Project

Bike Rental

Aditya Tiwari
Adi138250@gmail.com

Contents

1. Introduction

1.1 Problem Statement

The Bike Rental Data contains the daily count of bike rental between the year 2011 and 2012 with corresponding environmental and seasonal information. We would like to predict the daily count of rental count on daily based on the environmental and seasonal settings to automate the system.

1.2 Data

The variables are:

- instant: Record index
- dteday: Date
- season: Season (1:springer, 2:summer, 3:fall, 4:winter)
- yr: Year (0: 2011, 1:2012)
- mnth: Month (1 to 12)
- hr: Hour (0 to 23)
- holiday: weather day is holiday or not (extracted fromHoliday Schedule)
- weekday: Day of the week
- workingday: If day is neither weekend nor holiday is 1, otherwise is 0.
- weathersit: (extracted fromFreemeteo)

  1: Clear, Few clouds, Partly cloudy, Partly cloudy

  2: Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist

  3: Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds

  4: Heavy Rain + Ice Pallets + Thunderstorm + Mist, Snow + Fog

  temp: Normalized temperature in Celsius. The values are derived via (t-t_min)/(t_max-t_min),

  t_min=-8, t_max=+39 (only in hourly scale)

  atemp: Normalized feeling temperature in Celsius. The values are derived via (t-t_min)/(t_maxt_

  min), t_min=-16, t_max=+50 (only in hourly scale)
- hum: Normalized humidity. The values are divided to 100 (max)
- windspeed: Normalized wind speed. The values are divided to 67 (max)
- casual: count of casual users

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

  registered: count of registered users

  cnt: count of total rental bikes including both casual and registered

Table : Bike Rental Sample Data (Columns: 1-8)

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

Table : Bike Rental Sample Data (Columns: 9-14)

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

Below are the variables present in bike rentals dataset

Table : Bike Rental

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

2. Methodology

2.1 Pre-Processing

Any predictive modeling requires that we look at the data before we start modeling. We decided to simply remove few variables after loading data set. These variables are instant, dteday, casual and registered. The reason to remove instant has no information as it is record index. dteday has no meaning to us here as we are focusing on seasonal setting not dates of month or of year. yr variable also has no importance here. As we are interested in finding total count i.e. variable cnt which is our target variable and it is sum of casual and registered so we will remove casual and registered variable also.

However, in data mining terms looking at data refers to so much more than just looking. Looking at data refers to exploring the data, cleaning the data as well as visualizing the data through graphs and plots. This is often called as **Exploratory Data Analysis**. To start this process, we will first check the presence of missing values in data set.

2.1.1 Missing Value Analysis

Missing values Analysis is required to be done so that we can check if there is any missing data. In case data is missing at few places we will impute those missing values by different methods in order to generate appropiate results. In our case we have no missing values hence nothing further have to be done. Below table illustrate 0 missing value in all variables in the data.

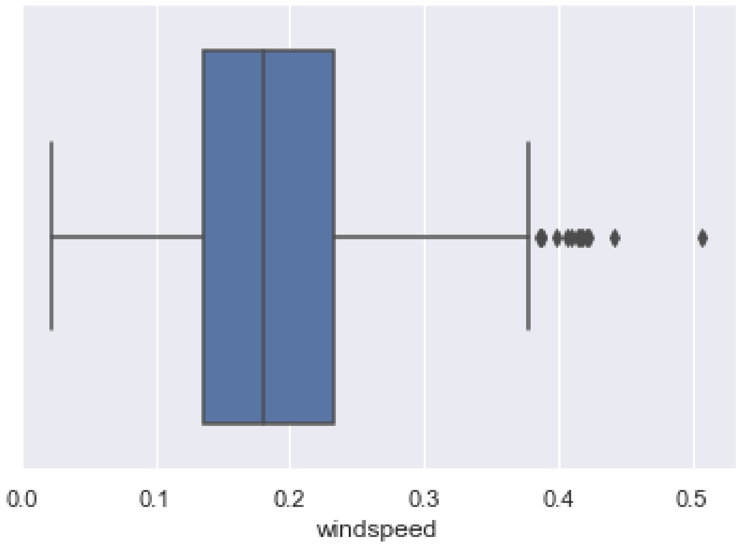
2.1 missing values

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

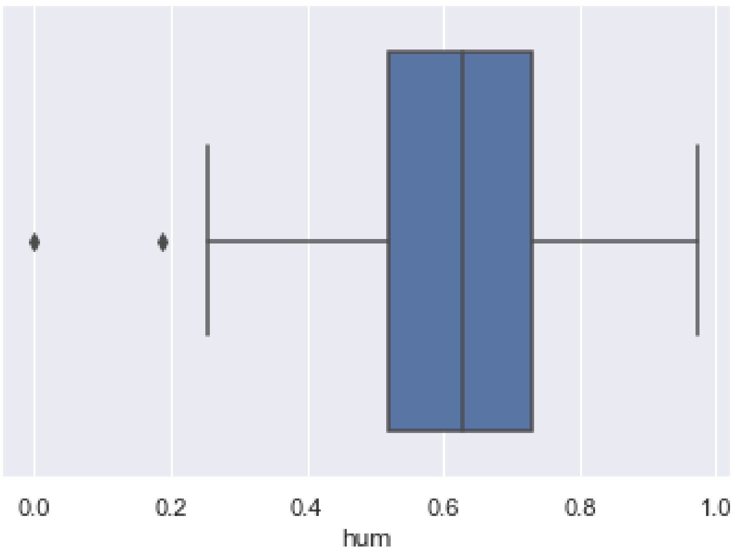
2.1.2 Outlier Analysis

The Other steps of Preprocessing Technique is Outliers analysis, an outlier is an observation point that is distant from other observations. Outliers in data can be good and it can be bad as well. Here in our case we don't want outliers the reason for removing these outliers instead of substituting them with other balancing values (such as mean, median or knn method) because we expect them to be relatively random values and replacing them with set values may cause inaccuracy in analysis later.
The outlier analysis is done by plotting the box plot. Boxplot is a method for graphically depicting groups of numerical data through their quartiles. Box plots may also have lines extending vertically from the boxes (whiskers) indicating variability outside the upper and lower quartiles.

Fig. 1.0




```
sns.boxplot(br_day_subset['windspeed'])
```

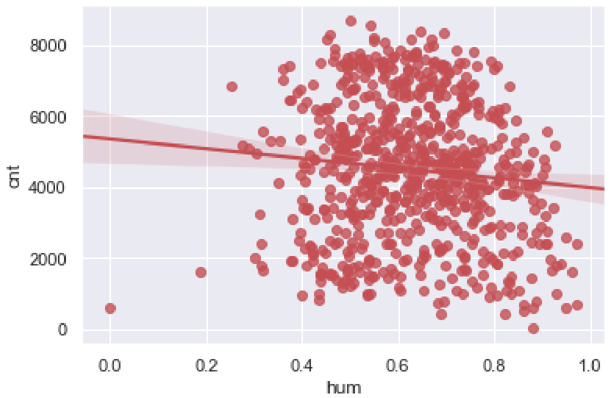



```
sns.boxplot(br_day_subset['hum'])
```

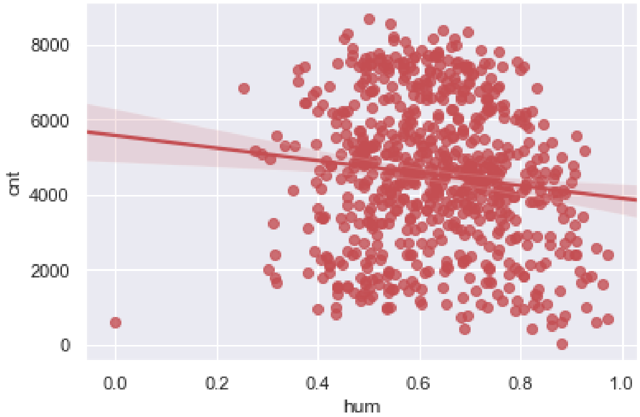
From above boxplots we can see outliers in windspeed and hum.

Fig. 2.0

Before outlier treatment

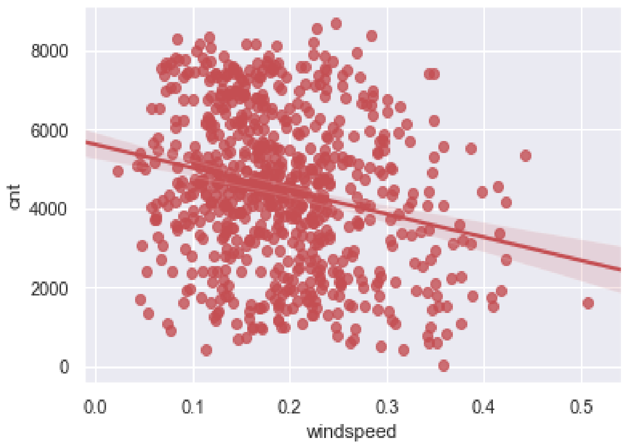


After Outlier treatment

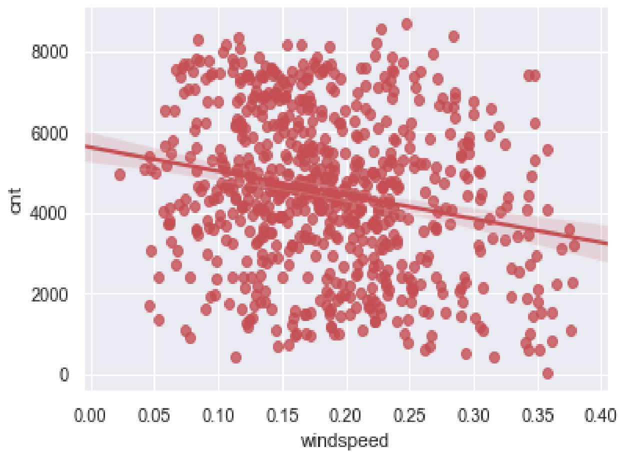



```
sns.regplot(x = 'hum', y = 'cnt', data = br_day_subset, color = 'r')
```

Before outlier treatment



After Outlier treatment




```
sns.regplot(x = 'windspeed' , y = 'cnt', data = br_day_subset, color = 'r')
```

The variable ‘hum’ must be dropped out due to collinearity in later processing so we are not going to delete the outliers. These outliers will not make the model bias.

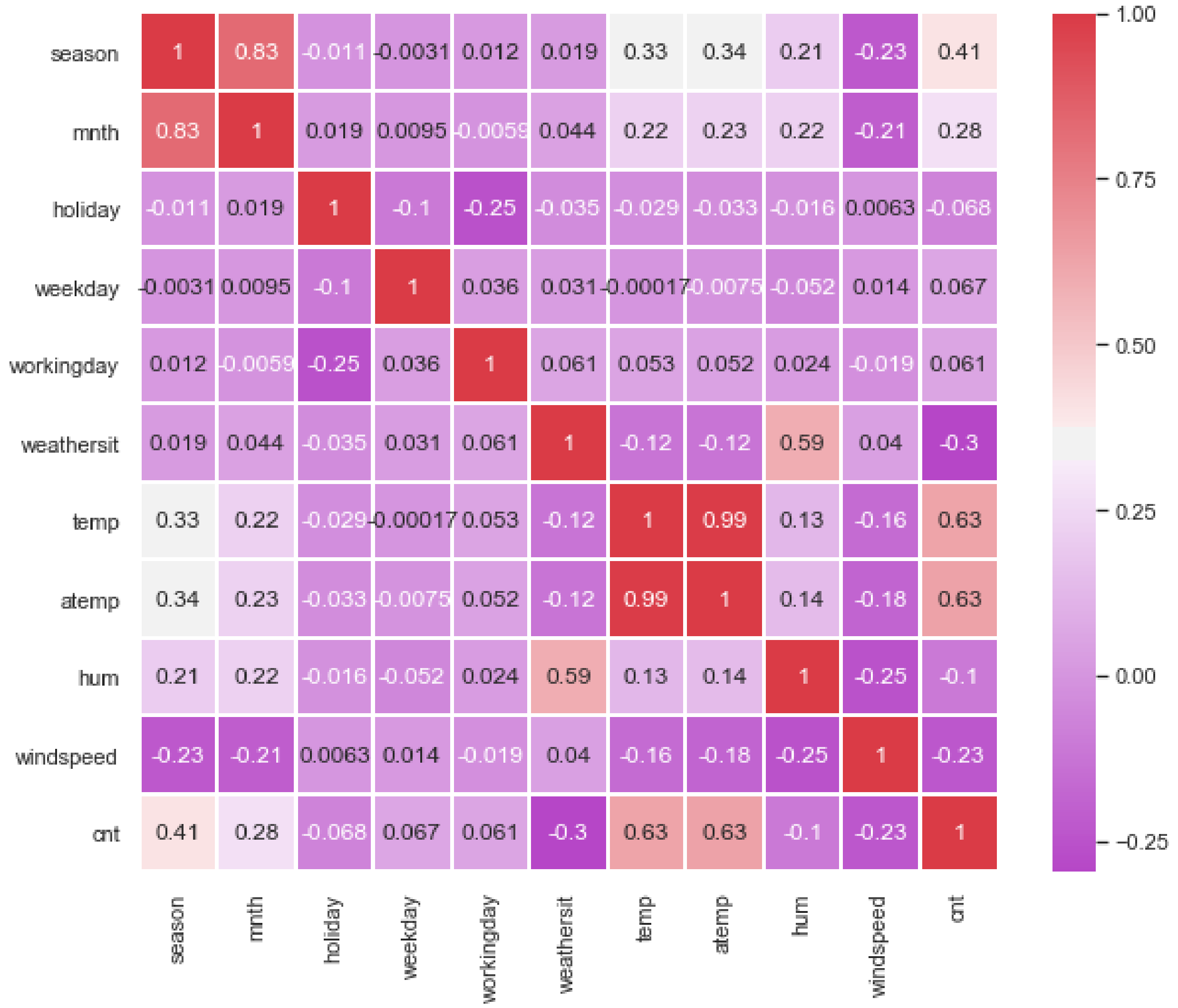
2.1.3 Feature Selection

Machine learning works on a simple rule of GIGO i.e. Garbage In Garbage Out. Here garbage refers to the noise or redundant values.

This becomes even 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 important. Feature subsets gives better results than complete set of features for the same algorithm or "Sometimes, less is better!".

We should consider the selection of feature for model keeping in mind that there should be low correlation between two independent variables otherwise there will be problem of multicollinearity.

Fig. 3.0



From above correlation plot we can clearly see that variable mnth and season have high correlation variable temp and atemp also have high correlation and variable weathersit and hum also have high correlation. It means that we must drop one variable out of two having high correlation. So in our study here we will drop variables temp , mnth and hum.

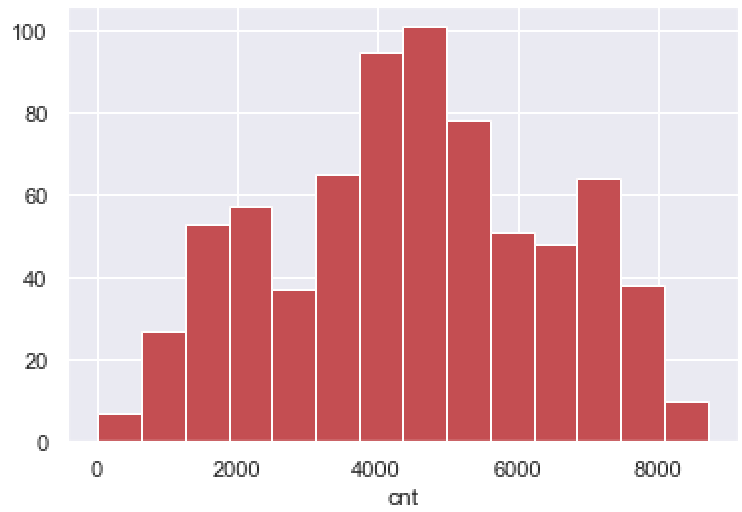
Color dark Red indicates there is strong positive correlation and if dark violet indicates negative correlation.

3. Data Distribution

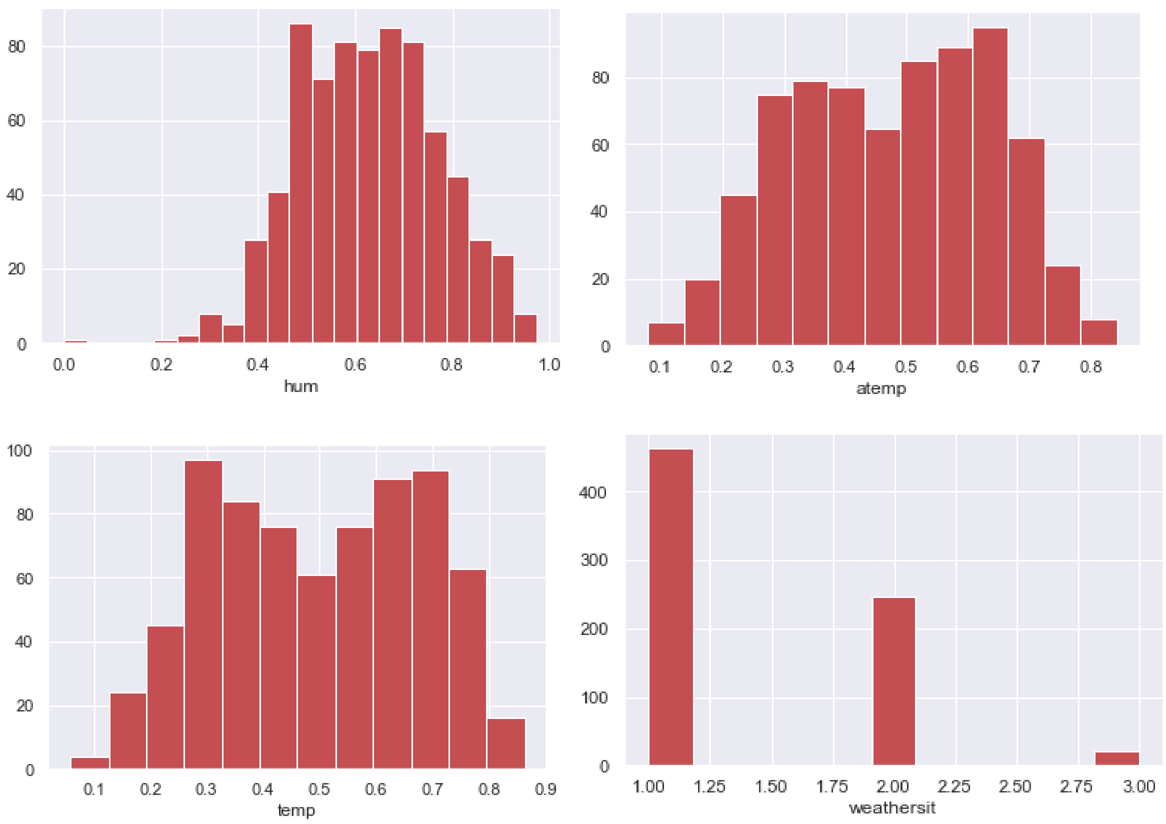
Checking distribution of variables with help of histograms

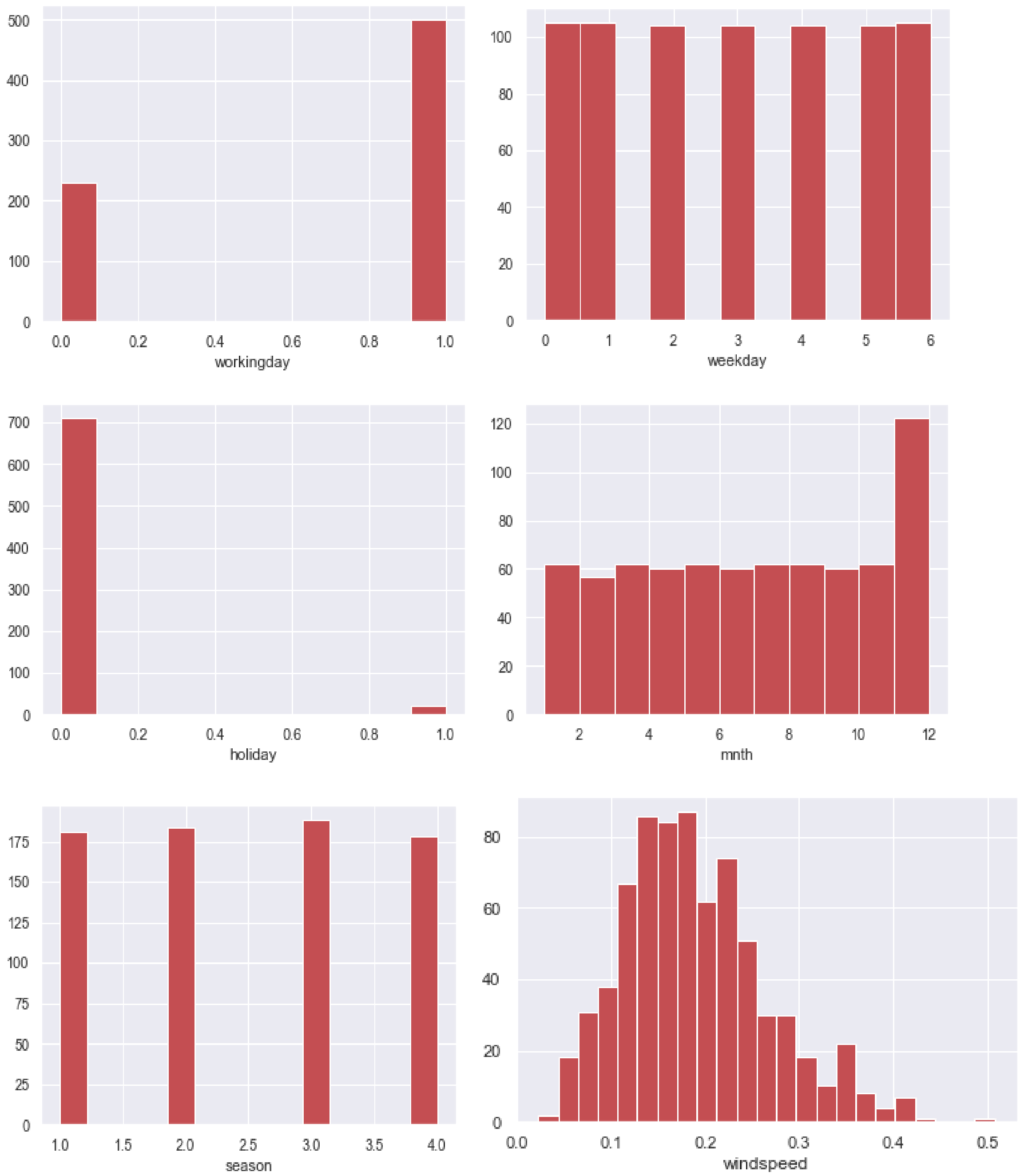
Distribution of target variable (cnt)

Fig. 4.0



From above grapgh we can see that our target variable cnt is normally distributed.



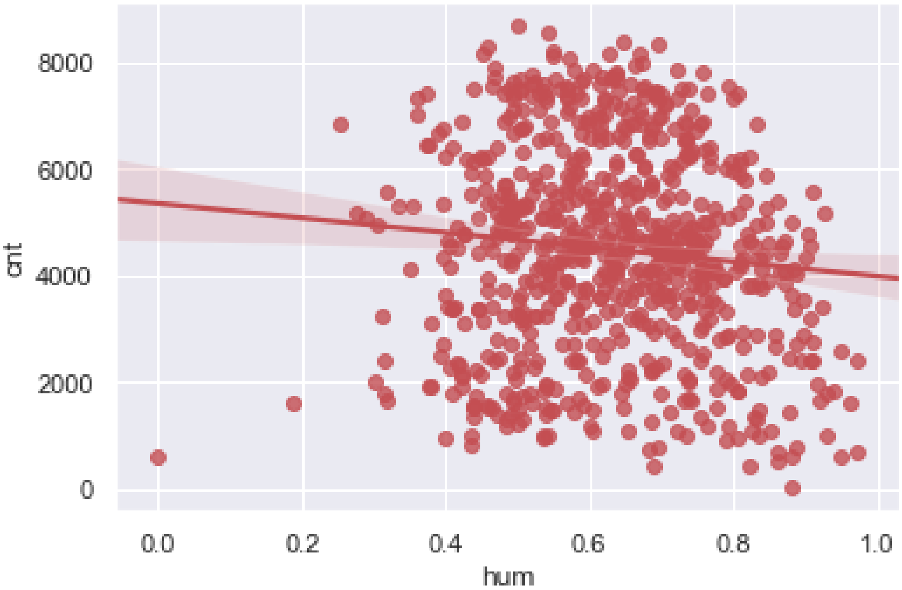

0 100 200 300 400 500
0.0 0.2 0.4 0.6 0.8 1.0
workingday
0 20 40 60 80 100
0 1 2 3 4 5 6
weekday
0 100 200 300 400 500 600 700
0.0 0.2 0.4 0.6 0.8 1.0
holiday
0 20 40 60 80 100 120
2 4 6 8 10 12
mnth
0 25 50 75 100 125 150 175
1.0 1.5 2.0 2.5 3.0 3.5 4.0
season
0 20 40 60 80
0.0 0.1 0.2 0.3 0.4 0.5
windspeed

4. Regression Scattered Plots

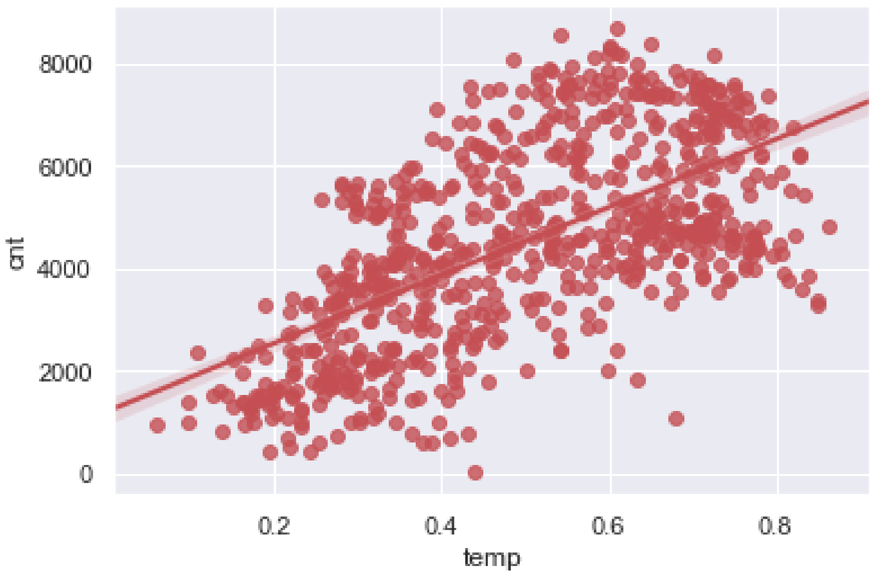
We have used seaborn library for Plotting regression scattered plot to see positive or negative relation of variables with target variable. . Regression scattered plots can be useful for quickly exploring the relationships between independent variable and dependentvariable in a data frame.

Below figures shows positive or negative relationship between independent variables and target variable.

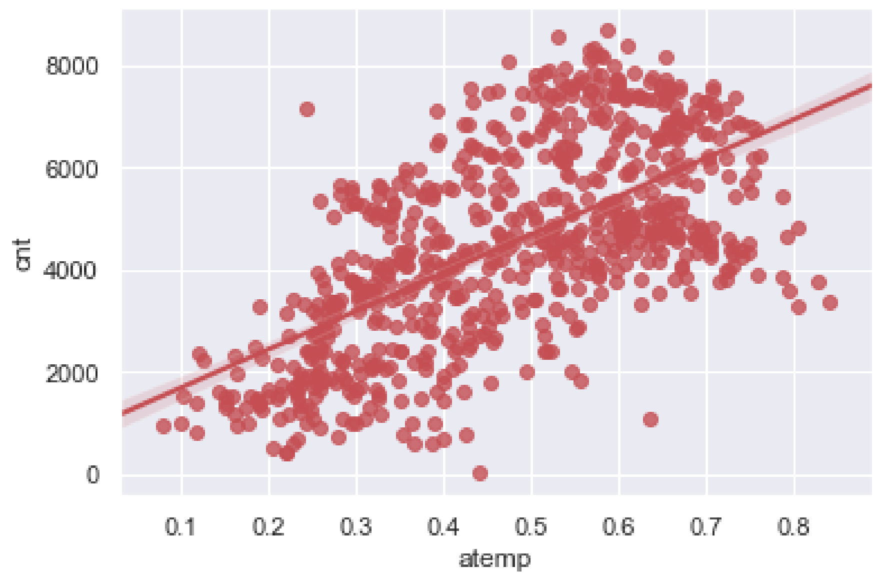
Fig. 5.0



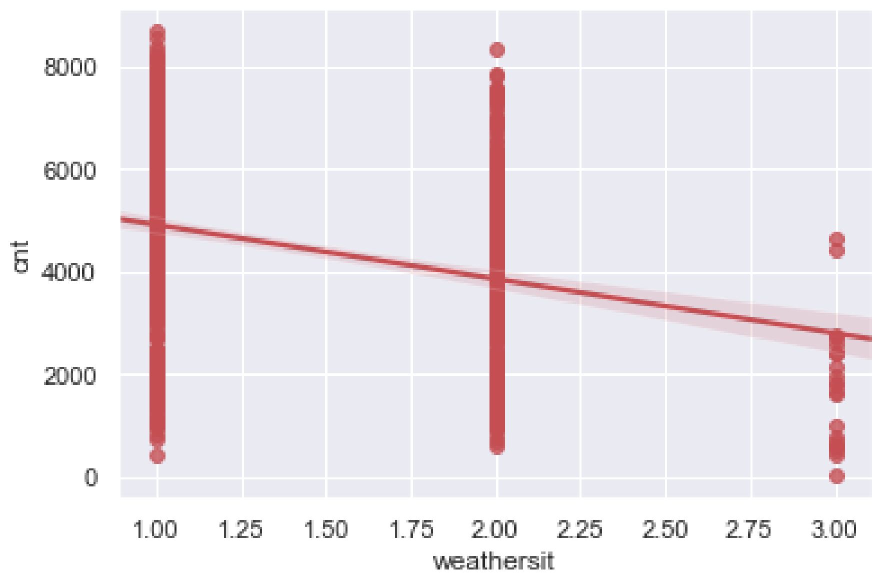
From above regression scattered plot we can see a negative relationship between hum and cnt. Which means increase in humidity leads to less bike cnt.



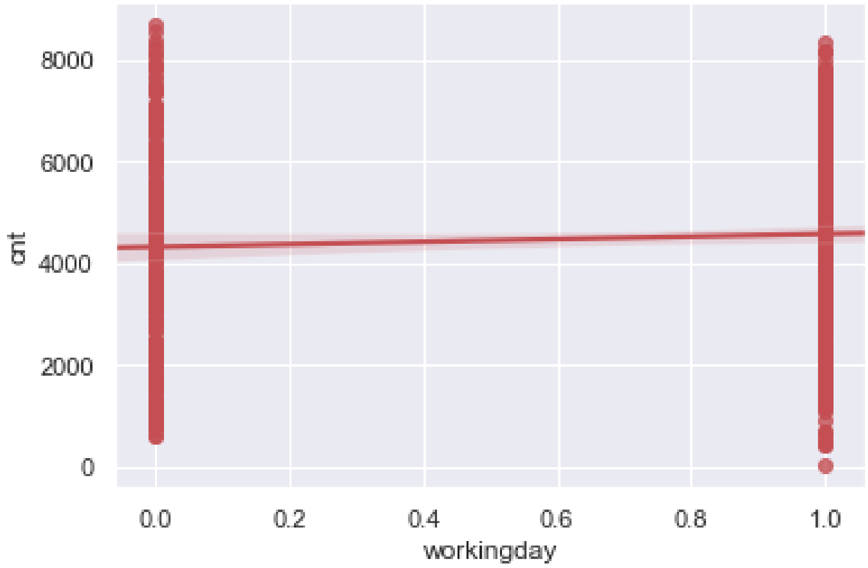
From above regression scattered plot we can see a positive relationship between temp and cnt. Which means increase in temperature leads to increase in bike cnt as we can see temp between 0.4 to 0.8 there is increase in bike count.



From above regression scattered plot we can see a positive relationship between atemp and cnt. Which means increase in actual temperature leads to increase in bike cnt as we can see atemp between 0.4 to 0.7 there is increase in bike count.

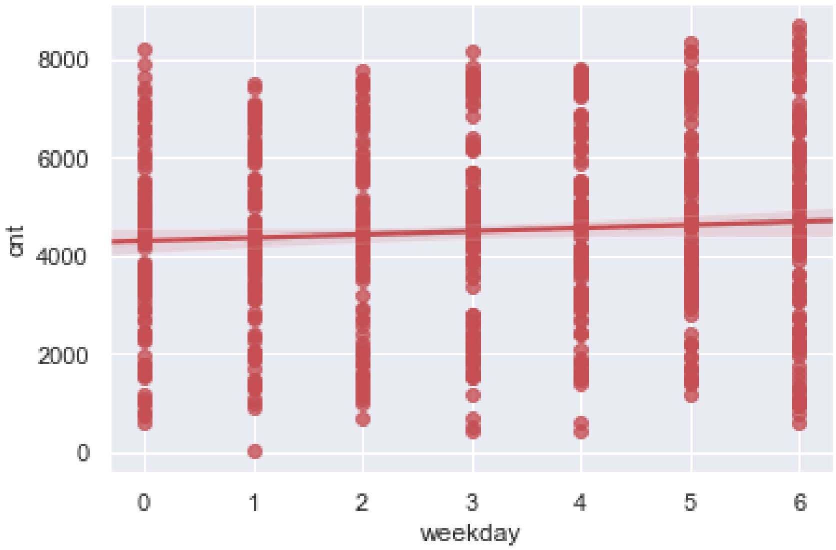


From above regression scattered plot we can see a negatice relationship between weathersit and cnt. Which means when weather situation is 3: Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds there is decrease in bike cnt where as when the weather situation in 1: Clear, Few clouds, Partly cloudy, Partly cloudy there is increased demand of bike. Which means the weather situation 3 has negative impact on bike count demand.

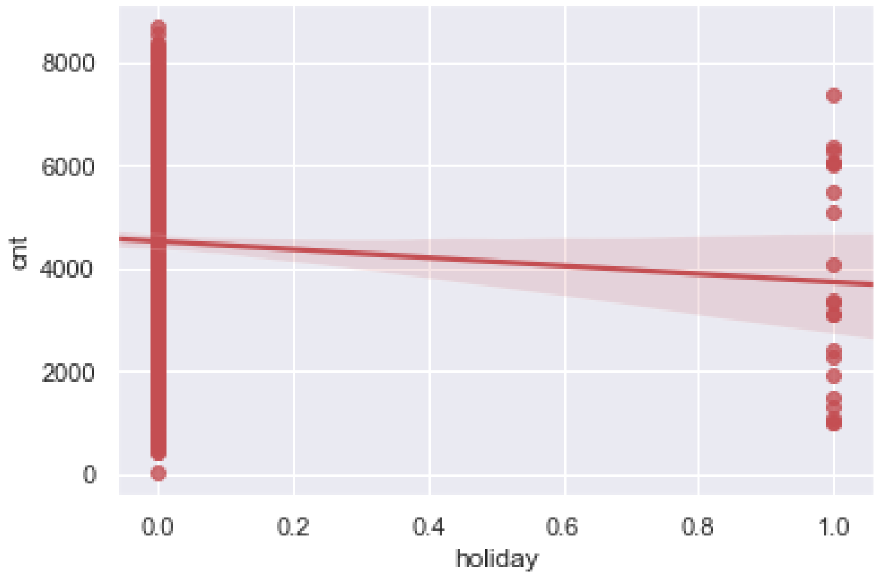


From above regression scattered plot we can see that bike count is almost same on 0 and 1 though we can see a slight positive relation when workingday is 1. Whereas 0 and 1 are as follows

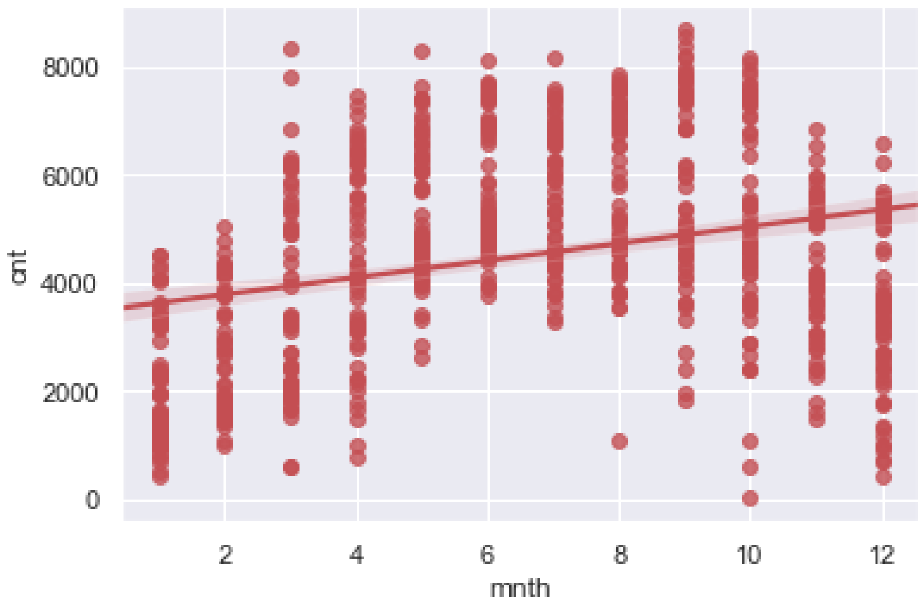
If day is neither weekend nor holiday is 1, otherwise is 0.



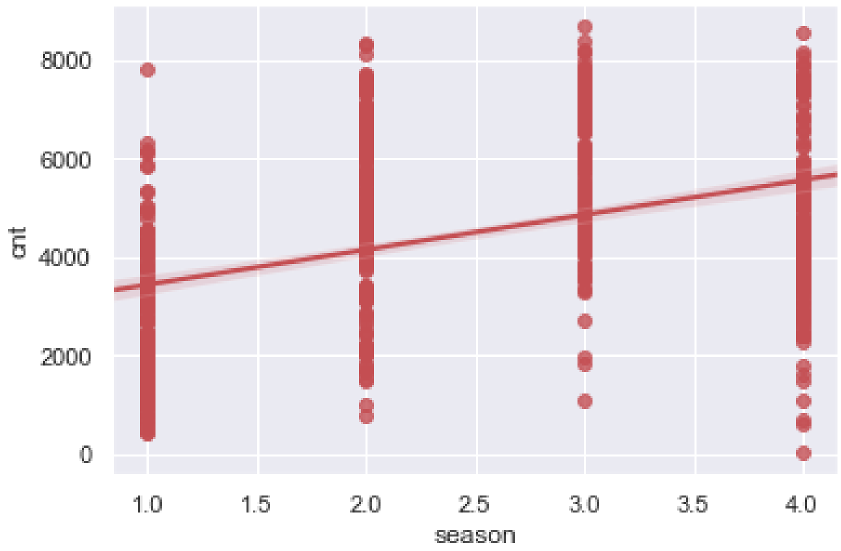
From above regression scattered plot we can see a slight increase in bike demand on working days. 0 represent Sunday 1- Monday 2- Tuesday 3- Wednesday 4- Thursday 5- Friday 6 – Saturday



From above regression scattered plot we can see that there is negative demand of bike when it's a holiday. 0- working day 1- holiday.

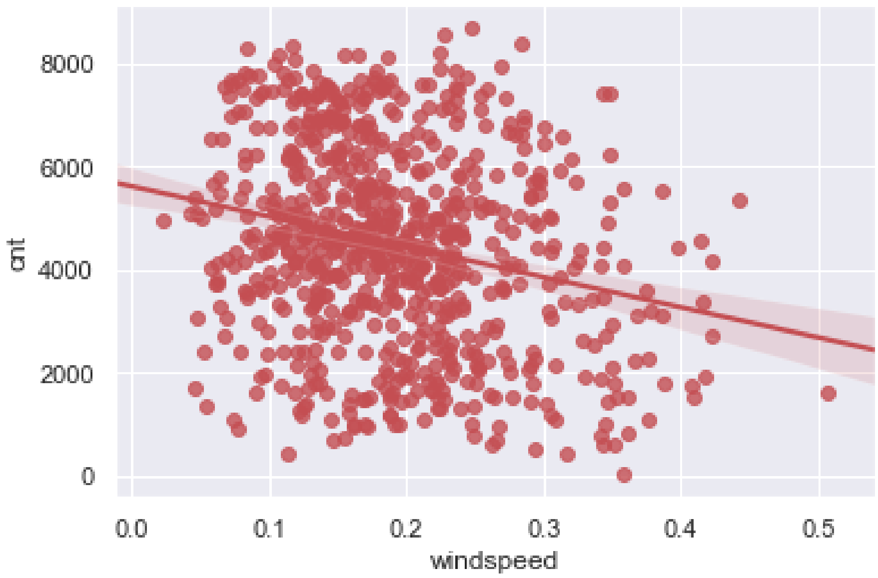


From above regression plot we see that during 9th and 10th month there is increase in bike demand hence the plot is showing a positive relationship. 0-12 are the names of month.



From above regression scattered plot we can see a positive relationship between season and cnt. Which means when season is 4 increase in bike cnt whereas when the season is 1 the bike demand is less It clearly shows that season 2,3 and 4 have positive impact on bike count demand. Season 3 having highest bike count.

Where Season (1:springer, 2:summer, 3:fall, 4:winter)



From above regression plot we can clearly see that there is a negative impact in bike rental count when windspeed is more.

5. Modelling

5.1 Model Selection

Model Selection is a process of selecting the model which have better accuracy and can work on train and test data. We must select a model where algorithm works well and shows low error rate.

5.1.1 Evaluating Regression Model

We are using MAPE methods to evaluate performance of model

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

$$\left(\frac{1}{n}\sum\frac{|Actual - Forecast|}{|Actual|}\right) * 100$$

5.2 Decision Tree

A tree has many analogies in real life and turns out that it has influenced a wide area of **machine learning**, covering both **classification and regression**. In decision 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.

Decision Tree Algorithm

```
In [32]: #############Decision Tree Regressor#############
         #select subset using stratified Sampling
         y = br_sliced['season']
         train, test = train_test_split(br_sliced, test_size = 0.2, stratify = y)

In [33]: train.shape
Out[33]: (584, 8)

In [34]: test.shape
Out[34]: (147, 8)

In [45]: br_fit_DT = DecisionTreeRegressor(max_depth =5).fit(train.iloc[:,0:7], train.iloc[:,7])

In [46]: #Apply above model on test data
         prediction_DT = br_fit_DT.predict(test.iloc[:,0:7])

In [47]: MAPE(test.iloc[:,7],prediction_DT)
Out[47]: 27.901736841891445

In [115]: #the error is 27.90% which means our model is 72.1% accurate when max depth is 5
```

Looking at above figure we have MAPE of 27.9% which means our model is 72.1% accurate.

5.3 Random Forest

Random forests is an ensemble learning method for classification and regression that operates by constructing a multitude of decision trees at training time and outputting 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' habit of overfitting to their training set.

Random forest functions in following way

- Draws a bootstrap sample from training data.
- 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.

Random Forest Implementation

```
In [57]: #Dividing data into train and test
         X = br_sliced.values[:, 0:7]
         Y = br_sliced.values[:,7]
         a = br_sliced['season']
         X_train, X_test, y_train, y_test = train_test_split( X, Y, test_size = 0.2, stratify = a)
```

```
In [62]: #Random Forest
         RF_model = RandomForestRegressor(n_estimators = 500, oob_score = True, n_jobs = -1,random_state =50,max_features =
                                          min_samples_leaf = 50).fit(X_train, y_train)
```

```
In [63]: RF_model

Out[63]: RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=None,
                    max_features='auto', max_leaf_nodes=None,
                    min_impurity_decrease=0.0, min_impurity_split=None,
                    min_samples_leaf=50, min_samples_split=2,
                    min_weight_fraction_leaf=0.0, n_estimators=500, n_jobs=-1,
                    oob_score=True, random_state=50, verbose=0, warm_start=False)
```

```
In [64]: RF_prediction = RF_model.predict(X_test)
```

```
In [65]: MAPE(y_test, RF_prediction)

Out[65]: 34.795475883613996
```

```
In [103]: # 34.79% is error rate which means 65.21% model is accurate
```

3.3.1 Evaluation of Random Forest

The above figure of Random Forest model has MAPE of 34.79% which means the model is 65.21% accurate which is less compared to decision tree. We will make use of Linear Regression to predict the 'cnt ' values and compare with decision tree.

5.4 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.

VIF (Variance Inflation factor) : It quantifies the multicollinearity between the independent variables.

As Linear regression will work well if multicollinearity between the Independent variables are less.

Multi collinearity between Independent variables

```
No variable from the 7 input variables has collinearity problem.

The linear correlation coefficients ranges between:
min correlation ( weekday ~ season ):  -0.003079881
max correlation ( atemp ~ season ):  0.3428756

---------- VIFs of the remained variables --------
   Variables      VIF
1     season 1.176995
2    holiday 1.079260
3    weekday 1.011595
4 workingday 1.074895
5 weathersit 1.026341
```

In the above figure it is showing there is that the data to input for train and test does not have collinearity problem.

Multiple Linear Regression Model

In [135]:
```
###Linear regression
linear_regression_model = sm.OLS(train.iloc[:,7], train.iloc[:,0:7]).fit()
```

In [136]:
```
#Summary of model
linear_regression_model.summary()
```

Out[136]:

OLS Regression Results

Dep. Variable:	cnt	R-squared:	0.918
Model:	OLS	Adj. R-squared:	0.917
Method:	Least Squares	F-statistic:	927.0
Date:	Sun, 30 Dec 2018	Prob (F-statistic):	4.53e-309
Time:	04:01:11	Log-Likelihood:	-5057.0
No. Observations:	584	AIC:	1.013e+04
Df Residuals:	577	BIC:	1.016e+04
Df Model:	7		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
season	458.2048	53.556	8.556	0.000	353.015	563.394
holiday	-201.5349	390.277	-0.516	0.606	-968.071	565.001
weekday	119.6636	28.628	4.180	0.000	63.436	175.891
workingday	251.1774	125.924	1.995	0.047	3.851	498.504
weathersit	-586.4024	93.963	-6.241	0.000	-770.954	-401.851
atemp	7118.5155	328.646	21.660	0.000	6473.027	7764.004
windspeed	1031.0747	643.352	1.603	0.110	-232.523	2294.673

Omnibus:	4.376	Durbin-Watson:	2.065
Prob(Omnibus):	0.112	Jarque-Bera (JB):	3.370
Skew:	-0.054	Prob(JB):	0.185
Kurtosis:	2.644	Cond. No.	51.4

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

In [137]:
```
#predict the  model

predict_LR = linear_regression_model.predict(test.iloc[:,0:7])
```

In [137]: #predict the model predict_LR = linear_regression_model.predict(test.iloc[:,0:7])

In [138]:
```
MAPE(test.iloc[:,7],predict_LR)
```

Out[138]: 32.33591489337019

In [94]:
```
#Error rate 32.33% hence accuracy of 67.67%
```

R-squared is a statistical measure of how close the data are to the fitted regression line. It is also known as the coefficient of determination, or the coefficient of multiple determination for multiple regression. 0% indicates that the model explains none of the variability of the response data around its mean.

R-squared: 0 .918, Adjusted R-squared: 0 .917

As R-Square value is 0.918 so, it means the model is well fitted on regression line.

But on other side the MAPE is 32.33% and accuracy is 67.67% which is less compared to decision tree model.

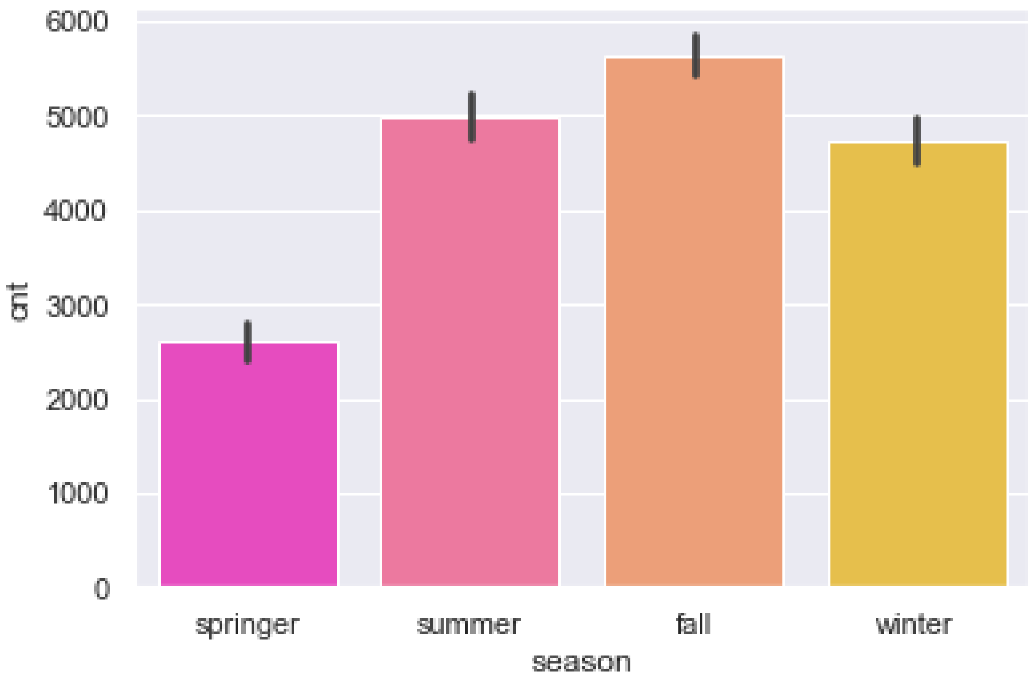
6. Model Selection

As we predicted counts for Bike Rental using three Models i.e. Decision Tree, Random Forest and Linear Regression as MAPE is low in decision tree and accuracy is also better than compared to random forest and linear regression, so we will go with decision tree model.

6.1 Conclusion: - For the Bike Rental Data Decision Tree Model is best model to predict the count.

Appendix - A

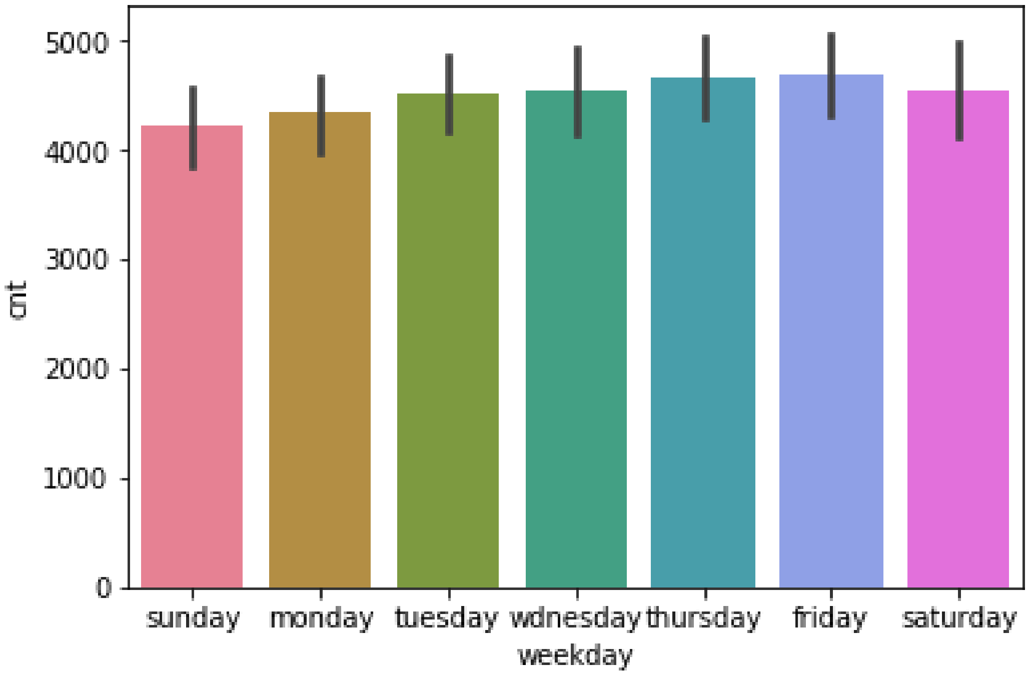
Bar plot between season and cnt




```
sns.barplot(x='season', y='cnt', palette = 'spring',  data = br_day_subset)
```

From above bar plot we can see that the bike count is high during fall season. And least is during springer season.

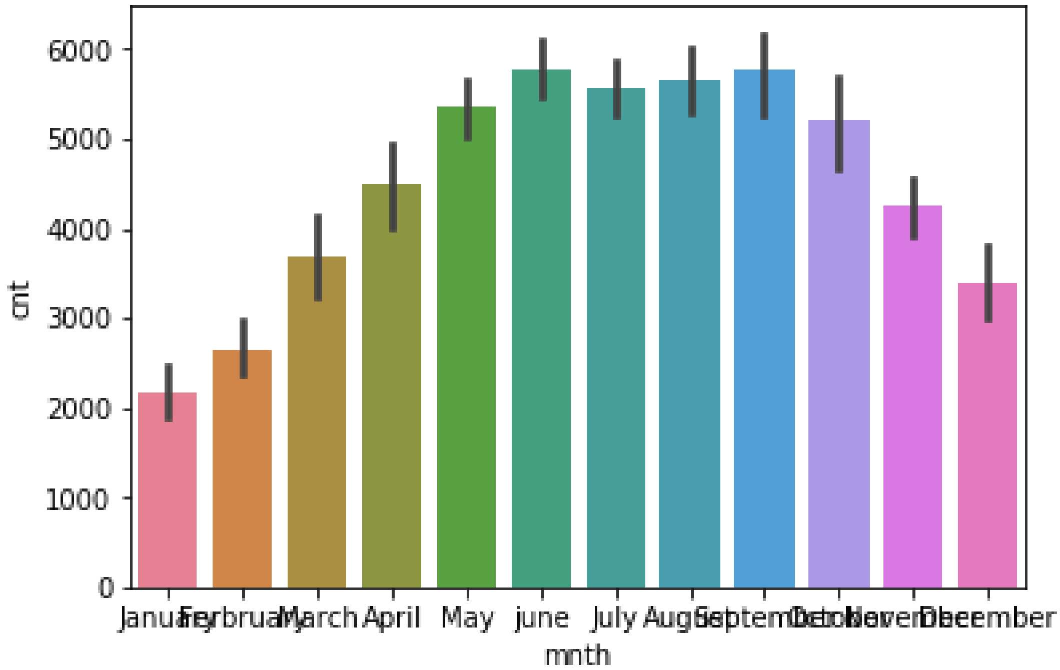
Bar plot between weekdays and cnt




```
sns.barplot(x='weekday', y='cnt', palette = 'husl', data = br_day_subset,order=['sunday','monday','tuesday','wdnesday',
                                                                                 'thursday', 'friday','saturday'])
```

As we can see from above bar plot bike count is more on weekdays compared to weekends

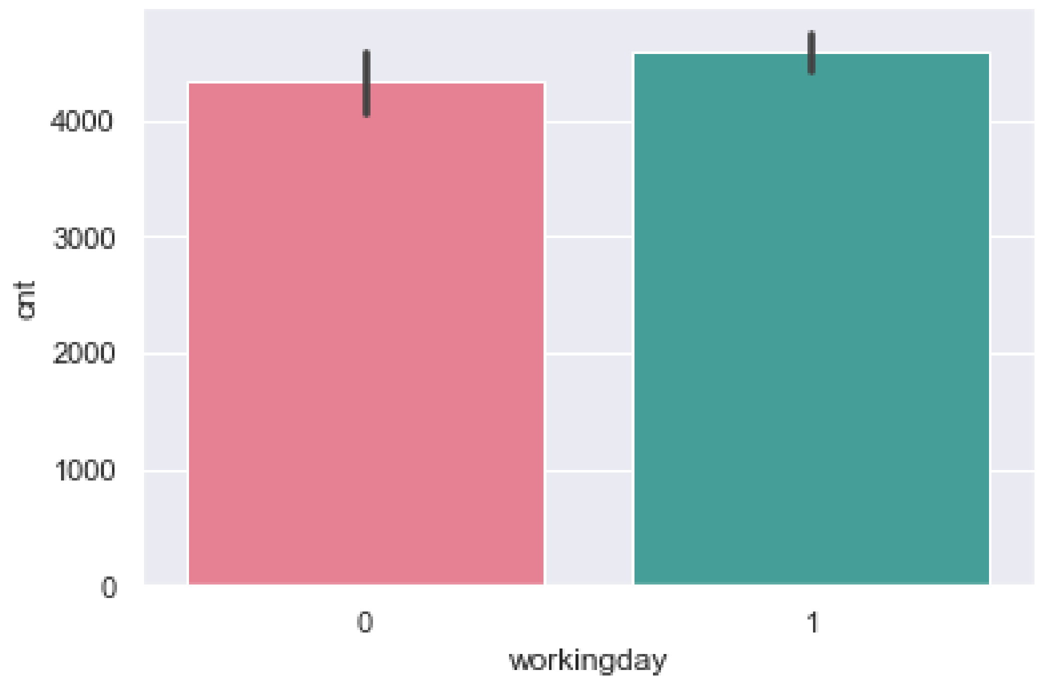
Bar plot between months and cnt




```
sns.barplot(x='mnth', y='cnt', palette = 'husl', data = br_day_subset)
```

From above bar plot bike count is high from month of June to September. It must be season of fall as in season bar plot we saw that during fall season the bike count is high.

Bar plot between workingday and cnt




```
sns.barplot(x='workingday', y='cnt', palette = 'husl', data = br_day_subset)
```

We can see from above bar plot that bike count is more on 1 than compared to 0.

If day is neither weekend nor holiday is 1, otherwise is 0.

Appendix- B - Python Code

Fig 3.0 Python Code

Feature selection

```
]: ####Feature selection
   cnames = ['season','mnth','holiday','weekday','workingday','weathersit','temp','atemp','hum','windspeed','cnt']
```

```
]: br_corr = br_day_subset.loc[:,cnames]
```

```
]: f, ax = plt.subplots(figsize=(10,8))

   corr_matrix = br_corr.corr()

   sns.heatmap(corr_matrix, mask = np.zeros_like(corr_matrix, dtype=np.bool), cmap = sns.diverging_palette(300,10, as_cmap= True),
               annot = True , linewidths = 0.9,square = True, ax=ax)
```

All under Fig 4.0 Python Code

```
#Checking distribution of season variable
sns.set()
plt.hist(br_day_subset['season'], bins = 'auto', color = 'r')
plt.xlabel('season')

plt.hist(br_day['mnth'], bins = 'auto', color = 'r')
plt.xlabel('mnth')

plt.hist(br_day['holiday'], bins = 'auto', color = 'r')
plt.xlabel('holiday')


plt.hist(br_day['weekday'], bins = 'auto', color = 'r')
plt.xlabel('weekday')

plt.hist(br_day['workingday'], bins = 'auto', color = 'r')
plt.xlabel('workingday')

plt.hist(br_day['weathersit'], bins = 'auto', color = 'r')
plt.xlabel('weathersit')

plt.hist(br_day['temp'], bins = 'auto', color = 'r')
plt.xlabel('temp')

plt.hist(br_day['atemp'], bins = 'auto', color = 'r')
plt.xlabel('atemp')


plt.hist(br_day['hum'], bins = 'auto', color = 'r')
plt.xlabel('hum')
```

```
plt.xlabel('hum')

plt.hist(br_day['windspeed'], bins = 'auto', color = 'r')
plt.xlabel('windspeed')

plt.hist(br_day['cnt'], bins = 'auto', color = 'r')
plt.xlabel('cnt')
```

All under Fig 5.0 Python Code

```
##Plotting regression scattered plot to see positive or negative relation of variables with target variable

sns.regplot(x = 'season', y = 'cnt', data = br_day_subset, color = 'r')

sns.regplot(x = 'mnth' , y = 'cnt', data = br_day_subset, color = 'r')

sns.regplot(x = 'holiday' , y = 'cnt', data = br_day_subset, color = 'r')

sns.regplot(x= 'workingday' , y = 'cnt', data = br_day_subset, color = 'r')

sns.regplot(x = 'weathersit' , y = 'cnt', data = br_day_subset, color = 'r')

sns.regplot(x = 'atemp' , y = 'cnt', data = br_day_subset, color = 'r')

sns.regplot(x= 'temp', y = 'cnt',  data= br_day_subset,  color = 'r')

sns.regplot(x= 'windspeed', y = 'cnt',  data= br_day_subset,  color = 'r')
```

Complete Python File

Loading libraries and data file

```
import os
import pandas as pd
import numpy as np
from ggplot import *
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.ensemble import RandomForestRegressor
import statsmodels.api as sm
from sklearn.cross_validation import train_test_split
from sklearn.tree import DecisionTreeRegressor
%matplotlib inline
```

```
#Loading the file
br_day = pd.read_csv('E:/Project/Bike rental/bike_rental_day.csv')
```

```
br_day.head(5)
```

```
#checking the data set
br_day.shape
```

```
br_day.dtypes
```

Checking missing Values in dataset

```
#Checking if any NA are there or not
br_day.isnull().sum()
```

```
#We can clearly see there are no missing values.
```

```
#Here we are removing few variables. instant has no information as it is record index. dteday has no meaning to us here
#as we are focusing on seasonal setting not dates of month or of year. yr variable also has no importance here. As we are
#interested in finding total count i.e. variable  cnt which is our target variable and it is sum of casual and registered so
#we will remove casual and registerd variable also.

br_day_subset = pd.DataFrame(br_day[['season','mnth', 'holiday', 'weekday', 'workingday','weathersit', 'temp', 'atemp',
                                     'hum','windspeed','cnt']])
```

```
br_day_subset.head(5)
```

Outlier Analysis

```
sns.boxplot(br_day_subset['season'])
```

```
sns.boxplot(br_day_subset['mnth'])
```

```
sns.boxplot(br_day_subset['holiday'])
```

```
sns.boxplot(br_day_subset['weekday'])
```

```
sns.boxplot(br_day_subset['weathersit'])
```

```
sns.boxplot(br_day_subset['temp'])
```

```
sns.boxplot(br_day_subset['atemp'])
```

```
sns.boxplot(br_day_subset['hum'])
```

```
sns.regplot(x = 'hum', y = 'cnt', data = br_day_subset, color = 'r')
```

```
sns.boxplot(br_day_subset['windspeed'])
```

```
sns.regplot(x = 'windspeed' , y = 'cnt', data = br_day_subset, color = 'r')
```

```
sns.boxplot(br_day_subset['cnt'])
```

```
cnames = ['hum', 'windspeed']
```

```
#Detecting and deleting outlier from data using boxplot method
for i in cnames:
    print(i)
    q75,q25 = np.percentile(br_day_subset.loc[:,i],[75,25])

    iqr = q75-q25

    min = q25-(iqr*1.5)
    max = q75+(iqr*1.5)
    print(min)
    print(max)

    br_day_subet = br_day_subset.drop(br_day_subset[br_day_subset.loc[:,i]<min].index)
    br_day_subset = br_day_subset.drop(br_day_subset[br_day_subset.loc[:,i]>max].index)
```

```
#boxplot after deleting outlier
sns.boxplot(br_day_subset['hum'])
```

```
sns.regplot(x = 'hum' , y = 'cnt', data = br_day_subset, color = 'r')
```

```
sns.boxplot(br_day_subset['windspeed'])
```

```
sns.regplot(x = 'windspeed' , y = 'cnt', data = br_day_subset, color = 'r')
```

```
#Checking distribution o variables with help of histogram
```

```
#Checking distribution of season variable
sns.set()
plt.hist(br_day_subset['season'], bins = 'auto', color = 'r')
plt.xlabel('season')
```

```
#Checking the distribution of mnth variable
plt.hist(br_day['mnth'], bins = 'auto', color = 'r')
plt.xlabel('mnth')
```

```
#Checking the distribution of holiday variable
plt.hist(br_day['holiday'], bins = 'auto', color = 'r')
plt.xlabel('holiday')
```

```
#Checking the distribution of weekday variable
plt.xlabel('weekday')
plt.hist(br_day['weekday'], bins = 'auto', color = 'r')
```

```
#Checking the distribution of workingday variable
plt.xlabel('workingday')
plt.hist(br_day['workingday'], bins = 'auto', color = 'r')
```

```
#Checking the distribution of weathersit variable
plt.xlabel('weathersit')
plt.hist(br_day['weathersit'], bins = 'auto', color = 'r')
```

```
#Checking distribution of temp variable
plt.xlabel('temp')
plt.hist(br_day['temp'], bins = 'auto', color= 'r')
```

```python
#Checking distribution of atemp variable
plt.xlabel('atemp')
plt.hist(br_day['atemp'], bins = 'auto', color = 'r')
```

```python
#Checking distribution of hum variable
plt.xlabel('hum')
plt.hist(br_day['hum'], bins = 'auto', color = 'r')
```

```python
#Checking distribution of windspeed variable
plt.xlabel('windspeed')
plt.hist(br_day['windspeed'], bins = 'auto', color = 'r')
```

```python
#Checking distribution of windspeed variable
plt.hist(br_day_subset['cnt'], bins = 'auto', color = 'r')
plt.xlabel('cnt')
```

Feature selection

```python
####Feature selection
cnames = ['season','mnth','holiday','weekday','workingday','weathersit','temp','atemp','hum','windspeed','cnt']
```

```python
br_corr = br_day_subset.loc[:,cnames]
```

```python
f, ax = plt.subplots(figsize=(10,8))

corr_matrix = br_corr.corr()

sns.heatmap(corr_matrix, mask = np.zeros_like(corr_matrix, dtype=np.bool), cmap = sns.diverging_palette(300,10, as_cmap= True),
            annot = True , linewidths = 0.9,square = True, ax=ax)
```

```python
##Plotting regression scattered plot to see positive or negative relation of variables with target variable
```

```python
sns.regplot(x = 'season', y = 'cnt', data = br_day_subset, color = 'r')
```

```python
sns.regplot(x = 'mnth' , y = 'cnt', data = br_day_subset, color = 'r')
```

```python
sns.regplot(x = 'holiday' , y = 'cnt', data = br_day_subset, color = 'r')
```

```python
sns.regplot(x = 'weekday', y = 'cnt', data = br_day_subset, color = 'r')
```

```python
sns.regplot(x= 'workingday' , y = 'cnt', data = br_day_subset, color = 'r')
```

```python
sns.regplot(x = 'weathersit' , y = 'cnt', data = br_day_subset, color = 'r')
```

```python
sns.regplot(x = 'atemp' , y = 'cnt', data = br_day_subset, color = 'r')
```

```python
sns.regplot(x= 'temp', y = 'cnt',  data= br_day_subset,  color = 'r')
```

```python
sns.regplot(x = 'hum' , y = 'cnt', data = br_day_subset, color = 'r')
```

```python
sns.regplot(x= 'windspeed', y = 'cnt',  data= br_day_subset,  color = 'r')
```

Model Development

```python
br_sliced = pd.DataFrame(br_day_subset[['season','holiday','weekday','workingday','weathersit','atemp','windspeed', 'cnt']])
```

```python
#Calculate MAPE ## We have defined a function here
## y_true --- actual value
## y_pred --- predicted value
## abs means absolute value. It rounds up the value to whole number
def MAPE(y_true, y_pred):
    mape = np.mean(abs((y_true-y_pred)/y_true))*100
    return mape
```

```
]: #############Decision Tree Regressor#############
   #select subset using stratified Sampling
   y = br_sliced['season']
   train, test = train_test_split(br_sliced, test_size = 0.2, stratify = y)
```

```
]: br_fit_DT = DecisionTreeRegressor(max_depth =5).fit(train.iloc[:,0:7], train.iloc[:,7])
```

```
]: #Apply above model on test data
   prediction_DT = br_fit_DT.predict(test.iloc[:,0:7])
```

```
]: MAPE(test.iloc[:,7],prediction_DT)
```

```
]: df=br_sliced.copy()
```

```
]: #Dividing data into train and test
   X = br_sliced.values[:, 0:7]
   Y = br_sliced.values[:,7]
   a = br_sliced['season']
   X_train, X_test, y_train, y_test = train_test_split( X, Y, test_size = 0.2, stratify = a)
```

```
]: #Random Forest
   RF_model = RandomForestRegressor(n_estimators = 500, oob_score = True, n_jobs = -1,random_state =50,max_features = "auto",
                                    min_samples_leaf = 50).fit(X_train, y_train)
```

```
]: RF_model
```

```
: RF_prediction = RF_model.predict(X_test)
```

```
: MAPE(y_test, RF_prediction)
```

```
: ###Linear regression
  y= br_sliced['season']
  train, test = train_test_split(br_sliced, test_size = 0.2, stratify = y)
```

```
: ###Linear regression
  linear_regression_model = sm.OLS(train.iloc[:,7], train.iloc[:,0:7]).fit()
```

```
: #Summary of model
  linear_regression_model.summary()
```

```
: #predict the  model

  predict_LR = linear_regression_model.predict(test.iloc[:,0:7])
```

```
: #we will put original values of season variable, weekday and mnth in data set so that it will be easy to understand the
  #graphs.
  df = br_day_subset.copy()
  #br_day_dubset = df.copy
```

```
]: #we will put original values of season variable, weekday and mnth in data set so that it will be easy to understand the
   #graphs.
   df = br_day_subset.copy()
   #br_day_dubset = df.copy
```

```
]: br_day_subset['season'] = br_day_subset['season'].replace(1, "springer")
   br_day_subset['season'] = br_day_subset['season'].replace(2, "summer")
   br_day_subset['season'] = br_day_subset['season'].replace(3, "fall")
   br_day_subset['season'] = br_day_subset['season'].replace(4, "winter")
```

```
]: br_day_subset['weekday'] = br_day_subset['weekday'].replace(0, "sunday")
   br_day_subset['weekday'] = br_day_subset['weekday'].replace(1, "monday")
   br_day_subset['weekday'] = br_day_subset['weekday'].replace(2, "tuesday")
   br_day_subset['weekday'] = br_day_subset['weekday'].replace(3, "wdnesday")
   br_day_subset['weekday'] = br_day_subset['weekday'].replace(4, "thursday")
   br_day_subset['weekday'] = br_day_subset['weekday'].replace(5, "friday")
   br_day_subset['weekday'] = br_day_subset['weekday'].replace(6, "saturday")
```

```
]: br_day_subset['mnth'] = br_day_subset['mnth'].replace(1, "January")
   br_day_subset['mnth'] = br_day_subset['mnth'].replace(2, "Ferbruary")
   br_day_subset['mnth'] = br_day_subset['mnth'].replace(3, "March")
   br_day_subset['mnth'] = br_day_subset['mnth'].replace(4, "April")
   br_day_subset['mnth'] = br_day_subset['mnth'].replace(5, "May")
   br_day_subset['mnth'] = br_day_subset['mnth'].replace(6, "june")
   br_day_subset['mnth'] = br_day_subset['mnth'].replace(7, "July")
   br_day_subset['mnth'] = br_day_subset['mnth'].replace(8, "August")
   br_day_subset['mnth'] = br_day_subset['mnth'].replace(9, "September")
   br_day_subset['mnth'] = br_day_subset['mnth'].replace(10, "October")
   br_day_subset['mnth'] = br_day_subset['mnth'].replace(11, "November")
   br_day_subset['mnth'] = br_day_subset['mnth'].replace(12, "December")
```

```
: sns.barplot(x='season', y='cnt', palette = 'spring',  data = br_day_subset)
```

```
: sns.barplot(x='weekday', y='cnt', palette = 'husl', data = br_day_subset,order=['sunday','monday','tuesday','wdnesday',
                                                                                 'thursday', 'friday','saturday'])
```

```
: sns.barplot(x='mnth', y='cnt', palette = 'husl', data = br_day_subset)
```

```
: sns.barplot(x='workingday', y='cnt', palette = 'husl', data = br_day_subset)
```

```
: sns.barplot(x='weathersit', y='cnt', palette = 'husl', data = br_day_subset)
```

```
:
```

```
:
```