Project: Bike Renting
Submitted By: Divyanshu
Date: 28/01/2020

Contents

1. Introduction

1.1 Problem Statement

The objective of this Case is to Predication of bike rental count on daily based on the environmental and seasonal settings. The details of data attributes in the dataset are as follows -

1.2 Data

Given below is a sample of the data set that we are using to predict the number of bikes:

Table 1.1: Bike Count Sample 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

As we can see in the table below we have the following 16 variables, using which we have to correctly predict the count of bikes:

Sr.No	Variables
1	Instant
2	Dteday
3	Season
4	Yr
5	Month
6	Holiday
7	Weekday
8	Workingday
9	Weathersit
10	Temp
11	Atemp
12	Hum
13	windspeed
14	casual
15	registered
16	cnt

1.3 Goal

Aim of the project is to build predictive modal to count of bike rental based on the environmental and seasonal settings. By predicting the count, it would be possible to help accommodate in managing the number of bikes required on a daily basis, and being prepared for high demand of bikes during peak periods. The goal is to build regression models which will predict the number of bikes used based on the environmental and season behaviour.

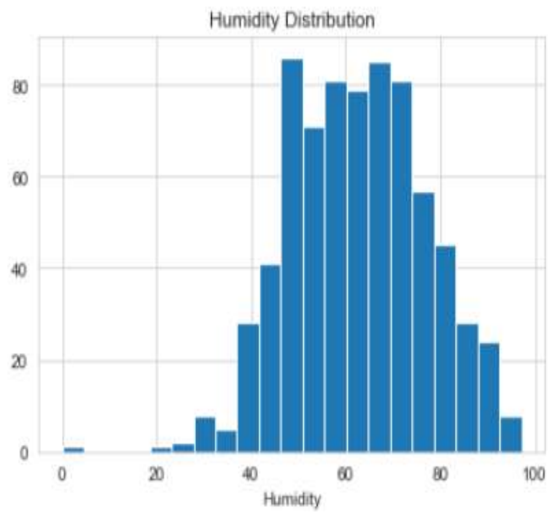
2. Methodology

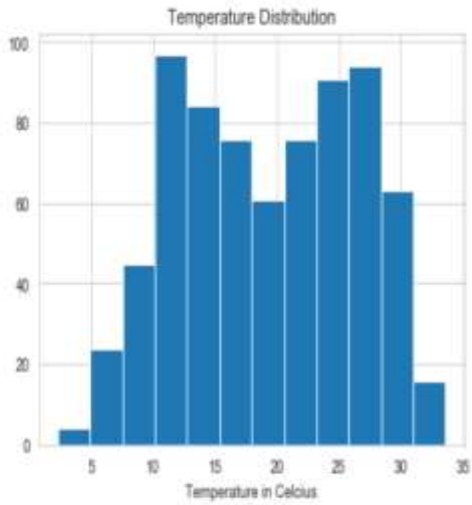
2.1 Pre-Processing

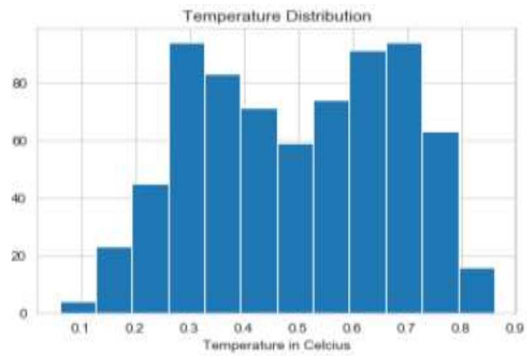
A predictive model requires that we look at the data before we start to create a model. However, in data mining, looking at data refers to exploring the data, cleaning the data as well as visualizing the data through graphs and plots. This is known as Exploratory Data Analysis.

2.2 Distribution of continuous variables

It can be observed from the below histograms is that temperature and feel temperature are normally distributed, where as the variables windspeed and humidity are slightly skewed. The skewness is likely because of the presence of outliers and extreme data in those variables.







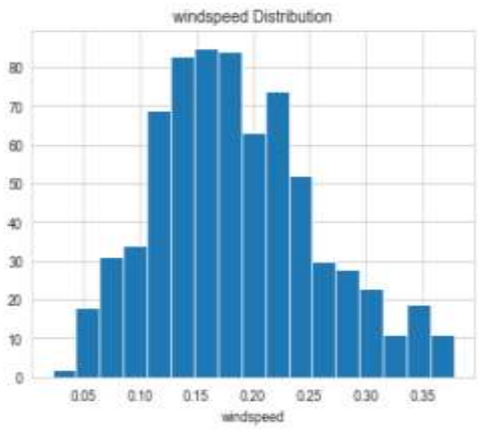
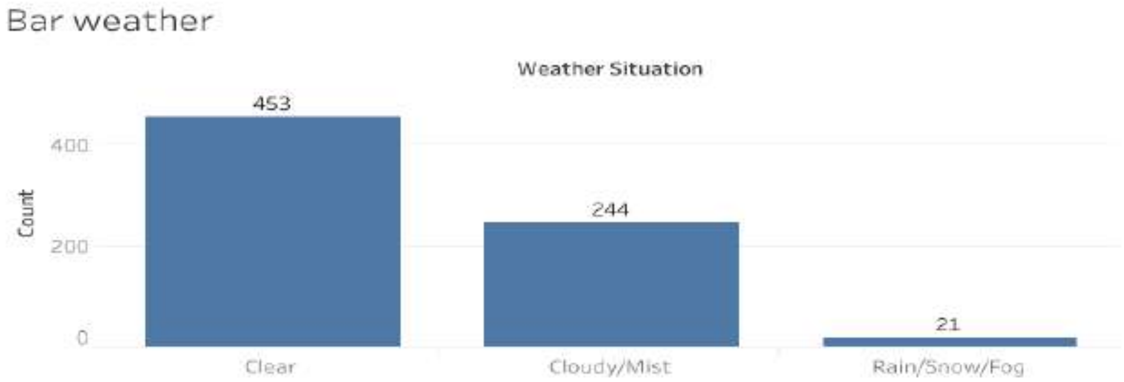
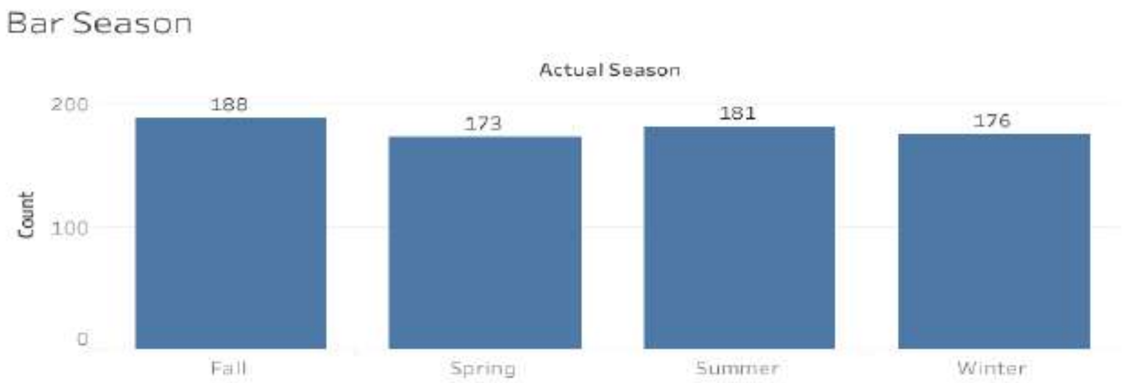


Fig2.1 Distribution of continuous variable using histograms

2.3 Distribution of categorical variables

The distribution of categorical variables is as shown in the below figure:





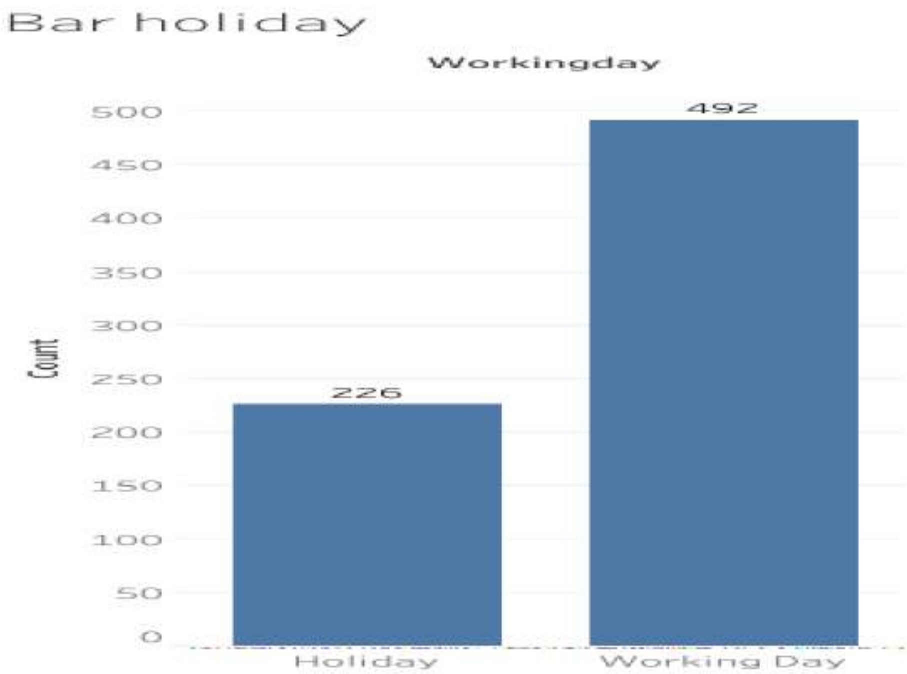


Fig 2.2 Distribution of categorical variable using bar plots

2.4 Relationship of Continuous variables against bike count

The below figure shows the relationship between continuous variables and the target variable using scatter plot. It can be observed that there exists a linear positive relationship between the variables temperature and feel temperature with the bike rental count. There also exists a negative linear relationship between the variable’s humidity and windspeed with the bike rental count.

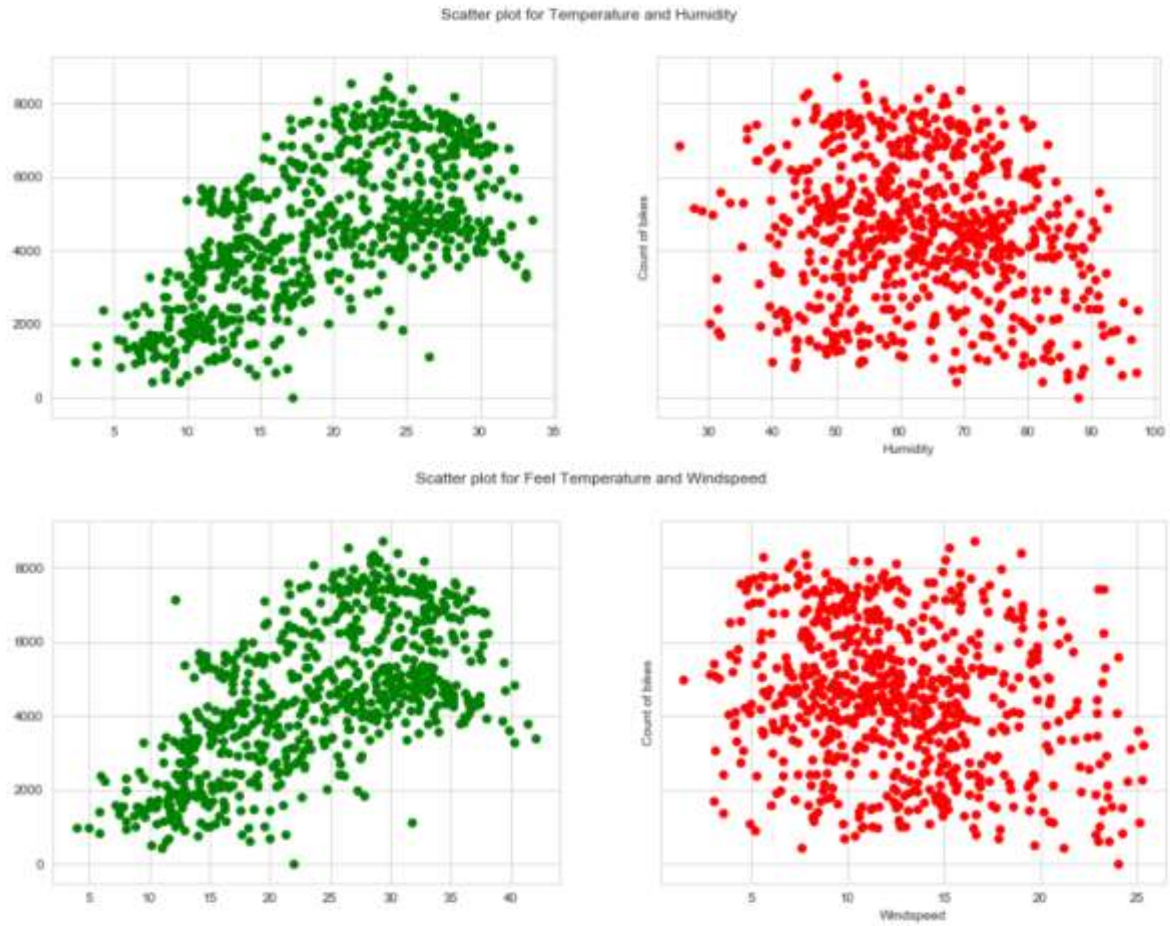


Fig 2.3 Scatter plot for continuous variable

2.5: Detection of outliers:

Outliers are detected using boxplots. Below figure illustrates the boxplots for all the continuous variables.

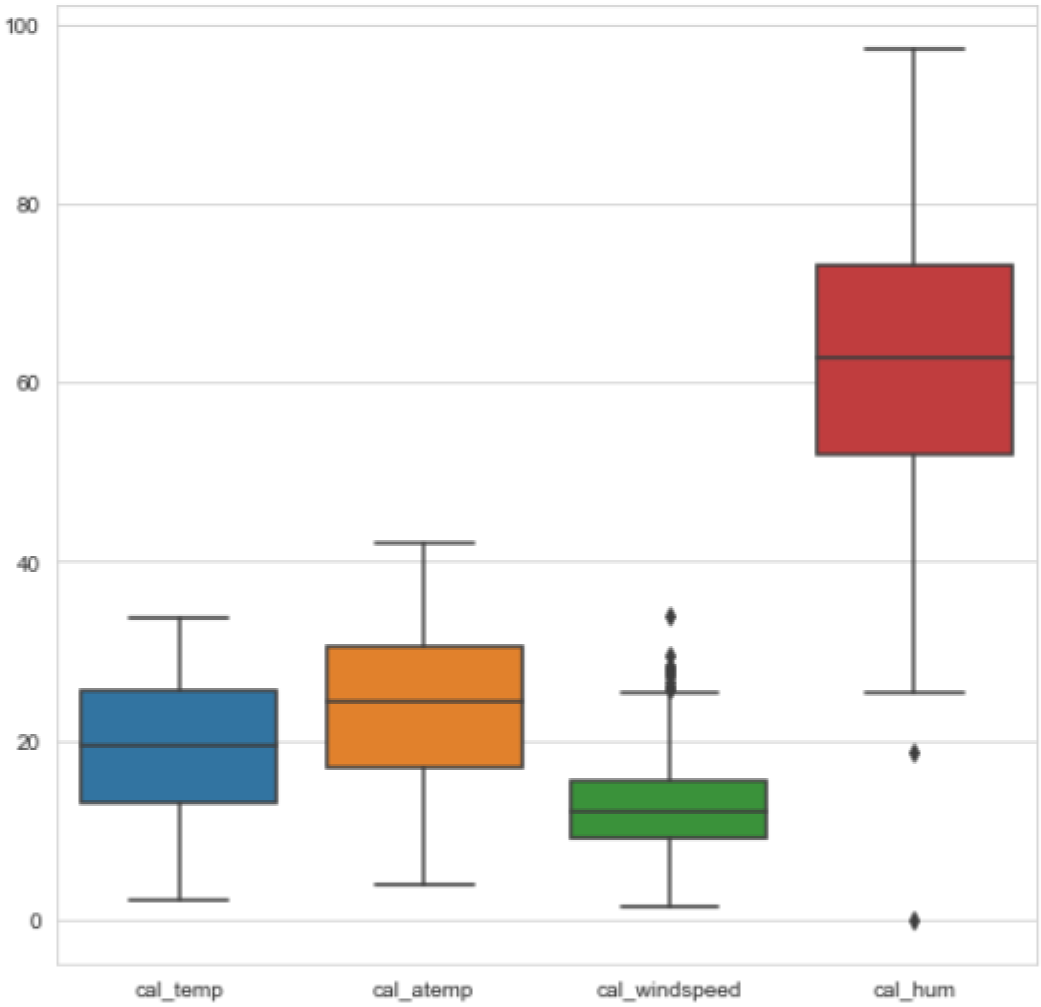


Fig 2.4 Boxplot of continuous variable

Outliers can be removed using the Boxplot stats method, wherein the Inter Quartile Range (IQR) is calculated and the minimum and maximum value are calculated for the variables. Any value ranging outside the minimum and maximum value are discarded. The boxplot of the continuous variables after removing the outliers is shown in the below figure:

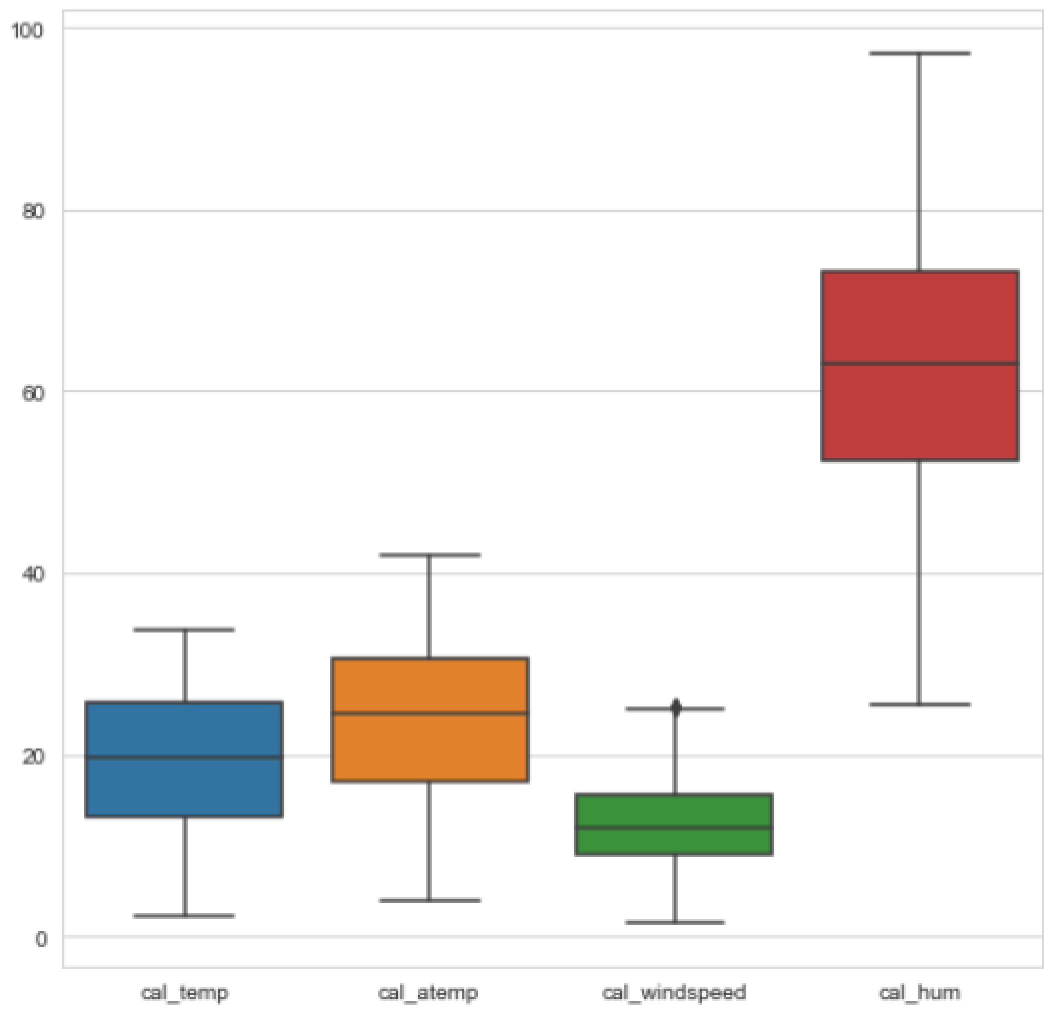
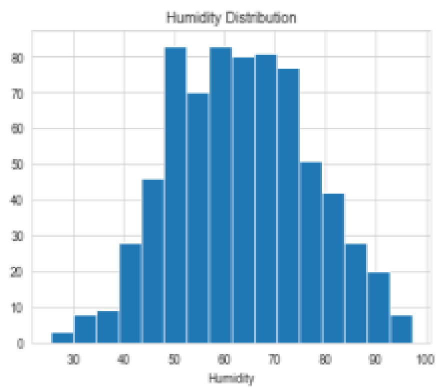
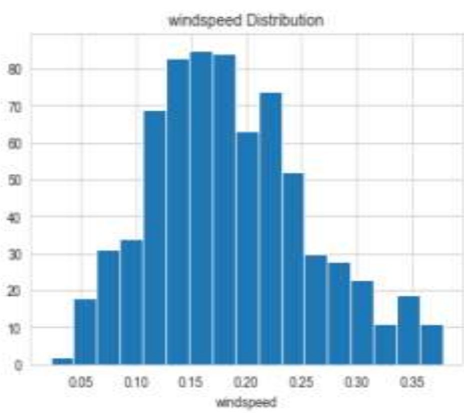


Fig 2.5 Boxplot of continuous variables after removal of outliers

It can be observed from the distribution of Windspeed and humidity after removal of outliers, is that data is not skewed as much as before the removal of outliers. The figure shown below illustrates the distribution of continuous variables using histograms.





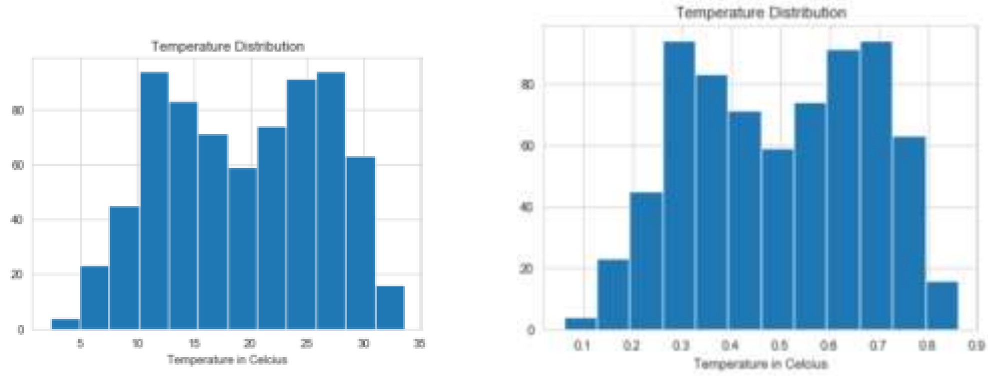


Fig 2.6 Distribution of numerical data using histograms after removal of outliers

2.6: Feature Selection

Feature Selection reduces the complexity of a model and makes it easier to interpret. It also reduces over fitting. Features are selected based on their scores in various statistical tests for their correlation with the outcome variable. Correlation plot is used to find out if there is any multi co-linearity between variables. The highly collinear variables are dropped and then the model is executed.

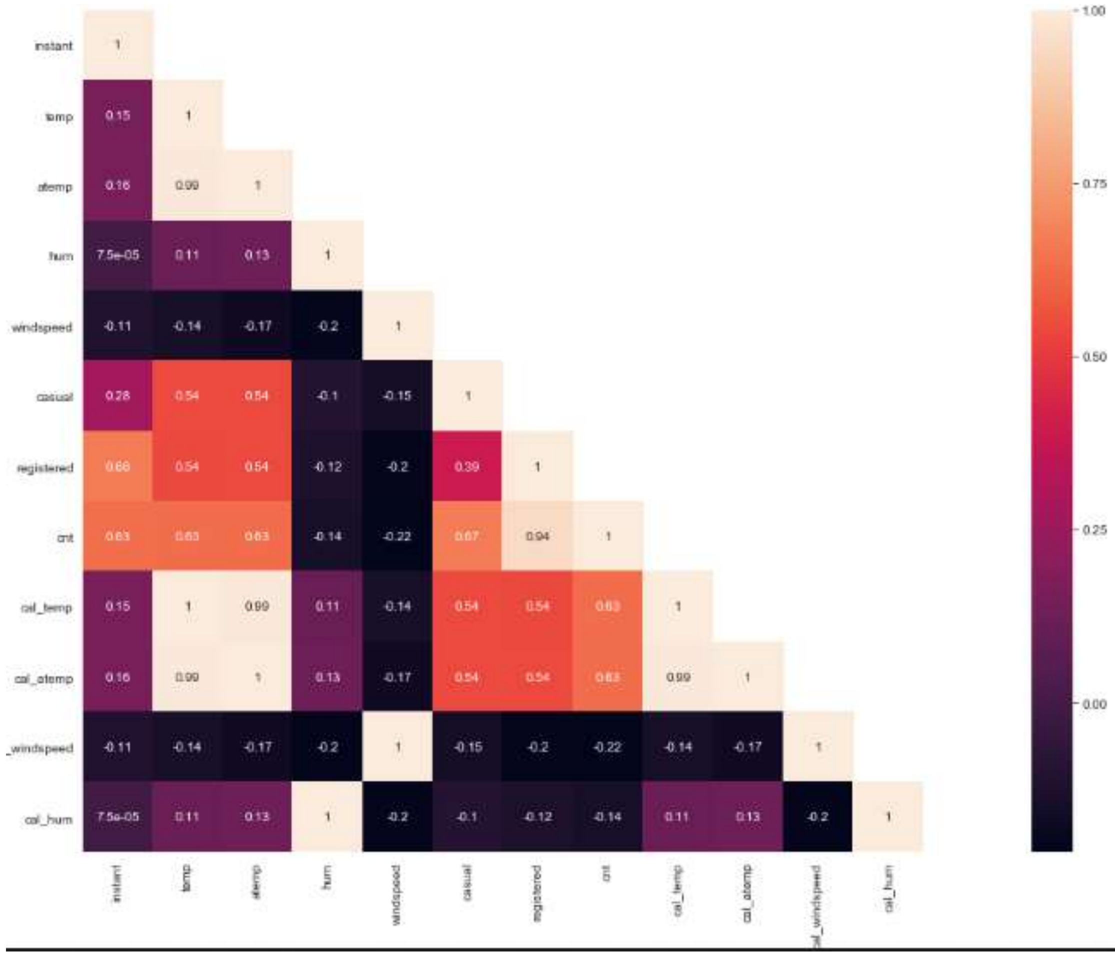


Fig 2.7 Correlation plot of all the variables

3. Modelling

3.1 Model Selection

The dependent variable in our model is a continuous variable i.e., Count of bike rentals. Hence the models that we choose are Linear Regression, Decision Tree and Random Forest. The error metric chosen for the problem statement is Mean Absolute Error (MAE).

3.2 Multiple Linear Regressions

Multiple linear regressions are the most common form of linear regression analysis. Multiple linear regressions are used to explain the relationship between one continuous dependent variable and two or more independent variables. The independent variables can be continuous or categorical.

OLS Regression Results

Dep. Variable:	cnt	R-squared:	0.967
Model:	OLS	Adj. R-squared:	0.966
Method:	Least Squares	F-statistic:	1601.
Date:	Tue, 28 Jan 2020	Prob (F-statistic):	0.00
Time:	15:47:07	Log-Likelihood:	-4115.3
No. Observations:	501	AIC:	8249.
Df Residuals:	492	BIC:	8287.
Df Model:	9		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
season	509.3154	68.345	7.452	0.000	375.031	643.600

OLS Regression Results

Dep. Variable:	cnt	R-squared:	0.858
Model:	OLS	Adj. R-squared:	0.850
Method:	Least Squares	F-statistic:	105.5
Date:	Tue, 28 Jan 2020	Prob (F-statistic):	3.49e-181
Time:	15:48:07	Log-Likelihood:	-4023.3
No. Observations:	501	AIC:	8103.
Df Residuals:	473	BIC:	8221.
Df Model:	27		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
temp	4787.8068	490.404	9.763	0.000	3824.167	5751.447

#MAPE:16.38%

#Accuracy: 83.62%

#Adjusted r Squared: 0.850

#F-stat: 105.5

As you can see the Adjusted R-squared value 0.850, we can explain 83.62% of the data using our multiple linear regression model. By looking at the F-statistic and combined p-value we can reject the null hypothesis that target variable does not depend on any of the predictor variables. This model explains the data very well and is considered to be good.

Even after removing the non-significant variables, the accuracy, Adjusted R-squared and F-statistic do not change by much; hence the accuracy of this model is chosen to be final. MAPE of this multiple linear regression model is 16.38%. Hence the accuracy of this model is 83.62%. This model performs very well for this test data.

3.3 Decision Tree:

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. Using decision tree; we can predict the value of bike count. MAE for this model is 684. The MAPE for this decision tree is 17.77%. Hence the accuracy for this model is 82.23%.

#MAPE: 17.77%

#Accuracy: 82.23%

3.4 Random Forest:

Using Classification for prediction analysis in this case is not normal, though it can be done. The number of decision trees used for prediction in the forest is 500. Using random forest, the MAPE was found to be 13.57%. Hence the accuracy is 86.43%.

#MAPE: 13.57%

#Accuracy:86.43%

4. Conclusion

Now that we have a few models for predicting the target variable, we need to decide which one to choose. There are several criteria that exist for evaluating and comparing models. We can compare the models using any of the following criteria:
1. Predictive Performance
2. Interpretability
3. Computational Efficiency
In our case of Bike count prediction Data, 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 Predictions of the models with real values of the target variables, and calculating some average error measure.

4.1 Mean Absolute Error (MAE)

MAE is one of the error measures used to calculate the predictive performance of the model. We will apply this measure to our models that we have generated in the previous section.
MAE <- function (actual, pred) { print(mean (abs (actual - pred))) }
Linear Regression Model: MAE 597

Decision Tree: MAE 688

Random Forest: MAE 508

Based on the above error metrics, Random Forest is the better model for our analysis. Hence Random Forest is chosen as the model for prediction of bike rental count.

5: Appendix

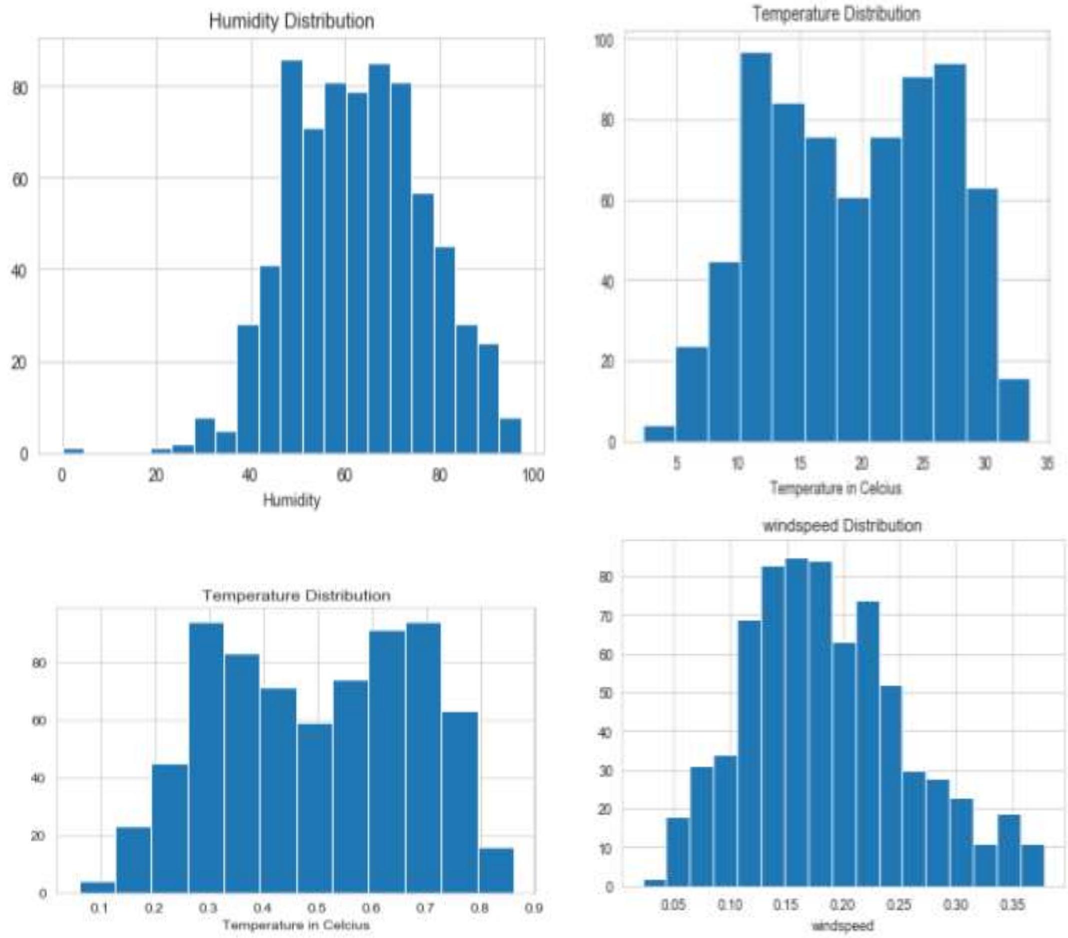
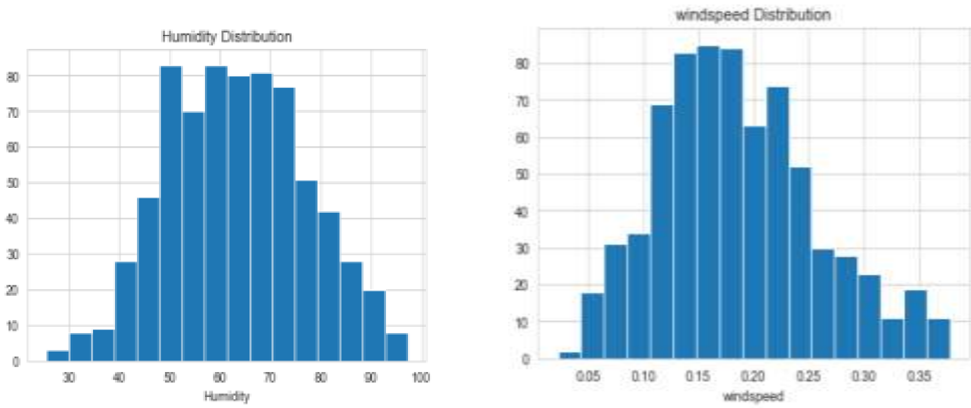


Fig2.1 Distribution of continuous variable using histograms



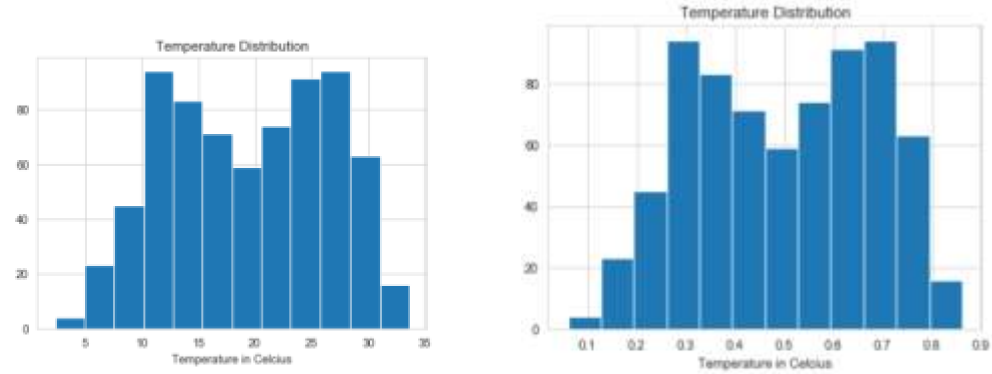
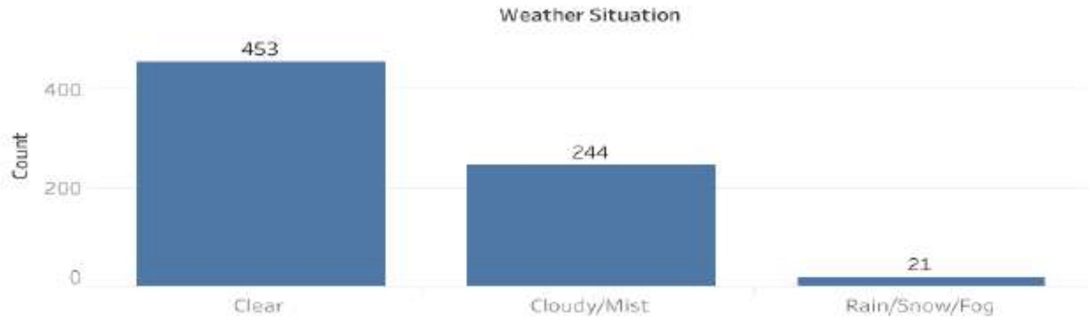
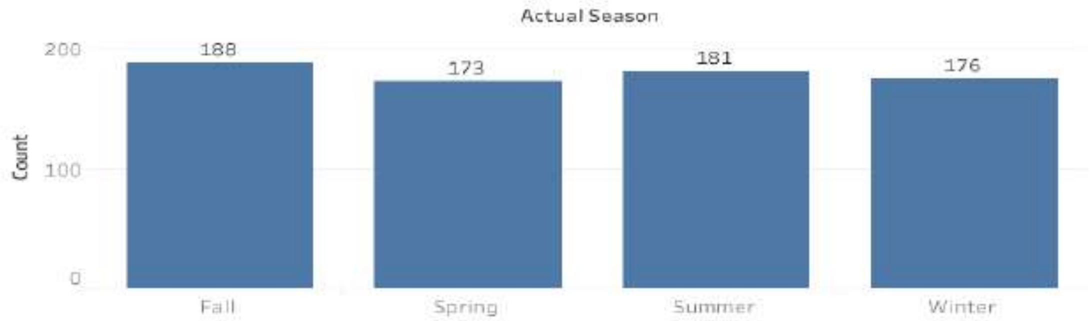


Fig 2.6 Distribution of numerical data using histograms after removal of outliers

Bar weather



Bar Season



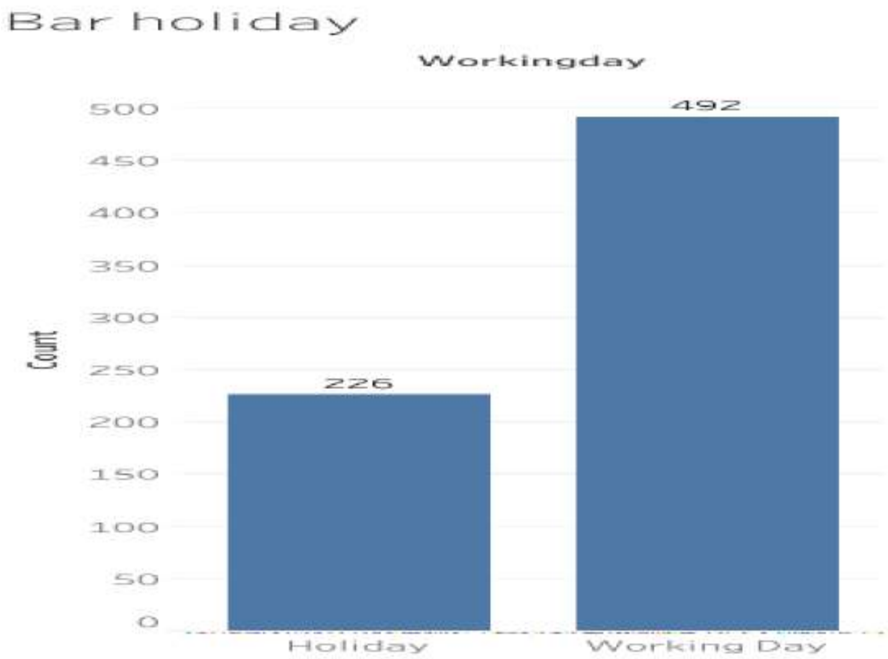


Fig 2.2 Distribution of categorical variable using bar plots

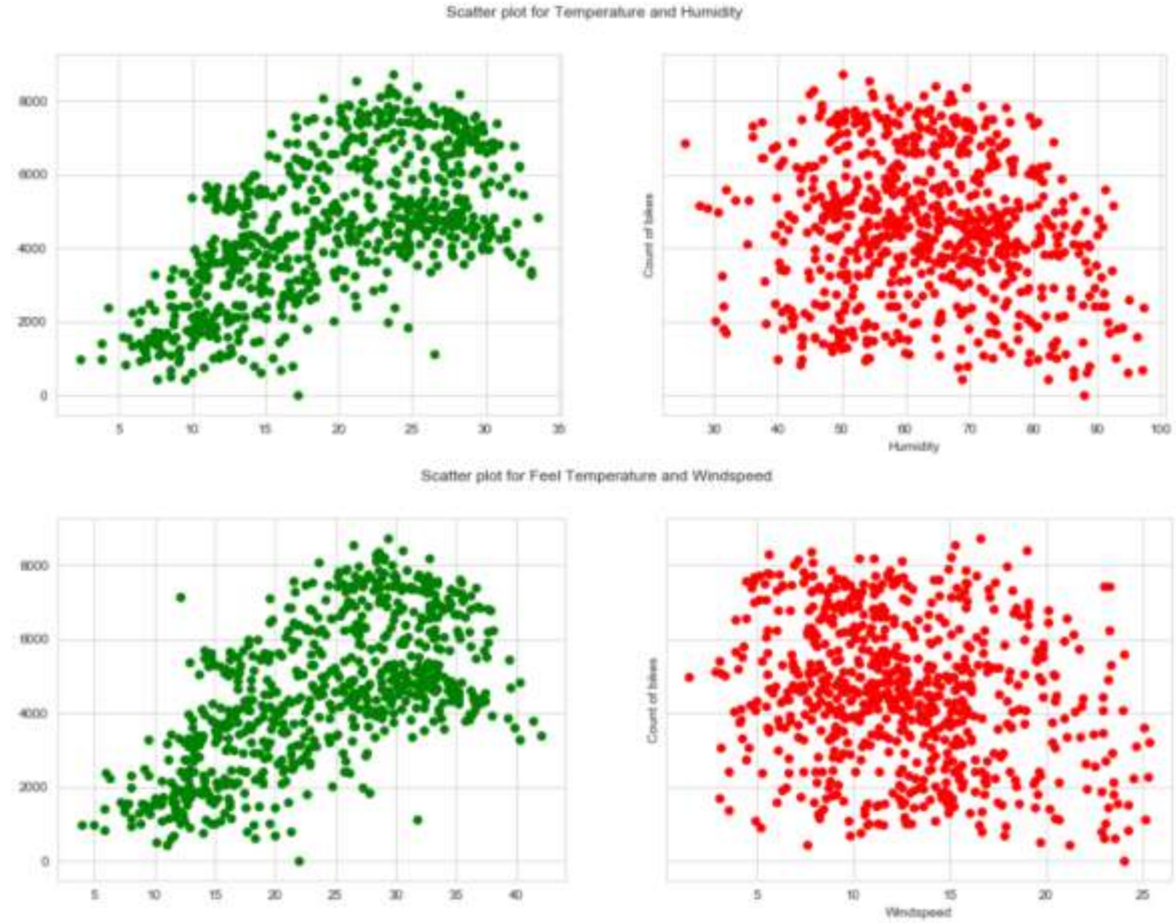


Fig 2.3 Scatter plot for continuous variable

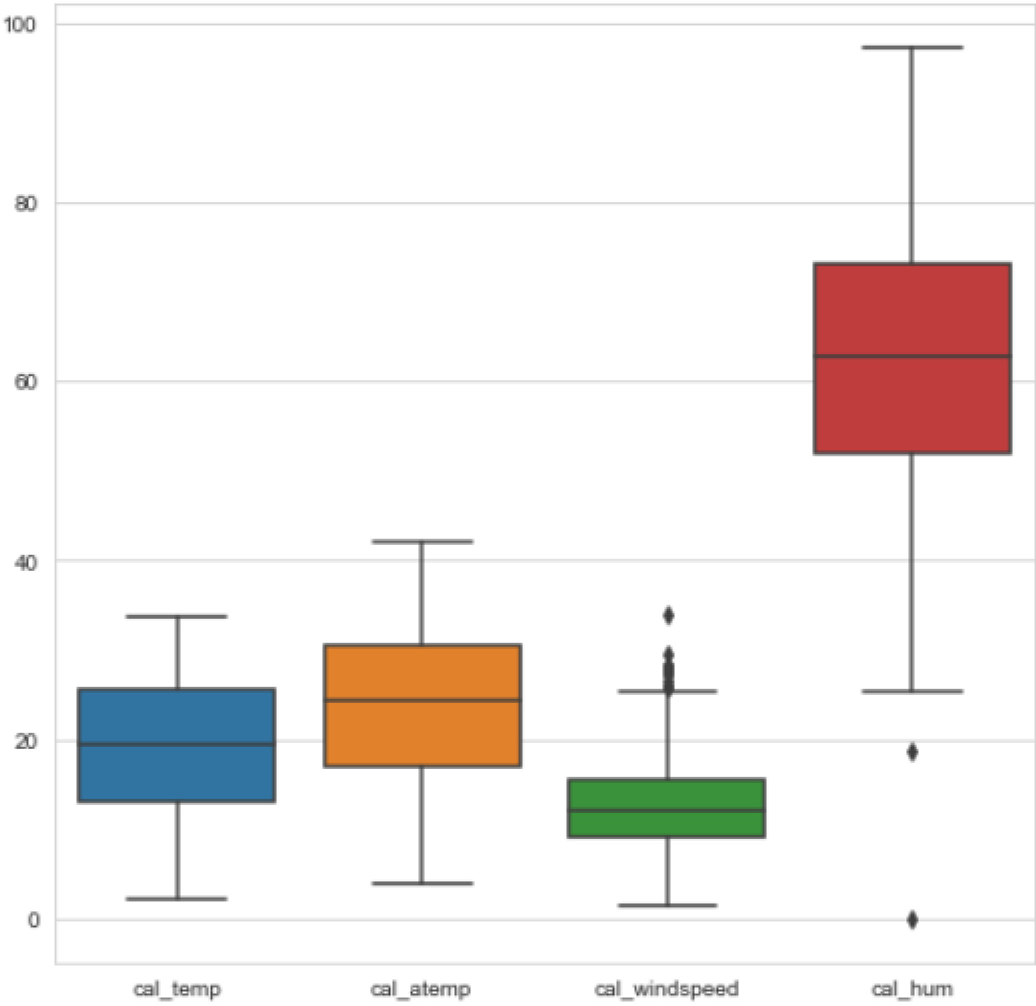


Fig 2.4 Boxplot of continuous variable

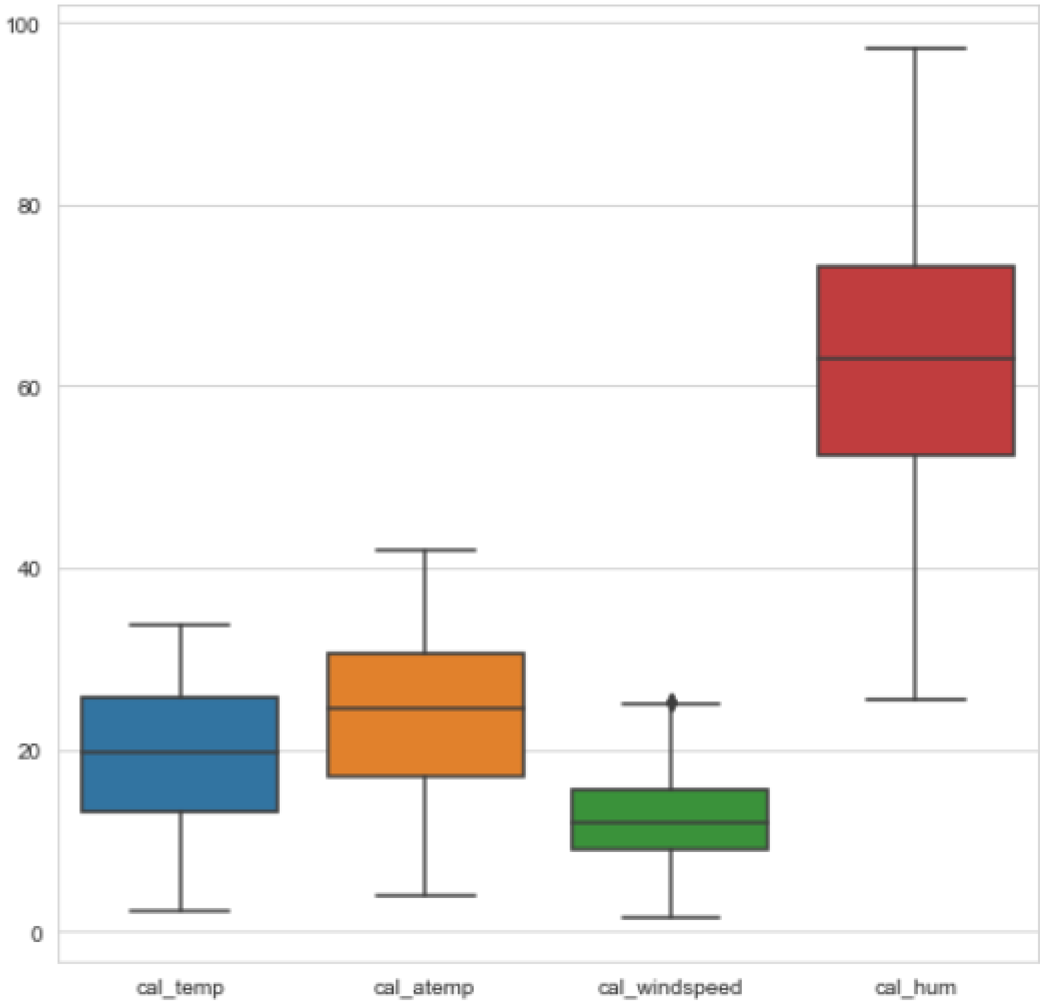


Fig 2.5 Boxplot of continuous variables after removal of outliers

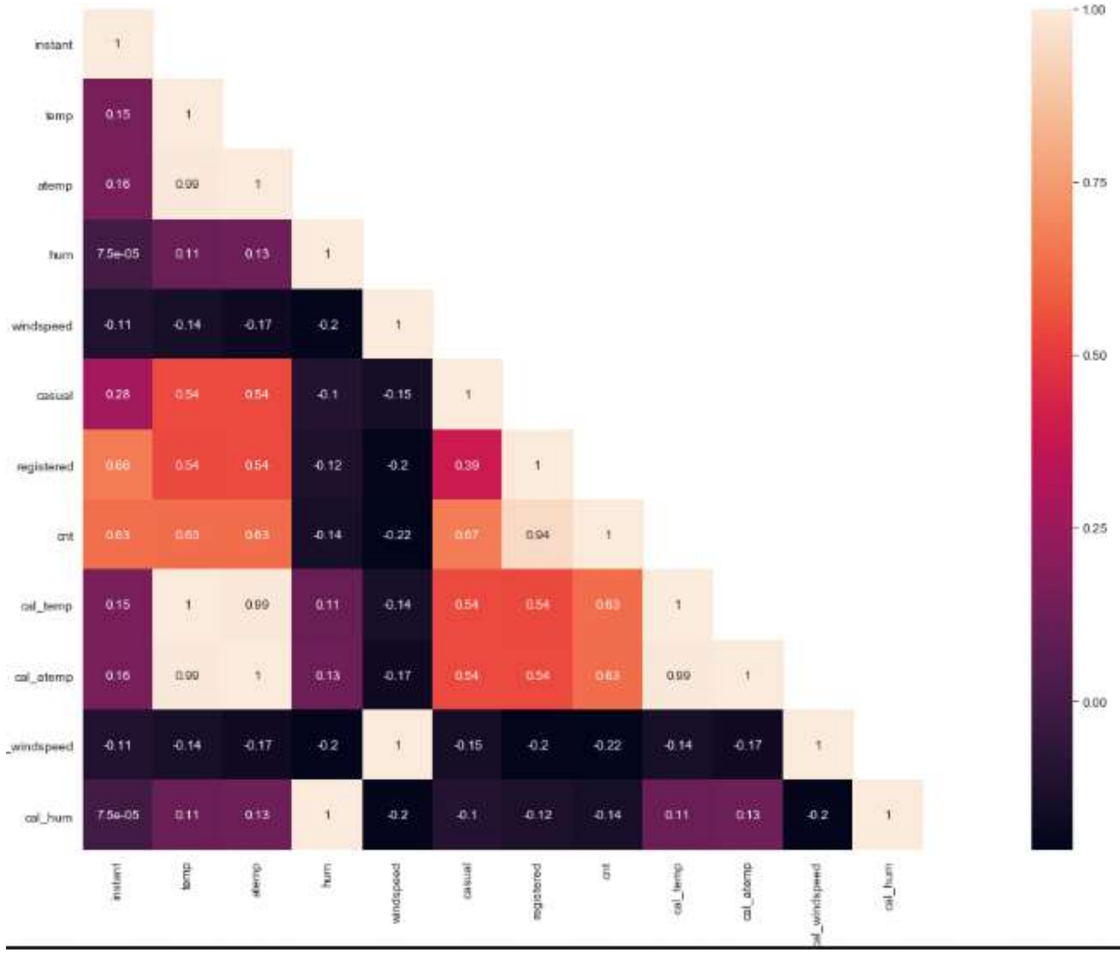


Fig 2.7 Correlation plot of all the variables

6. R code

```
#Clean the environment

rm(list = ls())


#Set working directory

setwd("C:/Users/Divyanshu/Desktop/Data Science_Edvisor")


#Load the librarires

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

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

rm(libraries)


#Read the csv file

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


########################################EXPLORE THE DATA#################################

#First few rows

head(D)


#Dimensions of data

dim(D)


#Column names

names(D)


#Structure of variables

str(D)


########################################FEATURE ENGINEERING##############################
```

```
#Create columns

D$cal_temp <- D$temp*39

D$cal_atemp <- D$atemp*50

D$cal_windspeed <- D$windspeed*67

D$cal_hum = D$hum * 100


D$new_season = factor(x = D$season, levels = c(1,2,3,4), labels = c("Spring","Summer","Fall","Winter"))

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

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

D$new_weathersit = factor(x = D$weathersit, levels = c(1,2,3,4),

                    labels = c("Clear","Cloudy/Mist","Rain/Snow/Fog","Heavy Rain/Snow/Fog"))


D$weathersit = as.factor(D$weathersit)

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

D$dteday = as.character(D$dteday)

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

D$weekday = as.factor(as.character(D$weekday))

D$workingday = as.factor(as.character(D$workingday))

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

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


#########################################MISSING VALUES####################################

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



#########################################EXPLORE USING GRAPHS#############################

#Check the distribution of categorical Data using bar graph

bar1 = ggplot(data = D, aes(x = new_season)) + geom_bar() + ggtitle("Count of Season")

bar2 = ggplot(data = D, aes(x = new_weathersit)) + geom_bar() + ggtitle("Count of Weather")
```

```
bar3 = ggplot(data = D, aes(x = new_holiday)) + geom_bar() + ggtitle("Count of Holiday")

bar4 = ggplot(data = D, aes(x = workingday)) + geom_bar() + ggtitle("Count of Working day")

# ## Plotting plots together

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


#Check the distribution of numerical data using histogram

hist1 = ggplot(data = D, aes(x =cal_temp)) + ggtitle("Distribution of Temperature") + geom_histogram(bins = 25)

hist2 = ggplot(data = D, aes(x =cal_hum)) + ggtitle("Distribution of Humidity") + geom_histogram(bins = 25)

hist3 = ggplot(data = D, aes(x =cal_atemp)) + ggtitle("Distribution of Feel Temperature") + geom_histogram(bins = 25)

hist4 = ggplot(data = D, aes(x =cal_windspeed)) + ggtitle("Distribution of Windspeed") + geom_histogram(bins = 25)

gridExtra::grid.arrange(hist1,hist2,hist3,hist4,ncol=2)


#Check the distribution of numerical data using scatterplot

scat1 = ggplot(data = D, aes(x =cal_temp, y = cnt)) + ggtitle("Distribution of Temperature") + geom_point() + xlab("Temperature") + ylab("Bike COunt")

scat2 = ggplot(data = D, aes(x =cal_hum, y = cnt)) + ggtitle("Distribution of Humidity") + geom_point(color="red") + xlab("Humidity") + ylab("Bike COunt")

scat3 = ggplot(data = D, aes(x =cal_atemp, y = cnt)) + ggtitle("Distribution of Feel Temperature") + geom_point() + xlab("Feel Temperature") + ylab("Bike COunt")

scat4 = ggplot(data = D, aes(x =cal_windspeed, y = cnt)) + ggtitle("Distribution of Windspeed") + geom_point(color="red") + xlab("Windspeed") + ylab("Bike COunt")

gridExtra::grid.arrange(scat1,scat2,scat3,scat4,ncol=2)


#Check for outliers in data using boxplot

cnames = colnames(D[,c("cal_temp","cal_atemp","cal_windspeed","cal_hum")])

for (i in 1:length(cnames))

{

  assign(paste0("gn",i), ggplot(aes_string(y = cnames[i]), data = D)+

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

```
        geom_boxplot(outlier.colour="red", fill = "green" ,outlier.shape=20,
                outlier.size=1, notch=FALSE) +
        theme(legend.position="bottom")+
        labs(y=cnames[i])+
        ggtitle(paste("Box plot for",cnames[i])))
}
gridExtra::grid.arrange(gn1,gn3,gn2,gn4,ncol=2)

#Remove outliers in Windspeed
val = D[,19][D[,19] %in% boxplot.stats(D[,19])$out]
D = D[which(!D[,19] %in% val),]

#Check for multicollinearity using VIF
df = D[,c("instant","temp","atemp","hum","windspeed")]
vifcor(df)

#Check for collinearity using corelation graph
corrgram(D, order = F, upper.panel=panel.pie, text.panel=panel.txt, main = "Correlation Plot")

#Remove the unwanted variables
D <- subset(D, select = -c(holiday,instant,dteday,atemp,casual,registered,cal_temp,cal_atemp,cal_windspeed,
                    cal_hum,new_season,new_yr,new_holiday,new_weathersit))

rmExcept(keepers = "D")
########################################DECISION TREE#####################################
#MAPE: 24.63%
#MAE: 688
#RMSE: 936.3
#Accuracy: 75.37%
```

```
#Divide the data into train and test

set.seed(123)

train_index = sample(1:nrow(D), 0.7 * nrow(D))

train = D[train_index,]

test = D[-train_index,]

#rpart for regression

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

#Predict the test cases

d_predictions = predict(d_model, test[,-10])

#Create dataframe for actual and predicted values

df = data.frame("actual"=test[,10], "pred"=d_predictions)

head(df)

#calculate MAPE

regr.eval(trues = test[,10], preds = d_predictions, stats = c("mae","mse","rmse","mape"))

#calculate MAPE

MAPE = function(actual, pred){

  print(mean(abs((actual - pred)/actual)) * 100)

}

MAPE(test[,10], d_predictions)


########################################RANDOM FOREST##################################

#MAPE: 19.14%

#MAE: 508

#RMSE: 718

#Accuracy: 80.86%

#Train the data using random forest

r_model = randomForest(cnt~., data = train, ntree = 500)

#Predict the test cases

r_predictions = predict(r_model, test[,-10])
```

```
#Create dataframe for actual and predicted values

df = cbind(df,r_predictions)

head(df)

#Calculate MAPE

regr.eval(trues = test[,10], preds = r_predictions, stats = c("mae","mse","rmse","mape"))

MAPE(test[,10], r_predictions)

########################################LINEAR REGRESSION############################

#MAPE: 19.87%

#RMSE: 825

#Accuracy: 80.13%

#MAE: 597

#Adjusted R squared: 0.8398

#F-statistic: 100

#Train the data using linear regression

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

#Check the summary of the model

summary(l_model)

#Predict the test cases

l_predictions = predict(l_model, test[,-10])

#Create dataframe for actual and predicted values

df = cbind(df,l_predictions)

head(df)

#Calculate MAPE

regr.eval(trues = test[,10], preds = l_predictions, stats = c("mae","mse","rmse","mape"))

MAPE(test[,10], lr_predictions)

#Plot a graph for actual vs predicted values

plot(test$cnt,type="l",lty=2,col="green")

lines(l_predictions,col="red")

#Predict a sample data
```

```
predict(l_model,test[2,])
```

7. Python code

```
#Import libraries

import os

import pandas as pd

import numpy as np


#import libraries for plots

import matplotlib.pyplot as plt

import seaborn as sns

%matplotlib inline

#Set working directory

os.chdir("C:/Users/Divyanshu/Desktop/Data Science_Edvisor")

print(os.getcwd())

#Read the csv file

day = pd.read_csv("day.csv", sep=",")

#Get the number of rows and columns

day.shape

#Get first 5 rows

day.head()

#Get the data types of variables

day.dtypes

#Create a new dataframe by copying the dataset in new dataset

df = day.copy()

#Create new columns with new calculated fields

df['cal_temp'] = day['temp'] * 39
```

```
df['cal_atemp'] = day['atemp'] * 50

df['cal_windspeed'] = day['windspeed'] * 67

df['cal_hum'] = day['hum'] * 100

#replcaing the catagorical data with proper catagorical name for processing and creating a new varibale

df['new_season'] = day['season'].replace([1,2,3,4],["Spring","Summer","Fall","Winter"])

df['new_yr'] = day['yr'].replace([0,1],["2011","2012"])

df['new_holiday'] = day['holiday'].replace([0,1],["Working day","Holiday"])

df['new_weathersit'] = day['weathersit'].replace([1,2,3,4],["Clear","Cloudy/Mist","Rain/Snow/Fog","Heavy Rain/Snow/Fog"])

#Check the data types and variables

df.dtypes

#Change the data types

df['weathersit'] = df['weathersit'].astype('category')

df['holiday'] = df['holiday'].astype('category')

df['yr'] = df['yr'].astype('category')

df['season'] = df['season'].astype('category')

df['workingday'] = df['workingday'].astype('category')

df['weekday'] = df['weekday'].astype('category')

df['mnth'] = df['mnth'].astype('category')

df['new_season'] = df['new_season'].astype('category')

df['new_yr'] = df['new_yr'].astype('category')

df['new_holiday'] = df['new_holiday'].astype('category')

df['new_weathersit'] = df['new_weathersit'].astype('category')

#Check the count of values of categorical variables

print(df.workingday.value_counts())

print(df.weekday.value_counts())

print(df.mnth.value_counts())
```

```
print(df.new_yr.value_counts())

print(df.new_holiday.value_counts())

print(df.new_weathersit.value_counts())

#Check if there are missing values

df.isnull().sum()

#Check the bar graph of categorical Data using factorplot

sns.set_style("whitegrid")

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

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

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

#Check the distribution of numerical data using histogram

plt.hist(data=df, x='cal_temp', bins='auto', label='Temperature')

plt.xlabel('Temperature in Celcius')

plt.title("Temperature Distribution")

#Check the distribution of numerical data using histogram

plt.hist(data=df, x='cal_hum', bins='auto', label='Temperature')

plt.xlabel('Humidity')

plt.title("Humidity Distribution")

#Check for outliers in data using boxplot

sns.boxplot(data=df[['cal_temp','cal_atemp','cal_windspeed','cal_hum']])

fig=plt.gcf()

fig.set_size_inches(8,8)

#Remove outliers in Humidity

q75, q25 = np.percentile(df['cal_hum'], [75 ,25])

print(q75,q25)

iqr = q75 - q25

print(iqr)
```

```
min = q25 - (iqr*1.5)

max = q75 + (iqr*1.5)

print(min)

print(max)


df = df.drop(df[df.iloc[:,19] < min].index)

df = df.drop(df[df.iloc[:,19] > max].index)

#Remove outliers in Windspeed

q75, q25 = np.percentile(df['cal_windspeed'], [75 ,25])

print(q75,q25)

iqr = q75 - q25

print(iqr)

min = q25 - (iqr*1.5)

max = q75 + (iqr*1.5)

print(min)

print(max)


df = df.drop(df[df.iloc[:,18] < min].index)

df = df.drop(df[df.iloc[:,18] > max].index)

sns.boxplot(data=df[['cal_temp','cal_atemp','cal_windspeed','cal_hum']])

fig=plt.gcf()

fig.set_size_inches(8,8)

#Check for collinearity using corelation matrix.

cor_mat= df[:].corr()

mask = np.array(cor_mat)

mask[np.tril_indices_from(mask)] = False

fig=plt.gcf()
```

```
fig.set_size_inches(30,12)

sns.heatmap(data=cor_mat,mask=mask,square=True,annot=True,cbar=True)

#Check the distribution of Temperature and Humdity against Bike rental count using scatter plot

fig, axs = plt.subplots(1,2, figsize=(15, 5), sharey=True)

axs[0].scatter(data=df, x='cal_temp', y='cnt', color = 'green')

axs[1].scatter(data=df, x='cal_hum', y='cnt', color = 'red')

fig.suptitle('Scatter plot for Temperature and Humidity')

plt.xlabel("Humidity")

plt.ylabel("Count of bikes")

#Check the distribution of Feel Temperature and Windspeed against Bike rental count using scatter plot

fig, axs = plt.subplots(1,2, figsize=(15, 5), sharey=True)

axs[0].scatter(data=df, x='cal_atemp', y='cnt', color = 'green')

axs[1].scatter(data=df, x='cal_windspeed', y='cnt', color = 'red')

fig.suptitle('Scatter plot for Feel Temperature and Windspeed')

plt.xlabel("Windspeed")

plt.ylabel("Count of bikes")

df =
df.drop(columns=['holiday','instant','dteday','atemp','casual','registered','cal_temp','cal_atemp',

'cal_windspeed','cal_hum','new_season','new_yr','new_holiday','new_weathersit'])

#Import Libraries for decision tree

from sklearn.model_selection import train_test_split

from sklearn.tree import DecisionTreeRegressor

#Divide data into train and test

train,test = train_test_split(df, test_size = 0.3, random_state = 123)

#Train the model

dt_model = DecisionTreeRegressor(random_state=123).fit(train.iloc[:,0:9], train.iloc[:,9])
```

```
#Predict the results of test data
dt_predictions = dt_model.predict(test.iloc[:,0:9])
df_dt = pd.DataFrame({'actual': test.iloc[:,9], 'pred': dt_predictions})
df_dt.head()
#Function for Mean Absolute Percentage Error
def MAPE(y_actual,y_pred):
    mape = np.mean(np.abs((y_actual - y_pred)/y_actual))
    return mape
#Calculate MAPE for decision tree
MAPE(test.iloc[:,9],dt_predictions)
#MAPE: 17.77%
#Accuracy: 82.23%
#Import library for RandomForestRegressor
from sklearn.ensemble import RandomForestRegressor
#Train the model
rf_model =
RandomForestRegressor(n_estimators=500,random_state=123).fit(train.iloc[:,0:9], train.iloc[:,9])
#Predict the results of test data
rf_predictions = rf_model.predict(test.iloc[:,0:9])
#Create a dataframe for actual values and predicted values
df_rf = pd.DataFrame({'actual': test.iloc[:,9], 'pred': rf_predictions})
df_rf.head()
#Calculate MAPE
MAPE(test.iloc[:,9],rf_predictions)
#MAPE: 13.57%
#Accuracy:86.43%
#import libraries for Linear regression
import statsmodels.api as sm
```

```
from sklearn.metrics import mean_squared_error

#Train the model

lr_model = sm.OLS(train.iloc[:,9].astype(float), train.iloc[:,0:9].astype(float)).fit()

#Check the summary of model

lr_model.summary()

#Predict the results of test data

lr_predictions = lr_model.predict(test.iloc[:,0:9])

##Create a dataframe for actual values and predicted values

df_lr = pd.DataFrame({'actual': test.iloc[:,9], 'pred': lr_predictions})

df_lr.head()

#Calclulate MAPE

MAPE(test.iloc[:,9],lr_predictions)

#MAPE:19.36%

#Accuracy: 80.64%

#Adjusted r Squared: 0.966

#F-stat: 1601

#Create continuous data. Save target variable first

train_lr = train[['cnt','temp','hum','windspeed']]

test_lr = test[['cnt','temp','hum','windspeed']]

##Create dummies for categorical variables

cat_names = ["season", "yr", "mnth", "weekday", "workingday", "weathersit"]


for i in cat_names:

   temp1 = pd.get_dummies(train[i], prefix = i)

   temp2 = pd.get_dummies(test[i], prefix = i)

   train_lr = train_lr.join(temp1)

   test_lr = test_lr.join(temp2)
```

```
#Train the model

lr_model = sm.OLS(train_lr.iloc[:,0].astype(float), train_lr.iloc[:,1:34].astype(float)).fit()

#summary of model

lr_model.summary()

#Predict the results of test data

lr_predictions = lr_model.predict(test_lr.iloc[:,1:34])

##Create a dataframe for actual values and predicted values

df_lr = pd.DataFrame({'actual': test_lr.iloc[:,0], 'pred': lr_predictions})

df_lr.head()

#Calclulate MAPE

MAPE(test_lr.iloc[:,0],lr_predictions)

#MAPE:16.38%

#Accuracy: 83.62%

#Adjusted r Squared: 0.850

#F-stat: 105.5
```