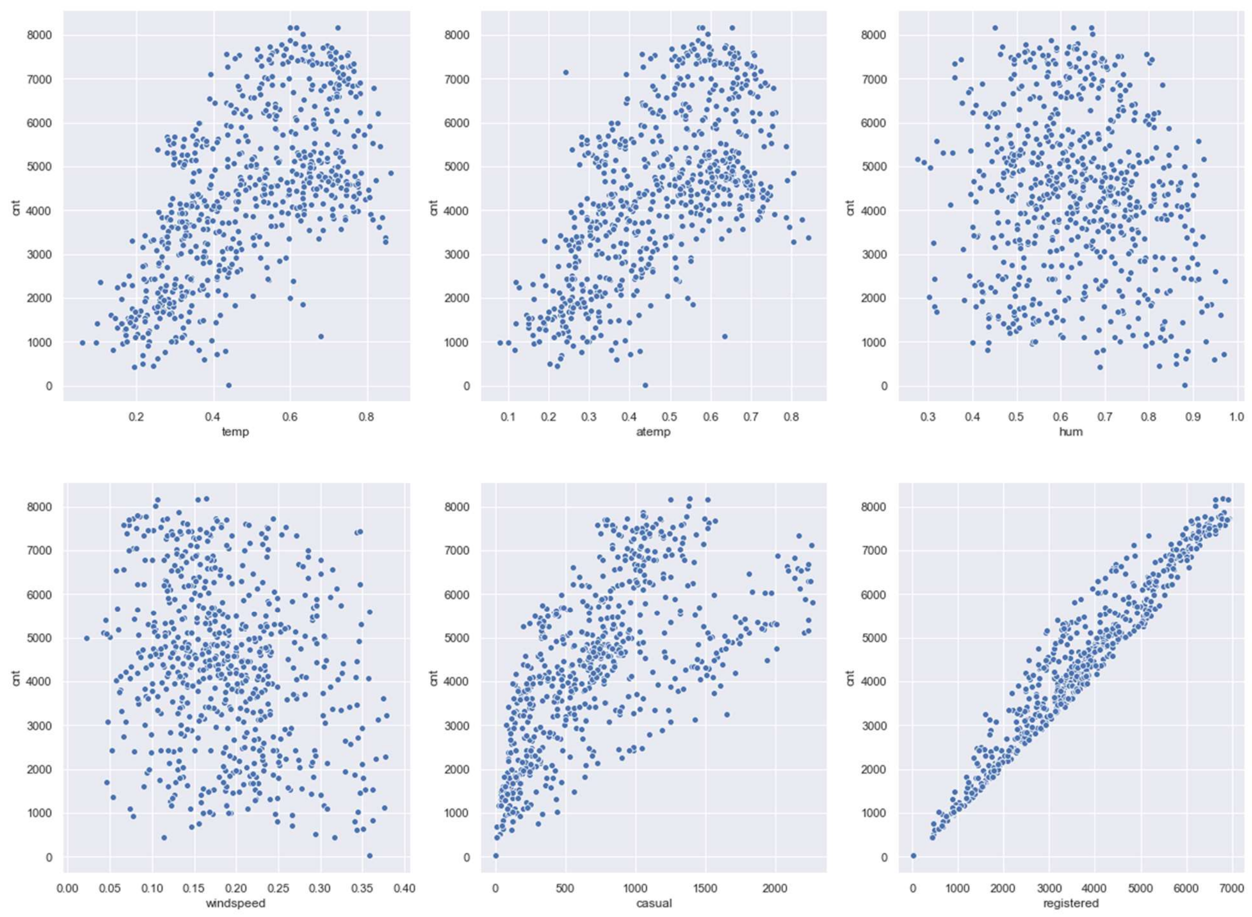
	temp	atemp	hum	windspeed	casual	registered	cnt
temp	1	0.991483	0.122486	-0.139599	0.595525	0.54512	0.629031
atemp	0.991483	1	0.135356	-0.167087	0.593962	0.54785	0.630906
hum	0.122486	0.135356	1	-0.206719	-0.0963499	-0.113078	-0.122854
windspeed	-0.139599	-0.167087	-0.206719	1	-0.184026	-0.212375	-0.231596
casual	0.595525	0.593962	-0.0963499	-0.184026	1	0.427474	0.64289
registered	0.54512	0.54785	-0.113078	-0.212375	0.427474	1	0.967266
cnt	0.629031	0.630906	-0.122854	-0.231596	0.64289	0.967266	1



Observations

1: temp and atemp is highly corelated so we should remove atemp
2: windspeed and cnt very less related to each other so this will not affect much the output
3: hum and cnt very less related to each other so this will not affect much the output

One cooler plot is the scatter plot lets plot scatterplot between cnt and all other variable

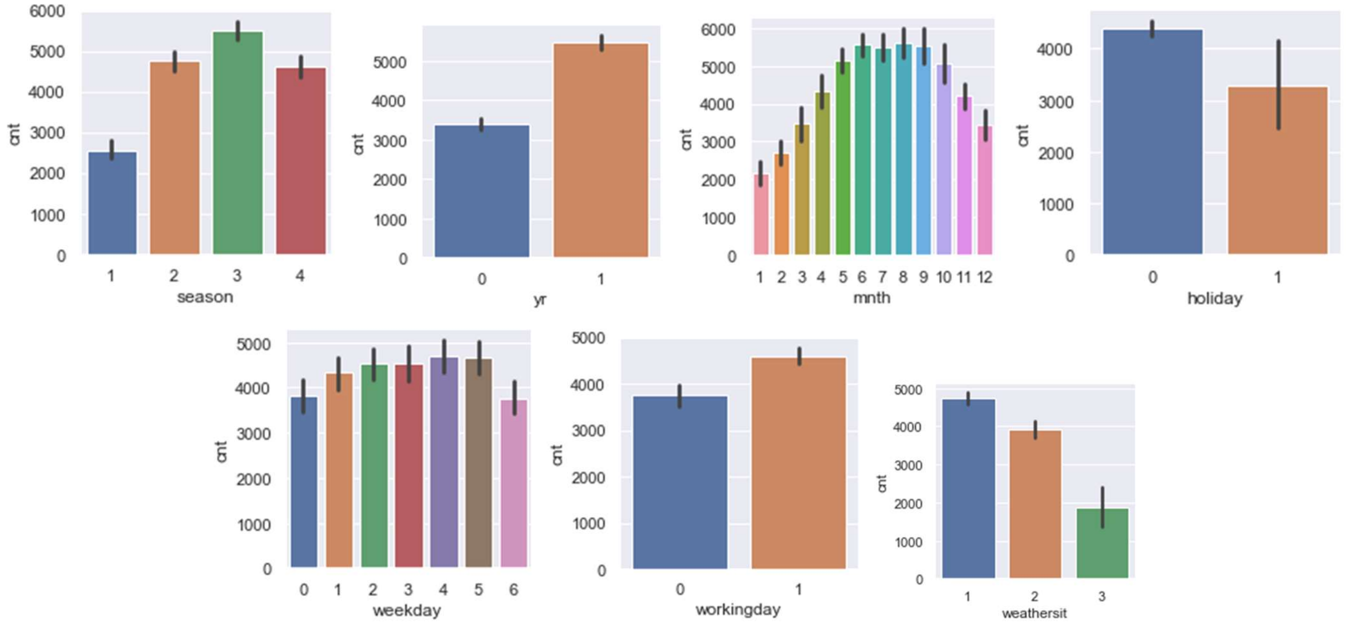


Observations from scatterplot

1:temp is highly positively related with cnt
2:atemp is highly positively related with cnt and same as temp
3:hm is very loosely negatively related with cnt
4:windspeed is also negatively related with cnt but relation is weak
5:casual is positively related with cnt with average relation
6:registered is very strongly positively related with cnt

So by all the observation in the above graphs and plots we come reduce our feature 15 to 12 by removing atemp,hum,windspeed

Now lets look on categorical variables to do so we will plot bar graph between categorical variable and cnt and check the tendency from graph



Observations from the above bar graphs

Season

1: the cnt is low for season 1

2:the cnt for season 2 and 3 is almost same

3:the 3 season has max no. of cnt

Year

the yr 1 has more count in respect with yr 0

Month

the count is less in starting months and increases in the middle of the year and then again decreases

Holiday

if it is a holiday the cnt is less

Weekday

the cnt is less on weekends and somewhat equivalent on the weekdays with slight variation

Working day

if it is a working day the cnt is more

Weather situation

the most cnt is for weather 1 then 2 and 3 has least cnt

2.1.4 Feature scaling

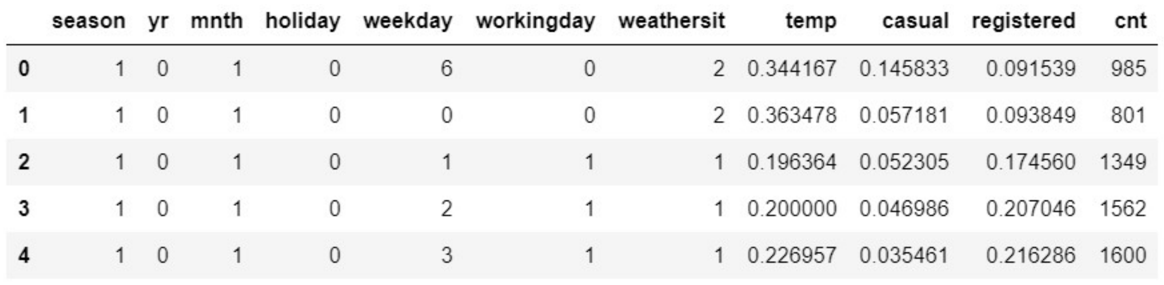
Feature scaling is a method used to standardize the range of independent variables or features of data. In data processing, it is also known as data normalization and is generally performed during the data preprocessing step.

Since the range of values of raw data varies widely, in some machine learning algorithms, objective functions will not work properly without normalization. For example, most classifiers calculate the distance between two points by the distance. If one of the features has a broad range of values, the distance will be governed by this feature. Therefore, the range of all features should be normalized so that each feature contributes approximately proportionately to the final distance.

Another reason why feature scaling is applied is that gradient descent converges much faster with feature scaling than without it.

in the given data we need only to normalize two column casual and registered

after normalization the data looks like



	season	yr	mnth	holiday	weekday	workingday	weathersit	temp	casual	registered	cnt
0	1	0	1	0	6	0	2	0.344167	0.145833	0.091539	985
1	1	0	1	0	0	0	2	0.363478	0.057181	0.093849	801
2	1	0	1	0	1	1	1	0.196364	0.052305	0.174560	1349
3	1	0	1	0	2	1	1	0.200000	0.046986	0.207046	1562
4	1	0	1	0	3	1	1	0.226957	0.035461	0.216286	1600

Modeling

3.1 Linear Regression

Linear regression is a basic and commonly used type of predictive analysis. The overall idea of regression is to examine two things:

(1) does a set of predictor variables do a good job in predicting an outcome (dependent) variable?

(2) Which variables are significant predictors of the outcome variable, and in what way do they–indicated by the magnitude and sign of the beta estimates–impact the outcome variable?

These regression estimates are used to explain the relationship between one dependent variable and one or more independent variables

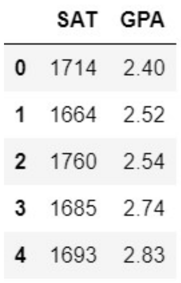
There are basically two type Linear regression

1. Simple Linear Regression

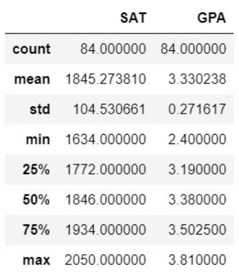
   The simplest form of the regression equation with one dependent and one independent variable is defined by the formula y = c + b*x, where y = estimated dependent variable score, c = constant, b = regression coefficient, and x = score on the independent variable.

   In try to fit the best line possible for the given data lets understand it by an example

   Let's use this dataset

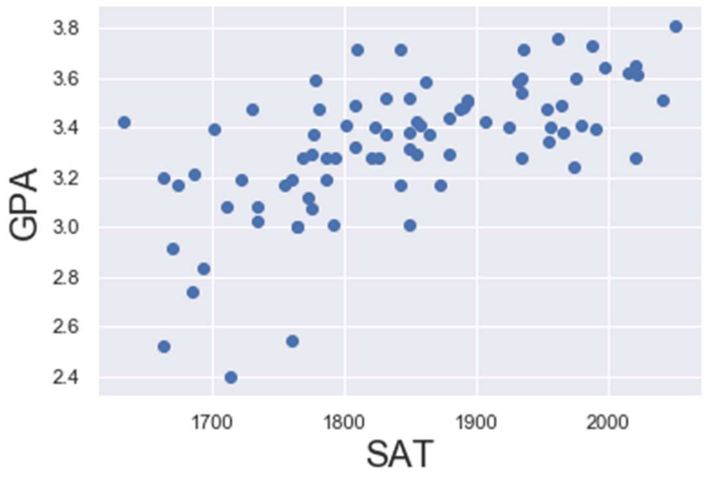


	SAT	GPA
0	1714	2.40
1	1664	2.52
2	1760	2.54
3	1685	2.74
4	1693	2.83

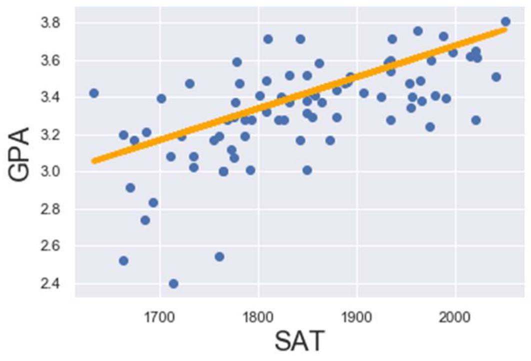


	SAT	GPA
count	84.000000	84.000000
mean	1845.273810	3.330238
std	104.530661	0.271617
min	1634.000000	2.400000
25%	1772.000000	3.190000
50%	1846.000000	3.380000
75%	1934.000000	3.502500
max	2050.000000	3.810000

The scatter plot between sat and gpa is



Let’s try to fit the line on the plot



2.Multiple linear regression

Multiple linear regression (MLR), also known simply as multiple regression, 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.

Multiple regression is the extension of ordinary least-squares (OLS) regression that involves more than one explanatory variable.

The Formula for Multiple Linear Regression Is

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip} + \epsilon$$

where, for $i = n$ observations:

$y_i =$ dependent variable

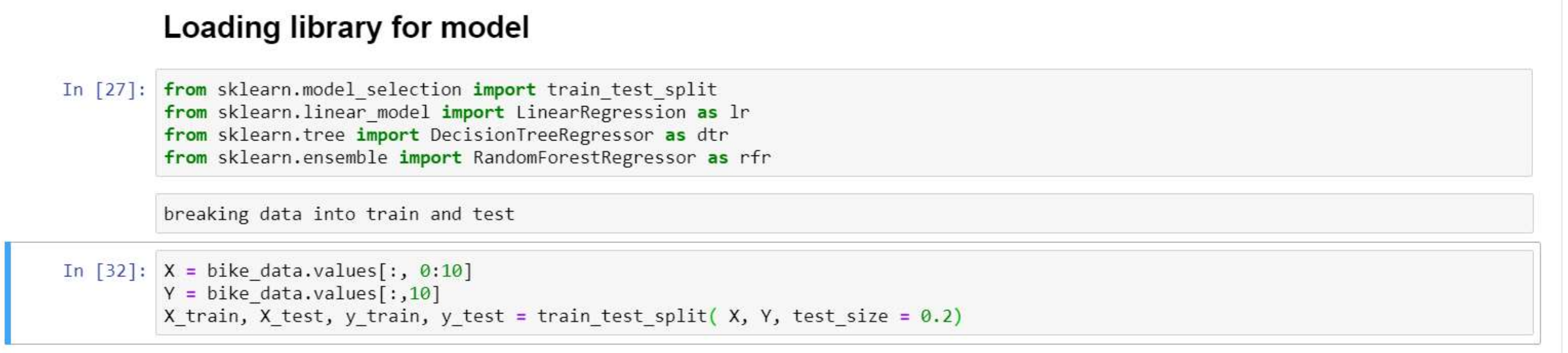
$x_i =$ expanatory variables

$\beta_0 =$ y-intercept (constant term)

$\beta_p =$ slope coefficients for each explanatory variable

$\epsilon =$ the model’s error term (also known as the residuals)

We have used MLR in our model lets check its performance

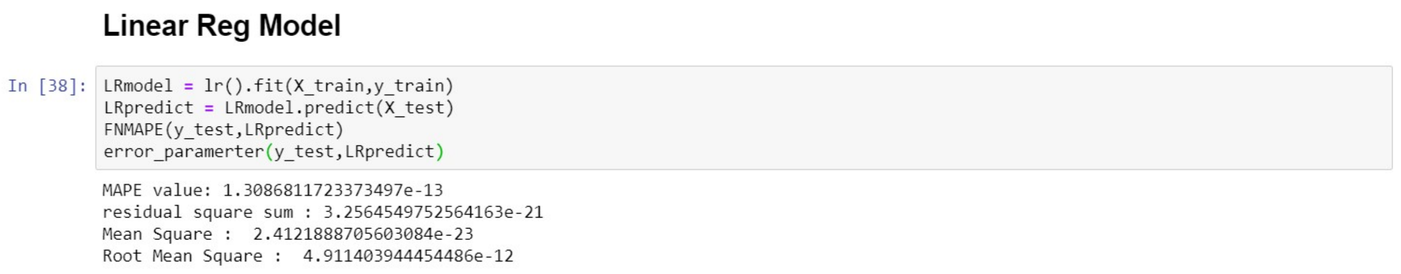


Loading library for model

```
In [27]: from sklearn.model_selection import train_test_split
         from sklearn.linear_model import LinearRegression as lr
         from sklearn.tree import DecisionTreeRegressor as dtr
         from sklearn.ensemble import RandomForestRegressor as rfr
```

```
breaking data into train and test
```

```
In [32]: X = bike_data.values[:, 0:10]
         Y = bike_data.values[:,10]
         X_train, X_test, y_train, y_test = train_test_split( X, Y, test_size = 0.2)
```



Linear Reg Model

```
In [38]: LRmodel = lr().fit(X_train,y_train)
         LRpredict = LRmodel.predict(X_test)
         FNMAPE(y_test,LRpredict)
         error_paramerter(y_test,LRpredict)

         MAPE value: 1.3086811723373497e-13
         residual square sum : 3.2564549752564163e-21
         Mean Square :  2.4121888705603084e-23
         Root Mean Square :  4.911403944454486e-12
```

We will discuss the result of all the model after discussing every model

3.2 Decision Tree

A **decision tree** is a decision support tool that uses a tree-like graph or model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility. It is one way to display an algorithm that only contains conditional control statements.

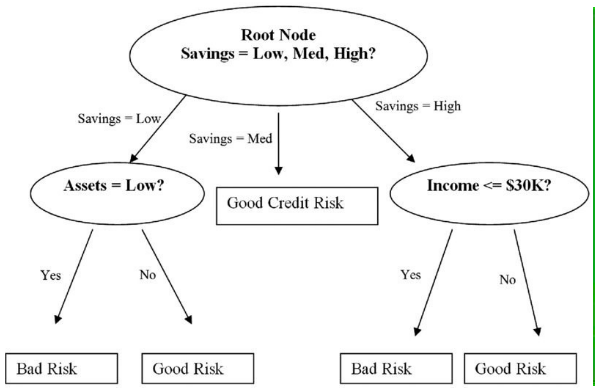
A decision tree is a flowchart-like structure in which each internal node represents a “test” on an attribute (e.g. whether a coin flip comes up heads or tails), each branch represents the outcome of the test, and each leaf node represents a class label (decision taken after computing all attributes). The paths from root to leaf represent classification rules.

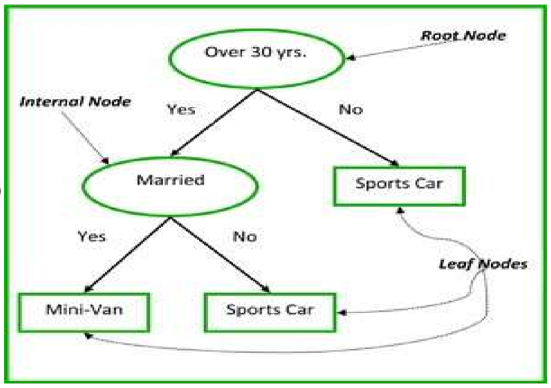
Tree based learning algorithms are considered to be one of the best and mostly used supervised learning methods. Tree based methods empower predictive models with high accuracy, stability and ease of interpretation. Unlike linear models, they map non-linear relationships quite well. They are adaptable at solving any kind of problem at hand (classification or regression). Decision Tree algorithms are referred to as **CART (Classification and Regression Trees)**.

“The possible solutions to a given problem emerge as the leaves of a tree, each node representing a point of deliberation and decision.”

- Niklaus Wirth (1934 —), Programming language designer

Methods like decision trees, random forest, gradient boosting are being popularly used in all kinds of data science problems.

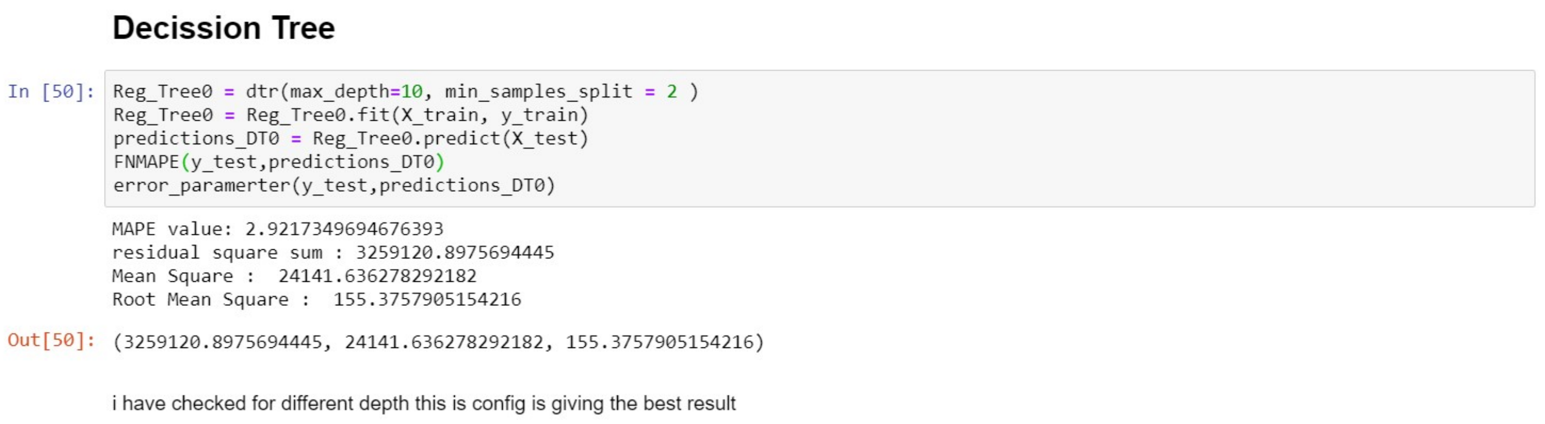




Common terms used with Decision trees:

1. **Root Node:** It represents entire population or sample and this further gets divided into two or more homogeneous sets.
2. **Splitting:** It is a process of dividing a node into two or more sub-nodes.
3. **Decision Node:** When a sub-node splits into further sub-nodes, then it is called decision node.
4. **Leaf/ Terminal Node:** Nodes do not split is called Leaf or Terminal node.
5. **Pruning:** When we remove sub-nodes of a decision node, this process is called pruning. You can say opposite process of splitting.
6. **Branch / Sub-Tree:** A sub section of entire tree is called branch or sub-tree.
7. **Parent and Child Node:** A node, which is divided into sub-nodes is called parent node of sub-nodes whereas sub-nodes are the child of parent node.

Let's check how our data performs on Decision tree



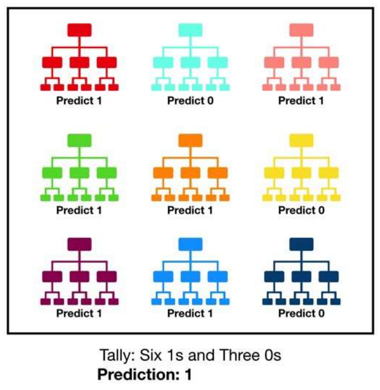
Decission Tree

```
In [50]: Reg_Tree0 = dtr(max_depth=10, min_samples_split = 2 )
         Reg_Tree0 = Reg_Tree0.fit(X_train, y_train)
         predictions_DT0 = Reg_Tree0.predict(X_test)
         FNMAPE(y_test,predictions_DT0)
         error_paramerter(y_test,predictions_DT0)

         MAPE value: 2.9217349694676393
         residual square sum : 3259120.8975694445
         Mean Square :  24141.636278292182
         Root Mean Square :  155.3757905154216

Out[50]: (3259120.8975694445, 24141.636278292182, 155.3757905154216)
```

i have checked for different depth this is config is giving the best result

3.3 Random Forrest

Random forest, like its name implies, consists of a large number of individual decision trees that operate as an ensemble. Each individual tree in the random forest spits out a class prediction and the class with the most votes becomes our model's prediction (see figure below).



Visualization of a Random Forest Model Making a Prediction

The fundamental concept behind random forest is a simple but powerful one — the wisdom of crowds. In data science speak, the reason that the random forest model works so well is:

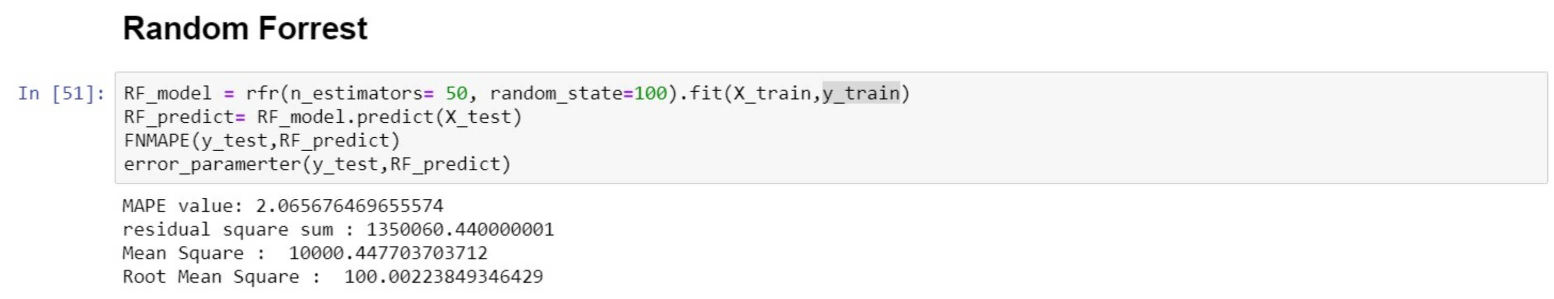
A large number of relatively uncorrelated models (trees) operating as a committee will outperform any of the individual constituent models.

The low correlation between models is the key. Just like how investments with low correlations (like stocks and bonds) come together to form a portfolio that is greater than the sum of its parts, uncorrelated models can produce ensemble predictions that are more accurate than any of the individual predictions. **The reason for this wonderful effect is that the trees protect each other from their individual errors** (as long as they don't constantly all err in the same direction). While some trees may be wrong, many other trees will be right, so as a group the trees are able to move in the correct direction. So the prerequisites for random forest to perform well are:

1. There needs to be some actual signal in our features so that models built using those features do better than random guessing.

2. The predictions (and therefore the errors) made by the individual trees need to have low correlations with each other.

Let's check our model performance for the same



Random Forrest

```
In [51]: RF_model = rfr(n_estimators= 50, random_state=100).fit(X_train,y_train)
         RF_predict= RF_model.predict(X_test)
         FNMAPE(y_test,RF_predict)
         error_paramerter(y_test,RF_predict)

         MAPE value: 2.065676469655574
         residual square sum : 1350060.440000001
         Mean Square :  10000.447703703712
         Root Mean Square :  100.00223849346429
```

Model Evaluation

4.1 Model Evaluation

Model	MAPE	RSS	MSE	RMSE
LR	1.30E-13	3.25E-21	2.41E-23	4.91E-12
DT	2.92	3259120.89	24141.63	155.37
RF	2.06567647	1350060.44	10000.44	100

As you can see the error rate is too low in case Linear reg model as compared to any other model it's because most of the independent variable is related to dependent variable with some certain extent of linearity

Appendix-A

Python Code

```
#pandas offers data structures and operations for manipulating numerical t
ables
import pandas as pd
#The OS module in python provides functions for interacting with the opera
ting system
import os
#numpy is for adding support for large, multi-dimensional arrays and matri
ces
import numpy as np
#Matplotlib is a plotting library
import matplotlib.pyplot as plt
#data visualization library based on matplotlib. It provides a high-level
interface for drawing attractive and informative statistical graphics.
import seaborn as sns
#overriding matplotlib with seaborn
sns.set()
```

Loading datafile and changing dtypes of necessary column

In [2]:

```
bike_data = pd.read_csv("day.csv")
cat_column = ['season', 'yr', 'mnth', 'holiday', 'weekday', 'workingday', 'we
athersit']
bike_data[cat_column] = bike_data[cat_column].apply(pd.Categorical)
```

here we can clearly see that instant column and dteday is usless as it will be unique for every data in the dataset so we will drop these two column¶

In [4]:

```
bike_data = bike_data.drop(['instant','dteday'],axis = 1)
```

here we can clearly see that instant column and dteday is usless as it will be unique for every data in the dataset so we will drop these two column

In [4]:

```
bike_data = bike_data.drop(['instant','dteday'],axis = 1)
```

lets check the datasets

In [5]:

```
bike_data.head(1)
```

Out[5]:

	season	yr	mnth	holiday	weekday	workingday	weathersit	temp	atemp	hum	windspeed	casual	registered	cnt
0	1	0	1	0	6	0	2	0.344167	0.363625	0.805833	0.160446	331	654	985

Missing value analysis

In [6]:

```
missing_data_table =  pd.DataFrame(bike_data.isnull().any())
missing_data_table.rename(columns = {'index': 'Variables', 0: 'missing_val_st
atus'}, inplace=True)
missing_data_table
```

Out[6]:

There are no missing value in any column so we don't have to worry about missing data

we will try to plot graphs for the data sets and try to get some perspective of the data sets

lets check the distribution for all the continuos column so that we can get a perspective of data distribution and the model that can choose best

```
cont_column = ['temp', 'atemp', 'hum', 'windspeed', 'casual', 'registered'
]
fig, axs = plt.subplots(figsize=(15,15), ncols=3, nrows=3)

for i,elem in enumerate(cont_column+['cnt']):
    sns.distplot(bike_data[elem],ax=axs[i//3,i%3], rug_kws = {"color": "w"
},kde_kws = {"color": "k", "lw": 3, "label": "KDE"})
```

Observations from graphs

1- temp is almost normally distributed
2- atemp is same as temp there might be corelation between temp and a temp
3- hum is almost normally distributed with some otliers in the left side
4- windspeed is slighty right skewed with some outliers in the right side
5- casual is highly right skewed implying that signifacnt amount of outliers are there in casual
6- registered is again almost normally distributed
7- cnt is slightly left skewed but not so significant

Till now the data is almost normally distributed with some outliers in windspeed and casual

Outlier Analysis

```
fig, axs = plt.subplots(figsize=(15,15), ncols=3, nrows=2)

for i,elem in enumerate(cont_column):
    sns.boxplot(bike_data[elem],ax=axs[i//3,i%3],color = 'red')
```

Observations from graphs
1-there are only two outliers in the hum column
2-there are few outliers in windspeed
3- there are many outliers in caual column

lets remove these outliers

In [9]:

```
cnames = ['casual','windspeed','hum']
old_data_count = len(bike_data)
for i in cnames:
    q75, q25 = np.percentile(bike_data.loc[:,i], [75 ,25])
    iqr = q75 - q25
    minline = q25 - (iqr*1.5)
    maxline = q75 + (iqr*1.5)

    count_outliers = len(bike_data[bike_data.loc[:,i] < minline].index)+le
n(bike_data[bike_data.loc[:,i] > maxline].index)
    print("outliers in "+i+" :",count_outliers)

    bike_data = bike_data.drop(bike_data[bike_data.loc[:,i] < minline].ind
ex)
    bike_data = bike_data.drop(bike_data[bike_data.loc[:,i] > maxline].ind
ex)


total_row_removed = (old_data_count-len(bike_data))
data_loss_per = round((total_row_removed/old_data_count)*100,2)

print("count of data entry lost : "+str(total_row_removed))
print("data loss percentage after removing outliers : "+ str(data_loss_per
)+"%" )
```

Feature selection

lets plot corelation plot for all numeric variable

In [10]:

```
numeric_column = bike_data.loc[:,cont_column+['cnt']]
numeric_column.corr().style.background_gradient(cmap=plt.get_cmap('Reds'))
sns.pairplot(numeric_column, height = 2,kind="reg")
plt.show();
```

Observations

1: temp and atemp is highgly corelated so we should remove atemp
2: windspeed and cnt very less related to each other so this will not give much data
3: windspeed and cnt very less related to each other so this will not give much data

lets check what kind of realation does continuous independent variable has with dependent variable

In [12]:

```
fig, axs = plt.subplots(figsize=(20,15), ncols=3, nrows=2)

for i,elem in enumerate(cont_column):
    sns.scatterplot(bike_data[elem],bike_data['cnt'],ax=axs[i//3,i%3]);
```

observations

1:temp is highly positeively related with cnt
2:atemp is highly positeively related with cnt and same as temp
3:hm is very loosely neagtively related with cnt
4:windspeed is also neagtively related with cnt but relation is weak
5:casual is positvely related with cnt with average relation
6:registered is very strongly positively related with cnt
Removing atemp,windspeed,hum from data

Lets have a look on tendency of categorical variables

In [13]:

```
for i,elem in enumerate(cat_column):
    fig, ax = plt.subplots(figsize=(3,3))
    sns.barplot(data=bike_data[[elem,
                                 'cnt']],
            x=elem, y='cnt')
```

Observation

Season

1:the cnt is low for season 1
2:the cnt for season 2 and 3 is almost same
3:the 3 season has max no. of cnt

Year

1: the yr 1 has more count in respect with yr 0

Month

the count is less in starting months and increases in the middle of the year and then again decreases

Holiday

if it is a holiday the cnt is less

Weekday

the cnt is less on weekends and somewhat equivalent on the weekdays with slight variation so we should remove it

Working day

if it is a working day the cnt is more

Weather situation

the most cnt is for weather 1 then 2 and 3 has least cnt

we can remove weekday variable also because its not affecting cnt that much

In [14]:

```
bike_data =bike_data.drop(['atemp','hum','weekday'], axis = 1)
```

Feature scaling

only two variable casual and registered need to be scaled

In [15]:

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

for i in cnames:
    bike_data[i] = (bike_data[i] - bike_data[i].min())/(bike_data[i].max() - 
bike_data[i].min())
bike_data.head(5)
```

Model Creation

with our analysis we can see that linear model is most probable fit for this dataset so we will try that first but before that lets make sum analytical function to check the performance of our model

In [16]:

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

In [17]:

```
def error_paramerter(y_test,y_predict):
    rss = ((y_predict-y_test)**2).sum()
    print("residual square sum :",rss)
    mse = np.mean((y_test-y_predict)**2)
    print("Mean Square : ",mse)
    rmse=np.sqrt(mse)
    print("Root Mean Square : ",rmse)
    return rss,mse,rmse
```

Loading library for model

In [18]:

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression as lr
from sklearn.tree import DecisionTreeRegressor as dtr
from sklearn.ensemble import RandomForestRegressor as rfr
```

breaking data into train and test

In [19]:

```
X = bike_data.values[:, 0:10]
Y = bike_data.values[:,10]
X_train, X_test, y_train, y_test = train_test_split( X, Y, test_size = 0.2)
```

Linear Reg Model

In [20]:

```
LRmodel = lr().fit(X_train,y_train)
LRpredict = LRmodel.predict(X_test)
FNMAPE(y_test,LRpredict)
error_paramerter(y_test,LRpredict)
```

```
MAPE value: 2.0692697398395713e-13
residual square sum : 4.425403567031793e-21
```

```
Mean Square :  3.278076716319847e-23
Root Mean Square :  5.7254490796092554e-12
```

Out[20]:

```
(4.425403567031793e-21, 3.278076716319847e-23, 5.7254490796092554e-12)
```

In [25]:

```
LRmodel.coef_
```

Out[25]:

```
array([-1.35183546e-12,  6.62225830e-12, -2.27373675e-13,  2.51909604e-13,
       -2.15205631e-12, -6.44040377e-13,  6.16395823e-13,  2.40296671e-12,
        2.25600000e+03,  6.92600000e+03])
```

Observation

MAPE value: 2.0692697398395713e-13
residual square sum : 4.425403567031793e-21
Mean Square : 3.278076716319847e-23
Root Mean Square : 5.7254490796092554e-12

Decission Tree

In [21]:

```
Reg_Tree0 = dtr(max_depth=10, min_samples_split = 2 )
Reg_Tree0 = Reg_Tree0.fit(X_train, y_train)
predictions_DT0 = Reg_Tree0.predict(X_test)
FNMAPE(y_test,predictions_DT0)
error_paramerter(y_test,predictions_DT0)
```

```
MAPE value: 20.256473661545463
residual square sum : 5364810.738282313
Mean Square :  39739.3388020912
Root Mean Square :  199.34728190294243
```

Out[21]:

```
(5364810.738282313, 39739.3388020912, 199.34728190294243)
```

i have checked for different depth this is config is giving the best result

Observation

MAPE value: 20.256473661545463
residual square sum : 5364810.738282313
Mean Square : 39739.3388020912
Root Mean Square : 199.34728190294243

Random Forrest

In [22]:

```
RF_model = rfr(n_estimators= 50, random_state=100).fit(X_train,y_train)
RF_predict= RF_model.predict(X_test)
FNMAPE(y_test,RF_predict)
error_paramerter(y_test,RF_predict)
```

```
MAPE value: 21.306310345968075
residual square sum : 1257282.5076000008
Mean Square :  9313.203760000006
Root Mean Square :  96.50494163513082
```

Out[22]:

```
(1257282.5076000008, 9313.203760000006, 96.50494163513082)
```

Observation

```
MAPE value: 21.306310345968075
residual square sum : 1257282.5076000008
Mean Square : 9313.203760000006
Root Mean Square : 96.50494163513082
```

from the results from all the model it is quite clear that the Linear reg model is best fitted for this dataset

In [23]:

```
data = X_test
columns = list(bike_data.columns)
columns.remove('cnt')
output_data= pd.DataFrame(data[0:,0:],columns=columns)
output_data['cnt'] = y_test
output_data['p_cnt_lr'] = LRpredict
output_data['p_cnt_dt'] = predictions_DT0
output_data['p_cnt_rf'] = RF_predict
```

Appendix-B

R-code

```
######clear the environment and set the directory

rm(list = ls())

setwd("C:/Users/Spathak/Desktop/Bike_Renting/Bike_Rent_R")



### check the required library is installed or not install if not and then load all the lib



x = c("ggplot2", "corrgram", "purrr", "caret", "randomForest", "tidyr","rpart", 'sampling', 'corrplot','GGally')

lib = x[! x %in% installed.packages()[,'Package']]

if (length(lib))

  install.packages(lib, dependencies = TRUE)

lapply(x, require, character.only = TRUE)

rm(x,lib)



###load the data into a df



bike_data = read.csv('day.csv')


## removing instatnt and dteday because they are always unique

bike_data = subset(bike_data, select = -c(instant,dteday))


##################################Missing Values
Analysis#################################################

missing_val = data.frame(apply(bike_data,2,function(x){sum(is.na(x))}))

missing_val$Columns = row.names(missing_val)

names(missing_val)[1] =  "Missing_percentage"
```

```
missing_val$Missing_percentage = (missing_val$Missing_percentage/nrow(bike_data)) * 100

missing_val = missing_val[order(-missing_val$Missing_percentage),]

row.names(missing_val) = NULL

missing_val = missing_val[,c(2,1)]

#there are no miising value in the dataframe so we don't have to go for missing data analysis

#lets convert column into thier proper datatype

bike_data$season=factor(bike_data$season)

bike_data$yr = factor(bike_data$yr)

bike_data$mnth = factor(bike_data$mnth)

bike_data$holiday =factor(bike_data$holiday)

bike_data$weekday = factor(bike_data$weekday)

bike_data$workingday = factor(bike_data$workingday)

bike_data$weathersit = factor(bike_data$weathersit)




##################################plotting various graph to get idea about
data################################################

#lets plot the distribution of numeric variable to get idea about data distribution

bike_data_num = bike_data %>% keep(is.numeric)


bike_data_num %>% gather() %>% ggplot(aes(value)) +                     # Plot the values

  facet_wrap(~ key, scales = "free") +                    # In separate panels

  geom_density()


ggpairs(bike_data_num)


p1 = ggplot(data = bike_data, aes(x=weekday, y= cnt))+

  geom_bar(stat="identity")


p2 = ggplot(data = bike_data, aes(x=season , y= cnt))+

  geom_bar(stat="identity")
```

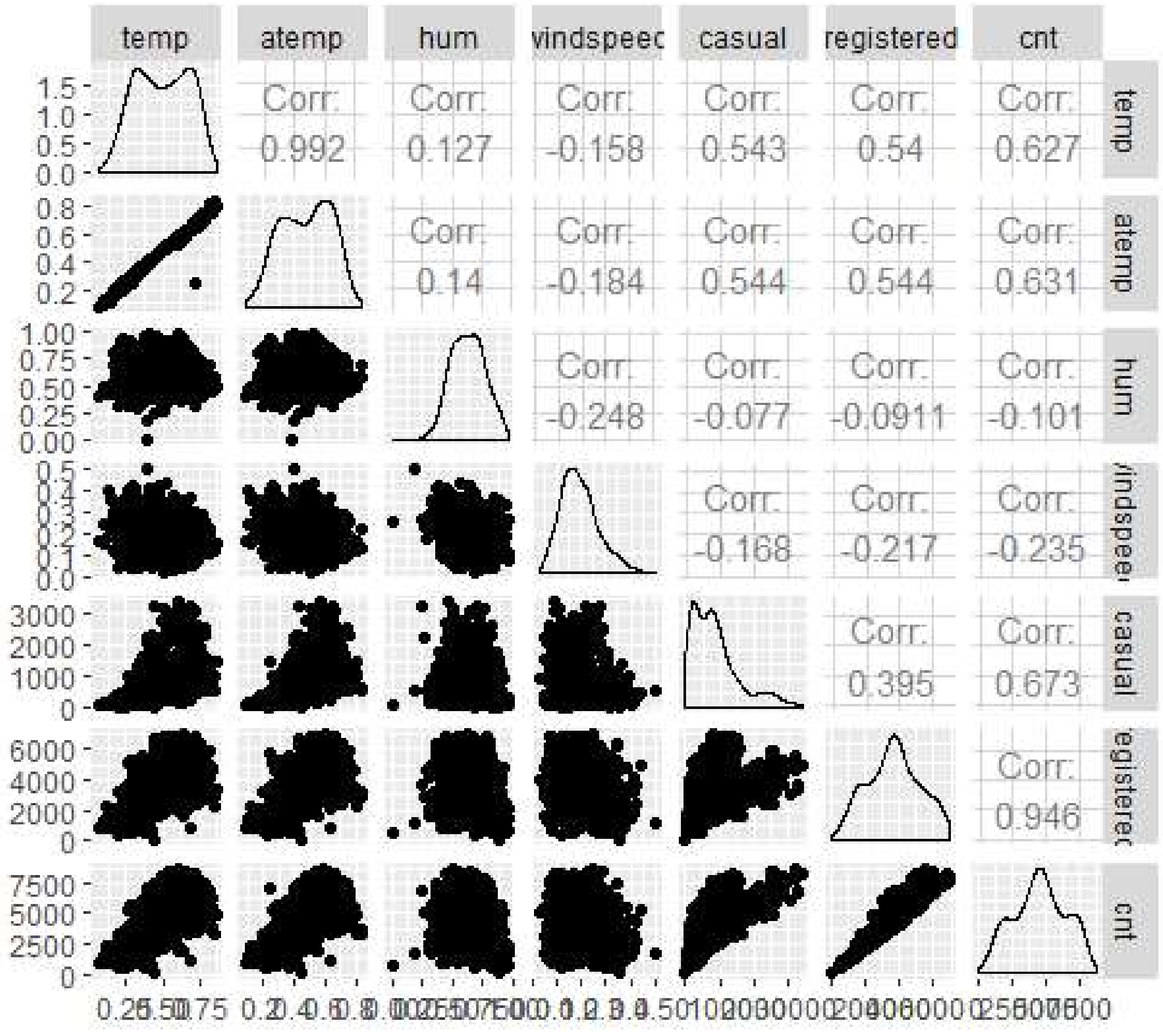
```
p3 = ggplot(data = bike_data, aes(x=yr , y= cnt))+
  geom_bar(stat="identity")


p4 = ggplot(data = bike_data, aes(x=mnth , y= cnt))+
  geom_bar(stat="identity")


p5 =  ggplot(data = bike_data, aes(x=holiday , y= cnt))+
  geom_bar(stat="identity")


p6 = ggplot(data = bike_data, aes(x=workingday , y= cnt))+
  geom_bar(stat="identity")


p7 = ggplot(data = bike_data, aes(x=weathersit , y= cnt))+
  geom_bar(stat="identity")


grid.arrange(p1,p2,p3,p4,p5,p6,p7)




##################################Outliers
Analysis##############################################

bike_data_num %>% gather()  %>% ggplot(aes(x="", y=value)) +                    # Plot the values
  facet_wrap(~ key, scales = "free") +   # In separate panels
  geom_boxplot(outlier.colour="red", fill = "grey" ,outlier.shape=18,outlier.size=1, notch=FALSE) +
  theme(legend.position="bottom")




# removing outliers from the boxlot method
#casual
val = boxplot.stats(bike_data_num$casual)$out
bike_data = bike_data[which(!bike_data$casual %in% val),]
#hum
val = boxplot.stats(bike_data_num$hum)$out
```

```
bike_data = bike_data[which(!bike_data$hum %in% val),]

#windspeed

val = boxplot.stats(bike_data_num$windspeed)$out

bike_data = bike_data[which(!bike_data$windspeed %in% val),]



##################################feature selection################################################

cor_bike_data = cor(bike_data_num)

corrplot(cor_bike_data, method="circle")


ggplot(bike_data, aes(x= casual,y=cnt)) +

  geom_point()


ggplot(bike_data, aes(x = hum, y=cnt)) +

  geom_point()


ggplot(bike_data, aes(x = windspeed , y=cnt)) +

  geom_point()


ggplot(bike_data, aes(x = temp , y=cnt)) +

  geom_point()


ggplot(bike_data, aes(x = atemp, y=cnt)) +

  geom_point()


ggplot(bike_data, aes(x = registered , y=cnt)) +

  geom_point()


#by the scatterplot for all numeric variable we can see windspeed and hum is very weakly related to cnt we will drop these column

#similarly corelaton between temp and atemp is very high so we will remove atemp as well

#there are no use of instant and dteday column  because they are always unique don't add any value so removing these two column
```

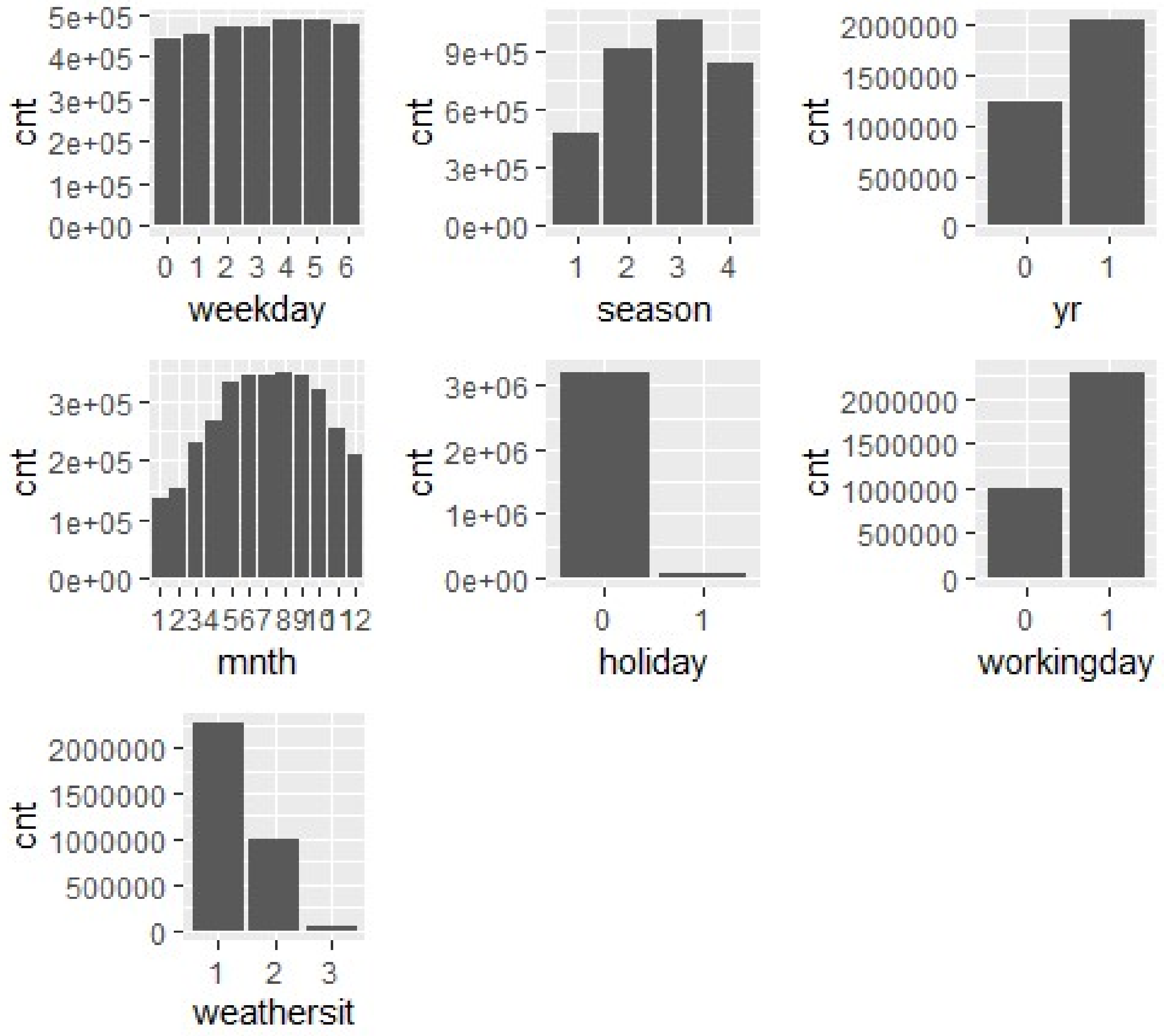
```
bike_data = subset(bike_data, select = -c(hum, windspeed ,atemp))




###################################feature
scaling################################################

cnames = c('registered','casual')

for(i in cnames){

  print(i)

  bike_data[,i] = (bike_data[,i] - min(bike_data[,i]))/

    (max(bike_data[,i] - min(bike_data[,i])))

}

###################################Model
Development########################################

#Clean the environment

rmExcept("bike_data")

#Divide data into train and test

set.seed(123)

index = sample(1:nrow(bike_data), 0.8*nrow(bike_data))

train = bike_data[index, ]

test = bike_data[-index, ]


MAPE = function(y, yper){

  mean(abs((y - yper)/y))

}




lm_model = lm(cnt ~., data = train)

summary(lm_model)

predictions_LR = predict(lm_model, test[,1:10])

#Calculate MAPE

errorlr = MAPE(test[,11], predictions_LR)

#error rate in LR model = 5.405056e-16 almost 100% accuracy
```

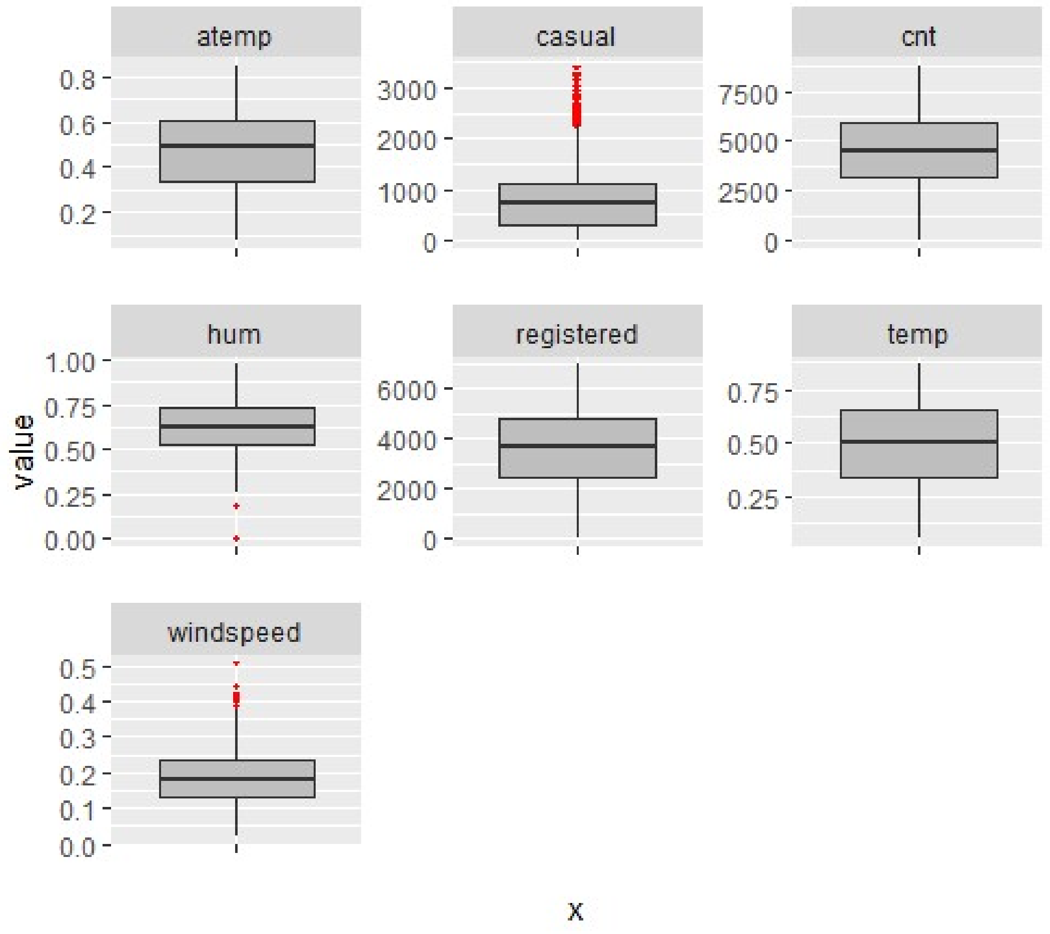
```
dr_model = rpart(cnt ~ ., data = train, method = "anova")

predictions_DT = predict(dr_model, test[,-11])

errordt = MAPE(test[,11],predictions_DT)

# error rate in dt model = 0.1410895 that means 99.85891% accuracy


RF_model = randomForest(cnt ~ ., data = train)

predictions_RF = predict(RF_model, test[,1:10])

errorrf = MAPE(test[,11], predictions_RF)

# error rate in rf model = 0.06399467 that means 99.93601% accuracy

modelval=test

modelval$lr_predict = predictions_LR

modelval$dt_predict = predictions_DT

modelval$RF_predict = predictions_RF


rmExcept(c("bike_data",'dr_model','lm_model','RF_model','errordt','errorlr','errorrf','modelval'))


write.csv(modelval,'OT_TEST_R.csv')
```

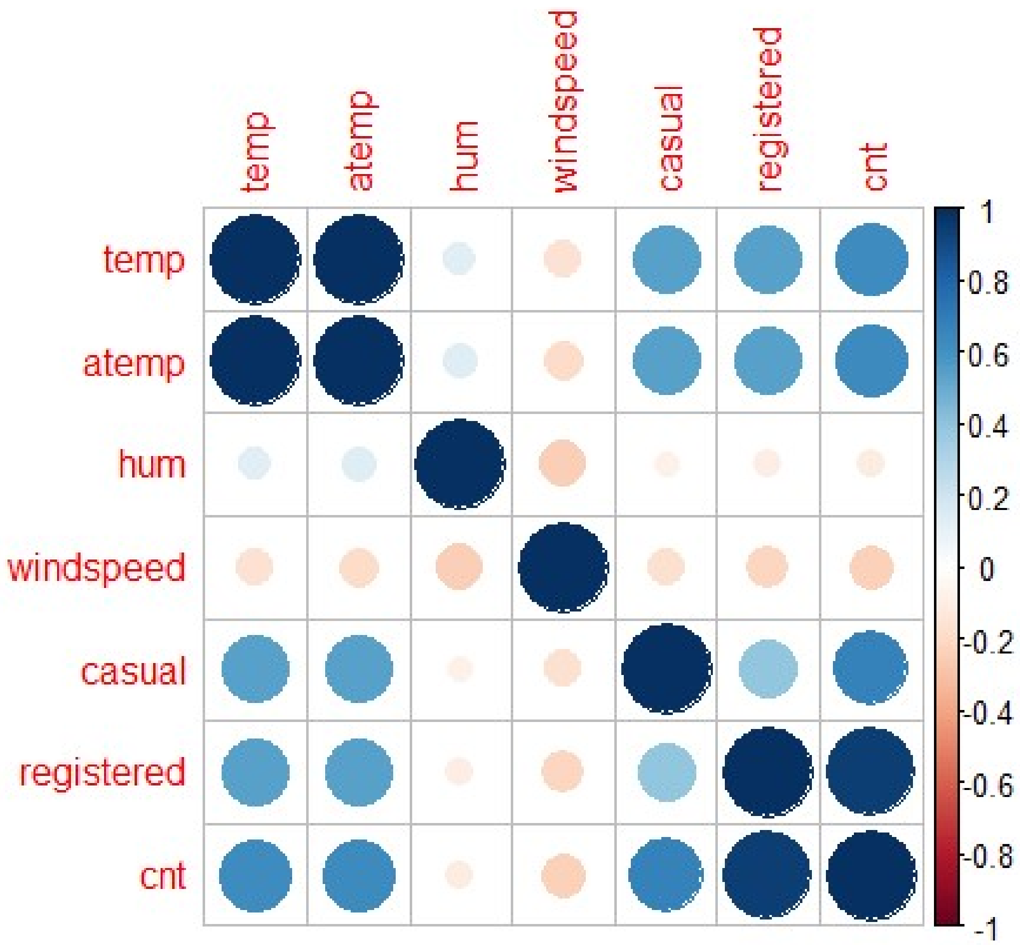
Appendix-c

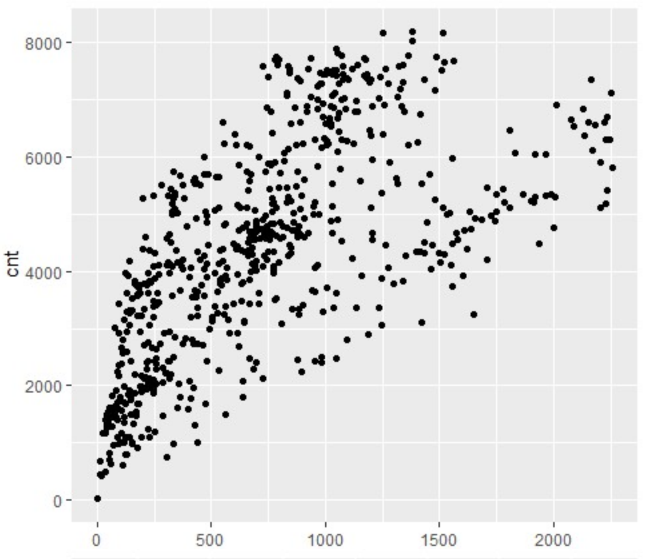


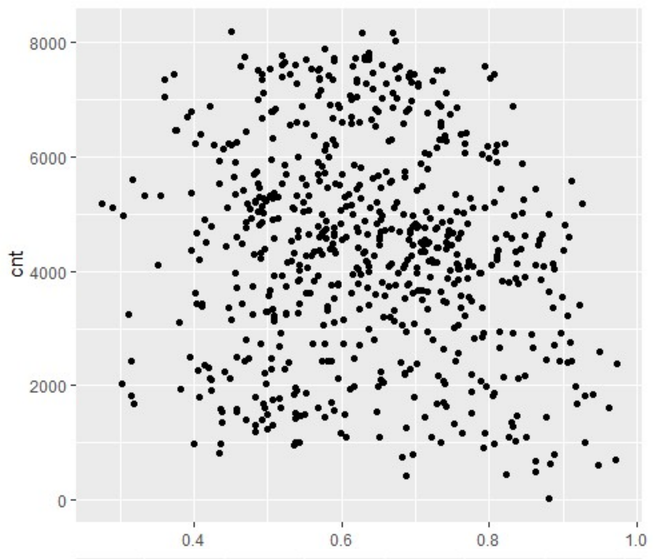
temp
atemp
hum
windspeed
casual
registered
cnt
Corr: 0.992
Corr: 0.127
Corr: -0.158
Corr: 0.543
Corr: 0.54
Corr: 0.627
Corr: 0.14
Corr: -0.184
Corr: 0.544
Corr: 0.544
Corr: 0.631
Corr: -0.248
Corr: -0.077
Corr: -0.0911
Corr: -0.101
Corr: -0.168
Corr: -0.217
Corr: -0.235
Corr: 0.395
Corr: 0.673
Corr: 0.946

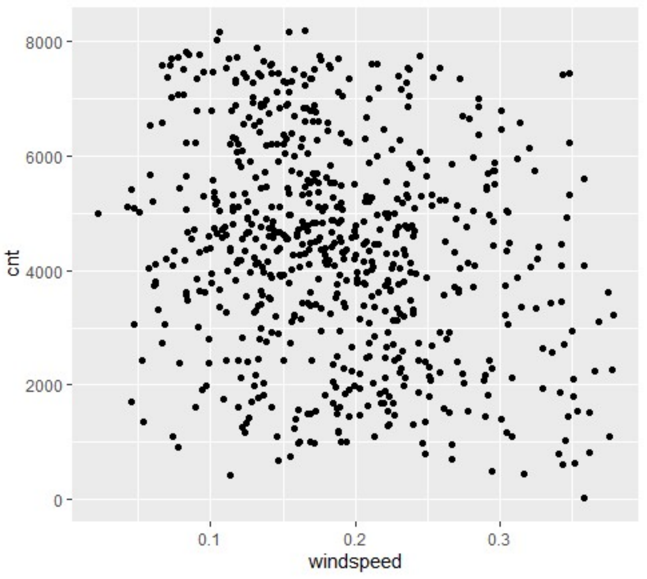

cnt
weekday
0 1 2 3 4 5 6
0e+00 1e+05 2e+05 3e+05 4e+05 5e+05
season
1 2 3 4
0e+00 3e+05 6e+05 9e+05
yr
0 1
0 500000 1000000 1500000 2000000
mnth
1 2 3 4 5 6 7 8 9 10 11 12
0e+00 1e+05 2e+05 3e+05
holiday
0 1
0e+00 1e+06 2e+06 3e+06
workingday
0 1
0 500000 1000000 1500000 2000000
weathersit
1 2 3
0 500000 1000000 1500000 2000000

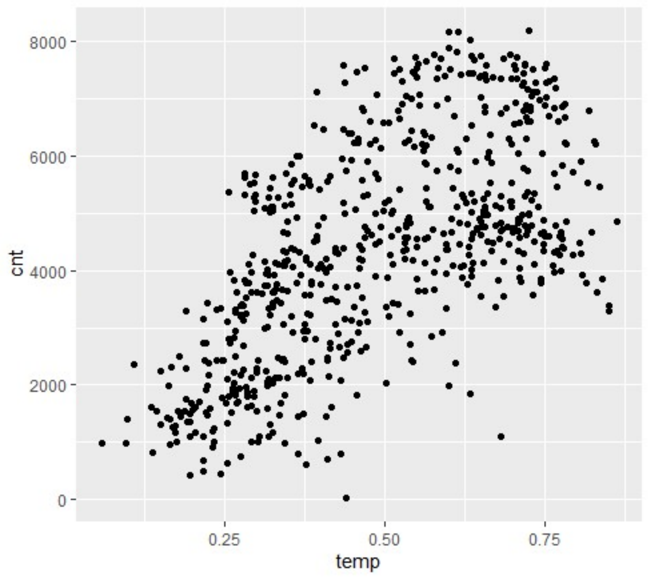

atemp
casual
cnt
hum
registered
temp
windspeed
value
x

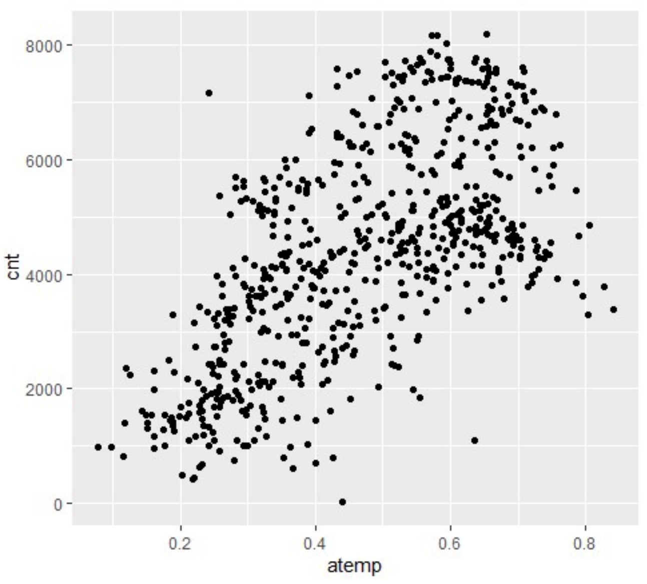

temp
atemp
hum
windspeed
casual
registered
cnt
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1

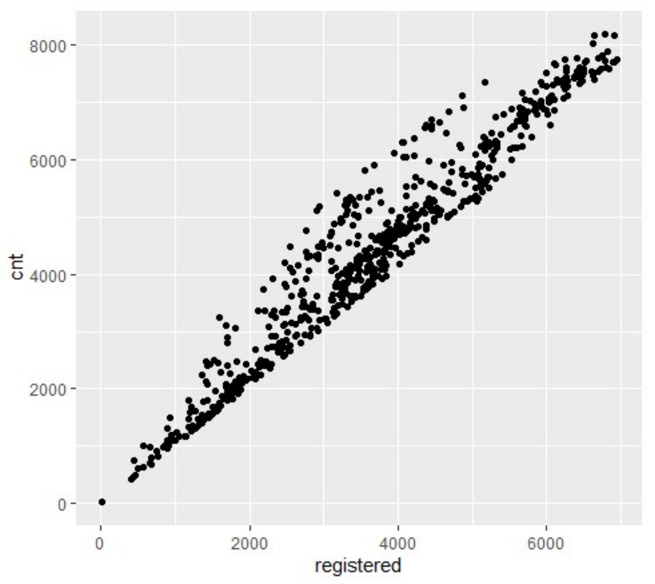

cnt
0
2000
4000
6000
8000
0
500
1000
1500
2000


cnt
0
2000
4000
6000
8000
0.4
0.6
0.8
1.0


cnt
0
2000
4000
6000
8000
0.1
0.2
0.3
windspeed


cnt
0
2000
4000
6000
8000
0.25
0.50
0.75
temp


cnt
0
2000
4000
6000
8000
0.2
0.4
0.6
0.8
atemp


cnt
0
2000
4000
6000
8000
0
2000
4000
6000
registered