Bike Renting

Submitted by

Lakshmipathi N

Contents

Chapter 1

Introduction

1.1 Problem Statement

The usage of bicycles as a mode of transportation has gained traction in recent years due to with environmental and health issues. The cities across the world have successfully rolled out bike sharing programs to encourage usage of bikes. Under such programs, the riders can rent bicycles using manual or automated stalls spread across the city for defined periods. In most cases, riders can pick up bikes from one location and returned them any other designated place.

The bike sharing programs from across the world are hotspots of all sorts of data, ranging from travel time, start and end location, demographics of riders, and so on. This data along with alternate sources of information such as weather, traffic, terrain, season and so on.

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

1.2 Data

In this project, our task is to build regression models which will used to predict the bike rental count on daily basis. Given below is a sample of the bike sharing dataset:

Table 1.1: Bike sharing Dataset (Columns: 1-8)

instant	dteday	season	yr	mnth	holiday	weekday	workingday
1		1	0	1	0	6	0
2		1	0	1	0	0	0
3		1	0	1	0	1	1
4		1	0	1	0	2	1
5		1	0	1	0	3	1

Table 1.2: Bike sharing Dataset (Columns: 9-16)

weathersit	temp	atemp	hum	windspeed	casual	registered	cnt
2	0.34416	0.36362	0.80583	0.16044	331	654	985
2	0.36347	0.35373	0.6960	0.24853	131	670	801
1	0.19636	0.1894	0.43727	0.24830	120	1229	1349
1	0.2	0.21212	0.59043	0.16029	108	1454	1562
1	0.22695	0.2292	0.43695	0.1869	82	1518	1600

From the table below we have the following 16 variables, using which we have to predict the bike rental count:

Table 1.3: Predictor Variables

SL.No.	Predictor
1	record id
2	datetime
3	season
4	year
5	month
6	holiday
7	weekday
8	workingday
9	weather_condition
10	temp
11	atemp
12	humidity
13	windspeed
14	casual
15	registered
16	total_count

Chapter 2

Methodology

2.1 Exploratory Data Analysis

Exploratory data analysis is one of the most important step in data mining in order to know features of data. It involves the loading dataset, data cleaning, normality test, typecasting of attributes, missing value analysis, outlier analysis, Attributes distributions and trends. So, we have to clean the data otherwise it will effect on performance of the model. Now we are going to explain one by one as follows.

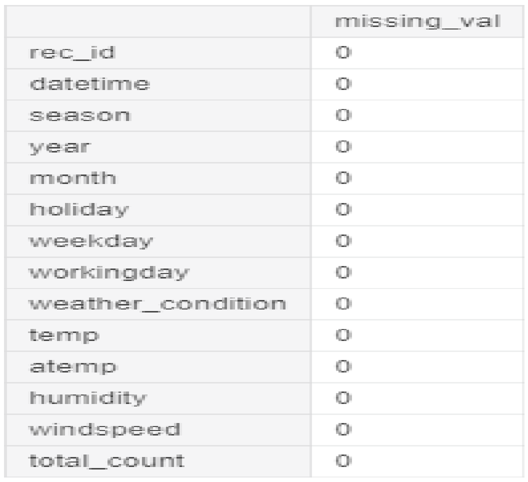
2.1.1 Missing value Analysis

In this, we have to find out any missing values are present in dataset. If it is present then either delete or Impute the values using mean, median and KNN imputation method. We have not found any missing values in this dataset.

R and Python code as follows,

```
#Missing values in dataset R code
missing_val<-data.frame(apply(bike_df,2,function(x){sum(is.na(x))}))
names(missing_val)[1]='missing_val'
missing_val

# Python code
bike_df.isnull().sum()
```

Missing values in dataset



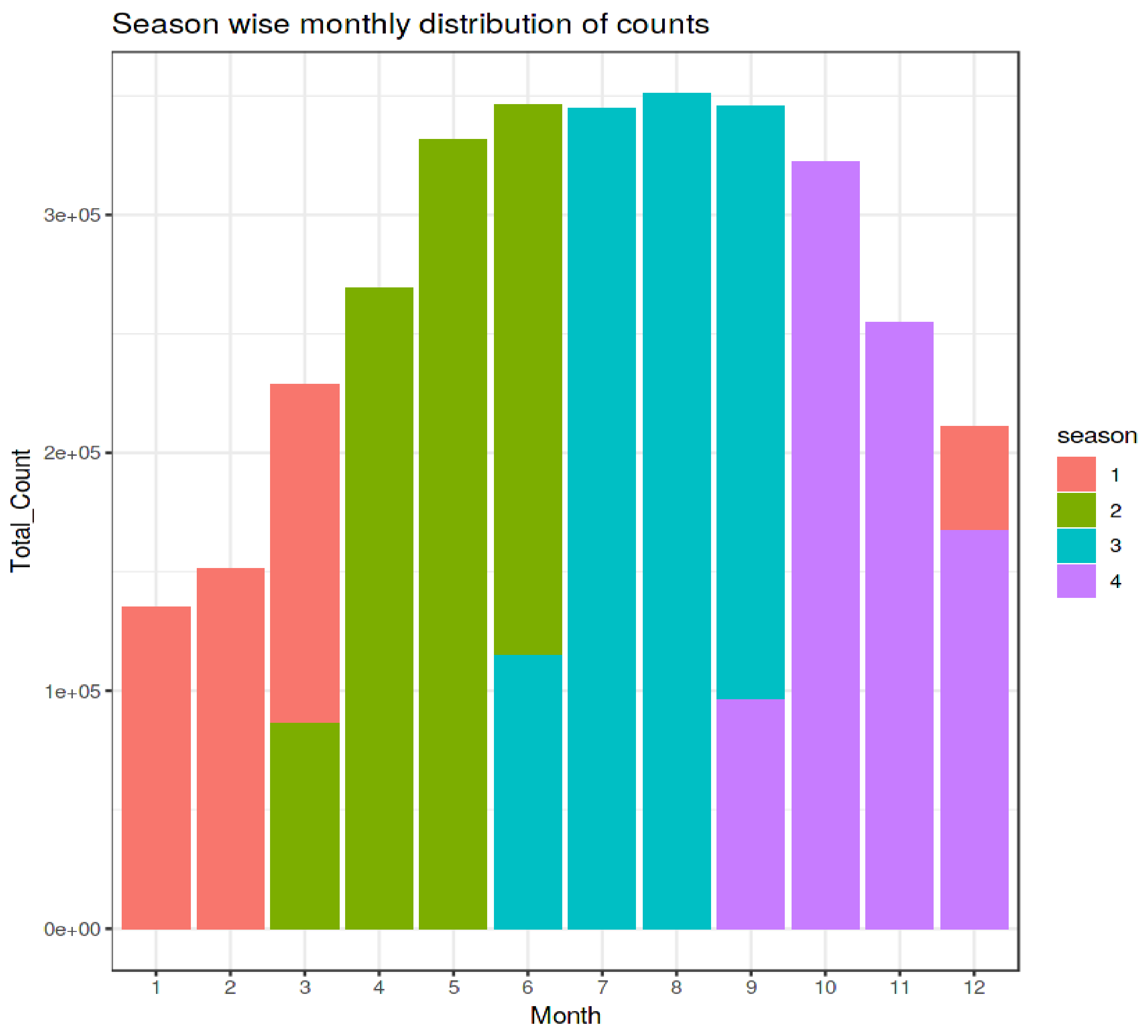
	missing_val
rec_id	0
datetime	0
season	0
year	0
month	0
holiday	0
weekday	0
workingday	0
weather_condition	0
temp	0
atemp	0
humidity	0
windspeed	0
total_count	0

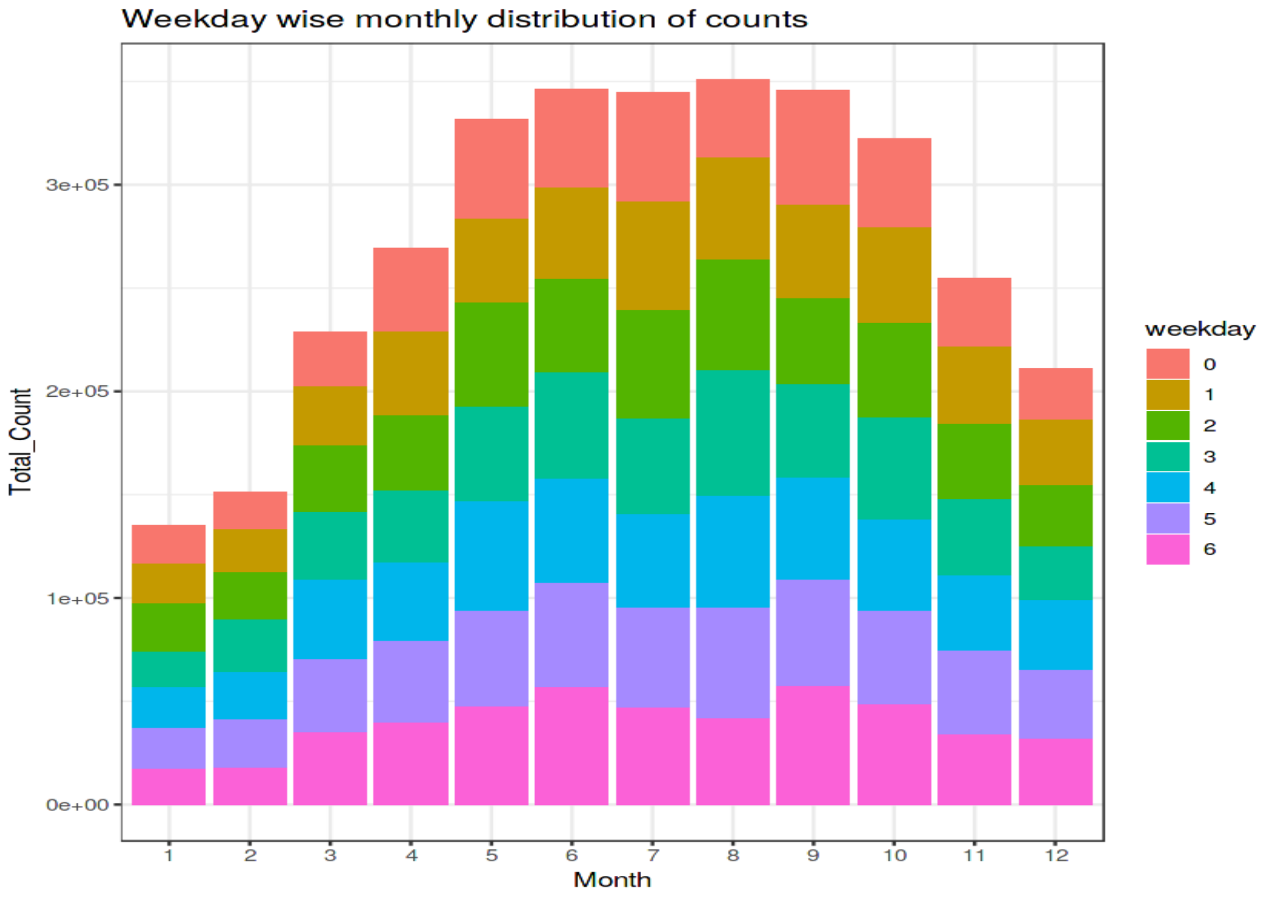
2.1.2 Attributes distributions and trends

Monthly distribution of counts

From the above plots, we can observed that increasing the bike rental count in spring and summer season and then decreasing the bike rental count in fall and winter season.

Here, season 1 -> spring, season 2 -> summer, season 3 -> fall , season 4 -> winter

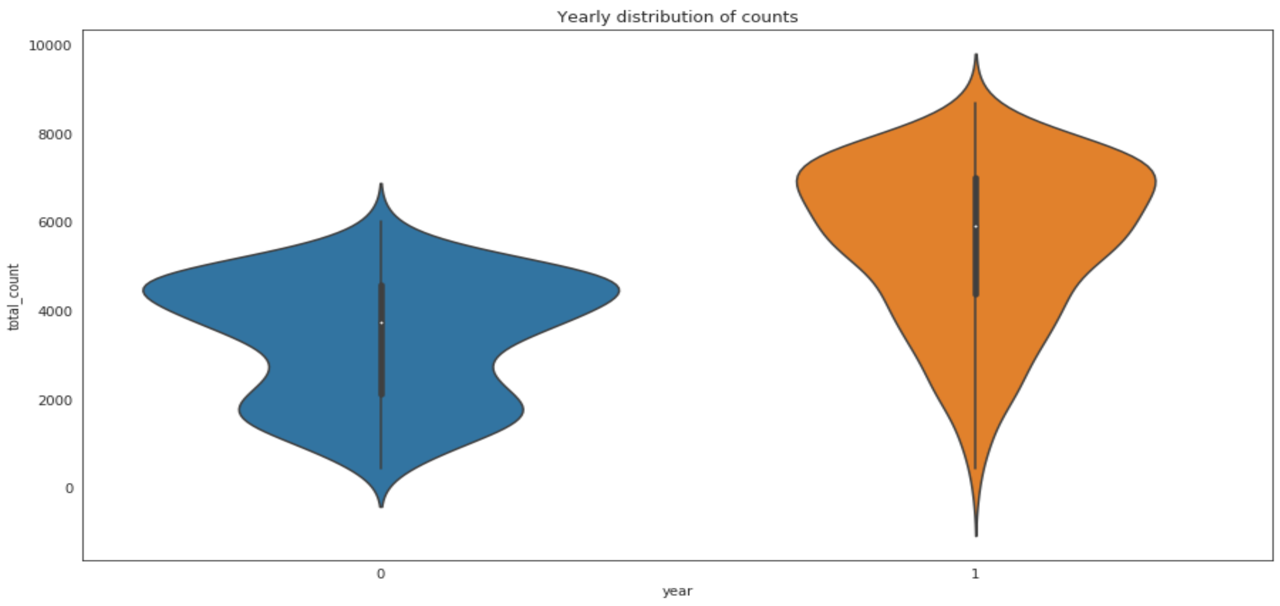




Yearly wise distribution of counts

From the violin plot, we can observed that the bike rental count distribution is highest in year 2012 then in the year 2011.

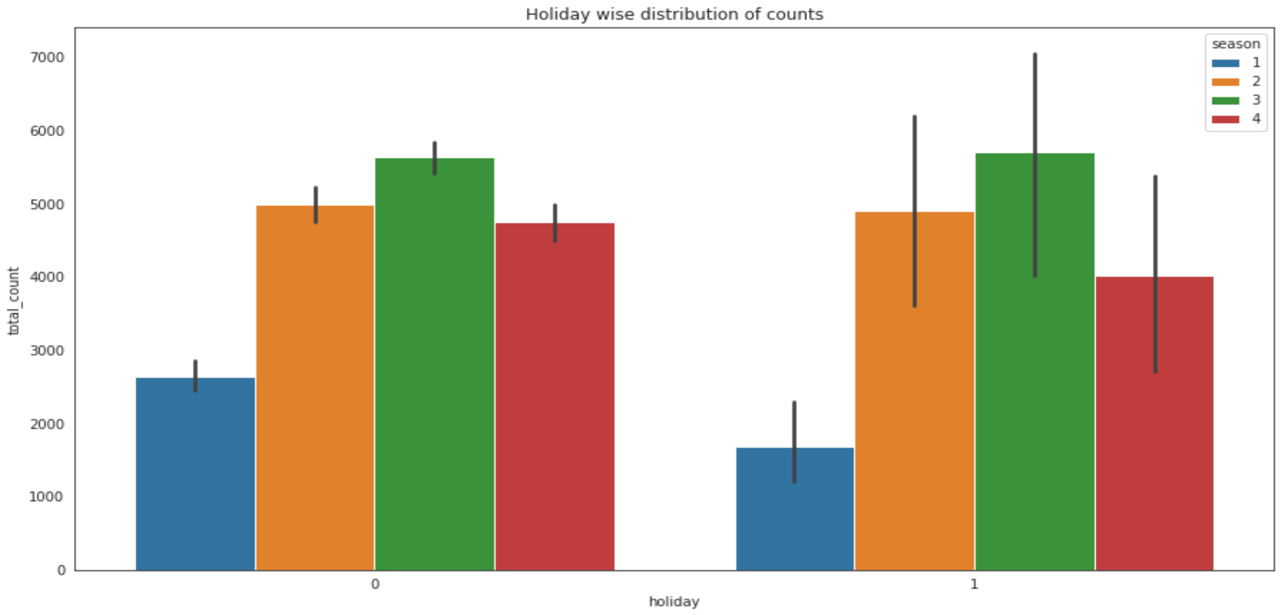
Here, year 0-> 2011, year 1-> 2012



Holiday wise distribution of counts

From the below bar plot, we can observed that during no holiday the bike rental counts is highest compared to during holiday for different seasons.

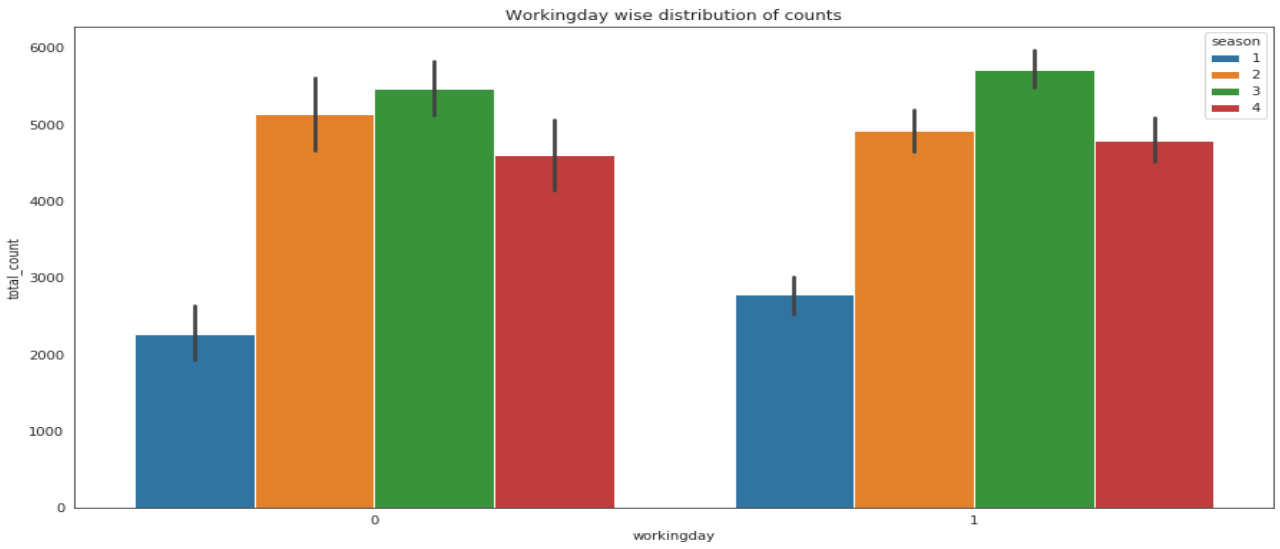
Here, 0->No holiday, 1-> holiday



Workingday wise distribution of counts

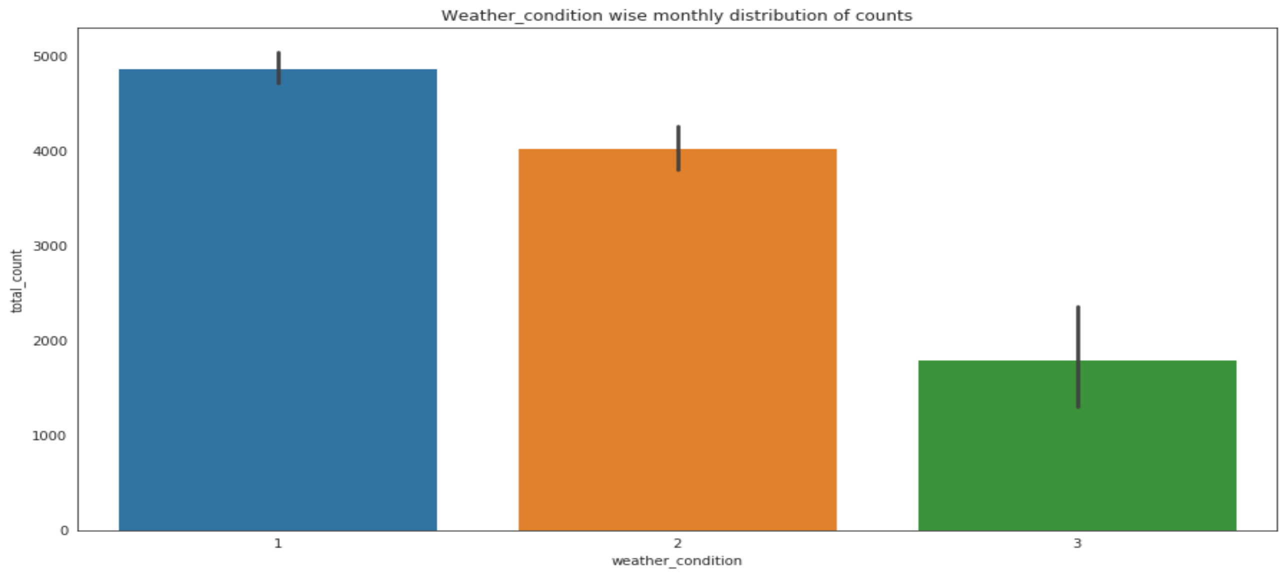
From the above bar plot, we can observed that during workingday the bike rental counts is quite highest compared to during no workingday for different seasons.

Here, 0-> No workingday, 1-> workingday



Weather_condition distribution of counts

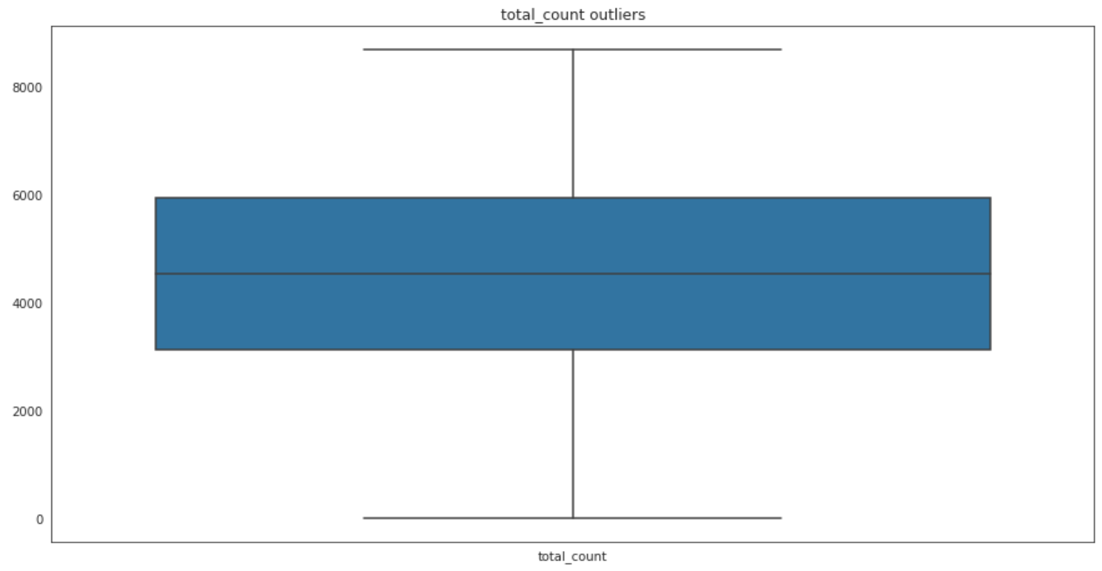
From the below bar plot, we can observed that during clear, partly cloudy weather the bike rental count is highest and the second highest is during mist cloudy weather and followed by third highest during light snow and light rain weather.



2.1.3 Outlier analysis

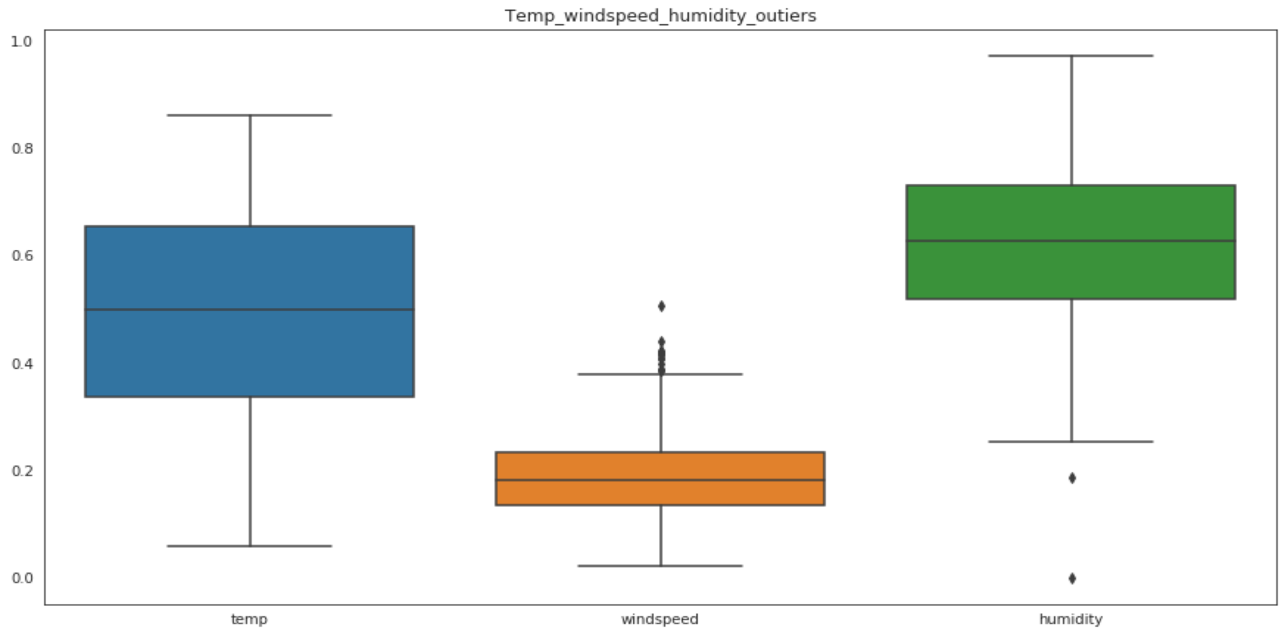
Total_Count_Outliers

From the box plot, we can observed that no outliers are present in total_count variable.



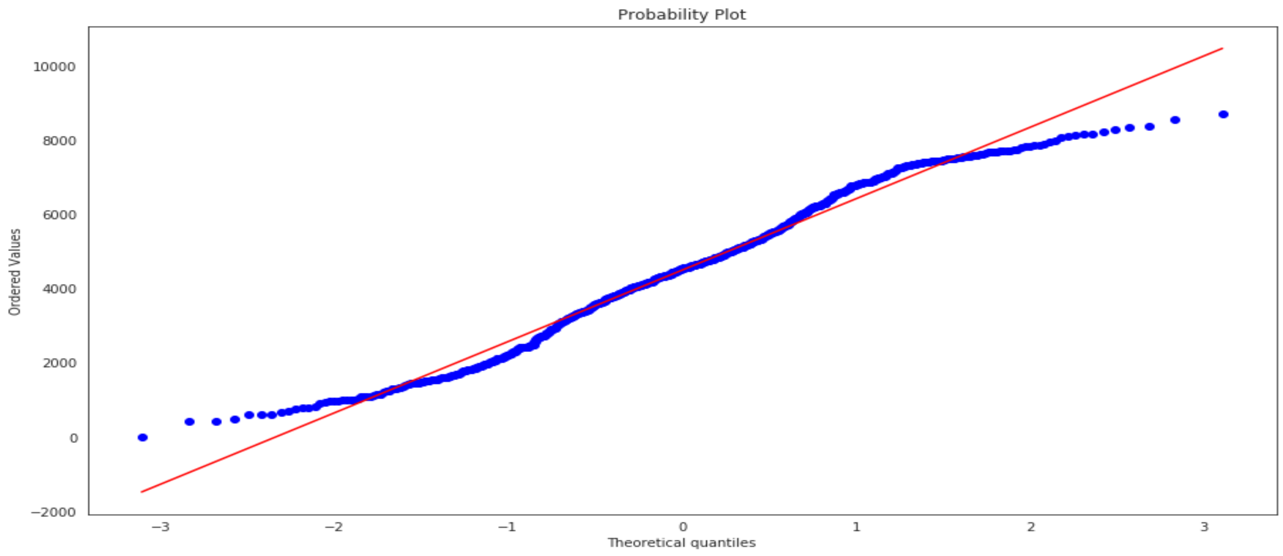
Temp_windspeed_humidity_outliers

From the box plot, we can observed that no outliers are present in normalized temp but few outliers are present in normalized windspeed and humidity variable. Finally we replaced and impute the outliers using mean imputation method.



Normality test

Normal probability plot is a graphical technique to identify substantive departures from normality and also it tells about goodness of fit.
From the below probability plot, we can observed that some target variable data points are deviates from normality.

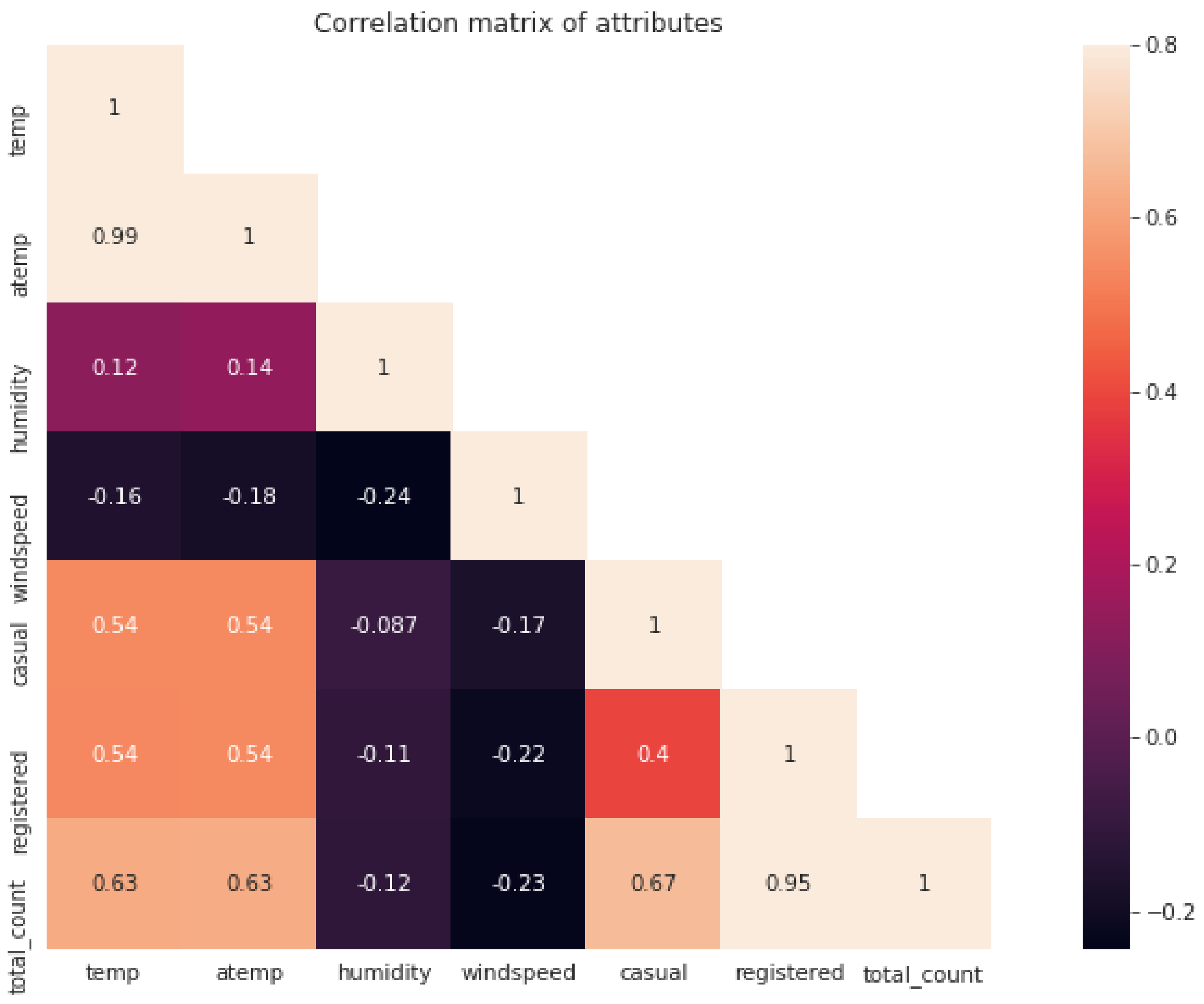


2.1.4 Feature Selection

Feature selection is very important for modelling the dataset. The every dataset have good and unwanted features. The unwanted features will effect on performance of model, so we have to delete those features. We have to select best features by using ANOVA, Chi-Square test and correlation matrix statistical techniques and so on. In this, we are selecting best features by using Correlation matrix.

Correlation matrix

Correlation matrix is tells about linear relationship between attributes and help us to build better models.
From the correlation plot, we can observed that some features are positively correlated and some are. negatively correlated to each other. The temp and atemp are highly positively correlated to each other, it means that both are carrying same information. So, we are going to ignore atemp, casual and registered variable for further analysis.



2.2 Modeling

2.2.1 Model Selection

After all early stages of preprocessing, then model the data. So, we have to select best model for this project with the help of some metrics.

The dependent variable can fall in either of the four categories:

1. Nominal
2. Ordinal
3. Interval
4. Ratio

If the dependent variable is Nominal the only predictive analysis that we can perform is **Classification**, and if the dependent variable is Interval or Ratio like this project, the normal method is to do a **Regression** analysis, or classification after binning.

We always start model building from the simplest to more complex.

2.2.2 Linear Regression

We will use a Linear Regression to predict the values of our target variable.

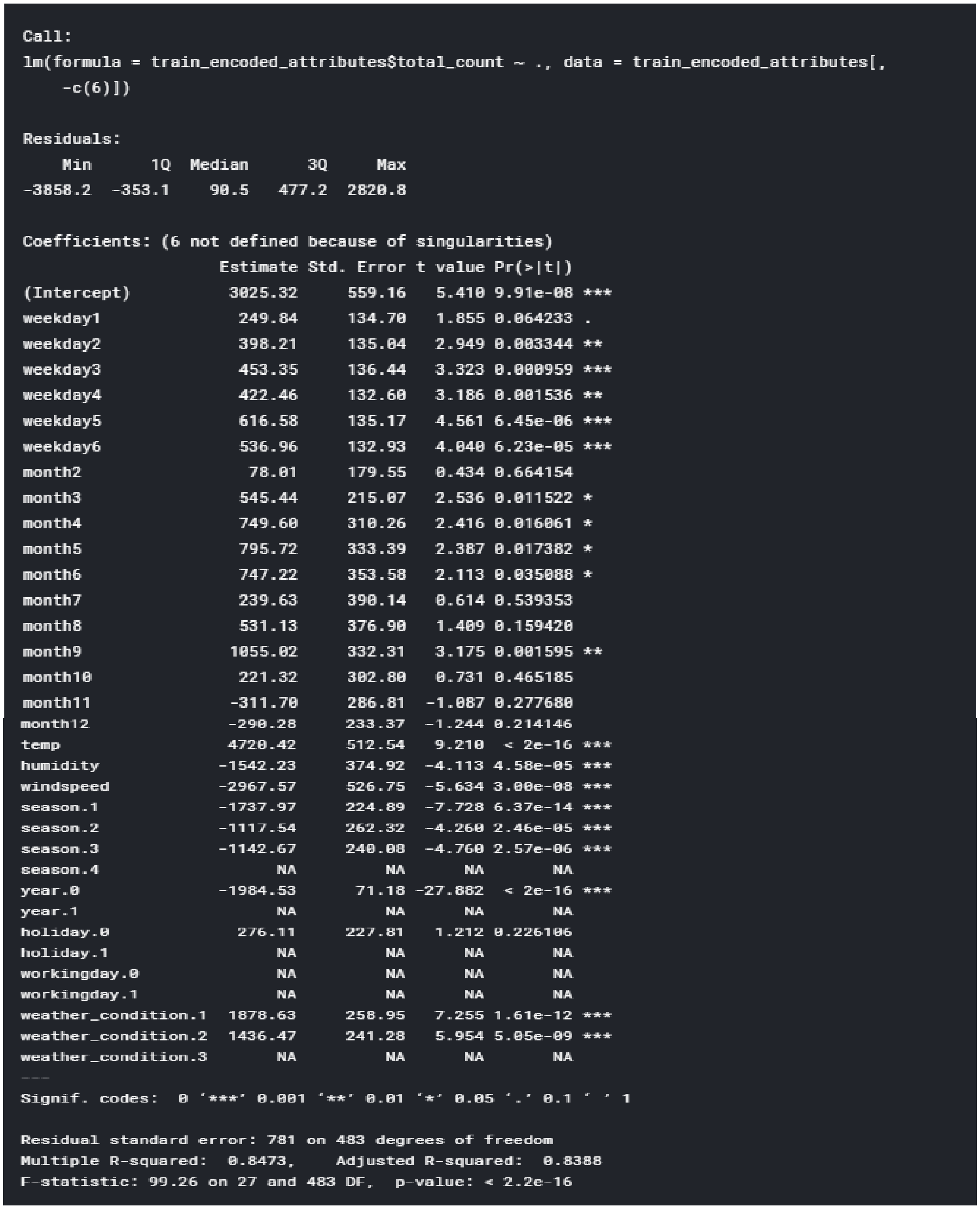
Python code

```
#Training dataset for modelling
X_train=train_encoded_attributes
y_train=y_train.total_count.values

#training model
lr_model=linear_model.LinearRegression()

#fit the trained model
lr_model.fit(X_train,y_train)

#Accuracy of the model
lr=lr_model.score(X_train,y_train)

Accuracy of the model : 0.8165

#Cross validation prediction
predict=cross_val_predict(lr_model,X_train,y_train,cv=3)

#R-squared scores
r2_scores = cross_val_score(lr_model, X_train, y_train, cv=3)
r2_scores=np.average(r2_scores)
r2_scores : 0.80
```

The R-squared or coefficient of determination is 0.80 on average for 3-fold cross validation , it means that predictor is only able to predict 80% of the variance in the target variable which is contributed by all the independent variables.

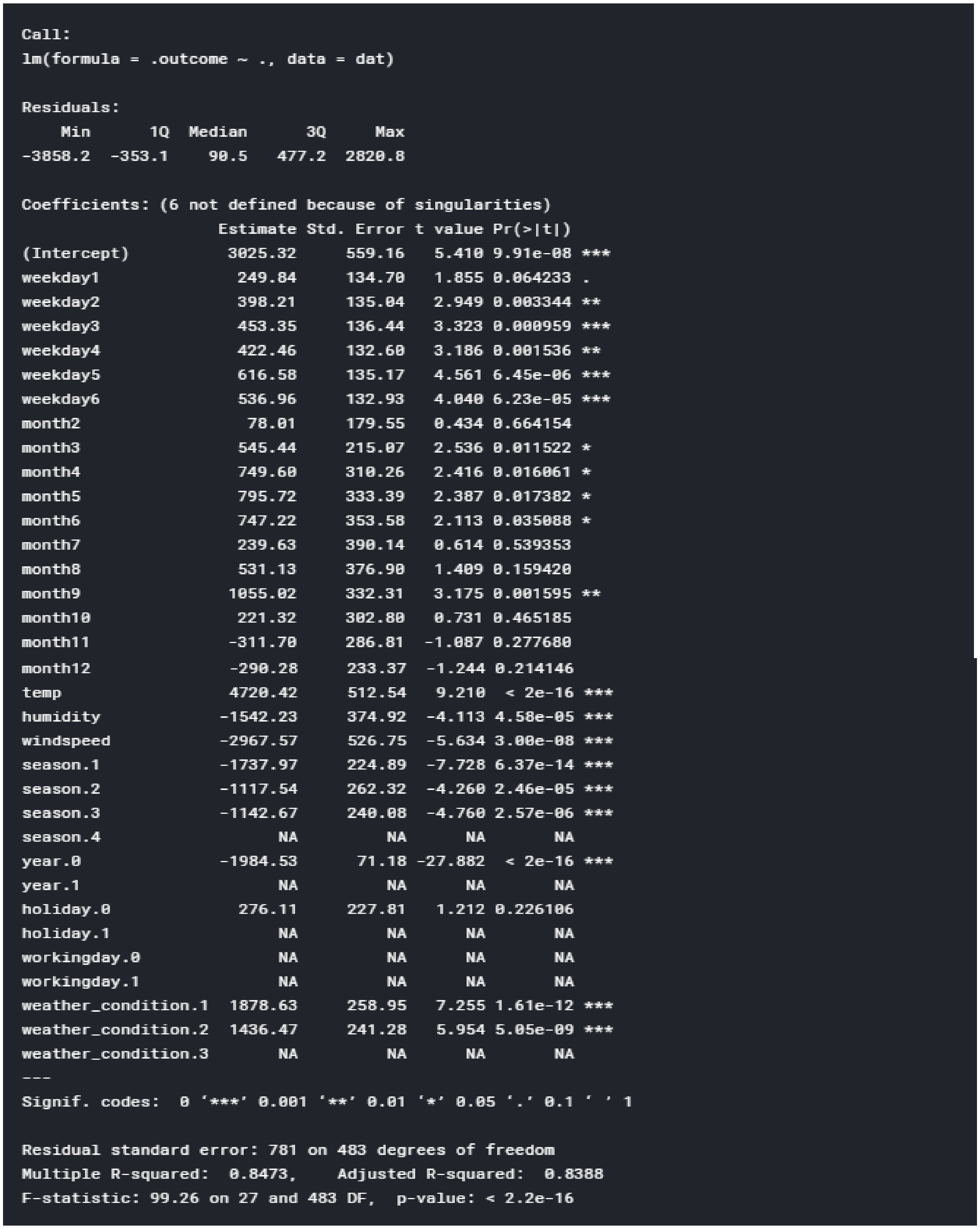
R code

Linear Regression model



```
Call:
lm(formula = train_encoded_attributes$total_count ~ ., data = train_encoded_attributes[,
    -c(6)])

Residuals:
    Min      1Q  Median      3Q     Max 
-3858.2  -353.1    90.5   477.2  2820.8 

Coefficients: (6 not defined because of singularities)
                     Estimate Std. Error t value Pr(>|t|)    
(Intercept)           3025.32     559.16   5.410 9.91e-08 ***
weekday1               249.84     134.70   1.855 0.064233 .  
weekday2               398.21     135.04   2.949 0.003344 ** 
weekday3               453.35     136.44   3.323 0.000959 ***
weekday4               422.46     132.60   3.186 0.001536 ** 
weekday5               616.58     135.17   4.561 6.45e-06 ***
weekday6               536.96     132.93   4.040 6.23e-05 ***
month2                  78.01     179.55   0.434 0.664154    
month3                 545.44     215.07   2.536 0.011522 *  
month4                 749.60     310.26   2.416 0.016061 *  
month5                 795.72     333.39   2.387 0.017382 *  
month6                 747.22     353.58   2.113 0.035088 *  
month7                 239.63     390.14   0.614 0.539353    
month8                 531.13     376.90   1.409 0.159420    
month9                1055.02     332.31   3.175 0.001595 ** 
month10                221.32     302.80   0.731 0.465185    
month11               -311.70     286.81  -1.087 0.277680    
month12               -290.28     233.37  -1.244 0.214146    
temp                  4720.42     512.54   9.210  < 2e-16 ***
humidity             -1542.23     374.92  -4.113 4.58e-05 ***
windspeed            -2967.57     526.75  -5.634 3.00e-08 ***
season.1             -1737.97     224.89  -7.728 6.37e-14 ***
season.2             -1117.54     262.32  -4.260 2.46e-05 ***
season.3             -1142.67     240.08  -4.760 2.57e-06 ***
season.4                   NA         NA      NA       NA    
year.0               -1984.53      71.18 -27.882  < 2e-16 ***
year.1                     NA         NA      NA       NA    
holiday.0              276.11     227.81   1.212 0.226106    
holiday.1                  NA         NA      NA       NA    
workingday.0               NA         NA      NA       NA    
workingday.1               NA         NA      NA       NA    
weather_condition.1  1878.63     258.95   7.255 1.61e-12 ***
weather_condition.2  1436.47     241.28   5.954 5.05e-09 ***
weather_condition.3        NA         NA      NA       NA    
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 781 on 483 degrees of freedom
Multiple R-squared:  0.8473,	Adjusted R-squared:  0.8388 
F-statistic: 99.26 on 27 and 483 DF,  p-value: < 2.2e-16
```

Cross validation prediction



```
Call:
lm(formula = .outcome ~ ., data = dat)

Residuals:
    Min      1Q  Median      3Q     Max 
-3858.2  -353.1    90.5   477.2  2820.8 

Coefficients: (6 not defined because of singularities)
                    Estimate Std. Error t value Pr(>|t|)    
(Intercept)          3025.32     559.16   5.410 9.91e-08 ***
weekday1              249.84     134.70   1.855 0.064233 .  
weekday2              398.21     135.04   2.949 0.003344 ** 
weekday3              453.35     136.44   3.323 0.000959 ***
weekday4              422.46     132.60   3.186 0.001536 ** 
weekday5              616.58     135.17   4.561 6.45e-06 ***
weekday6              536.96     132.93   4.040 6.23e-05 ***
month2                 78.01     179.55   0.434 0.664154    
month3                545.44     215.07   2.536 0.011522 *  
month4                749.60     310.26   2.416 0.016061 *  
month5                795.72     333.39   2.387 0.017382 *  
month6                747.22     353.58   2.113 0.035088 *  
month7                239.63     390.14   0.614 0.539353    
month8                531.13     376.90   1.409 0.159420    
month9               1055.02     332.31   3.175 0.001595 ** 
month10               221.32     302.80   0.731 0.465185    
month11              -311.70     286.81  -1.087 0.277680    
month12              -290.28     233.37  -1.244 0.214146    
temp                 4720.42     512.54   9.210  < 2e-16 ***
humidity            -1542.23     374.92  -4.113 4.58e-05 ***
windspeed           -2967.57     526.75  -5.634 3.00e-08 ***
season.1            -1737.97     224.89  -7.728 6.37e-14 ***
season.2            -1117.54     262.32  -4.260 2.46e-05 ***
season.3            -1142.67     240.08  -4.760 2.57e-06 ***
season.4                  NA         NA      NA       NA    
year.0              -1984.53      71.18 -27.882  < 2e-16 ***
year.1                    NA         NA      NA       NA    
holiday.0             276.11     227.81   1.212 0.226106    
holiday.1                 NA         NA      NA       NA    
workingday.0              NA         NA      NA       NA    
workingday.1              NA         NA      NA       NA    
weather_condition.1  1878.63     258.95   7.255 1.61e-12 ***
weather_condition.2  1436.47     241.28   5.954 5.05e-09 ***
weather_condition.3       NA         NA      NA       NA    
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 781 on 483 degrees of freedom
Multiple R-squared:  0.8473,	Adjusted R-squared:  0.8388 
F-statistic: 99.26 on 27 and 483 DF,  p-value: < 2.2e-16
```

The adjusted R-squared or coefficient of determination is 0.84 on average for 3-fold cross validation it means that predictor is only able to predict 84% of the variance in the target variable which is contributed by all the independent variables. The value of p-value is less than 0.05, means that it rejects the assumption of null hypothesis.

2.2.3 Decision Tree Regression

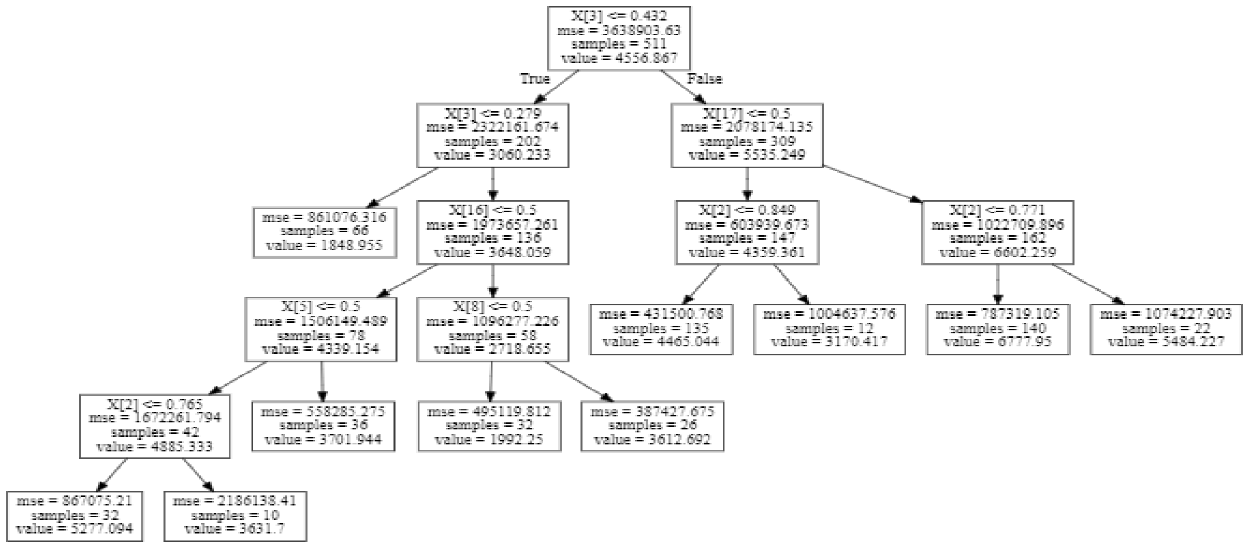
We will use a regression tree to predict the values of our target variable.

Python code

```
#training the model
dtr=DecisionTreeRegressor(min_samples_split=2,max_leaf_nodes=10)

#Fit the trained model
dtr.fit(X_train,y_train)

#Accuracy score of the model
dtr_score=dtr.score(X_train,y_train)
Accuracy score : 0.80

# Cross validation prediction
predict=cross_val_predict(dtr,X_train,y_train,cv=3)

#R-squared scores
r2_scores = cross_val_score(dtr, X_train, y_train, cv=3)
r2_scores=np.avearge(r2_scores)
r2_scores : 0.746
dot_data = tree.export_graphviz(dtr, out_file=None) # plot the learned model
graph = graphviz.Source(dot_data)
```

The R-squared or coefficient of determination is 0.74 on average for 3-fold cross validation, it means that predictor is only able to predict 74% of the variance in the target variable which is contributed by all the independent variables.

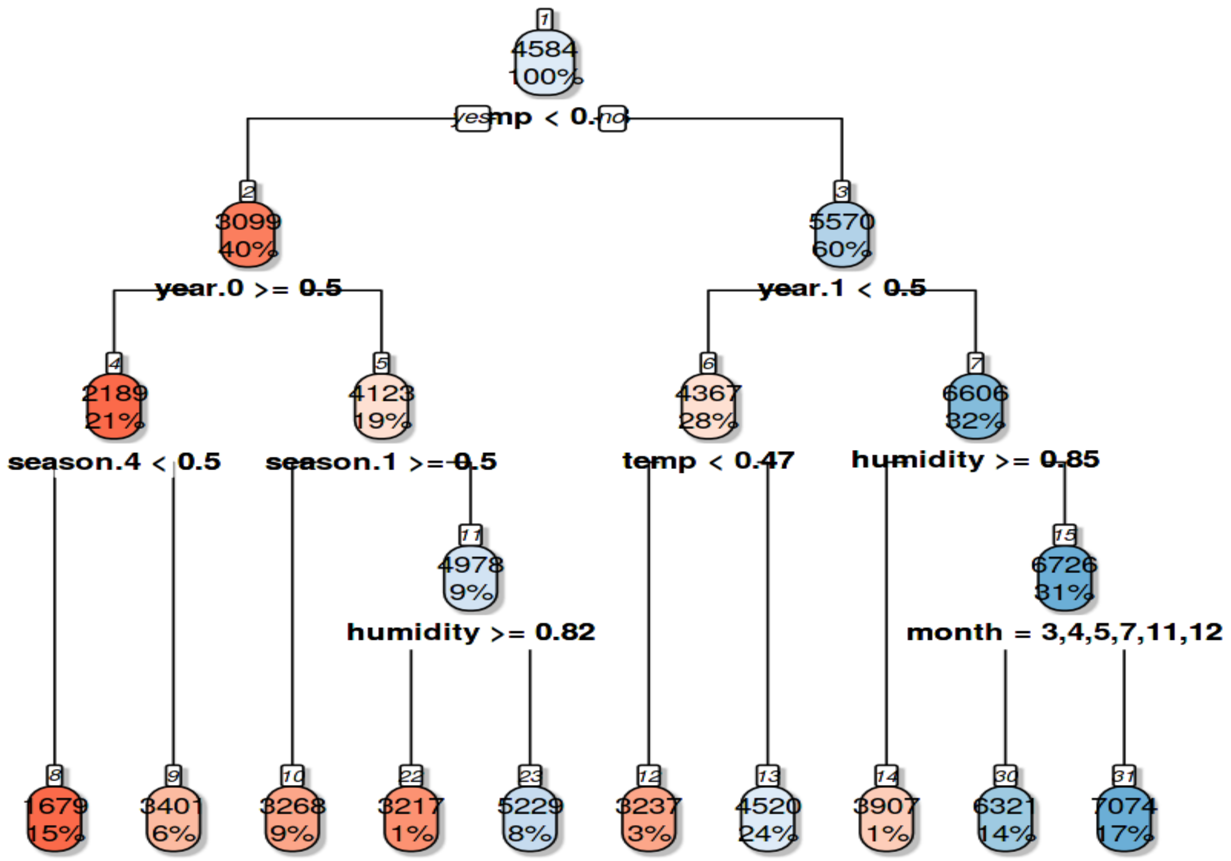
Decision tree plot



R code

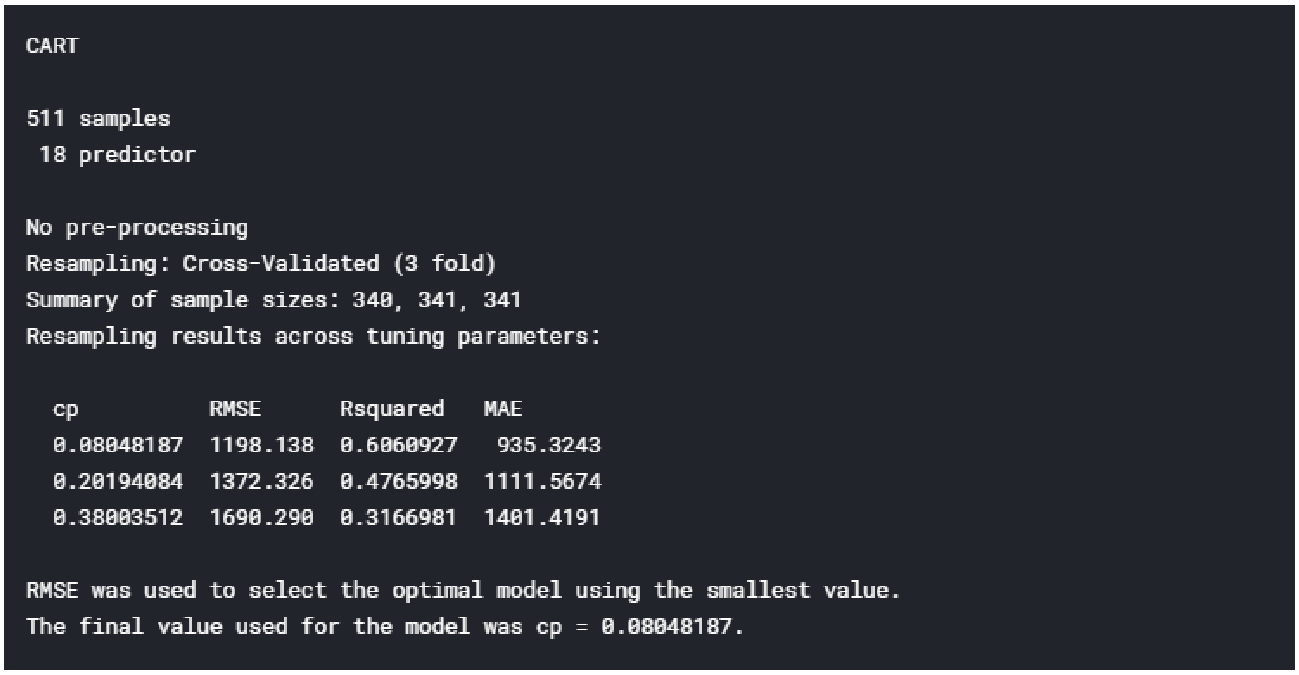
```
#Training model

set.seed(568)
#load the rpart library for decision trees
library(rpart)

#rpart.control to contro the performance of model
rpart.control<-rpart.control(minbucket = 2,cp = 0.01,maxcompete = 3,
maxsurrogate = 4, usesurrogate = 2, xval = 3,surrogatestyle = 0, maxdepth = 10)

#training the dtr model
dtr<-rpart(train_encoded_attributes$total_count~.,data=train_encoded_attributes
[,-c(6)],control=rpart.control,method='anova',cp=0.01)

#load the rpart.plot for plot the learned dtr model
library(rpart.plot)
rpart.plot(dtr, box.palette="RdBu", shadow.col="gray", nn=TRUE,roundint=FALSE)
```

Decision tree plot



Cross validation prediction

```
set.seed(5769)
#cross validation resampling method
train.control<-trainControl(method='CV',number=3)
#cross validation pred
dtr_CV_predict<-train(total_count~.,data=train_encoded_attributes,
method='rpart',trControl=train.control)
```



```
CART

511 samples
 18 predictor

No pre-processing
Resampling: Cross-Validated (3 fold)
Summary of sample sizes: 340, 341, 341
Resampling results across tuning parameters:

  cp          RMSE      Rsquared   MAE
  0.08048187  1198.138  0.6060927   935.3243
  0.20194084  1372.326  0.4765998  1111.5674
  0.38003512  1690.290  0.3166981  1401.4191

RMSE was used to select the optimal model using the smallest value.
The final value used for the model was cp = 0.08048187.
```

The R-squared or coefficient of determination is 0.60 on average for 3-fold cross validation , it means that predictor is only able to predict 60% of the variance in the target variable which is contributed by all the independent variables.

2.2.4 Random Forest

We will use a Random Forest to predict the values of our target variable.

Python code

```
#Training the model
 X_train=train_encoded_attributes
 rf=RandomForestRegressor(n_estimators=200)

 #Fit the trained model
 rf.fit(X_train,y_train)

 #accuracy of the model
 rf_score =rf.score(X_train,y_train)

 Accuracy of the model : 0.98
```

Cross validation prediction

```
#Cross validation prediction
predict=cross_val_predict(rf,X_train,y_train,cv=3)

#R-squared scores
r2_scores = cross_val_score(rf, X_train, y_train, cv=3)
r2_scores=np.average(r2_scores)

r2_scores : 0.85
```

The R-squared or coefficient of determination is 0.85 on average for 3-fold cross validation , it means that predictor is only able to predict 85% of the variance in the target variable which is contributed by all the independent variables.

After comparing all three models using some metrics, we finally choose random forest is best for predicting the bike rental count on daily basis.

Chapter 3

Conclusion

3.1 Model Evaluation

Now, we have a three models for predicting the target variable, but we need to decide which model better for this project. There are many metrics used for model evaluation.

Predictive performance can be measured by comparing predictions of the models with true values of the target variables, and calculating some average error measure.

In this project, we are using three metrics for model evaluation as follows,

1. R-squared or Coefficient of Determination (R^2) :- It explains the how much variance of dependent variable which is contributed by all the independent variables.

$$R^2 = 1 - \frac{\sum_{i=1}^{n}\left(y_{true} - y_{pred}\right)^2}{\sum_{i=1}^{n}(y_{true} - \bar{y}_{true})^2} = 1 - \frac{variance_{residuals}}{variance_{total}}$$

Here, $\bar{y}_{true}$ is the mean of true dependent variable values

2. Root Mean Square Error (RMSE) :- It is the square root of the mean of the square of difference between the true and predicted dependent variable values.

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{n}\left({y_{true}}^2 - {y_{pred}}^2\right)}{n}}$$

3. Mean Absolute Error (MEA) :- It is the mean of the absolute difference between the true and predicted dependent variable values.

$$\text{MAE} = \left|\frac{\sum_{i=1}^{n}\left(y_{true} - y_{pred}\right)}{n}\right|$$

3.1.1 Root Mean Squared Error (RMSE)

RMSE can be obtained by using R and Python as follows,

```
 Linear Regression

R code
 rmse = RMSE(lm_predict, y_test)
 [1] 803.0122

 Python code
 rmse=math.sqrt(metrics.mean_squared_error(y_test,lr_pred))
 [1] 783.061

 Decision Tree Regressor

 R code
 rmse<-RMSE(y_test,dtr_predict)
 [1] 952.7179

 Python code
 rmse=math.sqrt(metrics.mean_squared_error(y_test,dtr_pred))
 [1] 925.7976

 Random Forest

 R code
 rmse<-RMSE(y_test,rf_predict)
 [1] 683.9306

 Python code
 rmse = math.sqrt(metrics.mean_squared_error(y_test,rf_pred))
 [1] 632.5709
```

3.1.2 Mean Absolute Error (MAE)

MAE can be obtained using R code and Python code as follows

```
Linear Regression

R code
mae <-MAE(lm_predict, y_test)
1] 571.6949

Python code
mae=metrics.mean_absolute_error(y_test,lr_pred)
[1] 594.7368

Decision Tree Regressor

R code
mae<-MAE(y_test,dtr_predict)
[1] 694.0827

Python code
mae=metrics.mean_absolute_error(y_test,dtr_pred)
[1] 667.8168

Random Forest

R code
mae<-MAE(y_test,rf_predict)
[1] 487.8721

Python code
mae=metrics.mean_absolute_error(y_test,rf_pred)
[1] 422.0464
```

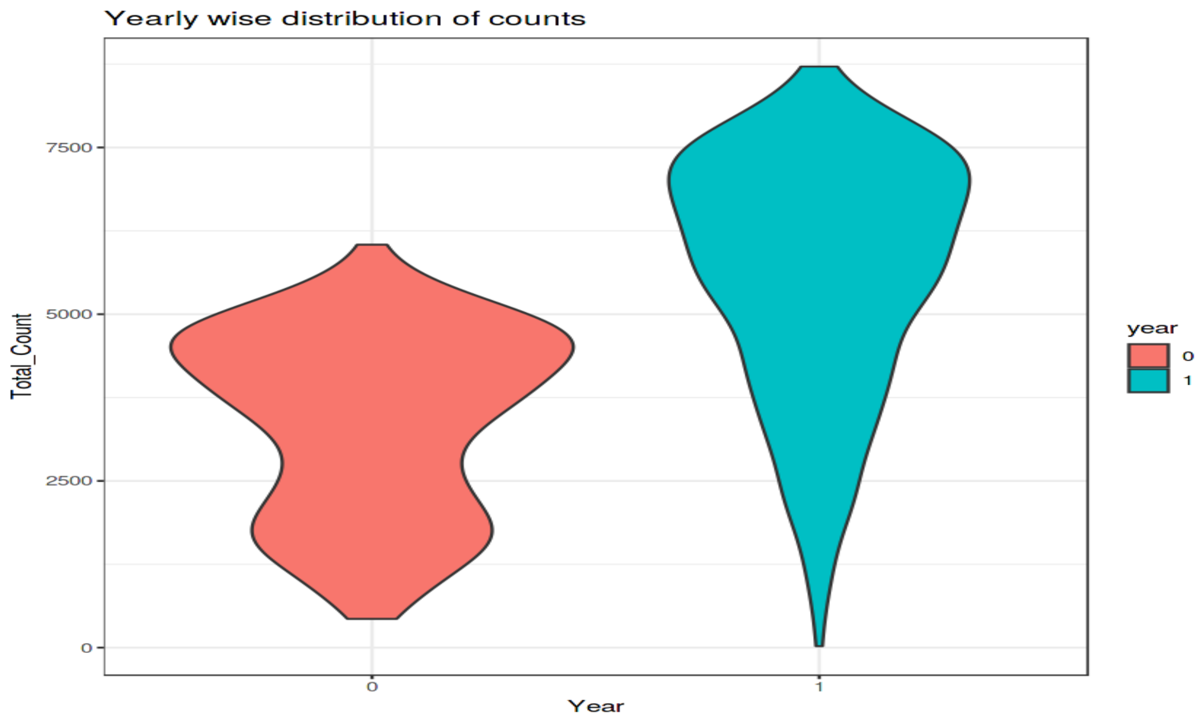
3.2 Model Selection

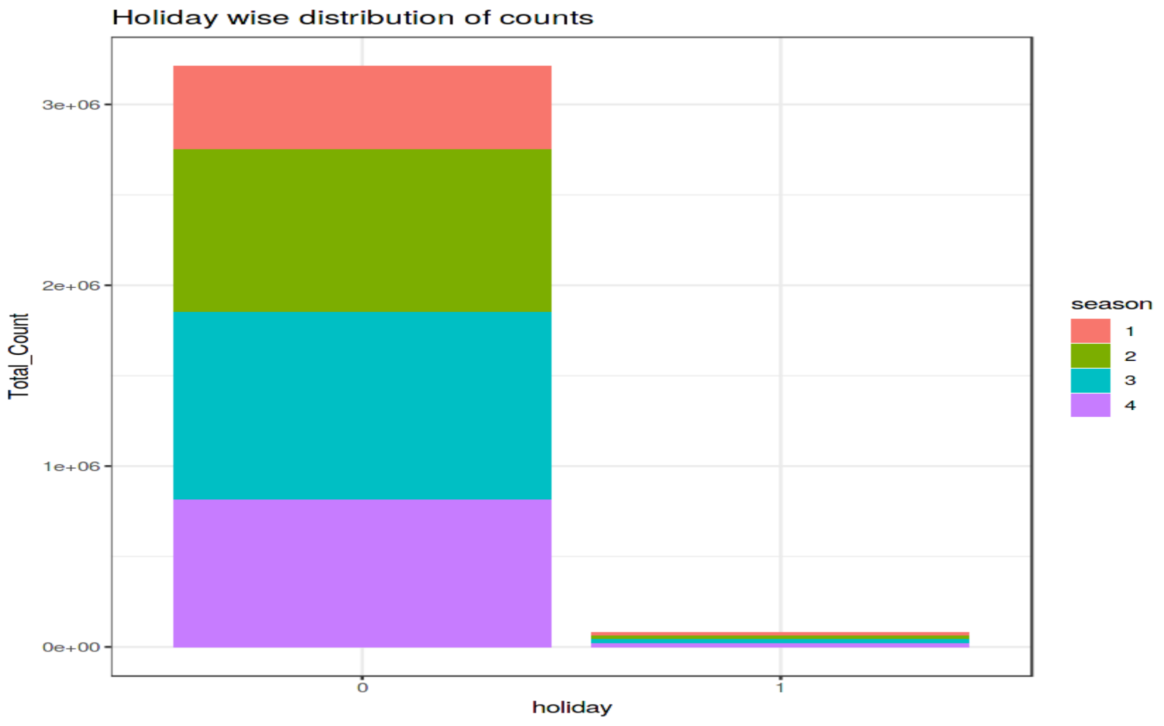
Model	**RMSE (Python)**	**RMSE (R)**	**%error**	**MAE (Python)**	**MAE (R)**	**%error**
Linear Regression	783.061	803.012	2.48	594.736	571.694	3.87
Decision Tree Regressor	925.797	952.717	2.82	667.816	694.082	3.78
Random Forest	632.57	683.93	7.5	422.046	487.872	13.5

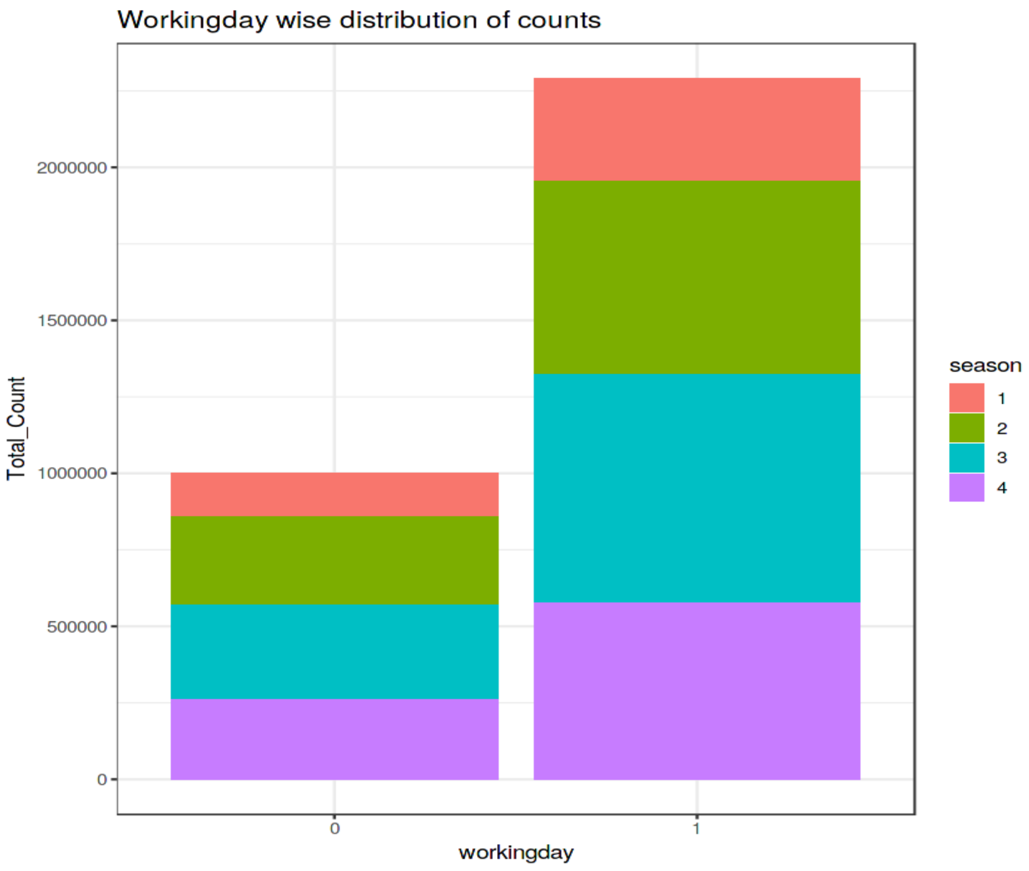
%error is the percentage of RMSE and MAE between python and R code for different models.

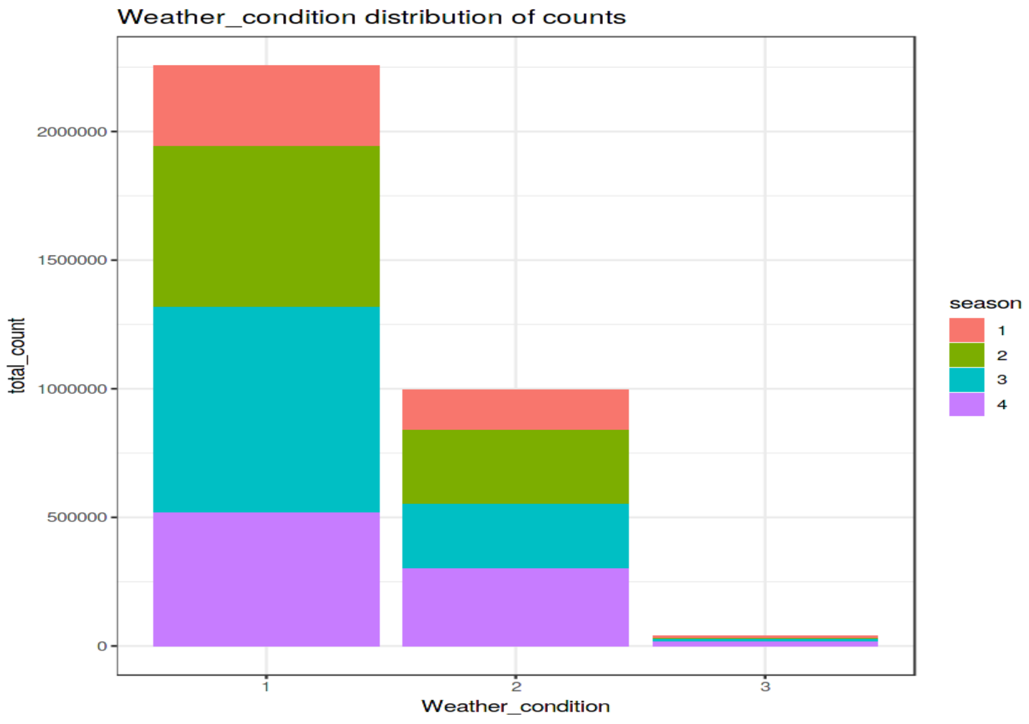
When we compare the root mean squared errors and mean absolute errors of all three models, the random forest model has less RMSE and MAE errors. So, finally random forest model is best for predicting the bike rental count on daily basis.

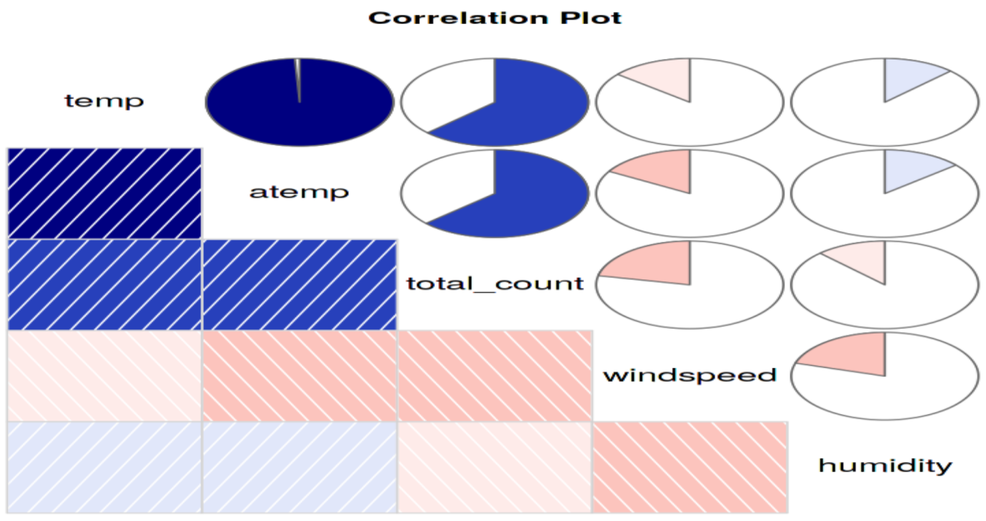
Appendix A - Extra Figures

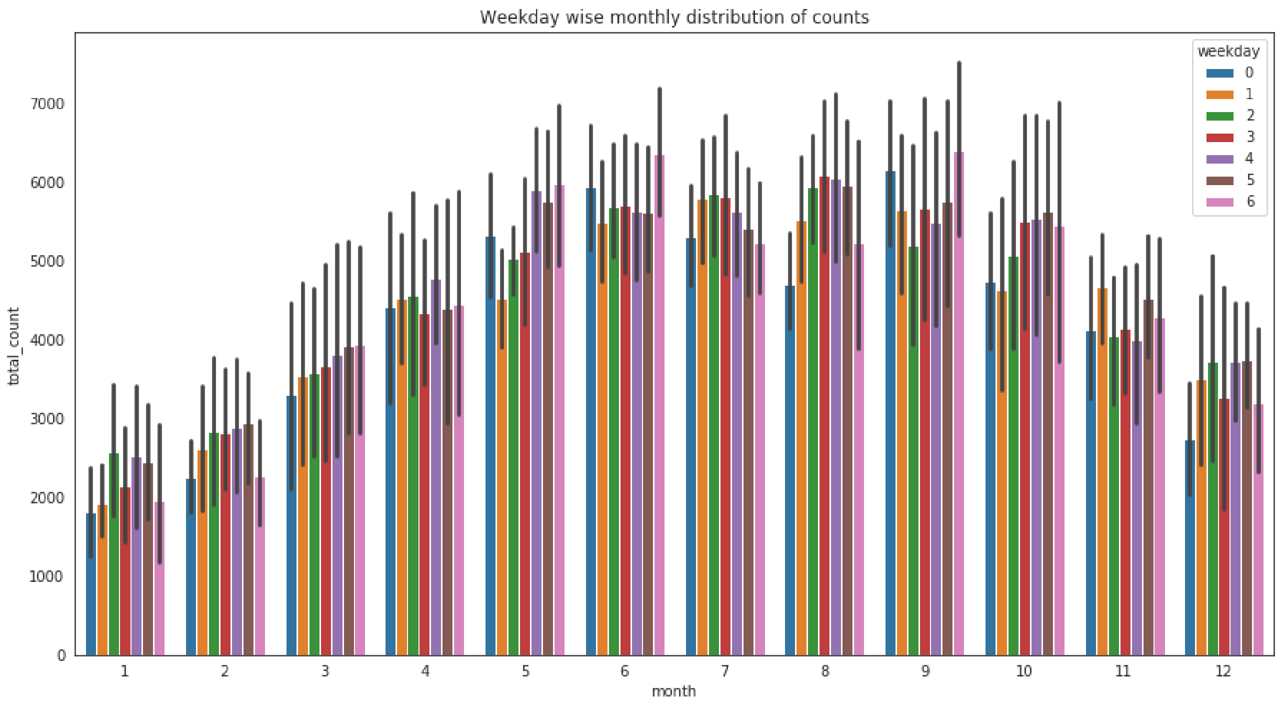


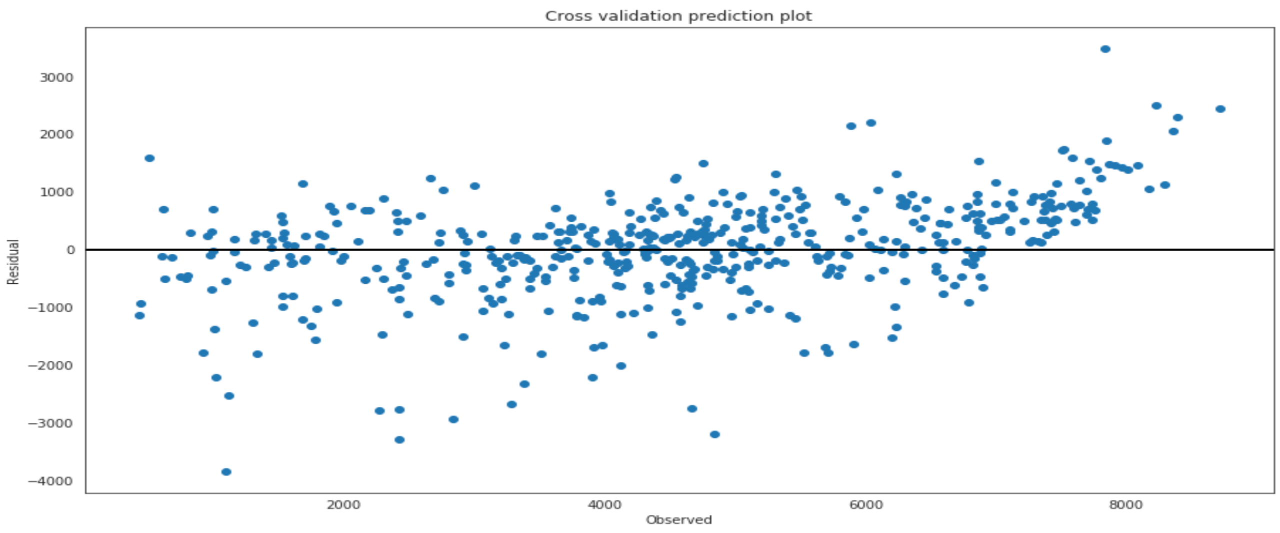


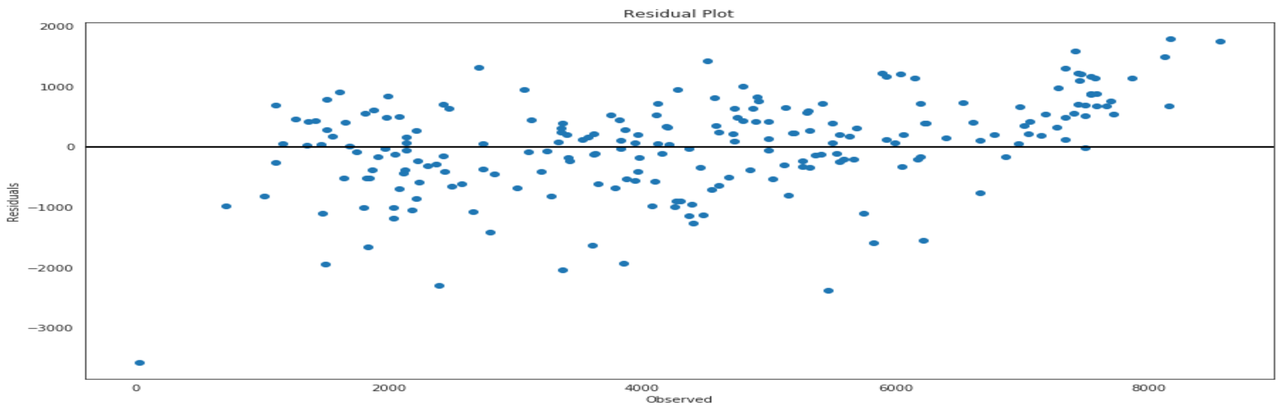

Workingday wise distribution of counts
Total_Count
2000000
1500000
1000000
500000
0
0
1
workingday
season
1
2
3
4

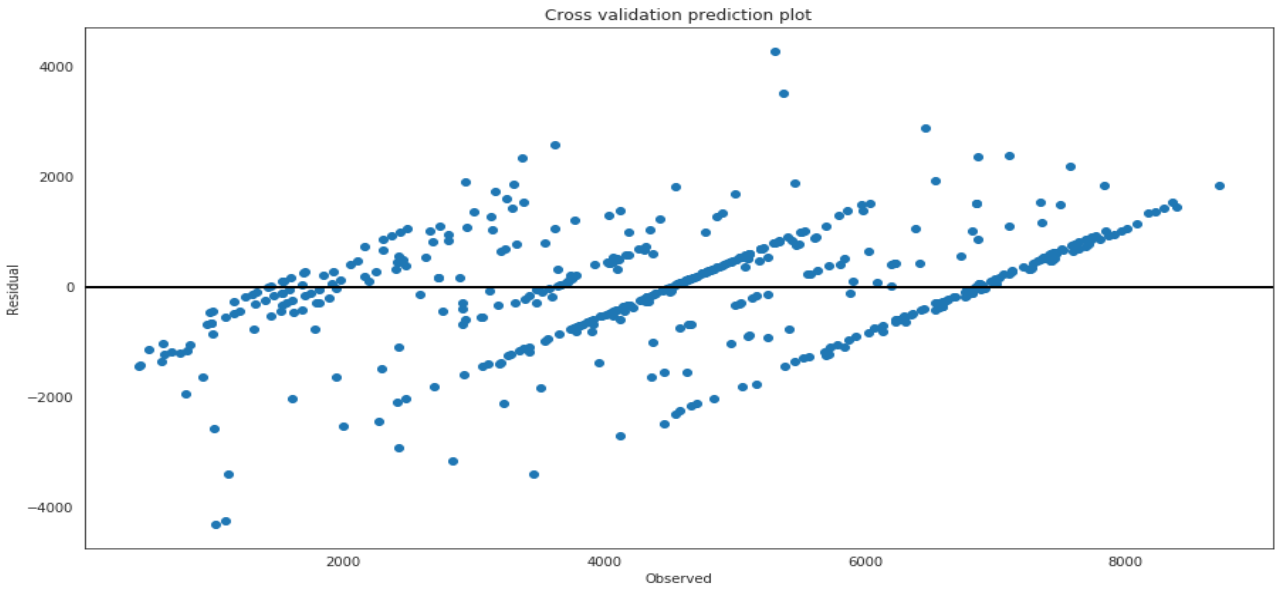

Weather_condition distribution of counts
total_count
2000000
1500000
1000000
500000
0
1
2
3
Weather_condition
season
1
2
3
4

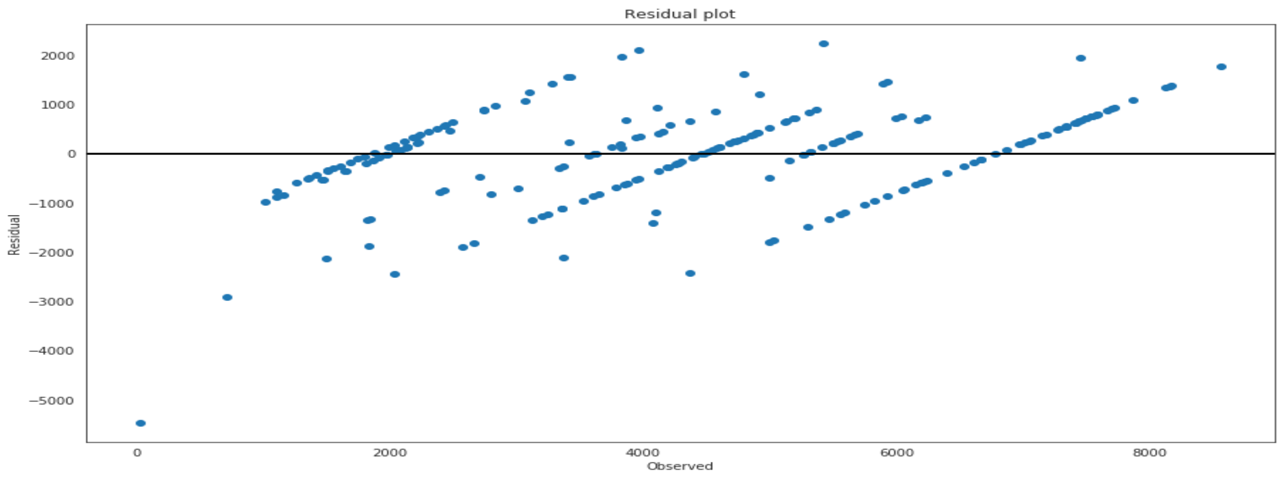

Correlation Plot
temp
atemp
total_count
windspeed
humidity

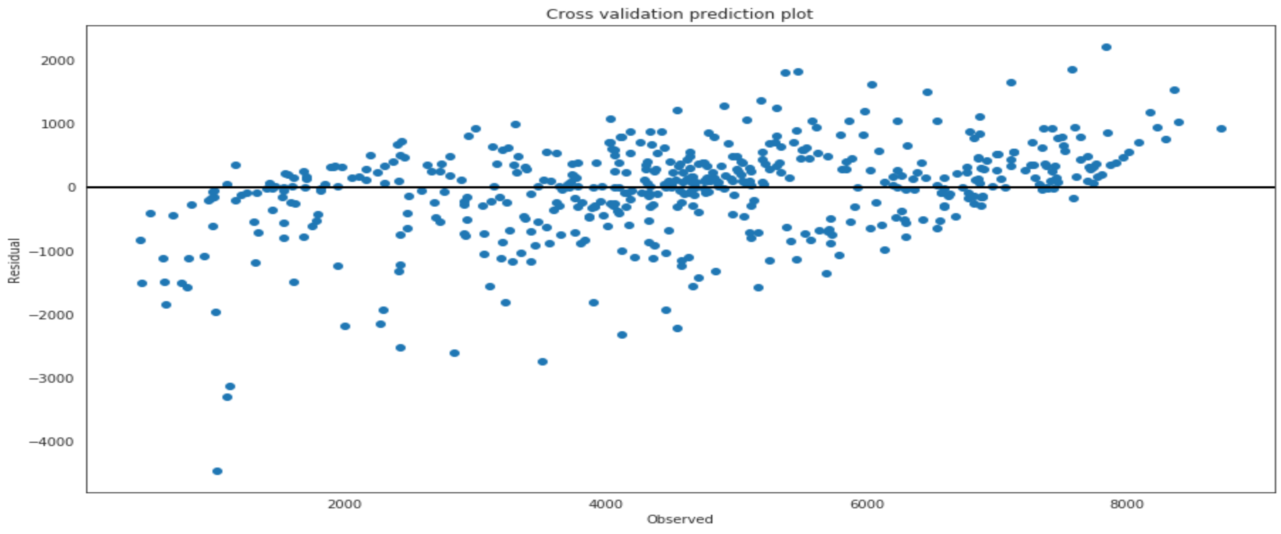

Weekday wise monthly distribution of counts
weekday
0
1
2
3
4
5
6
total_count
0
1000
2000
3000
4000
5000
6000
7000
month
1
2
3
4
5
6
7
8
9
10
11
12

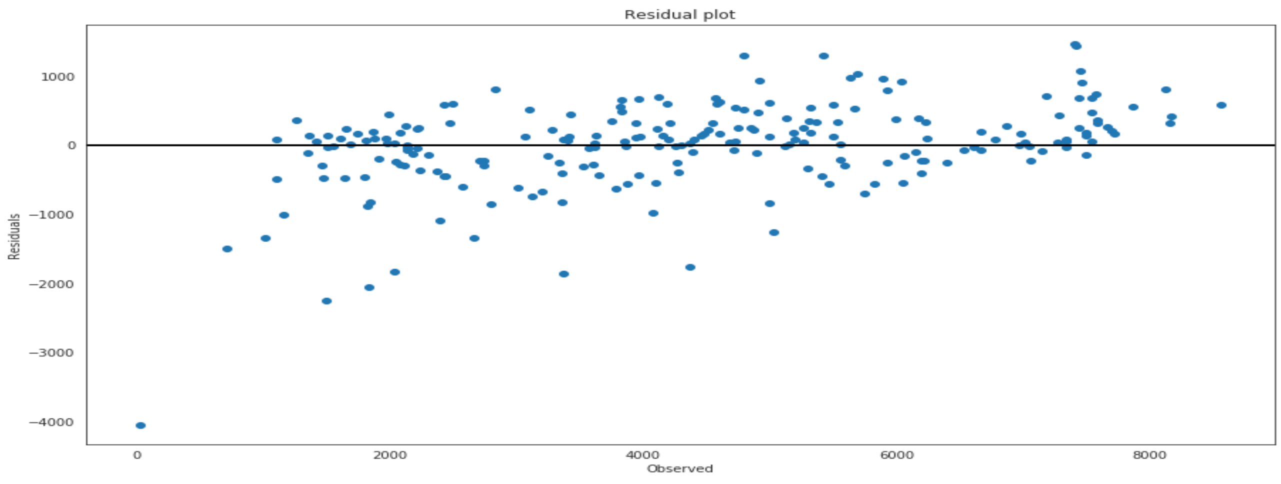

Cross validation prediction plot
Residual
Observed
3000
2000
1000
0
−1000
−2000
−3000
−4000
2000
4000
6000
8000


Residual Plot
Residuals
Observed
2000
1000
0
−1000
−2000
−3000
0
2000
4000
6000
8000


Cross validation prediction plot
Residual
Observed
4000
2000
0
−2000
−4000
2000
4000
6000
8000


Residual plot
Residual
Observed
2000
1000
0
−1000
−2000
−3000
−4000
−5000
0
2000
4000
6000
8000


Cross validation prediction plot
Residual
Observed
2000
1000
0
−1000
−2000
−3000
−4000
2000
4000
6000
8000


Residual plot
Residuals
Observed
1000
0
−1000
−2000
−3000
−4000
0
2000
4000
6000
8000

Appendix B – Complete Python and R Code

Python Code

Exploratory Data Analysis

```python
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn import preprocessing,metrics,linear_model
from sklearn.model_selection import cross_val_score,cross_val_predict


#import the csv file
bike_df=pd.read_csv("../input/day.csv")

#Shape of the dataset
bike_df.shape

#Data types
bike_df.dtypes

#Rename the columns
bike_df.rename(columns={'instant':'rec_id','dteday':'datetime','yr':'year',
'mnth':'month','weathersit':'weather_condition','hum':'humidity',
  'cnt':'total_count'},inplace=True)

#Type casting the datetime and numerical attributes to category

bike_df['datetime']=pd.to_datetime(bike_df.datetime)

bike_df['season']=bike_df.season.astype('category')
bike_df['year']=bike_df.year.astype('category')
bike_df['month']=bike_df.month.astype('category')
bike_df['holiday']=bike_df.holiday.astype('category')
bike_df['weekday']=bike_df.weekday.astype('category')
bike_df['workingday']=bike_df.workingday.astype('category')
bike_df['weather_condition']=bike_df.weather_condition.astype('category')

#Missing values in dataset
bike_df.isnull().sum()
```

Attributes distributions and trends

```
# Monthly distribution of counts
fig,ax=plt.subplots(figsize=(15,8))
sns.set_style('white')
#Bar plot for seasonwise monthly distribution of counts
sns.barplot(x='month',y='total_count',data=bike_df[['month','total_count',
'season ']],hue='season',ax=ax)
ax.set_title('Seasonwise monthly distribution of counts')
plt.show()
#Bar plot for weekday wise monthly distribution of counts
sns.barplot(x='month',y='total_count',data=bike_df[['month','total_count',
'weekday']],hue='weekday',ax=ax1)
ax1.set_title('Weekday wise monthly distribution of counts')
plt.show()

#Violin plot for yearly distribution of counts
sns.violinplot(x='year',y='total_count',data=bike_df[['year','total_count']])
ax.set_title('Yearly distribution of counts')
plt.show()

#Barplot for Holiday distribution of counts
sns.barplot(data=bike_df,x='holiday',y='total_count',hue='season')
ax.set_title('Holiday wise distribution of counts')
plt.show()

#Bar plot for workingday distribution of counts
sns.barplot(data=bike_df,x='workingday',y='total_count',hue='season')
ax.set_title('Workingday wise distribution of counts')
plt.show()

#Bar plot for weather_condition distribution of counts
sns.barplot(x='weather_condition',y='total_count',data=bike_df[['month',
'total_co unt','weather_condition']],ax=ax1)
ax1.set_title('Weather_condition wise monthly distribution of counts')
plt.show()
```

Outlier Analysis

```
#Boxplot for total_count outliers
sns.boxplot(data=bike_df[['total_count']])
ax.set_title('total_count outliers')
plt.show()

#Box plot for Temp,windspeed and humidity_outliers
sns.boxplot(data=bike_df[['temp','windspeed','humidity']])
ax.set_title('Temp_windspeed_humidity_outiers')
plt.show()

#Replace and imputate the outliers
from fancyimpute import KNN

#create dataframe for outliers
wind_hum=pd.DataFrame(bike_df,columns=['windspeed','humidity'])
#Cnames for outliers
cnames=['windspeed','humidity']

for i in cnames:
   q75,q25=np.percentile(wind_hum.loc[:,i],[75,25]) # Divide data into
             75%quantil  e and 25%quantile.
   iqr=q75-q25 #Inter quantile range
   min=q25-(iqr*1.5) #inner fence
   max=q75+(iqr*1.5) #outer fence
   wind_hum.loc[wind_hum.loc[:,i]<min,:i]=np.nan  #Replace with NA
   wind_hum.loc[wind_hum.loc[:,i]>max,:i]=np.nan  #Replace with NA

#Imputating the outliers by mean Imputation
wind_hum['windspeed']=wind_hum['windspeed'].fillna(wind_hum['windspeed'].mean())
wind_hum['humidity']=wind_hum['humidity'].fillna(wind_hum['humidity'].mean())

#Replacing the imputated windspeed
bike_df['windspeed']=bike_df['windspeed'].replace(wind_hum['windspeed'])
#Replacing the imputated humidity
bike_df['humidity']=bike_df['humidity'].replace(wind_hum['humidity'])
bike_df.head(5)
```

Normality test and correlation matrix

```
import scipy
from scipy import stats
stats.probplot(bike_df.total_count.tolist(),dist='norm',plot=plt)
plt.show()
#Create the correlation matrix
correMtr=bike_df[["temp","atemp","humidity","windspeed","casual","registered"
,"total_count"]].corr()
mask=np.array(correMtr)
mask[np.tril_indices_from(mask)]=False
#Heat map for correlation matrix of attributes
sns.heatmap(correMtr,mask=mask,vmax=0.8,square=True,annot=True,ax=ax)
ax.set_title('Correlation matrix of attributes')
plt.show
```

Splitting the dataset

```
#Split the dataset into the train and test data
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(bike_df.iloc[:,0:-3],
 bike_df.iloc[:,-1],test_size=0.3, random_state=42)

#Reset train index values
X_train.reset_index(inplace=True)
y_train=y_train.reset_index()

# Reset train index values
X_test.reset_index(inplace=True)
y_test=y_test.reset_index()
```

Encoding the categorical attributes

```
#Create a new dataset for train attributes
train_attributes=X_train[['season','month','year','weekday','holiday',
'workingday ','weather_condition','humidity','temp','windspeed']]
#Create a new dataset for test attributes
test_attributes=X_test[['season','month','year','weekday','holiday',
'workingday','humidity','temp','windspeed','weather_condition']]
#categorical attributes
cat_attributes=['season','holiday','workingday','weather_condition','year']
#numerical attributes
num_attributes=['temp','windspeed','humidity','month','weekday']

#To get dummy variables to encode the categorical features to numeric
train_encoded_attributes=pd.get_dummies(train_attributes,columns=cat_attribut
es)
test_encoded_attributes=pd.get_dummies(test_attributes,columns=cat_attributes
)
```

Linear Regression Model

```
#Training dataset for modelling
X_train=train_encoded_attributes
y_train=y_train.total_count.values

#training model
lr_model=linear_model.LinearRegression()

#fit the trained model
lr_model.fit(X_train,y_train)
#Accuracy of the model
lr=lr_model.score(X_train,y_train)
print('Accuracy of the model :',lr)
```

```
#Cross validation prediction
predict=cross_val_predict(lr_model,X_train,y_train,cv=3)

#Cross validation plot
fig,ax=plt.subplots(figsize=(15,8))
ax.scatter(y_train,y_train-predict)
ax.axhline(lw=2,color='black')
ax.set_title('Cross validation prediction plot')
ax.set_xlabel('Observed')
ax.set_ylabel('Residual')
plt.show()

#R-squared scores
r2_scores = cross_val_score(lr_model, X_train, y_train, cv=3)
print('R-squared scores :',np.average(r2_scores))

 #Test dataset for prediction

X_test=test_encoded_attributes
y_test=y_test.total_count.values

#predict the model
lr_pred=lr_model.predict(X_test)

import math
#Root mean square error
rmse=math.sqrt(metrics.mean_squared_error(y_test,lr_pred))
#Mean absolute error
mae=metrics.mean_absolute_error(y_test,lr_pred)

#Residual plot
fig, ax = plt.subplots(figsize=(15,8))
ax.scatter(y_test, y_test-lr_pred)
ax.axhline(lw=2,color='black')
ax.set_xlabel('Observed')
ax.set_ylabel('Residuals')
ax.title.set_text("Residual Plot")
plt.show()
```

Decision tree regression

```
#training the model
from sklearn.tree import DecisionTreeRegressor
dtr=DecisionTreeRegressor(min_samples_split=2,max_leaf_nodes=10)

#Fit the trained model
dtr.fit(X_train,y_train)
#Accuracy score of the model
dtr_score=dtr.score(X_train,y_train)

#Plot the learned model
from sklearn import tree
import pydot
import graphviz

# export the learned model to tree
dot_data = tree.export_graphviz(dtr, out_file=None)
graph = graphviz.Source(dot_data)

# Cross validation prediction
predict=cross_val_predict(dtr,X_train,y_train,cv=3)

# Cross validation prediction plot
fig,ax=plt.subplots(figsize=(15,8))
ax.scatter(y_train,y_train-predict)
ax.axhline(lw=2,color='black')
ax.set_title('Cross validation prediction plot')
ax.set_xlabel('Observed')
ax.set_ylabel('Residual')
plt.show()
#R-squared scores
r2_scores = cross_val_score(dtr, X_train, y_train, cv=3)
print('R-squared scores :',np.average(r2_scores))
#predict the model
dtr_pred=dtr.predict(X_test)

#Root mean square error
rmse=math.sqrt(metrics.mean_squared_error(y_test,dtr_pred))
#Mean absolute error
mae=metrics.mean_absolute_error(y_test,dtr_pred)
#Residual scatter plot
residuals = y_test-dtr_pred
fig, ax = plt.subplots(figsize=(15,8))
ax.scatter(y_test, residuals)
ax.axhline(lw=2,color='black')
ax.set_xlabel('Observed')
ax.set_ylabel('Residual')
ax.set_title('Residual plot')
plt.show()
```

Random Forest

```
#Training the model
from sklearn.ensemble import RandomForestRegressor
X_train=train_encoded_attributes
rf=RandomForestRegressor(n_estimators=200)

#Fit the trained model
rf.fit(X_train,y_train)
#accuracy of the model
rf_score =rf.score(X_train,y_train)

#Cross validation prediction
predict=cross_val_predict(rf,X_train,y_train,cv=3)

#Cross validation prediction plot

fig,ax=plt.subplots(figsize=(15,8))
ax.scatter(y_train,y_train-predict)
ax.axhline(lw=2,color='black')
ax.set_title('Cross validation prediction plot')
ax.set_xlabel('Observed')
ax.set_ylabel('Residual')
plt.show()

#predict the model
X_test=test_encoded_attributes
rf_pred=rf.predict(X_test)

#Root mean square error
rmse = math.sqrt(metrics.mean_squared_error(y_test,rf_pred))
print('Root mean square error :',rmse)
#Mean absolute error
mae=metrics.mean_absolute_error(y_test,rf_pred)
print('Mean absolute error :',mae)

#Residual scatter plot
fig, ax = plt.subplots(figsize=(15,8))
residuals=y_test-rf_pred
ax.scatter(y_test, residuals)
ax.axhline(lw=2,color='black')
ax.set_xlabel('Observed')
ax.set_ylabel('Residuals')
ax.set_title('Residual plot')
plt.show()

# Final model for predicting the bike rental on daily basis
Bike_df1=pd.DataFrame(y_test,columns=['y_test'])
Bike_df2=pd.DataFrame(rf_pred,columns=['rf_pred'])
Bike_predictions=pd.merge(Bike_df1,Bike_df2,left_index=True,right_index=True)
Bike_predictions.to_csv('Bike_Renting_Python.csv')
```

R Code

Exploratory Data Analysis

```r
#import tidyverse metapackage library
library(tidyverse)

list.files(path = "../input")

#Importing the csv file
bike_df<-read.csv("../input/day.csv")
head(bike_df,5)

#Create new dataset excluding casual and registered variables
bike_df<-subset(bike_df,select=-c(casual,registered))
head(bike_df,5)

#Dimension of dataset
dim(bike_df)

#Summary of the dataset
summary(bike_df)

#Structure of dataset
str(bike_df)

#Rename the columns
names(bike_df)<-c('rec_id','datetime','season','year','month','holiday',
'weekday','workingday','weather_condition','temp','atemp','humidity',
'windspeed','total_count')

#Typecasting the datetime and numerical attributes to category

bike_df$datetime<- as.Date(bike_df$datetime)
bike_df$year<-as.factor(bike_df$year)
bike_df$month<-as.factor(bike_df$month)
bike_df$season <- as.factor(bike_df$season)
bike_df$holiday<- as.factor(bike_df$holiday)
bike_df$weekday<- as.factor(bike_df$weekday)
bike_df$workingday<- as.factor(bike_df$workingday)
bike_df$weather_condition<- as.factor(bike_df$weather_condition)

#Missing values in dataset
missing_val<-data.frame(apply(bike_df,2,function(x){sum(is.na(x))}))
names(missing_val)[1]='missing_val'
```

Attributes distributions and trends

```
 #Monthly distribution of counts

 #column plot for season wise monthly distribution of counts
 ggplot(bike_df,aes(x=month,y=total_count,fill=season))+theme_bw()+geom_col()+
 labs(x='Month',y='Total_Count',title='Season wise monthly distribution of coun
ts')
 #column plot for weekday wise monthly distribution of counts
 ggplot(bike_df,aes(x=month,y=total_count,fill=weekday))+theme_bw()+geom_col()+
 labs(x='Month',y='Total_Count',title='Weekday wise monthly distribution of cou
nts')

 #Violin plot for Yearly wise distribution of counts
 ggplot(bike_df,aes(x=year,y=total_count,fill=year))+geom_violin()+theme_bw()+
 labs(x='Year',y='Total_Count',title='Yearly wise distribution of counts')

 #Column plot for holiday wise distribution of counts
 ggplot(bike_df,aes(x=holiday,y=total_count,fill=season))+geom_col()+theme_bw()
+labs(x='holiday',y='Total_Count',title='Holiday wise distribution of counts')

 #Column plot for workingday wise distribution of counts
 ggplot(bike_df,aes(x=workingday,y=total_count,fill=season))+geom_col()+
 theme_bw()+labs(x='workingday',y='Total_Count',
 title='Workingday wise distribution of counts')

 #Column plot for weather_condition distribution of counts
 ggplot(bike_df,aes(x=weather_condition,y=total_count,fill=season))+geom_col()+
 them e_bw()+labs(x='Weather_condition',y='total_count',
 title='Weather_condition distribution of counts')
```

Outlier analysis

```
 #boxplot for total_count_outliers
 par(mfrow=c(1, 1))#divide graph area in 1 columns and 1 rows
 boxplot(bike_df$total_count,main='Total_count',
 sub=paste(boxplot.stats(bike_df$total_count)$out))

 #box plots for outliers
 par(mfrow=c(2,2))
 #Box plot for temp outliers
 boxplot(bike_df$temp, main="Temp",sub=paste(boxplot.stats(bike_df$temp)$out))
 #Box plot for humidity outliers
 boxplot(bike_df$humidity,main="Humidity",
 sub=paste(boxplot.stats(bike_df$humidity)$out))
 #Box plot for windspeed outliers
 boxplot(bike_df$windspeed,main="Windspeed",
 sub=paste(boxplot.stats(bike_df$windspeed)$out))
```

```
Replace and imputate the outliers
#load the DMwR library
library(DMwR)
#create subset for windspeed and humidity variable
wind_hum<-subset(bike_df,select=c('windspeed','humidity'))
#column names of wind_hum
cnames<-colnames(wind_hum)
for(i in cnames){
   val=wind_hum[,i][wind_hum[,i] %in% boxplot.stats(wind_hum[,i])$out]
   wind_hum[,i][wind_hum[,i] %in% val]= NA  # Replace outliers with NA
}
#Imputating the missing values using mean imputation method
wind_hum$windspeed[is.na(wind_hum$windspeed)]<-mean(wind_hum$windspeed,na.rm=T
)
wind_hum$humidity[is.na(wind_hum$humidity)]<-mean(wind_hum$humidity,na.rm=T)

#Remove the windspeed and humidity variable in order to replace imputated data
new_df<-subset(bike_df,select=-c(windspeed,humidity))
#Combined new_df and wind_hum data frames
bike_df<-cbind(new_df,wind_hum)
head(bike_df,5)
```

Normality test and Correlation matrix

```
#Quintle-Quintle normal plot
qqnorm(bike_df$total_count)
#Quintle-Quintle line
qqline(bike_df$total_count)

#load the corrgram for correlation
library(corrgram)
#Correlation plot
corrgram(bike_df[,10:14],order=F,upper.panel=panel.pie,text.panel=panel.txt,
main=' Correlation Plot')
```

Splitting the dataset

```
#load the purrr library for functions and vectors
library(purrr)
#Split the dataset based on simple random resampling
train_index<-sample(1:nrow(bike_df),0.7*nrow(bike_df))
train_data<-bike_df[train_index,]
test_data<-bike_df[-train_index,]
dim(train_data)
dim(test_data)
```

Encoding the categorical attributes

```
 #Create a new subset for train attributes
 train<-subset(train_data,select=c('season','year','month','holiday', 'weekday'
,'workingday','weather_condition','temp','humidity','windspeed','total_count'))
 #Create a new subset for test attributes
 test<-subset(test_data,select=c('season','year','month','holiday','weekday',
 'workingday','weather_condition','temp','humidity','windspeed','total_count'))

 #create a new subset for train categorical attributes
 train_cat_attributes<-subset(train,select=c('season','holiday','workingday',
 'weather_condition','year'))
 #create a new subset for test categorical attributes
 test_cat_attributes<-subset(test,select=c('season','holiday','workingday',
 'weather_condition','year'))
 #create a new subset for train numerical attributes
 train_num_attributes<-subset(train,select=c('weekday','month','temp',
 'humidity','windspeed','total_count'))
 #create a new subset for test numerical attributes
 test_num_attributes<-subset(test,select=c('weekday','month','temp',
 'humidity','windspeed','total_count'))

 #load the caret library
 library(caret)
 #other variables along with target variable to get dummy variables
 othervars<-c('month','weekday','temp','humidity','windspeed','total_count')
 set.seed(2626)
 #Categorical variables
 vars<-setdiff(colnames(train),c(train$total_count,othervars))
 #formula pass through encoder to get dummy variables
 f <- paste('~', paste(vars, collapse = ' + '))
 #encoder is encoded the categorical variables to numeric
 encoder<-dummyVars(as.formula(f), train)
 #Predicting the encode attributes
 encode_attributes<-predict(encoder,train)
 #Binding the train_num_attributes and encode_attributes
 train_encoded_attributes<-cbind(train_num_attributes,encode_attributes)
 head(train_encoded_attributes,5)

 set.seed(5662)
 #Categorical variables
 vars<-setdiff(colnames(test),c(test$total_count,othervars))
 #formula pass through encoder to get dummy variables
 f<- paste('~',paste(vars,collapse='+'))
 #Encoder is encoded the categorical variables to numeric
 encoder<-dummyVars(as.formula(f),test)
 #Predicting the encoder attributes
 encode_attributes<-predict(encoder,test)
 #Binding the test_num_attributes and encode_attributes
 test_encoded_attributes<-cbind(test_num_attributes,encode_attributes)
 head(test_encoded_attributes,5)
```

Linear Regression Model

```
#Set seed to reproduce the results of random sampling
set.seed(672)
#training the lr_model
lr_model<-lm(train_encoded_attributes$total_count~.,
      train_encoded_attributes[,-c(6 )])
#Summary of the model
Summary(lr_model)

#Cross validation prediction
#Set seed to reproduce results of random sampling
set.seed(623)
#Cross validation resampling method
train.control<-trainControl(method='CV',number=3)
#Cross validation prediction
CV_predict<-train(total_count~.,data=train_encoded_attributes,method='lm',
trControl=train.control)
#Summary of cross validation prediction
summary(CV_predict)

#Cross validation prediction plot
residuals<-resid(CV_predict)
y_train<-train_encoded_attributes$total_count
plot(y_train,residuals,ylab=('Residuals'),xlab=('Observed'),
main=('Cross validation prediction plot'))
abline(0,0)

#predict the lr_model
lm_predict<- predict(lr_model,test_encoded_attributes[,-c(6)])
head(lm_predict,5)

set.seed(688)
#Root mean squared error
rmse<-RMSE(lm_predict, test_encoded_attributes$total_count)
print(rmse)
#Mean squared error
mae<-MAE(lm_predict, test_encoded_attributes$total_count)
print(mae)

#Residual plot
y_test<-test_encoded_attributes$total_count
residuals<-y_test-lm_predict
plot(y_test,residuals,xlab='Observed',ylab='Residuals',main='Residual plot')
abline(0,0)
```

Decision Tree Regression

```
set.seed(568)
#load the rpart library for decision trees
library(rpart)
#rpart.control to contro the performance of model
rpart.control<-rpart.control(minbucket = 2,cp = 0.01,maxcompete = 3,
maxsurrogate  = 4, usesurrogate = 2, xval = 3,surrogatestyle = 0,maxdepth =10)
#training the dtr model
dtr<-rpart(train_encoded_attributes$total_count~.,
data=train_encoded_attributes[,-c(6)],control=rpart.control,
method='anova',cp=0.01)

#load the rpart.plot for plot the learned dtr model

library(rpart.plot)
rpart.plot(dtr, box.palette="RdBu", shadow.col="gray", nn=TRUE,roundint=FALSE)

set.seed(5769)
#cross validation resampling method
train.control<-trainControl(method='CV',number=3)
#cross validation pred
dtr_CV_predict<-train(total_count~.,data=train_encoded_attributes,
 method='rpart',trControl=train.control)

#Cross validation prediction plot
residuals<-resid(dtr_CV_predict)
plot(y_train,residuals,xlab='Observed',ylab='Residuals',
 main='Cross validation plot')
abline(0,0)

set.seed(7882)
#predict the trained model
dtr_predict<-predict(dtr,test_encoded_attributes[,-c(6)])
head(dtr_predict,5)

set.seed(6889)
#Root mean squared error
rmse<-RMSE(y_test,dtr_predict)
print(rmse)
#Mean absolute error
mae<-MAE(y_test,dtr_predict)
print(mae)

#Residual plot
residuals<-y_test-dtr_predict
plot(y_test,residuals,xlab='Observed',ylab='Residuals',main='Residual plot')
abline(0,0)
```

Random Forest

```r
set.seed(6788271)
#load the randomForest library
library(randomForest)
#training the model
rf_model<-randomForest(total_count~.,train_encoded_attributes,importance=TRUE,
ntree=200)

set.seed(6772)
library(randomForest)
#load the ranger library for random forest CV
library(ranger)
#Cross validation resampling method
train.control<-trainControl(method='CV',number=3)
#Cross validation prediction
rf_CV_predict<-train(total_count~.,train_encoded_attributes,method='ranger',
trControl=train.control)

#Cross validation prediction plot
residuals<-resid(rf_CV_predict)
plot(y_train,residuals,xlab='Observed',ylab='Residuals',
main='Cross validation prediction plot')
abline(0,0)

set.seed(7889)
#Predicting the model
rf_predict<-predict(rf_model,test_encoded_attributes[,-c(6)])
head(rf_predict,5)

set.seed(667)
#Root mean squared error
rmse<-RMSE(y_test,rf_predict)
print(rmse)
mae<-MAE(y_test,rf_predict)
print(mae)

#Residual plot
residuals<-y_test-rf_predict
plot(y_test,residuals,xlab='Observed',ylab='Residuals',main='Residual plot')
abline(0,0)

Bike_predictions=data.frame(y_test,rf_predict)
write.csv(Bike_predictions,'Bike_Renting_R.CSV',row.names=F)
```

References

[1]. https://www.kaggle.com

[2]. http://rprogramming.net/

[3]. https://medium.com/

[4]. **Practical Machine learning with Python** book by by Dipanjan Sarkar , Raghav Bali and Tushar Sharma

[5]. https://stackoverflow.com/

[6]. https://www.rdocumentation.org