PROJECT REPORT

Bike Rental Count Prediction

Amol Pradhan

December 2019

Contents

1. Introduction

1.1. Problem Statement

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

1.2. Data Exploration

To solve above problem, we have one dataset named **day.csv** containing 16 variables and 731 observations. There are 15 predictor variables and one target variable. The description of all the variables is as following,

Sr. No.	Variable Name	Description
1	Instant	Record index
2	Dteday	Date variable
3	Season	1: springer 2: summer 3: Fall 4: Winter
4	Yr	0: 2011 1: 2012
5	Mnth	Month (1-12)
6	Holiday	Weather day holiday or not (extracted from Holiday Schedule)
7	Weekday	Day of the week
8	Workingday	0: Holiday 1: Working day
9	Weathersit	(extracted from Freemeteo) 1: Clear, Few clouds,Partly clouds 2: Mist+Cloudy,Mist+Broken clouds,Mist+few clouds, Mist 3: Light Snow,Light Rain+thunderstorm, Scatterd clouds,Light Rain+Scatterd clouds 4: Heavy Rain+Ice Pallets+thunderstorm+mist,snow+fog
10	temp	Normalized temperature in Celsius. t_min = -8, t_max=39
11	atemp	Normalized feeling temperature in Celsius. t_min = -16, t_max =+50
12	hum	Normalized humidity. The values are divided to 100(max)
13	Windspeed	Normalized windspeed. The values are divided to 67(max)
14	Casual	Count of casual user
15	Registered	Count of registered user
16	cnt	Count of total rental bikes including both casual and registered

Below is the small subset of total data

	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

Datatypes of All the variables are,

```
instant          int64
dteday          object
season           int64
yr               int64
mnth             int64
holiday          int64
weekday          int64
workingday       int64
weathersit       int64
temp           float64
atemp          float64
hum            float64
windspeed      float64
casual           int64
registered       int64
cnt              int64
dtype: object
```

Fig. Data Types

1.3. Defining problem category

In above problem statement we have to predict total no of bike count based on the environmental and seasonal settings. The bike count is a continuous integer variable. So, for continuous value prediction we will use Regression problem approach. Our task is to build a model which will predict Rental bike count on given environmental settings.

2. Methodology

2.1. Pre-processing

From Initial overview of dataset, it is found that some variables are not useful for model building. These variables are as follows

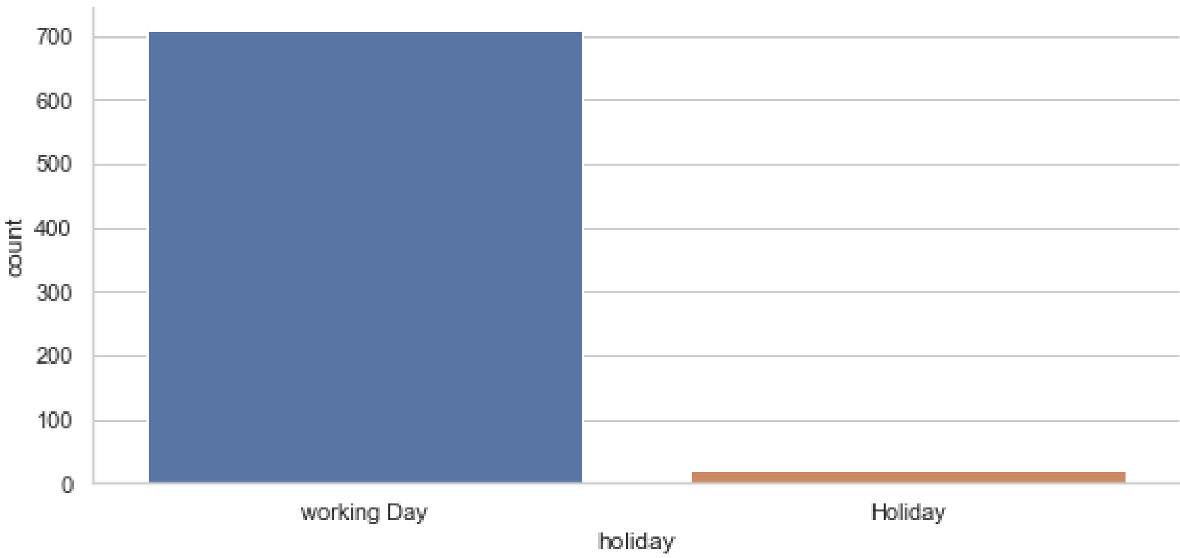
1. **Instant** – It is just an index number. It does not provide any meaningful information about the target variable
2. **dteday** – It is a date-stamp variable. But we have all the required values of date and time present in other variables.
3. **Casual**, **registered** - cnt variable is the addition of casual and registered variable

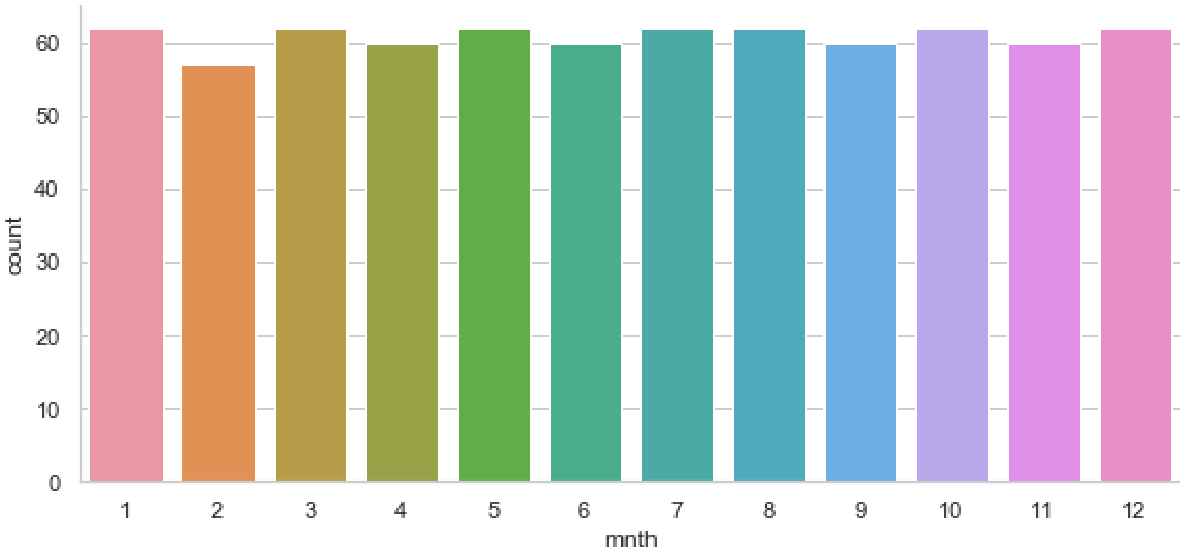
So, I removed these four variables as they do not contribute in model building.

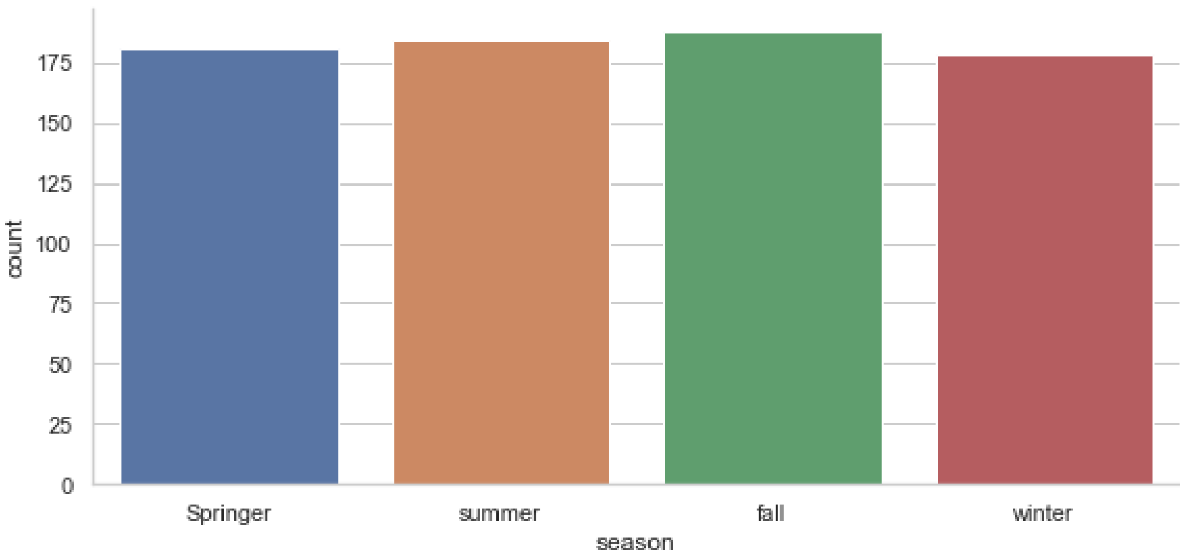
2.2. Univariate Analysis

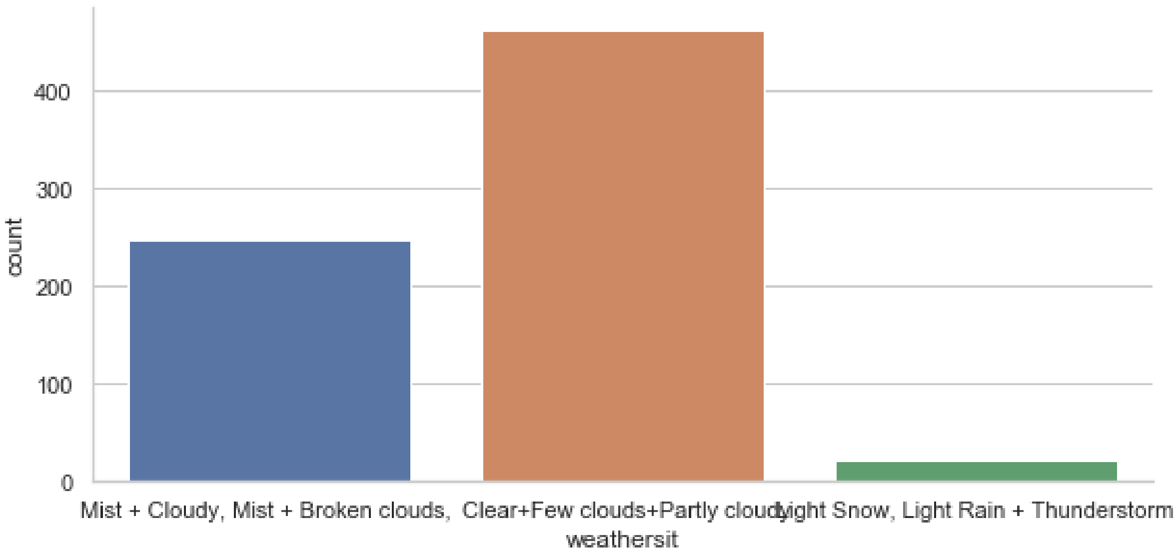
Univariate analysis is the simplest form of the data analysis where the data analysed contains only one variable. Since it is a single variable it does not deal with the cause or relationships. The main purpose of univariate analysis is to describe the data and find the patterns that exists within it. Univariate analysis is done on the categorical variables and continuous variables. For visualization I have use bar graph categorical variable and histogram for continuous variables.

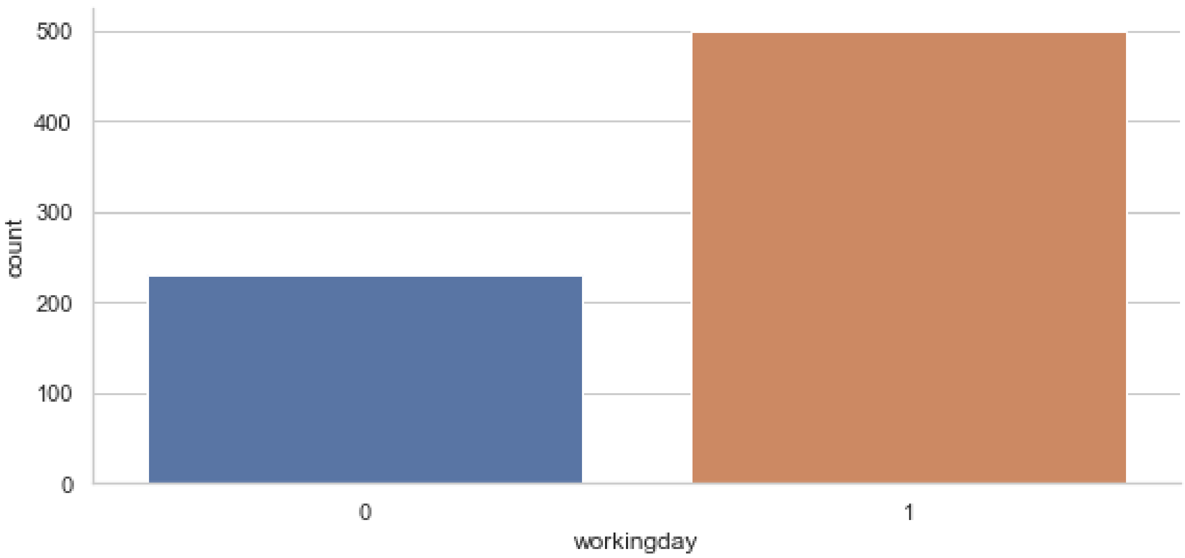
Following are the bar graph of categorial variables,

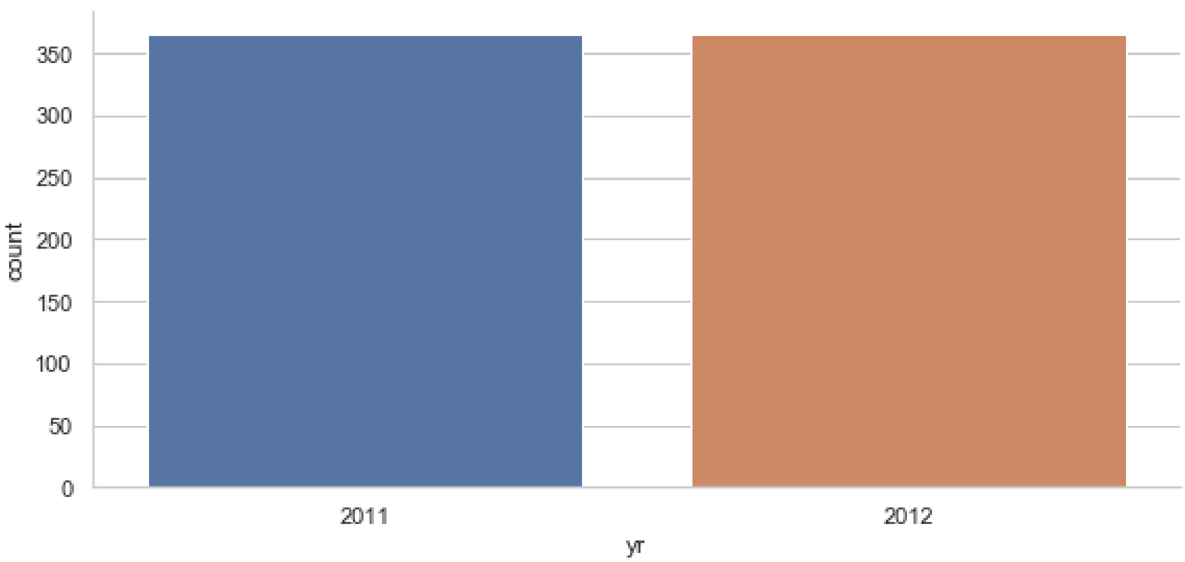

count
700
600
500
400
300
200
100
0
working Day
Holiday
holiday


count
60
50
40
30
20
10
0
1
2
3
4
5
6
7
8
9
10
11
12
mnth


count
175
150
125
100
75
50
25
0
Springer
summer
fall
winter
season





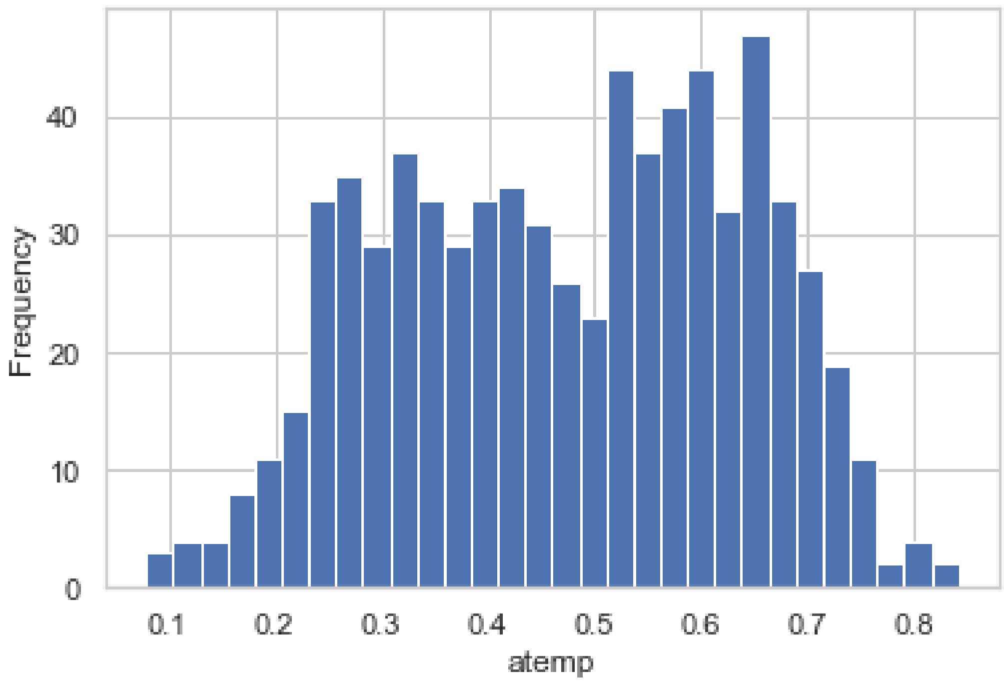


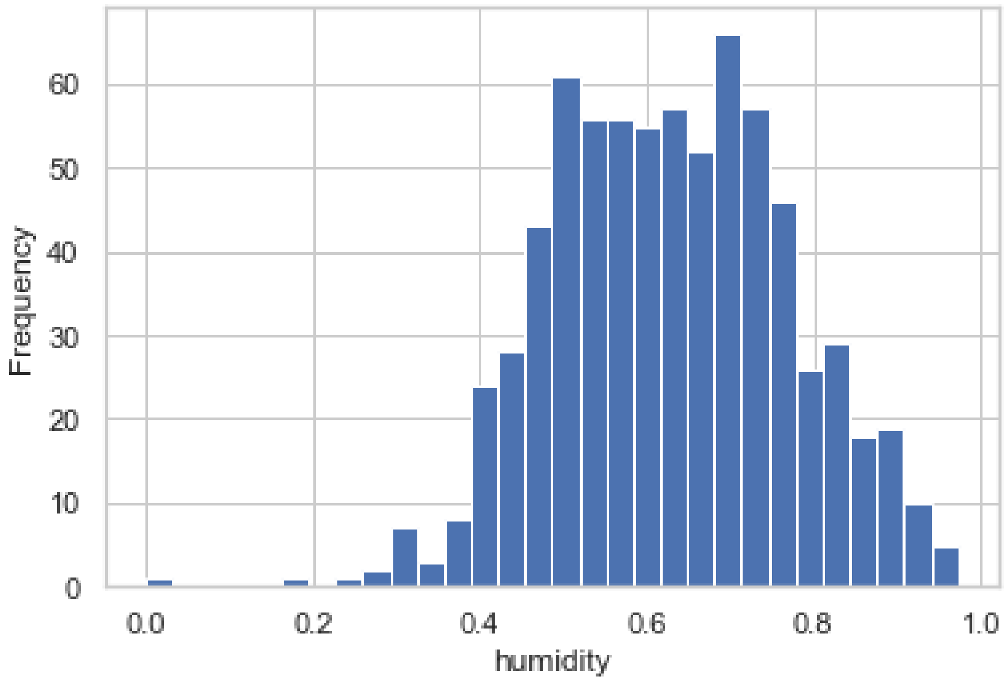
From Above Bar graphs w can say that,

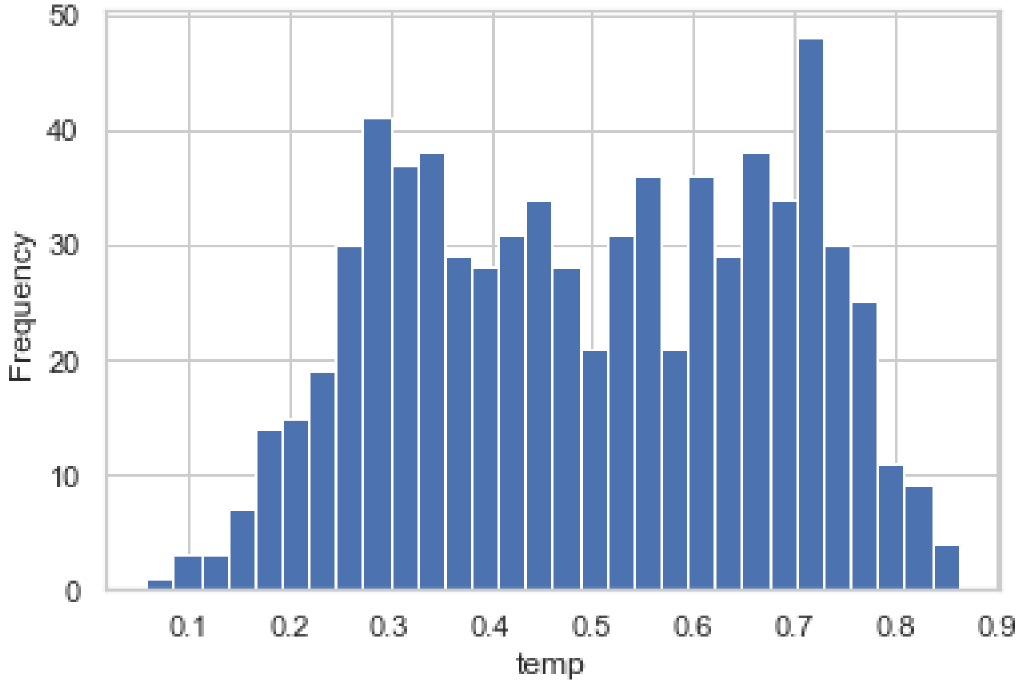
- The demand of bike renting is more on working day rather than holidays.

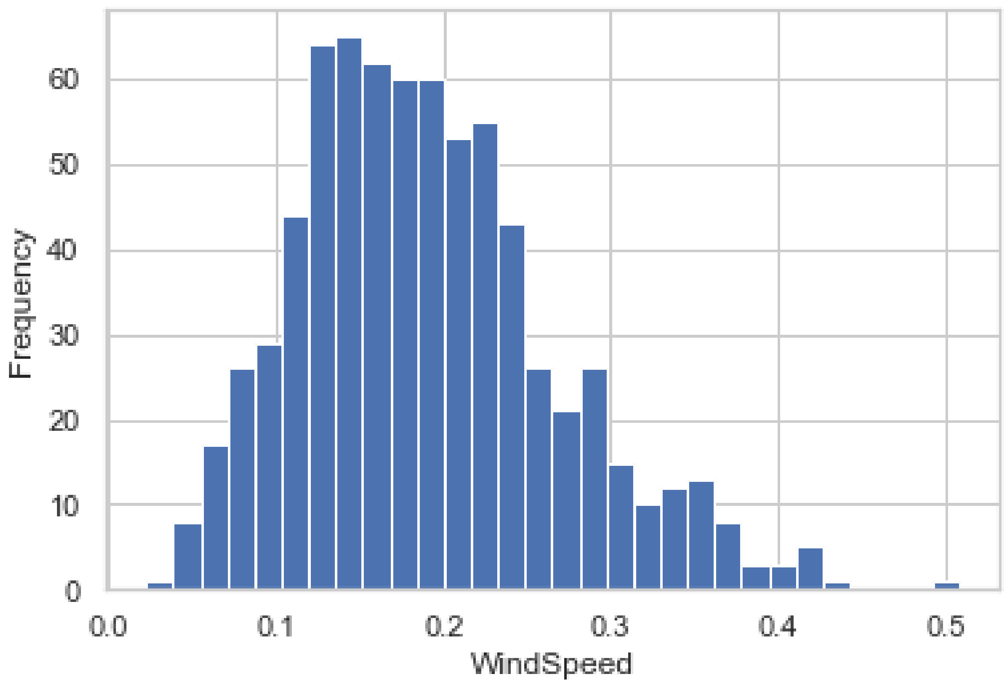
- There is no significant effect of month of year on demand of bikes.
- There is slightly increase in demand in fall season, otherwise in all the seasons demand is almost same.
- People prefer clear weather for bike riding.

Following are the Histogram plot of continuous variables,





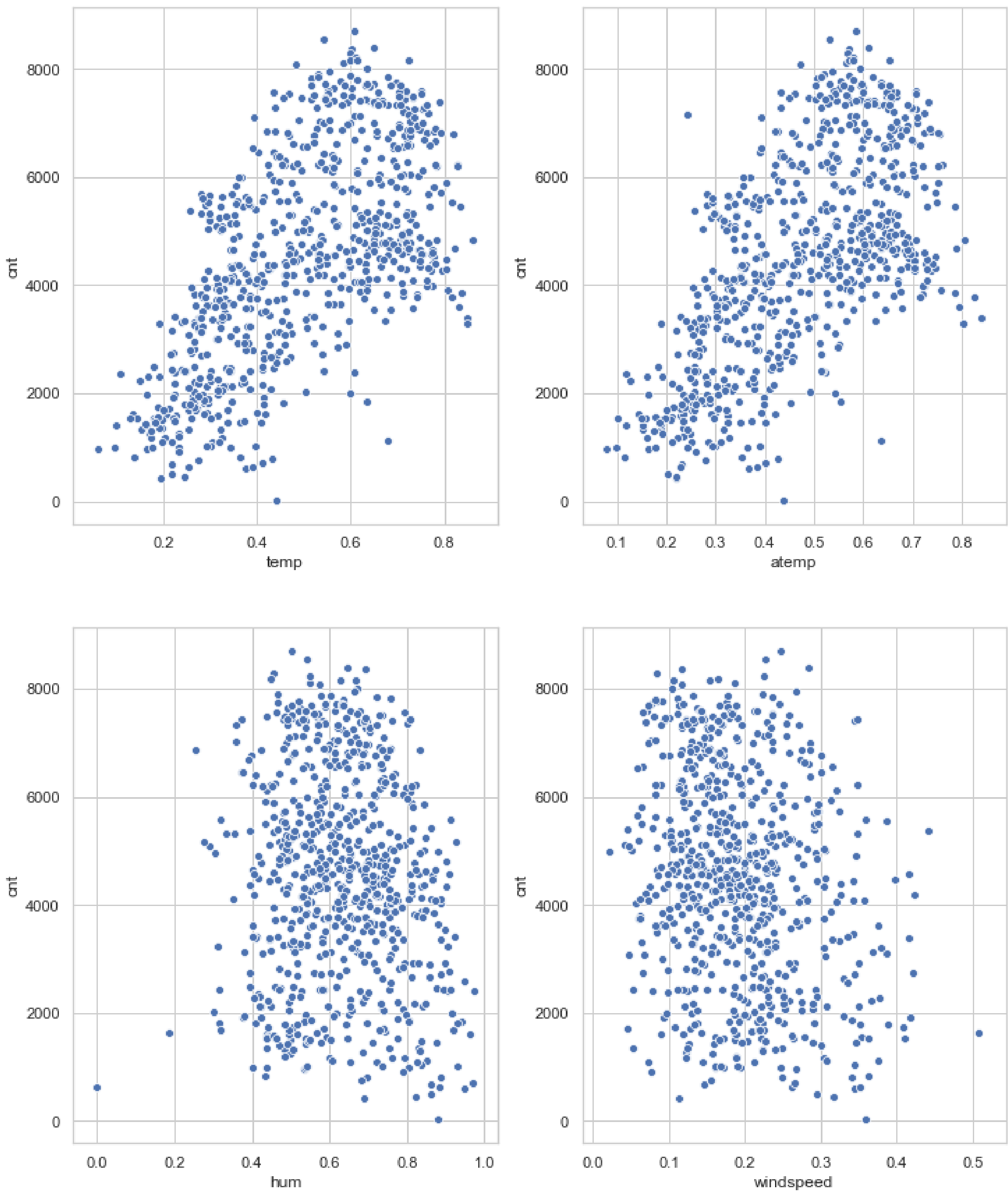




From Above histogram plots of continuous variables, it is seen that all the continuous variables are normally distributed.

2.3. Bivariate Analysis

In Bivariate analysis we check the relationship between all the continuous variables with the target variables. I have used scatter plot for bivariate analysis. It is done only on continuous variables

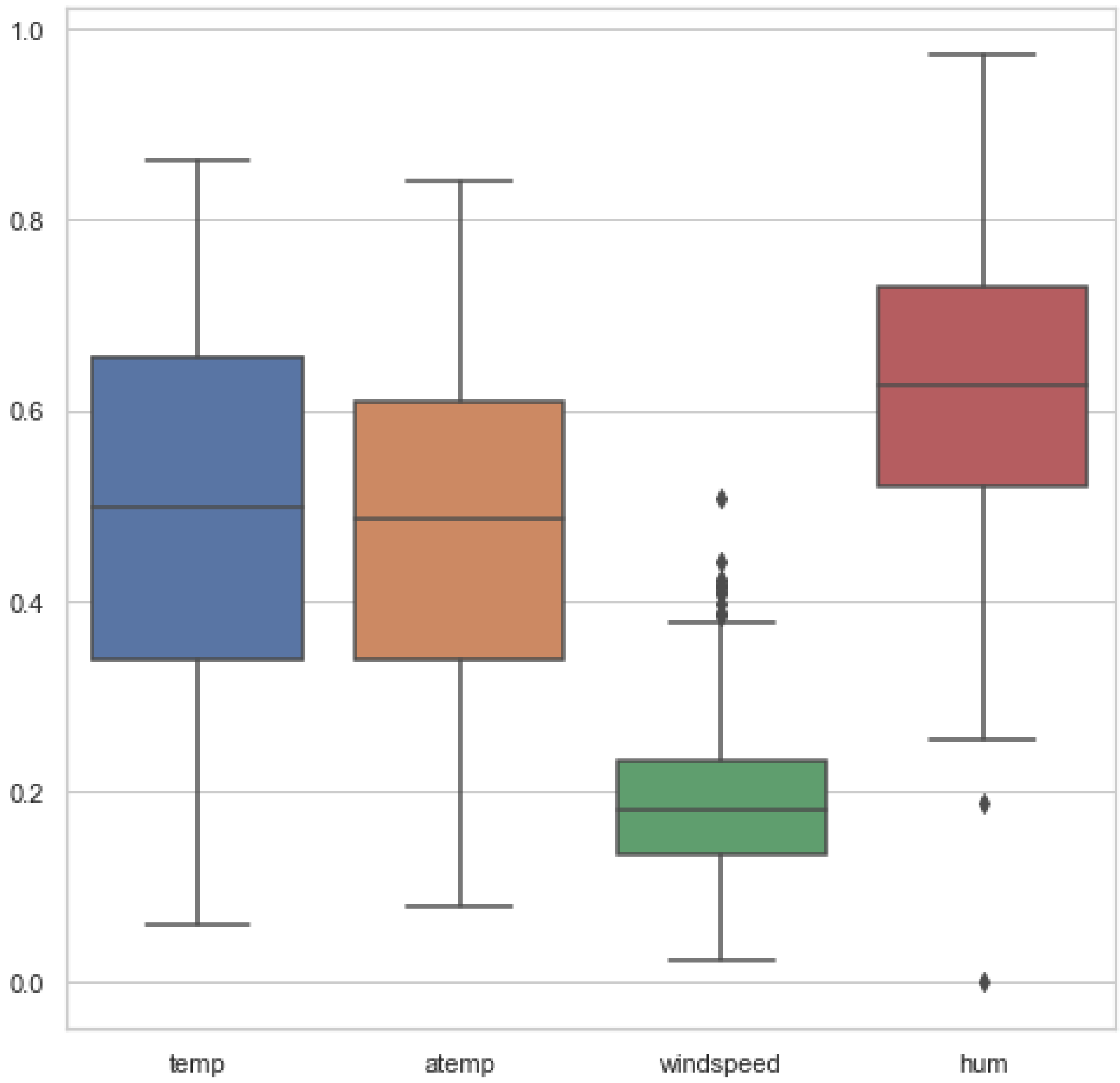


From above scatter plots it is seen that there exists a positive linear relation between temp and atemp variable with cnt target variable. There is slightly negative linear relationship in hum and windspeed variables with the target variable.

2.4. Outlier analysis

An outlier is an observation point that is distant from other observation. Outlier values affect the model development. It will change the mean of variables. When we plot the error, we might get big deviation if outliers are present in the dataset. Outlier analysis is done on continuous variables only.

To visualize outlier, I used box plot. If outlier is present in dataset it will be marked with dots outside the lower and upper fence of box plot.



From Above plots it is seen that windspeed and hum variables contains the outlier values. By setting the lower and upper limit to data we remove the outliers

Minimum value = q25-(1.5*iqr)

Maximum value = q75-(1.5*iqr)

q25 = 25^{th} percentile of data

q75 = 75^{th} percentile of data

iqr = Inter Quartile Range.

iqr = q75 – q25

2.5. Missing Value Analysis

Missing values are the values which are not present in the dataset. These missing values can affect the model as it will reduce the precision. These values can be introduced due to human errors. So, I done missing values analysis on given dataset, so I get following result

	Total	Percent
cnt	0	0.0
windspeed	0	0.0
hum	0	0.0
atemp	0	0.0
temp	0	0.0
weathersit	0	0.0
workingday	0	0.0
weekday	0	0.0
holiday	0	0.0
mnth	0	0.0
yr	0	0.0
season	0	0.0

Fig. Missing Value Result

From above result it is seen that there is no missing value present in the given dataset

2.6. Feature Selection

Feature selection is the process of selecting important features for model building. In this process we check the correlation between two independent variables. If there exist a high positive or negative correlation between two variables, the we drop one variable. Because multicollinearity will affect the output of model. So, for selecting features from all the data available we use correlation analysis.

2.6.1. Correlation analysis

In Correlation analysis we check following two criteria,

1. Relationship between two independent variables should be less
2. Relationship between independent variable and dependent/target variable should be high

To visualize correlation, I used heatmap. The output of heatmap is as follows,

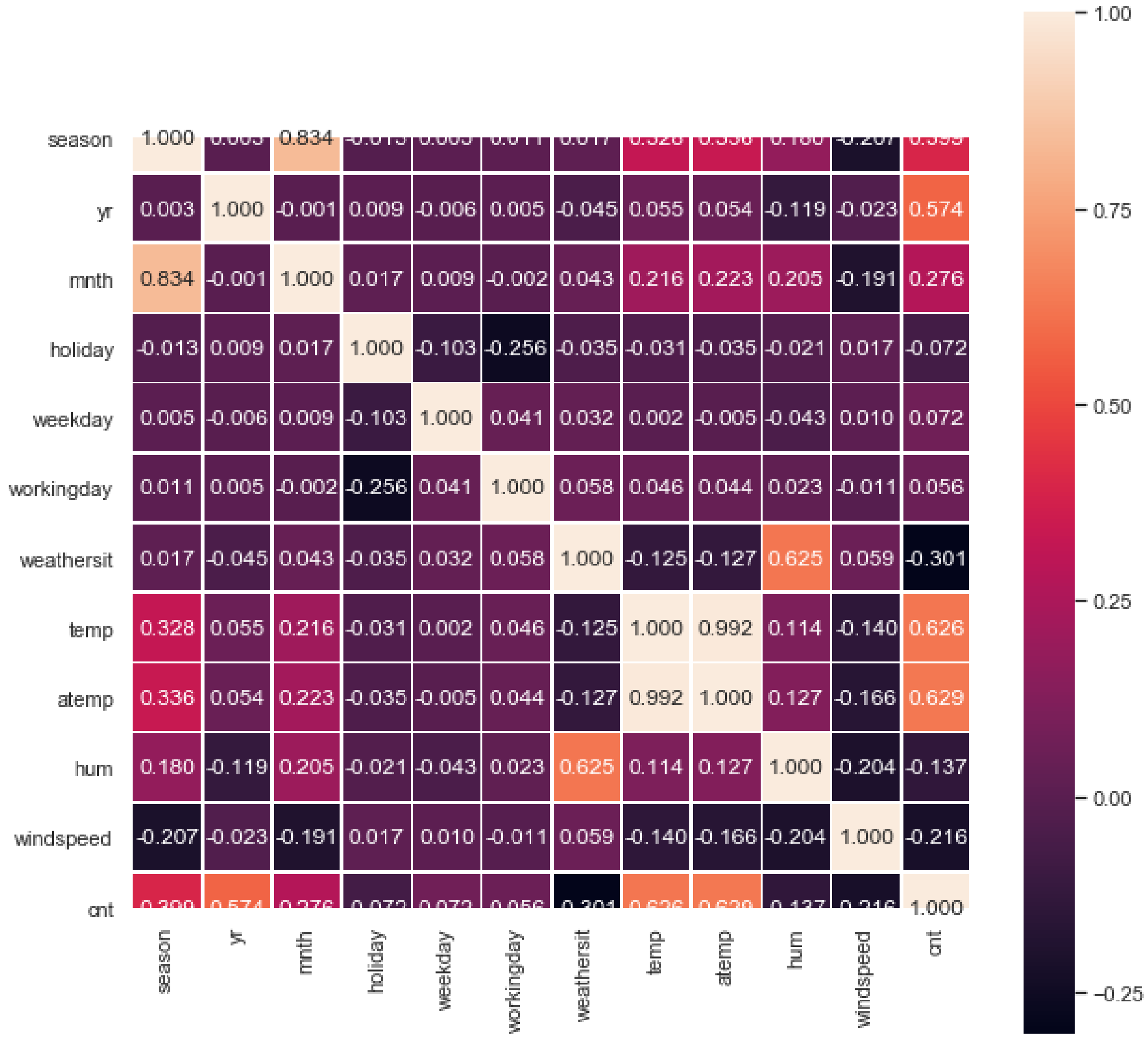


Fig. Correlation plot

From above heatmap plot we can say that,

- There exists high correlation between temp and atemp variable.
- There is very less correlation between holiday, weekday and workingday with target variable cnt.

So, I removed atemp, holiday, weekday and workingday variables.

So, the dataset after feature selection is as follows,

	season	yr	mnth	weathersit	temp	hum	windspeed	cnt
48	1	0	2	1	0.521667	0.516667	0.264925	2927
707	4	1	12	2	0.381667	0.911250	0.101379	5582
426	1	1	3	2	0.353333	0.657083	0.144904	3194
27	1	0	1	2	0.203478	0.793043	0.123300	1167
644	4	1	10	1	0.554167	0.664167	0.268025	7965

Fig. Dataset after feature selection

2.7. Feature Scaling

Feature scaling is used to make the variables in dataset scale free and for the ease of model development. If the scales between two variables is two high, then there is chance that our model will be biased towards large scale.

In our dataset all the continuous are already normalized. We do not need to do feature scaling on dataset

3. Modelling

In Modelling we have to create a model which will predict rental bike count based on environmental and seasonal settings. The target variable is a continuous variable. For continuous variable we can use various regression models. The model having less error rate and more accuracy will be our final model.

Models used for predictions are:

- Linear Regression
- Decision Tree
- Random Forest

Before training our model, we will split our data into train and test subset. Here I have taken 80% of total data as training and remaining 20% data as test data.

3.1. Performance Metrics

For evaluation of trained models, we are using two performance metrics

3.1.1. MAPE

MAPE stands for Mean Absolute Percentage Error. It measures the size of the error in terms of percentage. It is calculated as the average of the unsigned percentage error.

$$\text{MAPE} = \frac{\sum \frac{|A-F|}{A} \times 100}{N}$$

3.1.2. RMSE

RMSE stands for Root Mean Square Error. It is used to measure the difference between values predicted by a model and the values actually observed from the environment that is being modelled.

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

3.2. Linear Regression

Linear regression is most commonly used algorithm. Multiple linear regression is a method used to model the linear relationship between a dependent variable and more than one independent variables. Multiple Linear Regression is based on the ordinary least square (OLS), the model is fit such that the sum of square of differences of observed and predicted values is minimized.

Python Code

Training

```
LR_model = sm.OLS(train.iloc[:,7],train.iloc[:,0:6]).fit()

#Summary
print(LR_model.summary())

#Predict
LR_Model_predict = LR_model.predict(test.iloc[:,0:6])
```

Evaluation

```
MAPE(test.iloc[:,7],LR_Model_predict)
RMSE(test.iloc[:,7],LR_Model_predict)
# MAE is: 687.0163959661395
# MAPE is: 0.20885213428211127
# MSE:   872516.0100288191
# RMSE:   934.0856545461016
```

```
MAE is: 687.0163959661395
MAPE is: 0.20885213428211127
MSE:   872516.0100288191
RMSE:   934.0856545461016
```

R Code

Training

```
#Train
LR_model = lm(formula = cnt~.,data = train)

##summary

summary(LR_model)

LR_model_prediction = predict(LR_model,test[,-8])

df = data.frame("actual" =test[,8],"LR_model_predict"= LR_model_prediction)
head(df)
```

Evaluation

```
regr.eval(trues = test[,8],preds = LR_model_prediction,stats = c("mae","mse","rmse","mape"))

## mae          mse          rmse         mape
## 6.422606e+02 7.366696e+05 8.582946e+02 2.171891e-01
```

3.3. Decision Tree

Decision tree regression models in the form of tree structure. It breaks down a dataset into smaller and smaller subsets while at he same time an associated decision tree is incrementally developed. The final result is a tree with decision nodes and leaf nodes.

A decision node has two or more branches, each representing values for the attribute tested. Leaf node represents a decision on the numerical target. The topmost decision node in a tree corresponds to the best predictor is called root node. Decision tree can handle both categorical and numerical data.

Python code

Training

```
DT_model = DecisionTreeRegressor(random_state=100).fit(train.iloc[:,0:6],train.iloc[:,7])

#prediction
DT_model_predict = DT_model.predict(test.iloc[:,0:6],DT_model)
```

Evaluation

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

```
MAE is: 656.6944444444445
MAPE is: 0.1830472200529958
MSE:  871873.0277777778
RMSE:  933.7414137638845
```

R Code

Training

```
set.seed(12)
DT_model = rpart(cnt~.,data = train,method = "anova")

DT_model_predict = predict(DT_model,test[,-8])

df = data.frame(df,DT_model_predict)
head(df)
par(cex = 0.8)
plot(DT_model)
text(DT_model)
```

Evaluation

```
regr.eval(trues = test[,8],preds = DT_model_predict,stats = c("mae","mse","rmse","mape"))

## mae          mse          rmse         mape
## 7.329756e+02 9.471663e+05 9.732247e+02 2.831340e-01
```

3.4. Random Forest

Random Forest is an ensemble technique capable of performing both regression and classification tasks with multiple decision trees and a technique called Bootstrap Aggregation commonly known as bagging. The basic behind this is to combine multiple decision trees in determining the final output rather than relying on individual decision trees.

Python Code:

Training

```
RF_model = RandomForestRegressor(random_state=123)
np.random.seed(10)

arg_dict = {'max_depth':[2,4,6,8,10],
            'bootstrap':[True,False],
            'max_features':['auto','sqrt','log2',None],
            'n_estimators':[100,200,300,400,500]}

gs_randomForest = RandomizedSearchCV(RF_model,cv=10,param_distributions=arg_dict,
                                    n_iter=10)

gs_randomForest.fit(train.iloc[:,0:6],train.iloc[:,7])
print("Best Parameters using random Search",
     gs_randomForest.best_params_)
```

```
Best Parameters using random Search {'n_estimators': 500, 'max_features': 'sqrt', 'max_depth': 8, 'bootstrap': True}
```

```
RF_model.set_params(n_estimators = 500,
                   max_features='sqrt',
                   max_depth=8,
                   bootstrap=True)
RF_model.fit(train.iloc[:,0:6],train.iloc[:,7])
RF_model_predict = RF_model.predict(test.iloc[:,0:6])
```

Evaluation

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

```
MAE is: 498.20171825789504
MAPE is: 0.15936661694112925
MSE:  469755.8793950975
RMSE:  685.3873936651429
```

R Code

Training

```
RF_model = randomForest(cnt~.,data=train,ntree=500,nodesize=8,importance=TRUE)
RF_model_predict = predict(RF_model,test[,-8])
df = cbind(df,RF_model_predict)
head(df)
```

Evaluation

```
regr.eval(trues = test[,8],preds = RF_model_predict,stat = c("mae","mse","rmse","mape"))

## mae          mse          rmse         mape
## 6.033788e+02 6.486310e+05 8.053763e+02 2.436907e-01
```

3.5. Model Selection

Now we have three models for predicting the target variable, we need to decide which model to choose. There are several criteria exists for evaluating the models. We can compare models using following criteria,

1. Predictive Performance
2. Interpretability
3. Computational Efficiency

In our case, the interpretability and computation efficiency, do not hold much significance. Therefore, we will use predictive performance as the criteria to compare and evaluate models. Predictive performance can be measured by comparing prediction of the model with real values of the target variables and calculating some average error measure.

Performance Measurement:

In Python

Sr. No.	Algorithm	MAPE	RMSE
1.	Linear Regression	0.20	934.8
2.	Decision Tree	0.18	933.74
3.	Random Forest	0.15	685

As from the above table we can see that **Random Forest** preform better than Linear regression and decision tree algorithm in Python implementation.

In R

Sr. No.	Algorithm	MAPE	RMSE
1.	Linear Regression	0.21	858
2.	Decision Tree	0.28	973
3.	Random Forest	0.24	805

As from above table we can say see that **Random forest** perform better than linear regression and decision tree in R Implementation.

4. Conclusion

From the **RMSE** Error metric performance on Linear Regression, Decision Tree and Random Forest algorithm, I conclude that **Random Forest** algorithm works better for our problem statement of predicting of rental bike count.

Appendix A – Python Code

```
#!/usr/bin/env python

# coding: utf-8

# ## Importing Required libaries

# In[45]:

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

import numpy as np

from sklearn.model_selection import train_test_split,RandomizedSearchCV

from sklearn.metrics import mean_squared_error

from math import sqrt

import statsmodels.api as sm
```

```
from sklearn.linear_model import LinearRegression
from sklearn.neighbors import KNeighborsRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
# In[46]:
# Loading the dataset
data = pd.read_csv("day.csv")
# In[47]:
#Checking the dimensions of dataset
data.shape
# In[48]:
data.head()
# In[49]:
#Inital insight of datset
data.describe()

# In[50]:
data.dtypes
# In[51]:
## Removing unnessary variables from dataset
# instant - It is basically index number
# dteday - All the values from dteday are present in datset under differnet variables
# casual and registered - cnt is basically the sum of casual amd registerd variables
data = data.drop(['instant','dteday','casual','registered'],axis=1)
# In[52]:
#Creating a copy of original dataset
data_vis = data.copy()
data.head()
# In[53]:
##Converting the interger values into proper naming
data_vis['season'] = data_vis['season'].replace([1,2,3,4],['Springer','summer','fall','winter'])
```

```
data_vis['yr'] = data_vis['yr'].replace([0,1],[2011,2012])

data_vis['weathersit'] = data_vis['weathersit'].replace([1,2,3,4],[' Clear+Few clouds+Partly cloudy','Mist + Cloudy, Mist + Broken clouds, ',' Light Snow, Light Rain + Thunderstorm ','Heavy Rain + Ice Pallets '])

data_vis['holiday'] = data_vis['holiday'].replace([0,1],['working Day','Holiday'])

data_vis.head()

# In[54]:

print(data.dtypes)

print(data.head())

# ## Univarient Analysis

# In[55]:

## Bar Graph for Categorical data


sns.set_style("whitegrid")

sns.factorplot(data=data_vis,x='season',kind='count',size=4,aspect=2)

sns.factorplot(data=data_vis,x='yr',kind='count',size=4,aspect=2)

sns.factorplot(data=data_vis,x='mnth',kind='count',size=4,aspect=2)

sns.factorplot(data=data_vis,x='holiday',kind='count',size=4,aspect=2)

sns.factorplot(data=data_vis,x='workingday',kind='count',size=4,aspect=2)

sns.factorplot(data=data_vis,x='weathersit',kind='count',size=4,aspect=2)

# ## Outlier Analysis

# In[56]:

## Checking the presence of outlier in continous variables

sns.boxplot(data = data[['temp','atemp','windspeed','hum']])

fig  = plt.gcf()

fig.set_size_inches(8,8)


# In[57]:

## Removing outlier and checking correlation between target variable and independent continous variables

print(data.shape)

print(data['hum'].corr(data['cnt']))
```

```
print(data['windspeed'].corr(data['cnt']))
q75, q25 = np.percentile(data.loc[:,'hum'],[75,25])
iqr = q75 - q25
min = q25-(iqr*1.5)
max = q75+(iqr*1.5)
print(min)
print(max)
data = data.drop(data[data.loc[:,'hum']<min].index)
data = data.drop(data[data.loc[:,'hum']>max].index)
q75, q25 = np.percentile(data.loc[:,'windspeed'],[75,25])
iqr = q75 - q25
min = q25-(iqr*1.5)
max = q75+(iqr*1.5)
print(min)
print(max)
data = data.drop(data[data.loc[:,'windspeed']<min].index)
data = data.drop(data[data.loc[:,'windspeed']>max].index)

# In[58]:
#Dimensions of dataset after removing outliers
data.shape

# ## Missing Value Analysis
# In[59]:


total = data.isnull().sum().sort_values(ascending=False)
percent = (data.isnull().sum()/data.isnull().count()).sort_values(ascending=False)
missing_data = pd.concat([total, percent], axis=1, keys=['Total', 'Percent'])
missing_data.head(30)

# There are no missing vlaues present after outlier analysis
```

```
# In[60]:
plt.hist(data_vis['temp'],bins=30)
plt.xlabel('temp')
plt.ylabel('Frequency')
plt.show()


# In[61]:
plt.hist(data_vis['atemp'],bins=30)
plt.xlabel('atemp')
plt.ylabel('Frequency')
plt.show()


# In[62]:
plt.hist(data_vis['hum'],bins=30)
plt.xlabel('humidity')
plt.ylabel('Frequency')
plt.show()


# In[63]:
plt.hist(data_vis['windspeed'],bins=30)
plt.xlabel('WindSpeed')
plt.ylabel('Frequency')
plt.show()


# ## Bivariant Analysis


# In[64]:
##  Using Scatter Plot
# Index(['instant', 'season', 'yr', 'mnth', 'holiday', 'weekday', 'workingday',
#        'weathersit', 'temp', 'atemp', 'hum', 'windspeed', 'casual',
```

```
#       'registered', 'cnt'],
#      dtype='object')


# In[65]:
fig,x = plt.subplots(nrows= 2,ncols=2)
fig.set_size_inches(12,15)
sns.scatterplot(x="temp",y = "cnt",data = data_vis,palette="Set3",ax=x[0][0])
sns.scatterplot(x="atemp",y = "cnt",data = data_vis,palette="Set3",ax=x[0][1])
sns.scatterplot(x="hum",y = "cnt",data = data_vis,palette="Set3",ax=x[1][0])
sns.scatterplot(x="windspeed",y = "cnt",data = data_vis,palette="Set3",ax=x[1][1])


# ## Feature Selection
# In[66]:
def Correlation(df):
    df_corr = df.loc[:,df.columns]
    corr = df_corr.corr()
    sns.set()
    plt.figure(figsize=(10,10))
    sns.heatmap(corr,annot=True,fmt=".3f",square=True,linewidths=0.5)


Correlation(data)
# In[67]:
## There is high correlation between temp and atemp variable
## there is very weak relation between holiday, weekday and working day variables
## So we will drop those variables
data_fs = data.drop(['atemp','holiday','weekday','workingday'],axis=1)
data_fs.head()


# In[68]:
# Splitting Dataset into train and test dataset
```

```
train,test = train_test_split(data_fs,test_size=0.2,random_state=121)
# ## Feature Scaling
# In[69]:

## Data is normalized no need to do feature scaling
train.head()

# ## Error Metrics
# In[70]:
## Defining Performance Metrics

def MAPE(y_true, y_pred):
    MAE = np.mean(np.abs((y_true - y_pred)))
    mape = np.mean(np.abs((y_true - y_pred) / y_true))
    print("MAE is:", MAE)
    print("MAPE is:", mape)
    return mape

def RMSE(y_true, y_pred):
    mse = np.mean((y_true - y_pred)**2)
    rmse = np.sqrt(mse)
    print("MSE: ",mse)
    print("RMSE: ",rmse)
    return rmse

# ## Linear Regression
# In[71]:
LR_model = sm.OLS(train.iloc[:,7],train.iloc[:,0:6]).fit()
#Summary
print(LR_model.summary())
#Predict
```

```
LR_Model_predict = LR_model.predict(test.iloc[:,0:6])


# In[72]:

MAPE(test.iloc[:,7],LR_Model_predict)

RMSE(test.iloc[:,7],LR_Model_predict)

# MAE is: 687.0163959661395

# MAPE is: 0.20885213428211127

# MSE:  872516.0100288191

# RMSE:  934.0856545461016


# In[73]:

result = pd.DataFrame({'Actual':test.iloc[:,7],'Prediction':LR_Model_predict})

result.head()


# ## Desicion Tree


# In[74]:

DT_model = DecisionTreeRegressor(random_state=100).fit(train.iloc[:,0:6],train.iloc[:,7])

#prediction

DT_model_predict = DT_model.predict(test.iloc[:,0:6],DT_model)


# In[75]:

MAPE(test.iloc[:,7],DT_model_predict)

RMSE(test.iloc[:,7],DT_model_predict)


# In[76]:

# MSE:  924849.3888888889

# RMSE:  961.6909009078171


# In[77]:

result = pd.DataFrame({'Actual':test.iloc[:,7],'Prediction':DT_model_predict})
```

```
result.head()

# ## Random Forest
# In[78]:


RF_model = RandomForestRegressor(random_state=123)
np.random.seed(10)

arg_dict = {'max_depth':[2,4,6,8,10],
        'bootstrap':[True,False],
        'max_features':['auto','sqrt','log2',None],
        'n_estimators':[100,200,300,400,500]}

gs_randomForest = RandomizedSearchCV(RF_model,cv=10,param_distributions=arg_dict,
                    n_iter=10)

gs_randomForest.fit(train.iloc[:,0:6],train.iloc[:,7])
print("Best Parameters using random Search",
    gs_randomForest.best_params_)

# In[79]:
RF_model.set_params(n_estimators = 500,
            max_features='sqrt',
            max_depth=8,
            bootstrap=True)
RF_model.fit(train.iloc[:,0:6],train.iloc[:,7])
RF_model_predict = RF_model.predict(test.iloc[:,0:6])

# In[80]:
MAPE(test.iloc[:,7],RF_model_predict)
```

RMSE(test.iloc[:,7],RF_model_predict)

In[81]:

MSE: 469755.8793950975

RMSE: 685.3873936651429

In[82]:

result = pd.DataFrame({'Actual':test.iloc[:,7],'Prediction':RF_model_predict})

result.head()

From above models Random forest is performing well according to RMSE values

Appendix B – R Code

#Clean the environment

rm(list=ls())

##Set the working directory

setwd("D:/Amol_Data/Edwisor/Assignments/Project_2")

getwd()

##Load the required libraries

libraries = c("ggplot2","plyr","dplyr","rpart","gbm","DMwR","randomForest","usdm","corrgram","DataCombine")

lapply(X=libraries,require,character.only=TRUE)

#Load the dataset

data_day = read.csv(file="day.csv",header = T,sep=",",na.strings = c(" ","","NA"))

######################Exploraty Data Analysis##

head(data_day)

dim(data_day)

str(data_day)

##Droping data which is not essential

#instant- index number

```
#dteday - all the required values are present

#casual,registered - cnt variable is  the sum of casual and registered variable

data_day <- subset(data_day,select = -c(instant,dteday,casual,registered))


data_day$season = as.factor(data_day$season)

data_day$yr =as.factor(data_day$yr)

data_day$mnth = as.factor(data_day$mnth)

data_day$weekday =as.factor(data_day$weekday)

data_day$workingday = as.factor(data_day$workingday)

data_day$holiday = as.factor(data_day$holiday)


##################################Univarent Analysis####################################

data_vis <- data_day

data_vis$season = factor( x = data_vis$season,level = c(1,2,3,4),labels = c("Spring","Summer","Fall","Winter"))

data_vis$yr = factor(x=data_vis$yr,levels = c(0,1),labels = c("2011","2012"))

data_vis$holiday = factor(x=data_vis$holiday,levels = c(0,1),labels = c("Working","Holiday"))

data_vis$weathersit =factor(x=data_vis$weathersit,levels = c(1,2,3,4),labels = c("Clear","Cloudy/Mist","Rain/Snow?Fog","Heavy Rain/Snow/Fog"))

bar1 = ggplot(data = data_vis,aes(x = season)) +geom_bar() + ggtitle("Season")

bar2 = ggplot(data = data_vis,aes(x=workingday)) + geom_bar() + ggtitle("working dayr")

bar3 = ggplot(data = data_vis,aes(x=holiday)) + geom_bar() + ggtitle("Holiday")

bar4 = ggplot(data = data_vis,aes(x=weathersit)) + geom_bar() + ggtitle("weather")


gridExtra::grid.arrange(bar1,bar2,bar3,bar4,ncol=2)

#################################### Bivarient Analysis########################################

sct1 = ggplot(data=data_vis,aes(x=temp,y=cnt)) + ggtitle("Temp") +geom_point() + xlab("Temperature") +ylab("Bike count")

sct2 = ggplot(data=data_vis,aes(x=atemp,y=cnt)) + ggtitle("aTemp") +geom_point() + xlab("Temperature") +ylab("Bike count")
```

```
sct3 = ggplot(data=data_vis,aes(x=hum,y=cnt)) + ggtitle("Humidity") +geom_point() + xlab("Humidity") +ylab("Bike count")

sct4 = ggplot(data=data_vis,aes(x=windspeed,y=cnt)) + ggtitle("Windspeed") +geom_point() + xlab("windspeed") +ylab("Bike count")

gridExtra::grid.arrange(sct1,sct2,sct3,sct4,ncol=2)


####################################BOx plot for outlier analysis################################

cnames = colnames(data_day[,c("temp","atemp","hum","windspeed")])

cnames

 for( i in 1:length(cnames))

 {

   assign(paste0("gn",i),ggplot(aes_string(x=cnames[i],y='cnt'), data=data_day) +

           stat_boxplot(geom = "errorbar",width =0.5)+

           geom_boxplot(outlier.colour = "red",fill="grey",outlier.shape =18,

                        outlier.size=1,notch=FALSE)+

          theme(legend.position = "bottom")+

           labs(x=cnames[i])+

           ggtitle(paste("Box plot for",cnames[i])))

 }


gridExtra::grid.arrange(gn1,gn2,gn3,gn4,ncol=4)


cor(data_day$temp,data_day$cnt)

cor(data_day$hum,data_day$cnt)


##Remove outlier in windspeed

val = data_day$windspeed[data_day$windspeed %in% boxplot.stats(data_day$windspeed)$out]

val

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


val = data_day$hum[data_day$hum %in% boxplot.stats(data_day$hum)$out]
```

```
val

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


cnames = colnames(data_day[,c("hum","windspeed")])

cnames

for( i in 1:length(cnames))

{

  assign(paste0("gn",i),ggplot(aes_string(x=cnames[i],y='cnt'), data=data_day) +

         stat_boxplot(geom = "errorbar",width =0.5)+

         geom_boxplot(outlier.colour = "red",fill="grey",outlier.shape =18,

                      outlier.size=1,notch=FALSE)+

         theme(legend.position = "bottom")+

         labs(x=cnames[i])+

         ggtitle(paste("Box plot for",cnames[i])))

}

gridExtra::grid.arrange(gn1,gn2,ncol=2)

#####################################Missing Value Analysis#################################

missing_values = sapply(data_day,function(x){sum(is.na(x))})

print(missing_values)

## No missing Values

############################################ Feature scaling#################################

##In dataset numeric continous variables "temp","atemp","hum",and "windspeed" are in normalized form

## No need to feature scaling

########################################### Feature selection ###############################

df_vif = data_day[,c("temp","atemp","hum","windspeed")]

vifcor(df_vif)

corrgram(data_day,order =F,upper.panel = panel.pie, text.panel = panel.txt,main="Correlation Plot")

names(data_day)
```

```
data_day <- subset(data_day,select = c("season","yr","mnth","weekday","temp","hum","windspeed","cnt"))

rmExcept(keepers = "data_day")

##Split data into train and test

set.seed(123)

train_index = sample(1:nrow(data_day),0.8*nrow(data_day))

train = data_day[train_index,]

test = data_day[-train_index,]

############################################# Linear Regression ##############################

#Train

LR_model = lm(formula = cnt~.,data = train)

##summary

summary(LR_model)

LR_model_prediction = predict(LR_model,test[,-8])

df = data.frame("actual" =test[,8],"LR_model_predict"= LR_model_prediction)

head(df)

regr.eval(trues = test[,8],preds = LR_model_prediction,stats = c("mae","mse","rmse","mape"))


## mae      mse      rmse      mape

## 6.422606e+02 7.366696e+05 8.582946e+02 2.171891e-01


#######################################Decision tree ################################


set.seed(12)

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

DT_model_predict = predict(DT_model,test[,-8])

df = data.frame(df,DT_model_predict)

head(df)

par(cex = 0.8)

plot(DT_model)

text(DT_model)
```

regr.eval(trues = test[,8],preds = DT_model_predict,stats = c("mae","mse","rmse","mape"))

mae mse rmse mape

7.329756e+02 9.471663e+05 9.732247e+02 2.831340e-01

##################################### Random Forest ##################################

RF_model = randomForest(cnt~.,data=train,ntree=500,nodesize=8,importance=TRUE)

RF_model_predict = predict(RF_model,test[,-8])

df = cbind(df,RF_model_predict)

head(df)

regr.eval(trues = test[,8],preds = RF_model_predict,stat = c("mae","mse","rmse","mape"))

mae mse rmse mape

6.033788e+02 6.486310e+05 8.053763e+02 2.436907e-01

############ Result ##################

From Above three model's RMSE metrics we can say that Random Forest works better than Decision Tree and Linear Regression