Bike Renting

Chandini C

December 2019

Contents

1 Introduction **2**

1.1 Problem Statement 2

1.2 Data 2

2 Methodology **4**

2.1 Pre Processing 4

2.1.1 Exploratory Data Analysis 4

2.1.2 Missing Value Analysis 7

2.1.3 Outlier Analysis 7

2.1.4 Feature Selection 8

2.1.5 Feature scaling 9

2.2 Modeling 9

2.2.1 Model Selection 9

2.2.2 Linear Regression 9

2.2.3 Decision Tree 12

2.2.4 Random Forest 12

3. Conclusion 14

3.1 Model Evaluation 14

3.2 Model Selection 14

Appendix A – Extra Figures **17**

Appendix B - Python Code **21**

Appendix C- R Code **29**

References **36**

Chapter 1

Introduction

1.1 Problem Statement

A bike sharing system is a service in which users can rent bicycles on a short term basis for a price. It has become very popular nowadays as they eased and automated the process of renting a bike from one location and returning it at different location according to the convenience of customers make it simple to use. At present, there are over 500 bike sharing systems across the world. The number of users or the number of bikes rented per day vary extremely. Our aim of this project is to predict the count of bike rented using historical data over environmental and seasonal settings, so that the entities can oversee these systems to manage the required bikes and arrange them in cost-effective way based on environmental and seasonal factors.

1.2 Data

Our task is to build regression models to predict the daily count of bike rented based on weather and season. A sample dataset given below is used to predict the count.

Table 1.2(a): 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
4	1/4/2011	1	0	1	0	2
5	1/5/2011	1	0	1	0	3

Table 1.2(b): 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
1	0.196364	0.189405	0.437273	0.248309	120	1229	1349
1	0.2	0.212122	0.590435	0.160296	108	1454	1562
1	0.226957	0.22927	0.436957	0.1869	82	1518	1600

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

1.instant: Record index

2.dteday: Date

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

4.yr: Year (0: 2011, 1:2012)

5.mnth: Month (1 to 12)

6.hr: Hour (0 to 23)

7.holiday: weather day is holiday or not (extracted from Holiday Schedule)

8.weekday: Day of the week

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

10.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

11.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)

12.atemp: Normalized feeling temperature in Celsius.

The values are derived via (t-t_min)/(t_max- t_min), t_min=-16, t_max=+50 (only in hourly scale)

13.hum: Normalized humidity.

The values are divided to 100 (max)

14.windspeed: Normalized wind speed. The values are divided to 67 (max)

15.casual: count of casual users

Chapter 2

METHODOLOGY

2.1 Data Pre-processing

Before starting any predictive analysis, it is important that we look at the model. In data analysis, looking at the data does not refer to just looking and it is more than that. Looking at the data here refers to exploring the data, visualizing and cleaning the data using plots or graphs. This process is called as Exploratory data analysis.

2.1.1 Exploratory Data Analysis

Exploratory data analysis is an approach to analysing the datasets to summarize their main characteristics with visual methods. Primarily, it is used for seeing what data can tell us before modeling. Before we start any modeling process it is important that the variables are converted to proper data types and remove unwanted variables.

i. Converted 'season', 'mnth', 'workingday', 'weathersit' into categorical variables.
ii. Changed 'dteday' into day of date and categorical variable having 31 levels as 31 days.
iii. Deleted instant variable as it is just an index. Also omitted registered and casual as its sum gives the total count ('cnt' variable) which has to be predicted.

Most of the regression models need the data to be normally distributed. Now, let us look at the distributions of numeric variables. In Figure 2.1.1(a) and 2.1.1(b) we have plotted the probability density functions numeric variables present in the data including target variable 'cnt'. Target variable cnt and independent variables like 'temp', 'atemp', looks normally distributed but Independent variable 'hum' data is slightly skewed to the left.

Figure 2.1.1(a) Distribution of target variable 'cnt' (python code in Appendix B)

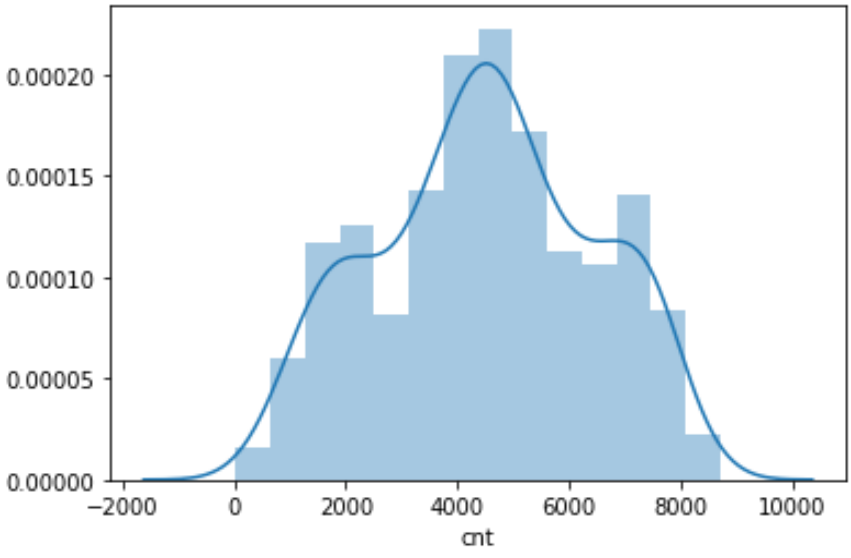
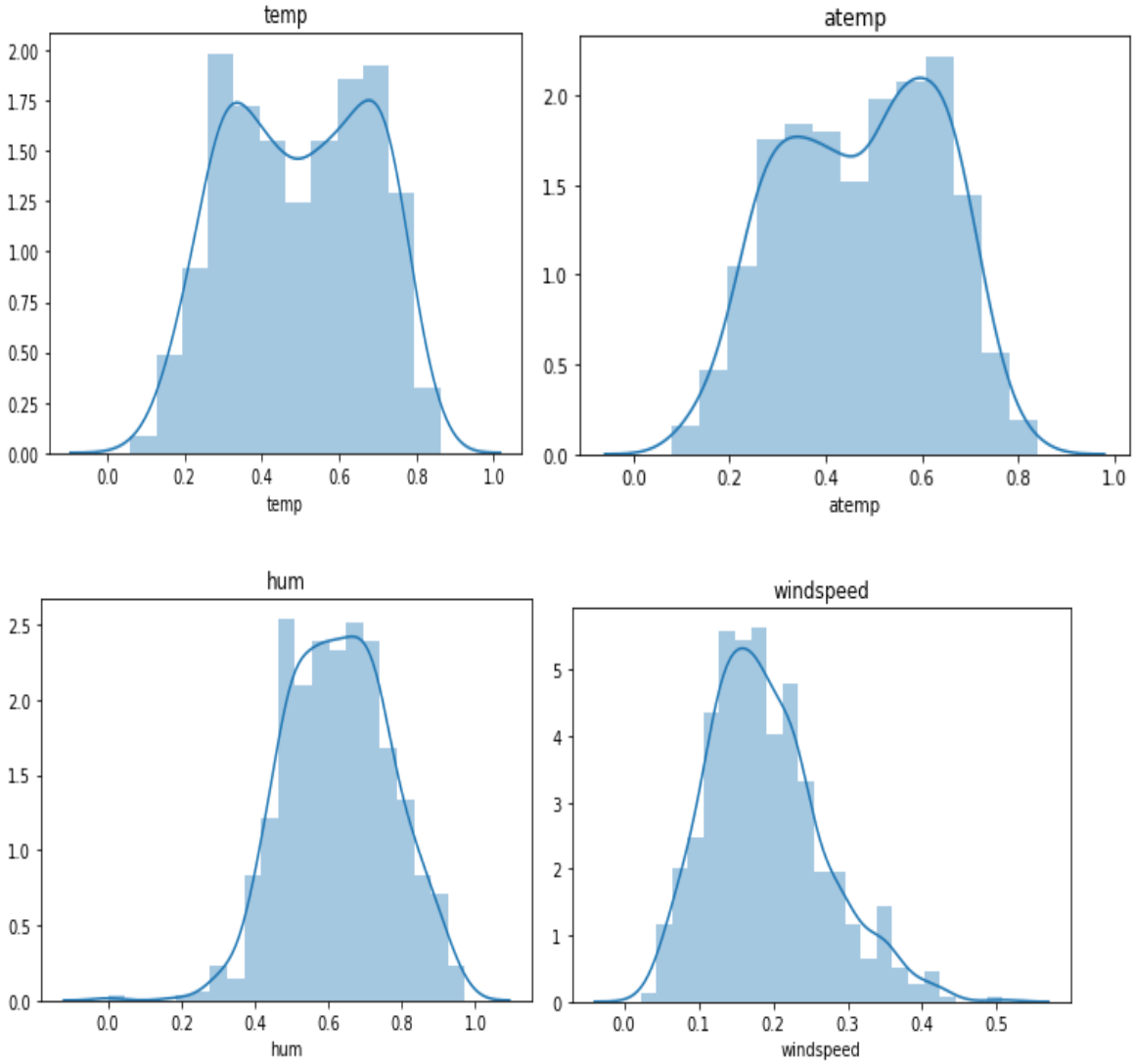


Figure 2.1.1(b) showing distribution of dependent variables (python code in Appendix B)



Now, let us look at the distribution of month and season with the target variable 'cnt'. Figure 2.1.1(c) and Figure 2.1.1(d) seasonal sales of total bike rented and the total count of bike w.r.t weather, holiday, working day and season.

Figure 2.1.1(c) distribution of sales based on seasons(python code in Appendix B)

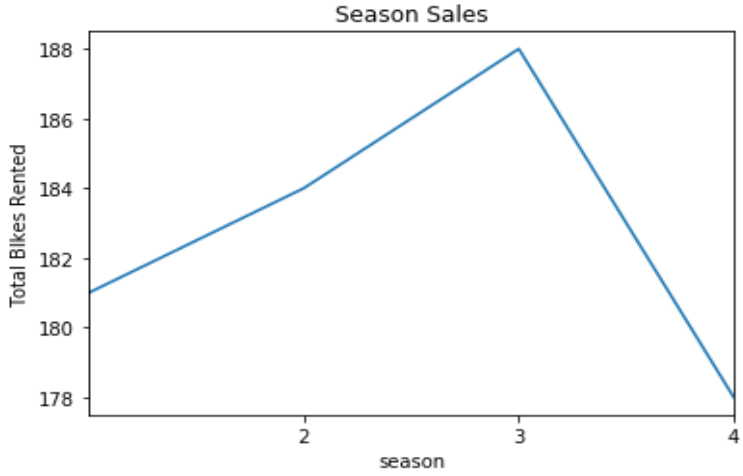
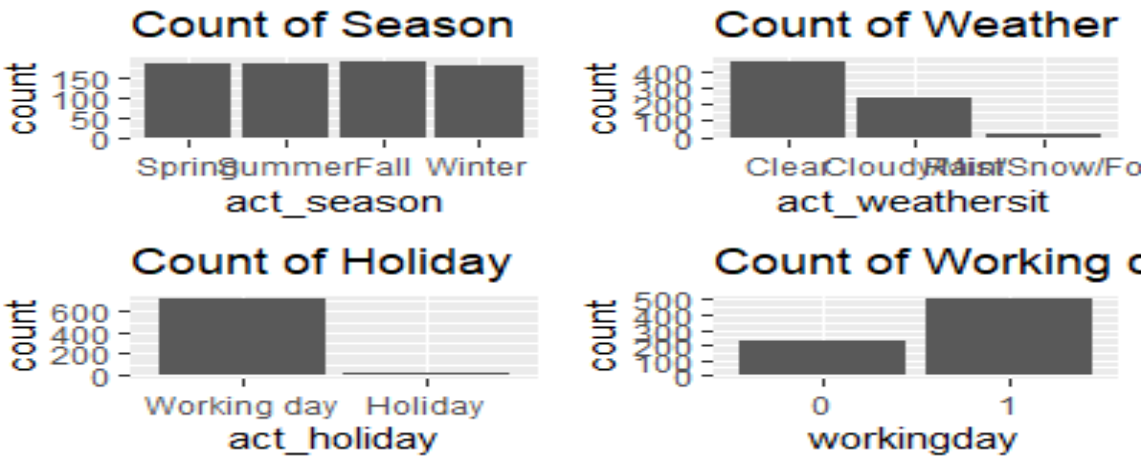


Figure 2.1.1(d) count of rental bike w.r.t season, weather,holiday and working day

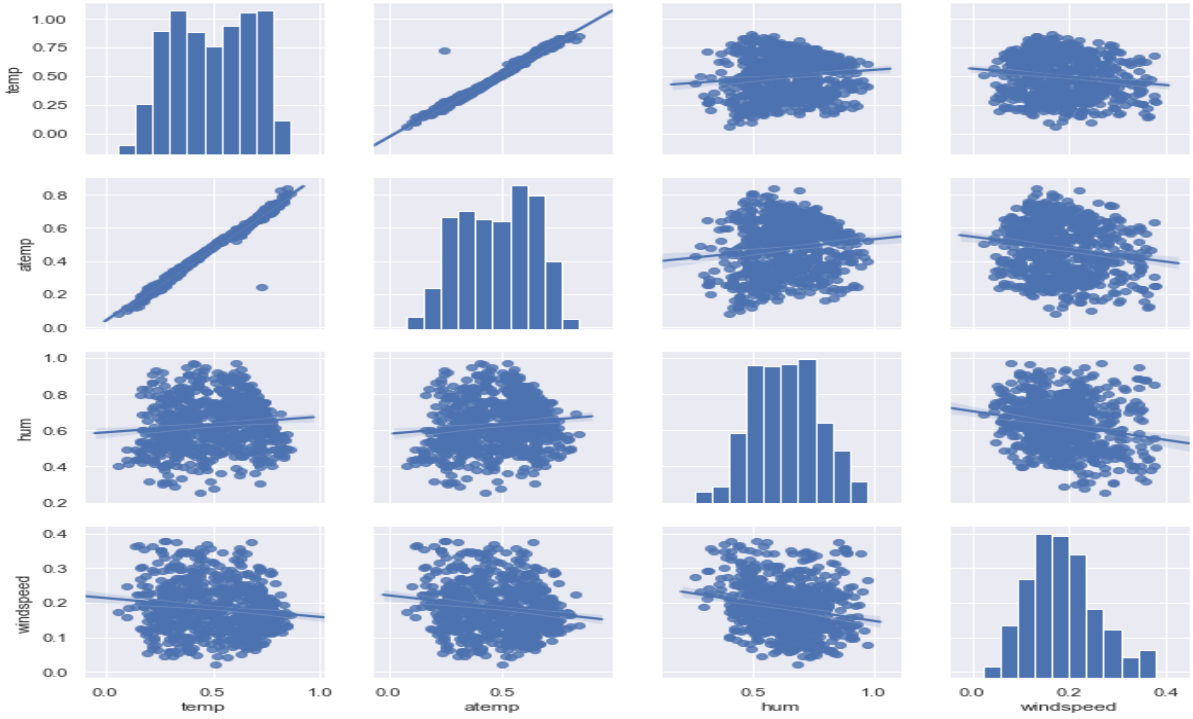


The above graph depicts that the sales are very less in spring and bike rental count is maximum in the fall. This tells that the total number of bikes rented is more during fall (i.e., September-November) and the weather plays an important role in bike rentals as users tend to rent bicycles during clear, few clouds, partly cloudy, partly cloudy weather. People hardly rent any bikes during rain, snow or thunderstorm.

By doing bivariate analysis can help us for quickly exploring the relationships between multiple columns of data in a data frame. The lower and upper arguments to the ggpairs function specifies the type of plot or data in each position of the lower or upper diagonal of the matrix, respectively. For continuous X and Y data, one can specify the smooth option to include a regression line.

Below figures(Figure 2.1.1(e)) illustrates relationship between numerical variables with target variable. Relationship between independent variables 'temp' and 'atemp' are very strong and relationship between 'hum' , 'windspeed' with target variable 'cnt' is less.

Figure 2.1.1(e) Relationship between numerical variables and target variable using bivariate analysis (python code in Appendix B)



2.1.2 Missing Value Analysis

Real world data consists of missing values. Hence, it is important that we do missing value analysis and perform required operations or treat them using mean,median, mode or knn method to impute missing values. Table 2.1.2 displays the count of missing values in the given dataset.

Table 2.1.2 Missing values in the data (python code in Appendix B)

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

There are no missing values in the given dataset.

2.1.3 Outlier Analysis

An outlier is a distanced observation from other observations. Outliers in data can distort predictions and affect the accuracy, if you don't detect and handle them appropriately. Outlier analysis can be done only on continuous variables. I chose boxplot analysis to check whether outliers are present in the data and by looking at the plot in Figure 2.1.3(a) it is clear that there are noisy data in 'windspeed' and 'hum' present in the data so we have to remove the outliers to get better model.

Figure 2.1.3(a) boxplot before outlier analysis in variable windspeed and hum (python code in Appendix B)

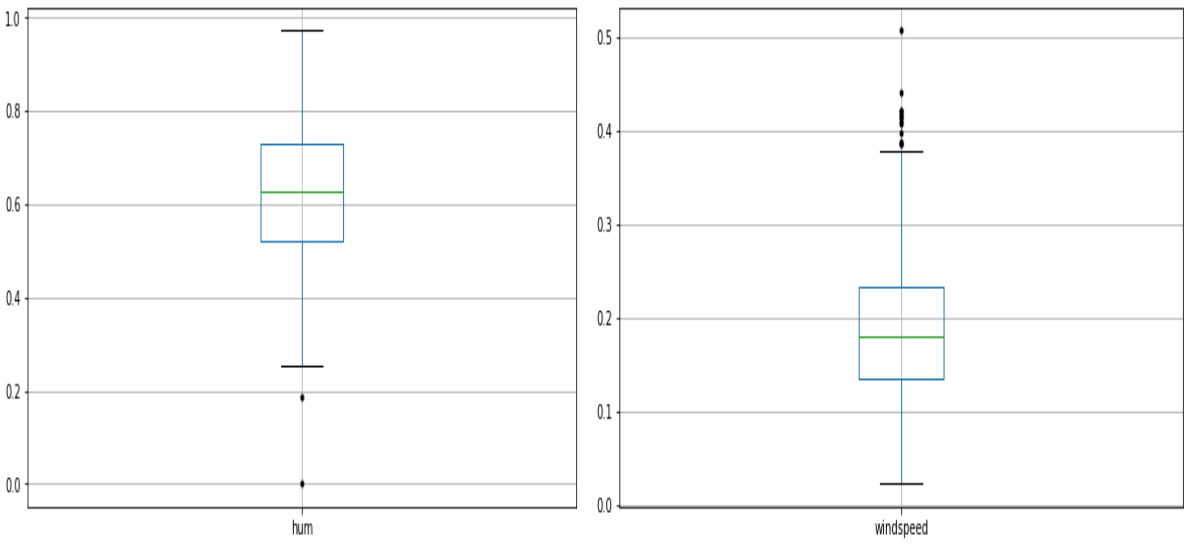
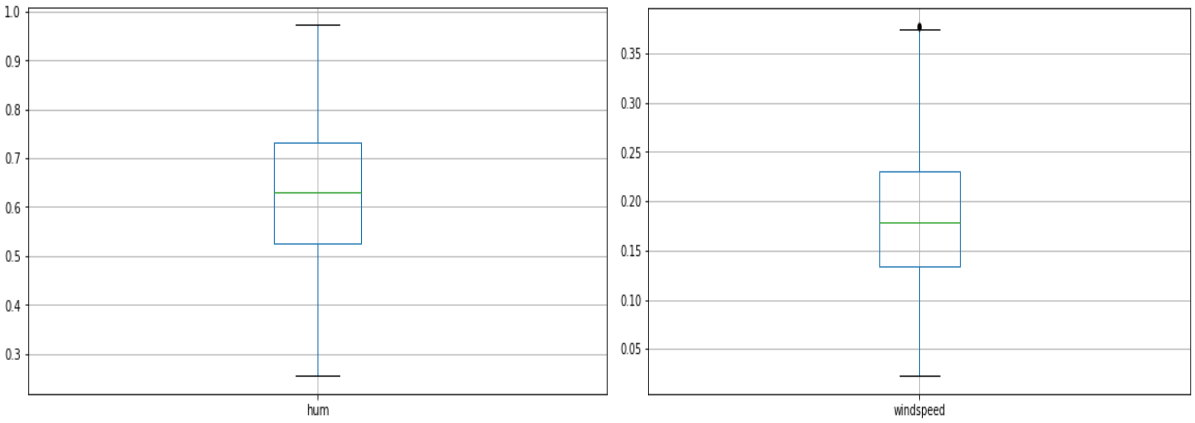


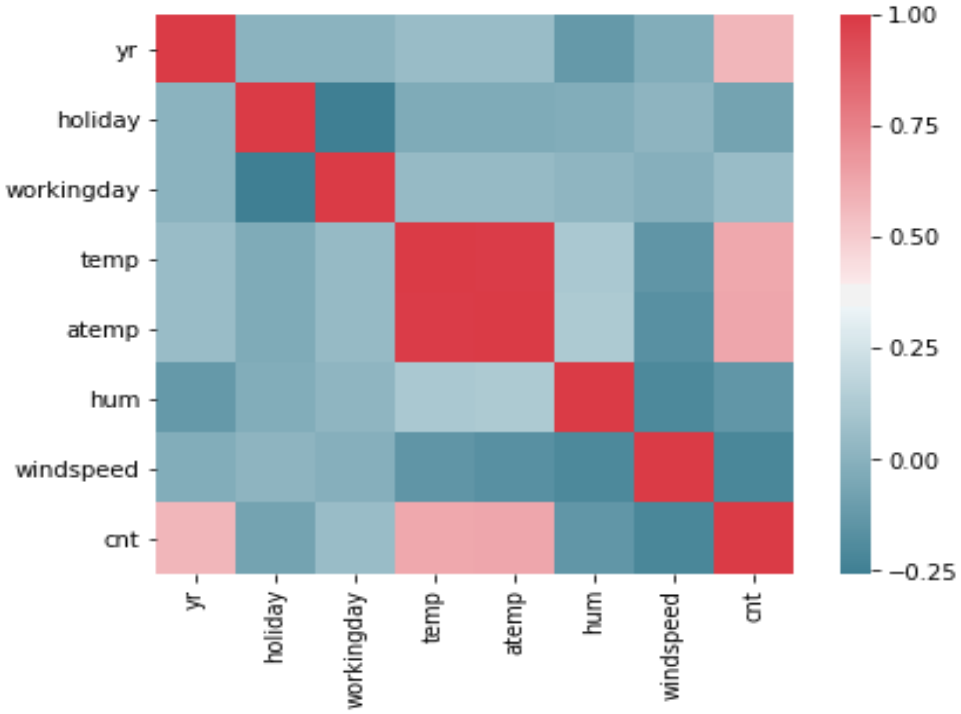
Figure 2.1.3(b) boxplot after outlier analysis in variable windspeed and hum (python code in Appendix B)



2.1.4 Feature Selection

Feature selection or attribute selection is a process of selecting a subset of relevant features (variable predictors) for use in model construction. As we are dealing with numeric data, correlation analysis is applied on the dataset. Fig. 2.1.4 depicts the correlation plot.

Fig. 2.1.4 Correlation plot (python code in Appendix B)



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

i. The relationship between two independent variables should be less and
ii. The relationship between Independent and Target variables should be high.

In the above plot there is a strong correlation between variables ‘temp’ and ‘atemp’.

Dimensionality Reduction:

As ‘temp’ and ‘atemp’ are highly correlated we can drop 'atemp'. There is almost no relationship between ‘hum’ and ‘cnt’, so it is not an important variable. Variable ‘workingday’ consists of ‘holiday’ and ‘workingday’, and holiday does not contribute to the target variable hence we can remove ‘holiday’ variable. (python code in Appendix B)

2.1.5 Feature Scaling

Feature scaling is a process of reducing unwanted variations within the variables and limit the range of variable on common ground. There are two types of feature scaling techniques:

a) Normalization: Rescales the range of values between [0,1]
b) Standardization: Rescales the data to have a mean of 0 or standard deviation of 1 unit variance

In given dataset all the numeric variables are normally distributed, so there is no need of standardizing or normalizing the data.

2.2 Modeling

Before feeding data to the model it is necessary that we split the data into test and train. We have to split the data with test_size=0.2(80% of train data and 20% of test data). This is done using train_test_split.

2.2.1 Model Selection:

In this case we have to predict the count of bike renting according to environmental and seasonal conditions. The target variable here is a continuous variable hence we have to use regression model to predict the count of bike renting. Model having less error rate and high accuracy will be selected.

The models which we implement are linear regression, decision tree and random forest. And the error metric used here is MAPE.

2.2.2 Linear Regression

For linear regression model we have divided the categorical variables containing more than 2 categories into dummy variable, so that all categorical variables are in binary classes form. On creating dummy variable there are 34 variables in the data set with 32nd variable as the target variable. This data is further divided into 80% train and 20% test data.

Python code:

```
#Train the model
lr_model = sm.OLS(train_lr.iloc[:,0].astype(float), train_lr.iloc[:,1:34].a
stype(float)).fit()
#summary of model
lr_model.summary()
```

Model Summary:

OLS Regression Results

Dep. Variable:	cnt	**R-squared:**	0.840
Model:	OLS	**Adj. R-squared:**	0.832
Method:	Least Squares	**F-statistic:**	110.3
Date:	Fri, 27 Dec 2019	**Prob (F-statistic):**	3.11e-198
Time:	17:13:20	**Log-Likelihood:**	-4622.3
No. Observations:	573	**AIC:**	9299.
Df Residuals:	546	**BIC:**	9416.
Df Model:	26		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
temp	3818.4793	467.751	8.163	0.000	2899.668	4737.291
windspeed	-2182.4235	498.453	-4.378	0.000	-3161.543	-1203.304
season_1	-465.5772	138.528	-3.361	0.001	-737.689	-193.465
season_2	432.0389	135.263	3.194	0.001	166.339	697.739
season_3	287.6943	144.415	1.992	0.047	4.018	571.371
season_4	931.6459	142.577	6.534	0.000	651.579	1211.713
yr_0	-435.8521	78.156	-5.577	0.000	-589.375	-282.329
yr_1	1621.6540	86.919	18.657	0.000	1450.916	1792.391
mnth_1	-429.9982	198.130	-2.170	0.030	-819.188	-40.808
mnth_2	-223.0144	191.206	-1.166	0.244	-598.605	152.576
mnth_3	187.2241	144.598	1.295	0.196	-96.812	471.261

mnth_4	283.4908	173.979	1.629	0.104	-58.258	625.240
mnth_5	375.3061	188.060	1.996	0.046	5.896	744.716
mnth_6	298.7420	174.937	1.708	0.088	-44.891	642.375
mnth_7	24.9066	207.730	0.120	0.905	-383.141	432.954
mnth_8	353.6132	193.228	1.830	0.068	-25.949	733.175
mnth_9	741.9026	153.585	4.831	0.000	440.213	1043.592
mnth_10	340.2359	174.444	1.950	0.052	-2.427	682.899
mnth_11	-380.7463	189.534	-2.009	0.045	-753.051	-8.442
mnth_12	-385.8604	159.519	-2.419	0.016	-699.207	-72.514
weekday_0	223.8312	164.515	1.361	0.174	-99.328	546.991
weekday_1	-53.2916	94.334	-0.565	0.572	-238.593	132.010
weekday_2	-50.6349	110.947	-0.456	0.648	-268.571	167.301
weekday_3	95.3518	108.612	0.878	0.380	-117.997	308.700
weekday_4	140.2671	111.451	1.259	0.209	-78.659	359.193
weekday_5	183.0553	109.674	1.669	0.096	-32.379	398.490
weekday_6	647.2229	163.225	3.965	0.000	326.597	967.849
workingday_0	353.5272	144.642	2.444	0.015	69.404	637.651
workingday_1	832.2747	116.015	7.174	0.000	604.383	1060.166
weathersit_1	1518.9562	82.347	18.446	0.000	1357.201	1680.711
weathersit_2	782.2625	84.609	9.246	0.000	616.064	948.461

weathersit_3	-1115.4169	165.635	-6.734	0.000	-1440.777	-790.057

Omnibus:	90.214	**Durbin-Watson:**	2.032
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	256.329
Skew:	-0.768	**Prob(JB):**	2.18e-56
Kurtosis:	5.895	**Cond. No.**	1.12e+16

By looking at the model summary, we can see that Adjusted R-Squared value is 83.2%. The F-Static and p-value indicates that we can reject the null hypothesis that the target variable is independent with other predictor variables.

The mean absolute percentage error (MAPE) is a statistical measure of how accurate a forecast system is and MAPE of this linear regression model is 17.47%. The accuracy of the model is 82.53%. RMSE is found to be 787.

2.2.3 Decision Tree

For decision tree analysis, we use 80% of train and 20% of test data from the given dataset with 10 variables and 10th variable as the target variable.

MAPE of this model is 18.67% which is high compared to the error value of linear regression model. Accuracy of the model is 81.33% and does not perform as good as linear regression model.

2.2.4 Random Forest

Random Forest is a bagging technique which uses ensemble learning method. We use the same train and test data used in decision tree.

Python code:

```
#implement random forest algorithm on train data
rf = RandomForestRegressor(n_estimators=500,random_state=123)
rf_model=rf.fit(train.iloc[:,0:9],train.iloc[:,9])

#Apply model on test data
rf_pred=rf_model.predict(test.iloc[:,0:9])

######Error Metrics-Random Forest Regression######
mape_rf=MAPE(test.iloc[:,9],rf_pred)
print('mape:',mape_rf)
```

```
# Calculate and display accuracy
accuracy = 100 - mape_rf
print('Accuracy:', round(accuracy, 2), '%.')
```

```
mape: 13.134553811126946
Accuracy: 86.87 %.
```

```
rmse = sqrt(mean_squared_error(test.iloc[:,9], rf_pred))

print(rmse)
```

```
599.4035250330578
```

The number of decision trees can be used is 500 trees in a forest. MAPE using this model was found to be 13.13%, hence the accuracy is 86.87%. The root mean squared error value is 599.

Chapter 3

Conclusion

3.1 Model Evaluation Metric

As we have done few models, now we have to decide which model to choose. There are multiple metrics on which we select the model. Few of the model evaluation regression metrics are given below:

1. Mean Squared Error (MSE)
2. Root Mean Squared Error (RMSE)
3. Mean Absolute Error (MAE)
4. R Squared (R^2)
5. Adjusted R Squared (R^2)
6. Mean Square Percentage Error (MSPE)
7. Mean Absolute Percentage Error (MAPE)
8. Root Mean Squared Logarithmic Error (RMSLE)

The MAPE method measure the absolute error. The negative and the positive errors, cancel out each other. RMSE method is more accurate. By Squaring the errors we can get more accurate results as the negative and positive errors don't cancel out each other and stay in existence till the end of commutation, thus adding more accuracy to the result.

3.2 Model Selection

When we look at the MAPE and RMSE values of these models, error value is less in random forest model. Hence, we choose Random Forest model to predict the test cases and get more accurate predicted values.

Output is saved in the csv format:

```
#Create a dataframe for actual values and predicted values
df_rf = pd.DataFrame({'actual': test.iloc[:,9], 'pred': rf_pred})
df_rf.to_csv('submission_rf.csv',index=False)
df_rf.head()
```

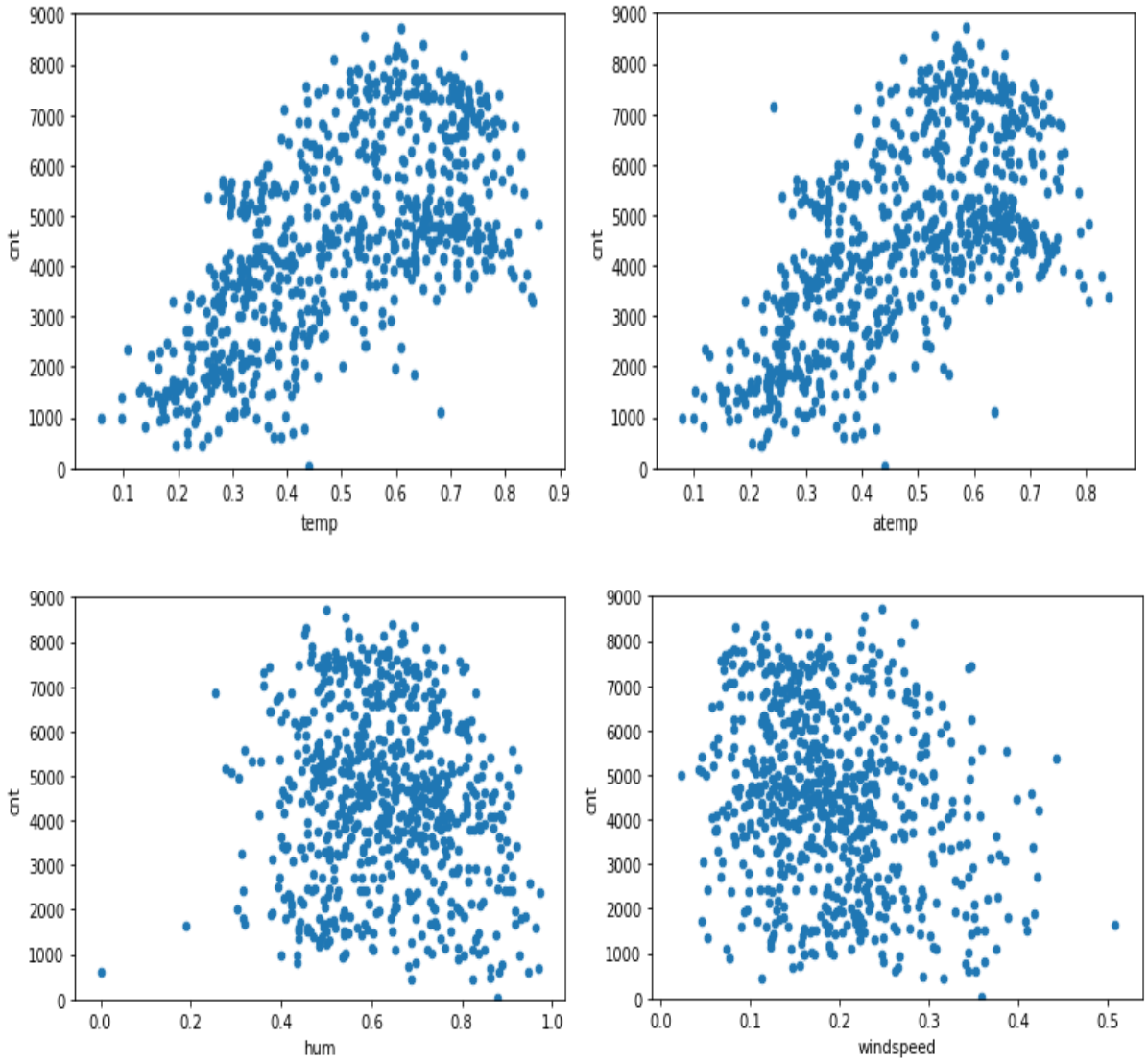
	actual	pred
706	5008	5278.878
73	2056	1932.118
630	8395	7922.820
55	1461	1444.022
161	4966	5143.978

Table 3.2 Actual and Predicted value

The above Tables 3.2 shows the actual and predicted value of the test cases using random forest.

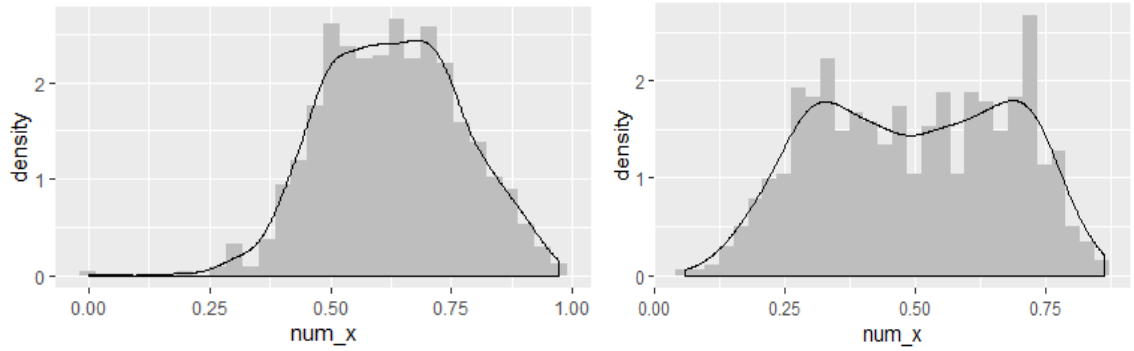
Appendix A – Extra Figure

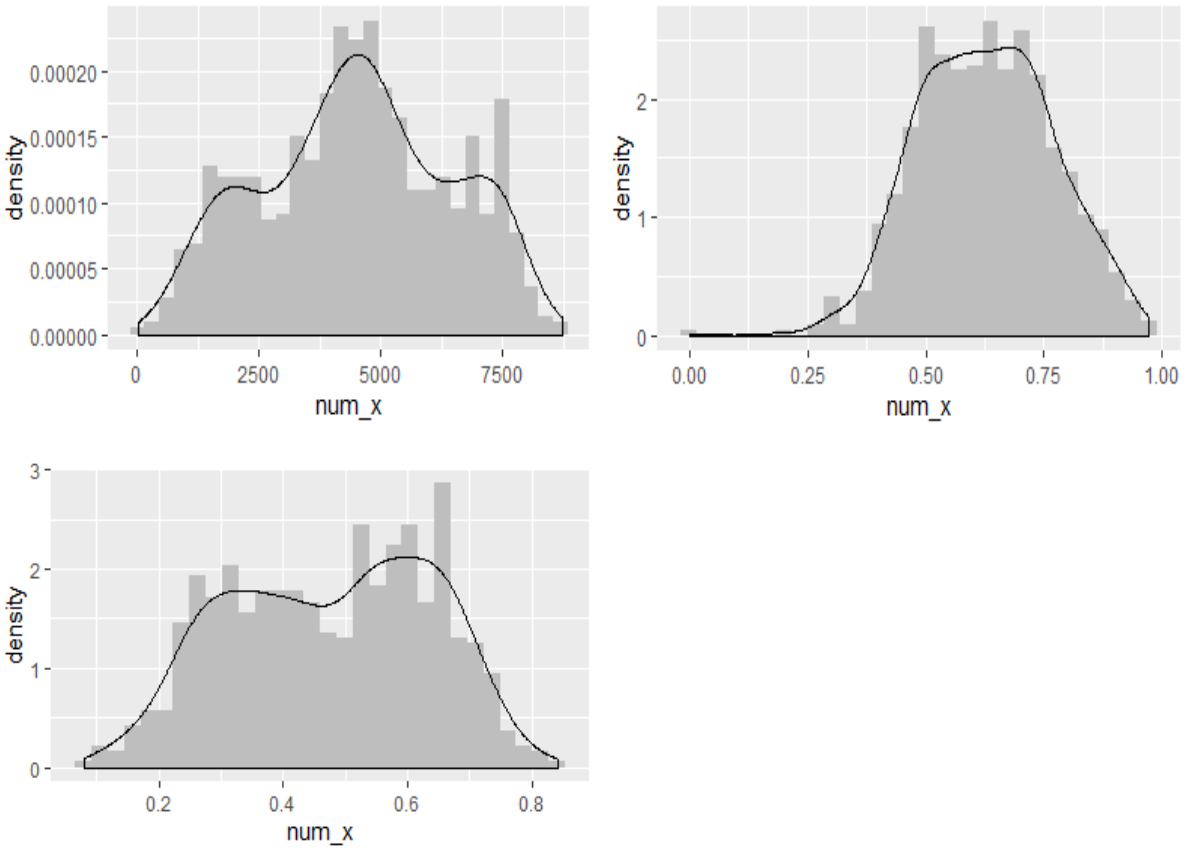
Scatter plots of numeric variable with target variable cnt using python (See python code in Appendix)



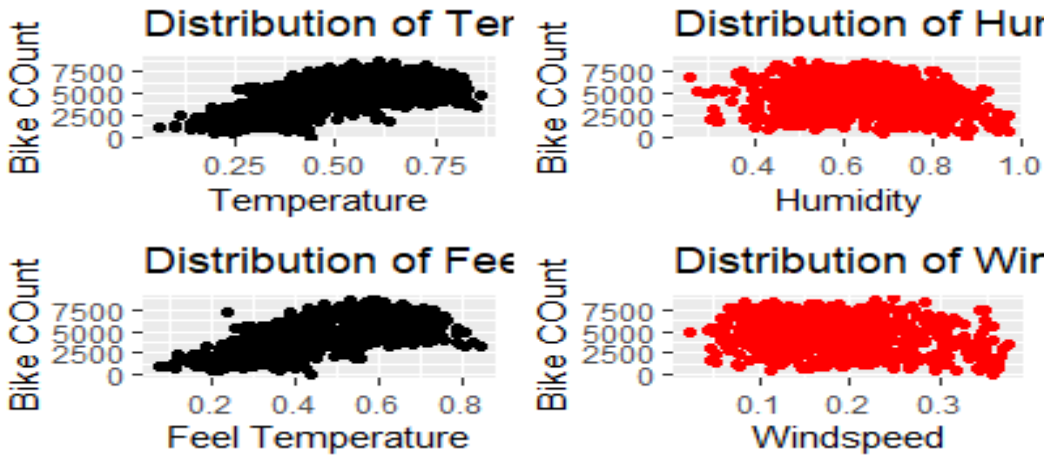
R plots (See R code in Appendix)

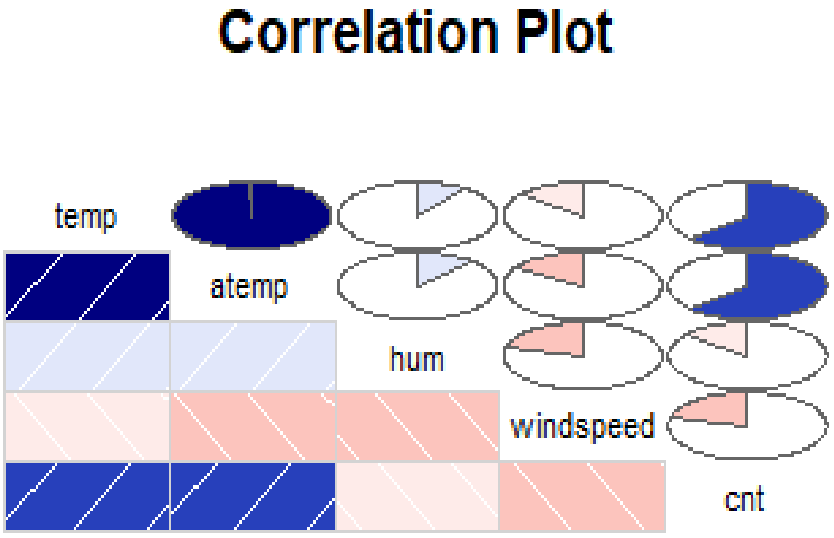
Distribution of numerical variables:



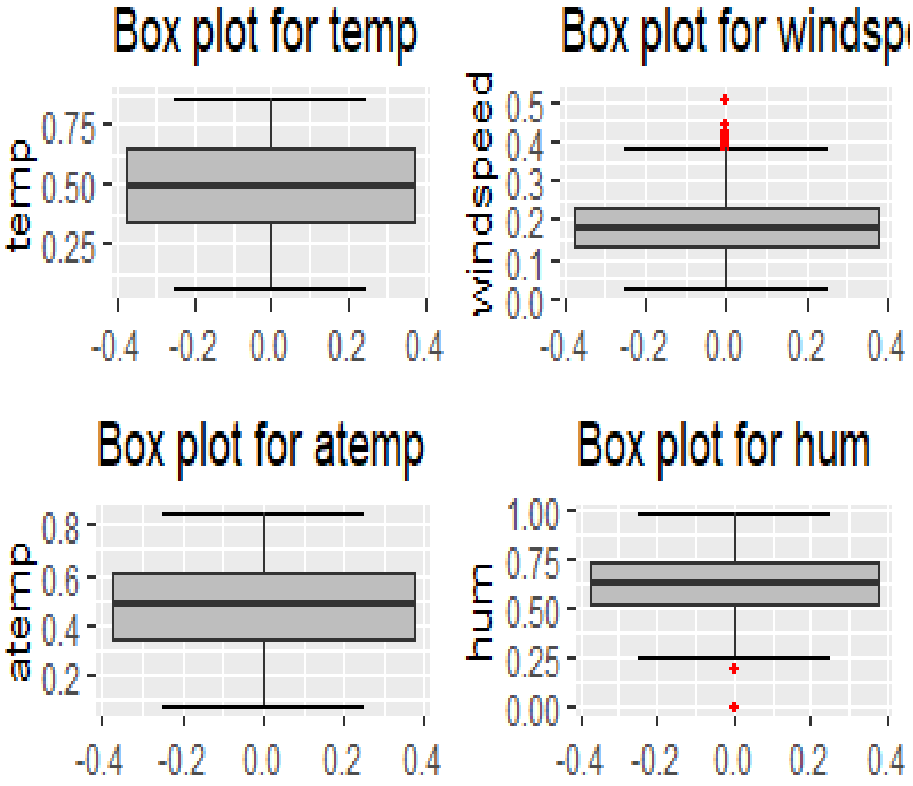


Scatter plot of numerical variables with target variable:

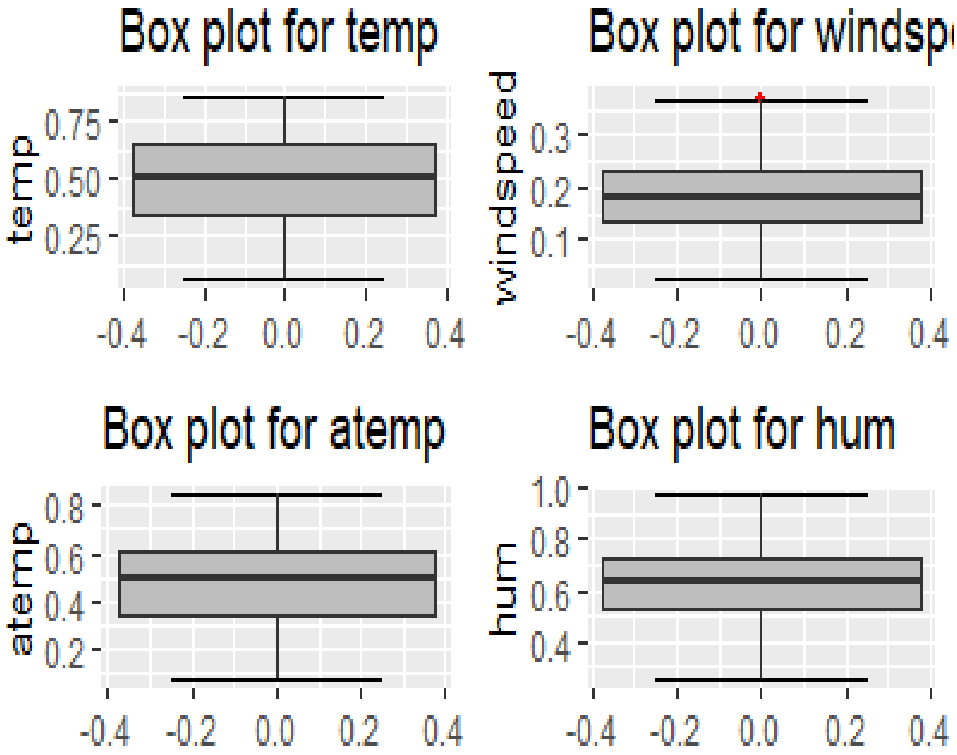




Boxplot before removal of outliers:



Boxplot after removing outliers



Appendix B -Python Code

Figure 2.1.1(a) & (b) Distribution of target variable 'cnt' and all numerical variables

```
##target variable analysis##
day['cnt'].describe()
#check whether it is normally distributed#
sns.distplot(day['cnt']);
##distribution of independent numerical varaibles##
numerical_feature=['temp','atemp','hum','windspeed']
print("Distribution of Numerical Variables")
for i,col in enumerate(numerical_feature):
    plt.subplots(1)
    sns.distplot(day[col]);
    plt.title(col)
```

Figure 2.1.1(c) distribution of sales based on seasons

```
#Sales by Season
sales_by_season = day.groupby('season').size()
print(sales_by_season)
plot_by_day = sales_by_season.plot(title='Season Sales',xticks=(range(1
,4)))
plot_by_day.set_xlabel('season')
plot_by_day.set_ylabel('Total BIkes Rented')
```

Figure 2.1.1(e) Relationship between numerical variables and target variable using bivariate analysis

```
##check realtionship with scatter plots##
sns.set()
sns.pairplot(day[numerical_feature],size=2.5,kind='reg')
plt.show();
```

Table 2.1.2 Missing values in the data

```
#Check if there are missing values
day.isnull().sum()
```

Figure 2.1.3(a) &(b) boxplot before outlier analysis in variable windspeed and hum

```
#displaying a boxplot for numerical variables before removal of outlier
s
for i in numerical_feature:
    plt.show(day.boxplot(column = i, sym='k.', figsize=(10,5)))
```

Fig. 2.1.4 Correlation plot

```
dfcorr=day
#dfcorr.shape
f,ax=plt.subplots(figsize=(7,5))
corr=dfcorr.corr()
sns.heatmap(corr,mask=np.zeros_like(corr,dtype=np.bool),cmap=sns.diverg
ing_palette(220,10,as_cmap=True),square=True,ax=ax)
```

Fig. 2.1.4 Dimensionality Reduction

```
day = day.drop(['atemp','hum','holiday'], axis=1)
```

Complete Python code:

```
############importing libraries#############
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import chi2_contingency
import seaborn as sns
from random import randrange, uniform
import datetime as dt
from sklearn.tree import DecisionTreeRegressor
from sklearn.model_selection import train_test_split
import statsmodels.api as sm
from sklearn.metrics import mean_squared_error,mean_squared_log_error
from math import sqrt
from sklearn.ensemble import RandomForestRegressor
from  matplotlib import pyplot
%matplotlib inline

############working directory################
os.chdir("C:/Users/chandini c/Desktop")
os.getcwd()
```

```
############Read the csv file################
day = pd.read_csv("day.csv")
```

```
##########shape of train and test data#########
day.shape
```

```
(731, 16)
```

```
############observing data#################
day.head(5)
```

```
############checking datatypes#############
day.dtypes
```

Exploratory Data Analysis

```
############exploratory data analysis#######################
day['season']= day['season'].astype('category')
day['yr']=day['yr'].astype('int')
day['mnth']=day['mnth'].astype('category')
day['holiday']=day['holiday'].astype('int')
day['workingday']=day['workingday'].astype('int')
day['weekday']=day['weekday'].astype('category')
day['weathersit']=day['weathersit'].astype('category')
d1=day['dteday'].copy()
for i in range (0,d1.shape[0]):
    d1[i]=dt.datetime.strptime(d1[i], '%Y-%m-%d').strftime('%d')
day['dteday']=d1
day['dteday']=day['dteday'].astype('category')
day = day.drop(['instant','casual', 'registered'], axis=1)
```

```
day.dtypes
```

```
############target variable analysis############
day['cnt'].describe()
############check whether it is normally distributed###########
sns.distplot(day['cnt']);
```

```
#############distribution of independent numerical varaibles#############
numerical_feature=['temp','atemp','hum','windspeed']
print("Distribution of Numerical Variables")
for i,col in enumerate(numerical_feature):
    plt.subplots(1)
    sns.distplot(day[col]);
    plt.title(col)
```

```
#Sales by Season
sales_by_season = day.groupby('season').size()
print(sales_by_season)
plot_by_day = sales_by_season.plot(title='Season Sales',xticks=(range(1,4))
)
```

```
plot_by_day.set_xlabel('season')
plot_by_day.set_ylabel('Total BIkes Rented')
```

```
season
1    181
2    184
3    188
4    178
dtype: int64
```

Bivariate relationship between the numerical variable and target variable

```
#############relation between numerical variable 'temp' and target variable
'cnt'###########
day['cnt'].value_counts()
##draw scatter plot between numerical variable 'temp' and target variable '
cnt'##
var='temp'
df=pd.concat([day['cnt'],day[var]],axis=1)
df.plot.scatter(x=var,y='cnt',ylim=(0,9000));
#there is a good relationship between 'temp' and 'cnt'

#############relation between numerical variable 'atemp' and target variabl
e 'cnt'###########
day['cnt'].value_counts()
##draw scatter plot between numerical variable 'atemp' and target variable
'cnt'##
var='atemp'
df=pd.concat([day['cnt'],day[var]],axis=1)
df.plot.scatter(x=var,y='cnt',ylim=(0,9000));


#############relation between numerical variable 'hum' and target variable
'cnt'###########
day['cnt'].value_counts()
##draw scatter plot between numerical variable 'atemp' and target variable
'cnt'##
var='hum'
df=pd.concat([day['cnt'],day[var]],axis=1)
df.plot.scatter(x=var,y='cnt',ylim=(0,9000));

#############relation between numerical variable 'windspeed' and target var
iable 'cnt'###########
day['cnt'].value_counts()
##draw scatter plot between numerical variable 'windspeed' and target varia
ble 'cnt'##
var='windspeed'
```

```
df=pd.concat([day['cnt'],day[var]],axis=1)
df.plot.scatter(x=var,y='cnt',ylim=(0,9000));
```

Missing Value Analysis

```
#Check if there are missing values
day.isnull().sum()
```

Outlier Analysis

```
#############displaying a boxplot for numerical variables before removal of
outliers##############
for i in numerical_feature:
    plt.show(day.boxplot(column = i, sym='k.', figsize=(10,5)))


##after analysis we can say that variables 'hum' and 'windspeed' has outlie
rs##
##Removal Of Outliers##
#outliers in each variable in train data
train_outliers = dict()
for col in [col for col in numerical_feature]:
    q75,q25=np.percentile(day.loc[:,col],[75,25])
    Q=q75-q25
    min=q25-(Q*1.5)
    max=q75+(Q*1.5)
    #print(min)
    #print(max)
    day=day.drop(day[day.loc[:,col]<min].index)
    day=day.drop(day[day.loc[:,col]>max].index)

# displaying a boxplot for numerical variables after removal of outliers
for i in numerical_feature:
    plt.show(day.boxplot(column = i, sym='k.', figsize=(10,5)))
```

Feature Selection

```
################Correlation Analysis#######################
dfcorr=day
#dfcorr.shape
f,ax=plt.subplots(figsize=(7,5))
corr=dfcorr.corr()
sns.heatmap(corr,mask=np.zeros_like(corr,dtype=np.bool),cmap=sns.diverging_
palette(220,10,as_cmap=True),square=True,ax=ax)
```

```
#as per the correlation graph there is a strong relation between variables
'temp' and 'atemp'

##check realtionship with scatter plots##

sns.set()
sns.pairplot(day[numerical_feature],size=2.5,kind='reg')
plt.show();

#as per the scatter plots there is poor relation between 'hum' and 'cnt'
#there is a poor relation between 'windspeed' and 'cnt'

#as 'temp' and 'atemp' are highly correlated we can drop 'atemp'
#poor relation between 'hum' and 'cnt'
#'holiday' does not contribute much to the independent variable.
day = day.drop(['atemp','hum','holiday'], axis=1)
```

Modeling

```
#Divide data into train and test
train, test = train_test_split(day, test_size=0.2)

##########################################Linear Regression############
##################################
#implement linear regression algorithm on train data
lr_model = sm.OLS(train.iloc[:,9].astype(float), train.iloc[:,0:9].astype(f
loat)).fit()

#Check the summary of model
lr_model.summary()

#Predict the results of test data
lr_pred = lr_model.predict(test.iloc[:,0:9].astype(float))

######Error Metrics-Linear Regression######

#defining MAPE function
def MAPE(y_true, y_pred):
    mape = np.mean(np.abs((y_true - y_pred) / y_true))*100
    return mape
mape_lr=MAPE(test.iloc[:,9],lr_pred)
print('mape:',mape_lr)

# Calculate and display accuracy
accuracy = 100 - mape_lr
print('Accuracy:', round(accuracy, 2), '%.')

mape: 17.697765131857857
Accuracy: 82.3 %.

rmse = sqrt(mean_squared_error(test.iloc[:,9], lr_pred))
print(rmse)
```

```
887.5354856177063
#mape=17.69
#Accuracy=82.3 %.
#rmse=887
#Adj. R-Squared=0.966
#F-stat=1796

######linear regression###########
#Create continuous data. Save target variable first
train_lr = train[['cnt','temp','windspeed']]
test_lr = test[['cnt','temp','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].a
stype(float)).fit()

#summary of model
lr_model.summary()

#Predict the results of test data
lr_predictions = lr_model.predict(test_lr.iloc[:,1:34])

######Error Metrics-Linear Regression######
mape_lr=MAPE(test.iloc[:,9],lr_predictions)
print('mape:',mape_lr)

# Calculate and display accuracy
accuracy = 100 - mape_lr
print('Accuracy:', round(accuracy, 2), '%.')

mape: 17.472874253080665
Accuracy: 82.53 %.
rmse = sqrt(mean_squared_error(test.iloc[:,9], lr_predictions))

print(rmse)

787.0603399855446
#Adj. R-squared=0.832
#F-statistic=110.3
#MAPE=17.47
#Accuracy=82.53 %
#rmse=787
```

```
##Decision Tree Regression
#implement decision tree algorithm on train data
dt=DecisionTreeRegressor()
dt_model=dt.fit(train.iloc[:,0:9],train.iloc[:,9])

#Apply model on test data
dt_pred=dt_model.predict(test.iloc[:,0:9])
######Error Metrics-Decision Tree Regression######
mape_dt=MAPE(test.iloc[:,9],dt_pred)
print('mape:',mape_dt)

# Calculate and display accuracy
accuracy = 100-mape_dt
print('Accuracy:', round(accuracy, 2), '%.')
```

```
mape: 18.673181688926196
Accuracy: 81.33 %.
```

```
rmse = sqrt(mean_squared_error(test.iloc[:,9], dt_pred))

print(rmse)
```

```
872.2209319700294
```

```
#mape=18.67
#Accuracy=81.33 %
#rmse=872
```

```
##Random Forest Regression##
#implement random forest algorithm on train data
rf = RandomForestRegressor(n_estimators=500,random_state=123)
rf_model=rf.fit(train.iloc[:,0:9],train.iloc[:,9])

#Apply model on test data
rf_pred=rf_model.predict(test.iloc[:,0:9])
######Error Metrics-Random Forest Regression######
mape_rf=MAPE(test.iloc[:,9],rf_pred)
print('mape:',mape_rf)

# Calculate and display accuracy
accuracy = 100 - mape_rf
print('Accuracy:', round(accuracy, 2), '%.')
```

```
mape: 13.134553811126946
Accuracy: 86.87 %.
```

```
rmse = sqrt(mean_squared_error(test.iloc[:,9], rf_pred))
print(rmse)
```

```
599.4035250330578
```

```
#mape=13.13
#Accuracy=86.87 %
```

```
#rmse=599

#Create a dataframe for actual values and predicted values
df_rf = pd.DataFrame({'actual': test.iloc[:,9], 'pred': rf_pred})
df_rf.to_csv('submission_rf.csv',index=False)
df_rf.head()
```

file:///C:/Users/chandini%20c/Downloads/Project%202%20(Bike%20Rental%20Prediction)%20(1).html

Python output file:

submission_rf.csv

Appendix C -R Code

```
#set working directory

rm(list=ls())

setwd("C:/Users/chandini c/Desktop")

getwd()


#read csv file

day=read.csv("day.csv")


#structure of the data

str(day)

#Data overview

head(day)


#import libraries

x = c("ggplot2", "corrgram", "DMwR", "caret", "randomForest", "unbalanced", "C50", "dummies",
"e1071", "Information","Metrics","MASS", "rpart", "gbm", "ROSE", 'sampling', 'DataCombine',
'inTrees','fastDummies',"GGally","usdm","dplyr","plyr","data.table")

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

rm(x)


##Feature Engineering##

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

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

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

day$act_weathersit = factor(x = day$weathersit, levels = c(1,2,3,4),

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

```
day$weathersit = as.factor(day$weathersit)
day$season = as.factor(day$season)
day$dteday = as.character(day$dteday)
day$mnth = as.factor(day$mnth)
day$weekday = as.factor(as.character(day$weekday))
day$workingday = as.factor(as.character(day$workingday))
day$yr = as.factor(day$yr)
day$holiday = as.factor(day$holiday)

#remove column 'instant' as it is record index and does not help us to predict any value
#remove 'casual' and 'registered' as its sum is equal to target variable 'cnt
day=subset(day,select = -c(instant,casual,registered))
dim(day)
#structure of the data
str(day)

##Exploratory Data Analysis

# Analyze variables  by visualize

# function to create univariate distribution of numeric  variables
univariate_numeric = function(num_x) {



ggplot(day)+
geom_histogram(aes(x=num_x,y=..density..),
fill= "grey")+
geom_density(aes(x=num_x,y=..density..))

}
```

```
# analyze the distribution of  target variable 'cnt'
univariate_numeric(day$cnt)

# analyse the distrubution of  independence variable 'temp'
univariate_numeric(day$temp)

# analyse the distrubution of  independence variable 'atemp'
univariate_numeric(day$atemp)

# analyse the distrubution of  independence variable 'hum'
univariate_numeric(day$hum)

# analyse the distrubution of  independence variable 'windspeed'
univariate_numeric(day$windspeed)

#Check the distribution of categorical Data using bar graph
bar1 = ggplot(data = day, aes(x = act_season)) + geom_bar() + ggtitle("Count of Season")
bar2 = ggplot(data = day, aes(x = act_weathersit)) + geom_bar() + ggtitle("Count of Weather")
bar3 = ggplot(data = day, aes(x = act_holiday)) + geom_bar() + ggtitle("Count of Holiday")
bar4 = ggplot(data = day, aes(x = workingday)) + geom_bar() + ggtitle("Count of Working day")
# Plotting plots together
gridExtra::grid.arrange(bar1,bar2,bar3,bar4,ncol=2)


##Missing Values Analysis
#checking for missing value
missing_val = data.frame(apply(day,2,function(x){sum(is.na(x))}))
missing_val

##Outlier Analysis
#Check for outliers in data using boxplot
```

```
cnames = colnames(day[,c("temp","atemp","windspeed","hum")])
for (i in 1:length(cnames))
{
assign(paste0("gn",i), ggplot(aes_string(y = cnames[i]), 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(y=cnames[i])+
ggtitle(paste("Box plot for",cnames[i])))
}
gridExtra::grid.arrange(gn1,gn3,gn2,gn4,ncol=2)
#'hum' and 'windspeed' has outliers, which has to be removed
#Removal of outliers in numerical variables:
for(i in cnames){
val=day[,i][day[,i]%in%boxplot.stats(day[,i])$out]
day=day[which(!day[,i]%in%val),]
}

#Feature selection
#Check for collinearity using corelation analysis
corrgram(day, order = F, upper.panel=panel.pie, text.panel=panel.txt, main = "Correlation Plot")
# correlation matrix  stating  'temp' and 'atemp' having strong relationship
# and there is no  relationship between 'hum' and 'cnt'


#Check the distribution of numerical data using scatterplot
scat1 = ggplot(data = day, aes(x =temp, y = cnt)) + ggtitle("Distribution of Temperature") +
geom_point() + xlab("Temperature") + ylab("Bike COunt")
scat2 = ggplot(data = day, aes(x =hum, y = cnt)) + ggtitle("Distribution of Humidity") +
geom_point(color="red") + xlab("Humidity") + ylab("Bike COunt")
```

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

scat4 = ggplot(data = day, aes(x =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)


#Remove the unwanted variables


day=subset(day, select = -c(dteday,hum,atemp,act_season,act_yr,act_holiday,act_weathersit))


rmExcept("day")



##MODELING##

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

train = day[train_index,]

test = day[-train_index,]


############Decision tree regression  ##################

#mae=640.7

#rmse=856.9

#mape=17.36%

#Accuracy=82.64%


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

predictions_DT = predict(fit, test[,-10])


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

##############Random Forest Model############################

#mae=463.4

#rmse=629.9

#mape=13.54%
```

```
#accuracy=86.46%
RF_model = randomForest(cnt ~ ., train, importance = TRUE, ntree = 200)
predictions_RF = predict(RF_model, test[,-10])

regr.eval(trues = test[,10], preds = predictions_RF, stats = c("mae","mse","rmse","mape"))
#################Linear Regression#################
#mae=562.76
#rmse=722.17
#mape=17.668%
#accuracy=82.3%

#converting multilevel categorical variable into binary dummy variable
cnames= c("season","mnth","weekday","weathersit")
data_lr=day[,cnames]
cnt=data.frame(day$cnt)
names(cnt)[1]="cnt"
data_lr=fastDummies::dummy_cols(data_lr)
data_lr= subset(data_lr,select = -c(season,mnth,weekday,weathersit))
d3 = cbind(data_lr,day)
d3= subset(d3,select = -c(season,mnth,weekday,weathersit,cnt))
data_lr=cbind(d3,cnt)


#dividind data into test and train
train_index = sample(1:nrow(data_lr), 0.8 * nrow(data_lr))
train_lr = data_lr[train_index,]
test_lr = data_lr[-train_index,]

#Linear regression model making
lm_model = lm(cnt ~., data = train_lr)
summary(lm_model)
```

```
predictions_LR = predict(lm_model,test_lr[,-32])

regr.eval(trues = test_lr[,32], preds = predictions_LR, stats = c("mae","mse","rmse","mape"))

output=data.frame(test, pred_cnt = predictions_RF)

write.csv(output, file = 'RF output R .csv', row.names = FALSE, quote=FALSE)
```

R Output file:

RF output R .csv

References

https://towardsdatascience.com/predicting-no-of-bike-share-users-machine-learning-data-visualization-project-using-r-71bc1b9a7495

https://chartio.com/learn/data-analytics/what-is-exploratory-data-analysis/

https://www.statisticshowto.datasciencecentral.com/mean-absolute-percentage-error-mape/

https://towardsdatascience.com/random-forest-and-its-implementation-71824ced454f

https://www.quora.com/Why-we-use-Root-mean-square-error-RMSE-Mean-absolute-and-mean-absolute-percent-errors-for-forecasting-time-series-models