Bike Renting

Riya Mittal

04 November 2018

Contents

1. Introduction

1.1 [Problem Statement](#_1.1_Problem_Statement) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

4

4

6

6

8

9

10

11

12

12

13

15

15

15

16

16

17

1.2 [Data](#_1.2_Data) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

1. Methodology

2.1 [Pre Processing](#_2.1_Pre_Processing) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

2.1.1 [Missing Value Analysis](#_2.1.1_Missing_Value) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

2.1.2 [Outlier Analysis](#_2.1.2_Outlier_Analysis) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

2.1.3 [Feature Dimension Reduction](#_2.1.3_Feature_Dimension)

2.1.3.1 [Correlation Analysis](#_2.1.3.1_Correlation_Analysis)

* 1. [Modeling](#_2.2_Modeling) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

* + 1. [Model Selection](#_2.2.1_Model_Selection) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

2.2.2 [Multiple Linear Regression](#_2.2.2_Multiple_Linear) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . ....

2.2.3 [Random Forest Regression](#_2.2.3_Random_Forest). . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

2.2.4 [Extreme Gradient Boost Regression](#_2.2.4_Extreme_Gradient)…………………………………

2.2.5 [K Neighbors Regression](#_2.2.5_K_Neighbors)………………………………………………..

1. [Conclusion](#_Conclusion)

3.1 [Model Evaluation](#_3.1_Model_Evaluation) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

3.2 [Model Selection](#_3.2_Model_Selection_1) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

[Appendix B - R Code](#_Appendix_B_-)

[Complete Python File](#_Complete_R_File) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . … 18

[References 21](#_References)

Chapter 1

# Introduction

## 1.1 Problem Statement

The objective of this project is the prediction of bike rental count daily based on the environmental and seasonal conditions. This aims at determining the future trends for a company so that they can plan accordingly on how they need to do the setup for renting bikes.

## 1.2 Data

We would build a regression model here which will predict the count of rented bikes on daily basis (target variable) based on multiple factors. Below is the sample of the dataset being used for this purpose:

Table 1.1: Sample Data (Columns: 1-8)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ***instant*** | ***dteday*** | ***season*** | ***yr*** | ***mnth*** | ***holiday*** | ***weekday*** | ***workingday*** |
| *1* | *40544* | *1* | *0* | *1* | *0* | *6* | *0* |
| *2* | *40545* | *1* | *0* | *1* | *0* | *0* | *0* |
| *3* | *40546* | *1* | *0* | *1* | *0* | *1* | *1* |
| *4* | *40547* | *1* | *0* | *1* | *0* | *2* | *1* |
| *5* | *40548* | *1* | *0* | *1* | *0* | *3* | *1* |

Table 1.2: Sample Data (Columns: 9-16)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ***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* |

This entire prediction model will be based on 730 X 16 dataset. Following is the bifurcation for predictor and target variables:

* ***Predictor Variables*** : dteday, season, yr, mnth, holiday, weekday, workingday, weathersit, temp, atemp, hum, windspeed, casual, registered
* ***Target Variable :*** cnt

Also below is the list of categorical variables and continuous variables amongst them:

* ***Categorical Variables***: dteday, season, yr, mnth, holiday, weekday, workingday, weathersit
* ***Continuous Variables:*** temp, atemp, hum, windspeed, casual, registered, cnt

Chapter 2

# Methodology

## 2.1 Pre Processing

Pre-processing is a technique through which we make the data fit to be applied to any algorithm. Raw data undergoes a number of transformations before we feed it into an actual model. A data scientist roughly spends 80% of his time in pre-processing. It gives us an idea about how important this process is. Basic pre-processing steps involved before every model implementation is as shown in Figure 2.1 below:

Figure 2.1: Pre-processing Steps

**Missing Value Analysis**

**Outlier Analysis**

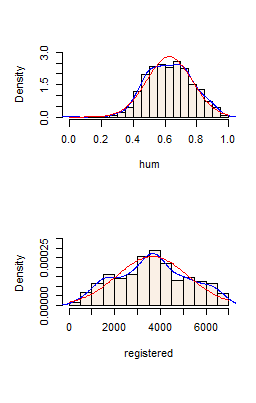
**Feature Selection**

**Feature Scaling**

**Feature Sampling**

The Bike Renting is a regression problem. This means we would have to predict a quantity. In this case the quantity is cnt. Most of the regression problems analysis need normally distributed data. So we would have to look upon probability distributions or probability density functions of the continuous variables as shown in Figure 2.2. The blue lines indicate Kernel Density Estimations (KDE) of the variable. The red lines represent the normal distribution. The data is not normally distributed. This needs to be worked upon.

Figure 2.2: Probability Density Function of Bike Rental Data



### 

### 2.1.1 Missing Value Analysis

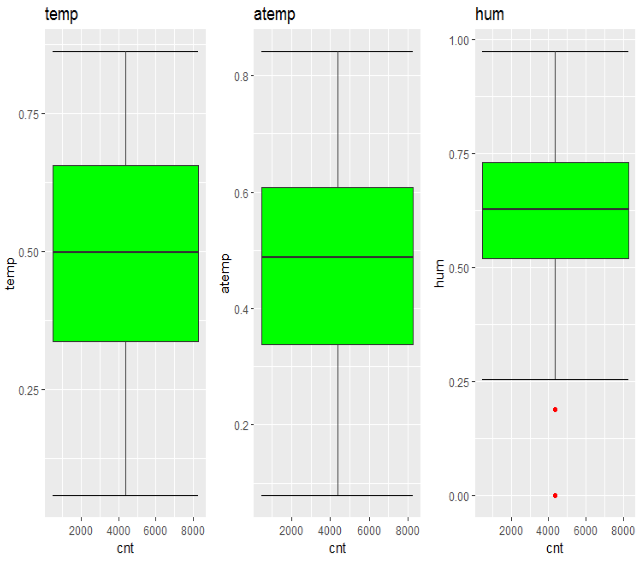
Missing values are the most common problem encountered while dealing with the real world data. There could be multiple reasons for this like human error, skipped entry etc. To handle the missing values, we have a number of ways :

* Deleting the rows or columns
* Replacing with mean/median/mode
* Imputation techniques such as KNN-Imputation

In our case, we don’t have any missing values

### 2.1.2 Outlier Analysis

Often we come across another common issue in the process of data exploration called Outliers. Outliers are observations which stand out from a normal range of a particular variable. It is of utter importance to analyze the reason of their deviation as they could many times lead to false predictions. It’s sometimes observed that we may have valid and possible outliers. Outliers generally can be observed using Boxplots. Here we are performing multivariate analysis plotting each continuous variable against target variable as shown in Figure 2.3.



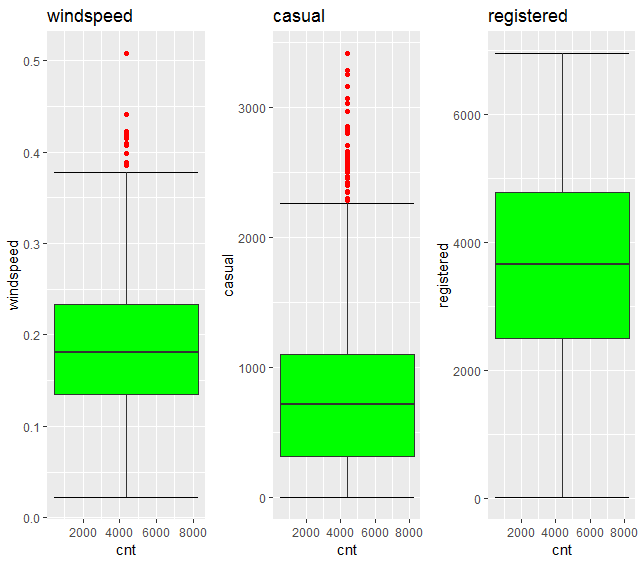


Figure 2.3 Outlier Analysis for Continuous Variables

We can see a number of outliers in different predictors. For removing the outliers we are using the maximum and minimum values in case of python. In case of R, we are using the library mice.

### 2.1.3 Feature Dimension Reduction

Feature Selection is one of the most important steps which decide the quality of our model. For this problem, our aim is to find out the count of bikes that will be rented daily. This means we need to find out the prime features affecting the target variable. One approach to find correlation between continuous variables is using correlation analysis. Let’s first look upon it.

### 2.1.3.1 Correlation Analysis

As we are interested in variable importance, correlation analysis can help a lot in reducing the dimension of features leaving us with the most important ones. As we have continuous target variable, we can go with heatmap to find out correlation between variables. Figure 2.4 shows the heatmap obtained.

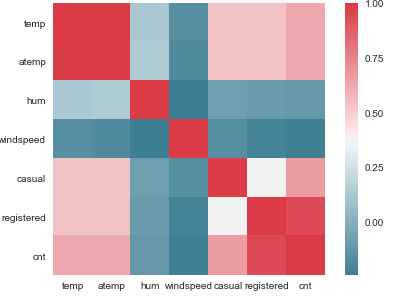


Figure 2.4 HeatMap between continuous variables

Heatmap shows there is a strong correlation between temp and atemp. Hence, one amongst the two can be chosen. Also whether the count is casual or registered would be count only any day. So, casual and registered can also be dropped and we kept count which is sum of casual and registered. Apart from this, we had also used dummies for season and weather. The 4 seasons and 3 weathers would be individual columns. So this would give 7 columns rather than 2 and help us give better results. Below is how the new table looks like :

.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ***yr*** | ***mnth*** | ***holiday*** | ***weekday*** | ***workingday*** | ***atemp*** | ***hum*** | ***windspeed*** |
| *0* | *1* | *0* | *6* | *0* | *0.363625* | *0.81* | *0.160446* |
| *0* | *1* | *0* | *0* | *0* | *0.353739* | *0.7* | *0.248539* |
| ***0*** | ***1*** | ***0*** | ***1*** | ***1*** | ***0.189405*** | ***0.44*** | ***0.248309*** |
| ***0*** | ***1*** | ***0*** | ***2*** | ***1*** | ***0.212122*** | ***0.59*** | ***0.160296*** |
| *0* | *1* | *0* | *3* | *1* | *0.22927* | *0.44* | *0.1869* |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ***cnt*** | ***df\_season1*** | ***df\_season2*** | ***df\_season3*** | ***df\_season4*** | ***df\_weathersit1*** | ***df\_weathersit2*** | ***df\_weathersit3*** |
| *985* | *1* | *0* | *0* | *0* | *0* | *1* | *0* |
| *801* | *1* | *0* | *0* | *0* | *0* | *1* | *0* |
| ***1349*** | ***1*** | ***0*** | ***0*** | ***0*** | ***1*** | ***0*** | ***0*** |
| ***1562*** | ***1*** | ***0*** | ***0*** | ***0*** | ***1*** | ***0*** | ***0*** |
| *1600* | *1* | *0* | *0* | *0* | *1* | *0* | *0* |

Table 2.1 Final Dataset after Feature Selection

### 2.2 Modeling

### 2.2.1 Model Selection

Now since we are ready with our features, we can apply models for predicting the daily counts. Here our dependent variable is a quantity hence we need to regression model. Let’s first go with Multiple Linear Regression Model.

### 

### 2.2.2 Multiple Linear Regression

Call:

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

Residuals:

Min 1Q Median 3Q Max

-3516.2 -374.7 66.9 465.4 2988.7

Coefficients: (2 not defined because of singularities)

Estimate Std. Error t value Pr(>|t|)

(Intercept) 1096.28 473.08 2.317 0.020838 \*

yr 2034.61 67.81 30.004 < 2e-16 \*\*\*

mnth -18.76 18.94 -0.990 0.322522

holiday -518.39 207.01 -2.504 0.012550 \*

weekday 90.30 16.81 5.372 1.13e-07 \*\*\*

workingday 72.98 74.11 0.985 0.325222

atemp 5635.06 366.76 15.364 < 2e-16 \*\*\*

hum -1447.09 345.15 -4.193 3.19e-05 \*\*\*

windspeed -2557.23 494.47 -5.172 3.21e-07 \*\*\*

df\_season1 -1646.32 179.27 -9.183 < 2e-16 \*\*\*

df\_season2 -511.95 152.64 -3.354 0.000849 \*\*\*

df\_season3 -660.85 139.50 -4.737 2.73e-06 \*\*\*

df\_season4 NA NA NA NA

df\_weathersit1 1821.32 245.82 7.409 4.58e-13 \*\*\*

df\_weathersit2 1408.14 227.82 6.181 1.21e-09 \*\*\*

df\_weathersit3 NA NA NA NA

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 805 on 573 degrees of freedom

Multiple R-squared: 0.8312, Adjusted R-squared: 0.8274

F-statistic: 217.1 on 13 and 573 DF, p-value: < 2.2e-16

Adjusted R square value shows that we can predict 82.74 % of our data using this model.

Analysis of Variance Table

Response: cnt

Df Sum Sq Mean Sq F value Pr(>F)

yr 1 724747828 724747828 1118.4261 < 2.2e-16 \*\*\*

mnth 1 178028107 178028107 274.7318 < 2.2e-16 \*\*\*

holiday 1 18216662 18216662 28.1118 1.637e-07 \*\*\*

weekday 1 9663866 9663866 14.9132 0.0001254 \*\*\*

workingday 1 1068287 1068287 1.6486 0.1996722

atemp 1 669450871 669450871 1033.0922 < 2.2e-16 \*\*\*

hum 1 65865222 65865222 101.6428 < 2.2e-16 \*\*\*

windspeed 1 33585420 33585420 51.8288 1.910e-12 \*\*\*

df\_season1 1 72422912 72422912 111.7626 < 2.2e-16 \*\*\*

df\_season2 1 3022 3022 0.0047 0.9455768

df\_season3 1 17495473 17495473 26.9989 2.835e-07 \*\*\*

df\_weathersit1 1 13221292 13221292 20.4030 7.621e-06 \*\*\*

df\_weathersit2 1 24757175 24757175 38.2051 1.210e-09 \*\*\*

Residuals 573 371307940 648007

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

According to this table we have all columns as significant factors except df\_season2 and workingday.

LM\_model2 = update(LM\_model,. ~ . - df\_season2-workingday)

summary(LM\_model2)

Call:

lm(formula = cnt ~ yr + mnth + holiday + weekday + atemp + hum +

windspeed + df\_season1 + df\_season3 + df\_season4 + df\_weathersit1 +

df\_weathersit2 + df\_weathersit3, data = train)

Residuals:

Min 1Q Median 3Q Max

-3492.7 -384.1 74.9 469.7 2933.9

Coefficients: (1 not defined because of singularities)

Estimate Std. Error t value Pr(>|t|)

(Intercept) 656.99 447.08 1.470 0.142243

yr 2034.47 67.81 30.002 < 2e-16 \*\*\*

mnth -18.56 18.94 -0.980 0.327548

holiday -569.09 200.50 -2.838 0.004694 \*\*

weekday 90.26 16.81 5.370 1.15e-07 \*\*\*

atemp 5644.01 366.64 15.394 < 2e-16 \*\*\*

hum -1468.33 344.46 -4.263 2.36e-05 \*\*\*

windspeed -2579.91 493.92 -5.223 2.46e-07 \*\*\*

df\_season1 -1133.29 129.47 -8.753 < 2e-16 \*\*\*

df\_season3 -149.17 118.30 -1.261 0.207861

df\_season4 511.83 152.63 3.353 0.000851 \*\*\*

df\_weathersit1 1809.58 245.53 7.370 5.97e-13 \*\*\*

df\_weathersit2 1403.01 227.75 6.160 1.37e-09 \*\*\*

df\_weathersit3 NA NA NA NA

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 805 on 574 degrees of freedom

Multiple R-squared: 0.8309, Adjusted R-squared: 0.8274

F-statistic: 235.1 on 12 and 574 DF, p-value: < 2.2e-16

No improvement in Adjusted R-square is observed though this shows df\_season2 and workingday does not affect the results.

### 2.2.3 Random Forest Regression

Ensemble technique in which we collect number of decision trees to predict values.

#Random Forest

from sklearn.ensemble import RandomForestRegressor

RF\_model = RandomForestRegressor(n\_estimators = 20).fit(X\_train, y\_train)

RF\_Predictions = RF\_model.predict(X\_test)

np.sqrt(metrics.mean\_squared\_log\_error(y\_test,RF\_Predictions))

metrics.r2\_score(y\_test,RF\_Predictions)

### 2.2.4 Extreme Gradient Boost Regression

from xgboost.sklearn import XGBRegressor

model = XGBRegressor(n\_estimators = 100).fit(X\_train, y\_train)

Predictions = model.predict(X\_test)

np.sqrt(metrics.mean\_squared\_log\_error(y\_test,Predictions))

metrics.r2\_score(y\_test,Predictions)

### 2.2.5 K Neighbors Regression

from sklearn.neighbors import KNeighborsRegressor

model = KNeighborsRegressor(n\_neighbors = 7).fit(X\_train, y\_train)

Predictions = model.predict(X\_test)

np.sqrt(metrics.mean\_squared\_log\_error(y\_test,Predictions))metrics.r2\_score(y\_test,Predictions)

Chapter 3

# Conclusion

# 3.1 Model Evaluation

Evaluation metrics help in explaining the performance of a model. The most popular error metric to evaluate any regression model is Root Mean Square Error (RMSE).

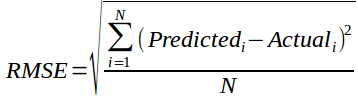


Table 3.1 RMSLE

|  |  |
| --- | --- |
|  | **RMSLE** |
| **Linear Regression** | 0.21 |
| **XGBoost** | 0.0332 |
| **Random Forest** | 0.011 |
| **KNeighbors** | 0.039 |

**Root Mean Square Logarithmic Error,**

***RMSLE=log((pi+1)/(ai+1))***

# 3.2 Model Selection

We can see from the results that Random Forest models are better for this regression problem. This is when we have kept training and testing data ratio as 80:20.

##### **Appendix B - R Code**

### Complete Python File

***# data visualisation and manipulation***

***import numpy as np***

***import pandas as pd***

***import matplotlib.pyplot as plt***

***from matplotlib import style***

***import seaborn as sns***

***import os***

***import ggplot***

***# sets matplotlib to inline and displays graphs below the corressponding cell.***

***% matplotlib inline***

***style.use('fivethirtyeight')***

***sns.set(style='whitegrid',color\_codes=True)***

***#splitting***

***from sklearn.cross\_validation import train\_test\_split***

***#regression***

***from sklearn.linear\_model import LinearRegression***

***from sklearn.ensemble import RandomForestRegressor***

***from xgboost.sklearn import XGBRegressor***

***from sklearn.neighbors import KNeighborsRegressor***

***#evaluation metrics***

***from sklearn import metrics***

***from sklearn.metrics import mean\_squared\_log\_error,mean\_squared\_error, r2\_score,mean\_absolute\_error # for regression***

***#Set working directory***

*os.chdir("C:/Users/DELL/Desktop/edwisor/project2")*

***# Load dataset***

*df = pd.read\_csv("C:/Users/DELL/Desktop/edwisor/project2/day.csv",encoding='latin-1')*

***#structure of data***

*df.info()*

***# KDE plot , probability distribution and histogram for continuous variables***

*f, axes = plt.subplots(4, 2, figsize=(14, 17))*

*sns.distplot( df["temp"] , color="olive", ax=axes[0, 0])*

*sns.distplot( df["atemp"] , color="green", ax=axes[0, 1])*

*sns.distplot( df["hum"] , color="red", ax=axes[1, 0])*

*sns.distplot( df["windspeed"] , color="black", ax=axes[1, 1])*

*sns.distplot( df["casual"] , color="orange", ax=axes[2, 0])*

*sns.distplot( df["registered"] , color="blue", ax=axes[2, 1])*

*sns.distplot( df["cnt"] , color="brown", ax=axes[3, 0])*

***#count visualisation for categorical variables***

*f, axes = plt.subplots(4, 2, figsize=(14, 17))*

*sns.factorplot( x='season',data=df,kind='count', ax=axes[0, 0])*

*sns.factorplot( x='mnth',data=df,kind='count', ax=axes[0, 1])*

*sns.factorplot(x='yr',data=df,kind='count', ax=axes[1, 0])*

*sns.factorplot(x='holiday',data=df,kind='count', ax=axes[1, 1])*

*sns.factorplot(x='weekday',data=df,kind='count', ax=axes[2, 0])*

*sns.factorplot(x='workingday',data=df,kind='count', ax=axes[2, 1])*

*sns.factorplot(x='weathersit',data=df,kind='count',ax=axes[3,0])*

***#boxplots of all continuous variables***

*f, axes = plt.subplots(4, 2, figsize=(10, 17))*

*sns.boxplot( y=df["temp"] , color="olive", ax=axes[0, 0])*

*sns.boxplot( df["atemp"] , color="green", ax=axes[0, 1])*

*sns.boxplot( df["hum"] , color="red", ax=axes[1, 0])*

*sns.boxplot( y=df["windspeed"] , color="black", ax=axes[1, 1])*

*sns.boxplot( y=df["casual"] , color="orange", ax=axes[2, 0])*

*sns.boxplot( y=df["registered"] , color="blue", ax=axes[2, 1])*

*sns.boxplot( y=df["cnt"] , color="brown", ax=axes[3, 0])*

***#Extract quartiles for continuous variables with outliers***

*Windspeedq75, Windspeedq25 = np.percentile(df['windspeed'], [75 ,25])*

*humq75, humq25 = np.percentile(df['hum'], [75 ,25])*

***#Interquartile Region***

*Windspeediqr = Windspeedq75 - Windspeedq25*

*humiqr = humq75 - humq25*

***#Whiskers value***

*Windspeedminimum = Windspeedq25 - (Windspeediqr\*1.5)*

*Windspeedmaximum = Windspeedq75 + (Windspeediqr\*1.5)*

*humminimum = humq25 - (humiqr\*1.5)*

*hummaximum = humq75 + (humiqr\*1.5)*

***#Replacing windspeed outliers below IQR with minimum values***

*for i in range(len(df)):*

*if(df['windspeed'].iloc[i] < 0.0124):*

*df['windspeed'].iloc[i]=0.0124*

***#Replacing windspeed outliers above IQR with maximum values***

*for i in range(len(df)):*

*if(df['windspeed'].iloc[i] > Windspeedmaximum):*

*df['windspeed'].iloc[i]=Windspeedmaximum*

***#Replacing humidity outliers above IQR with maximum values***

*for i in range(len(df)):*

*if(df['hum'].iloc[i] > hummaximum):*

*df['hum'].iloc[i]=hummaximum*

***#Replacing humidity outliers below IQR with minimum values***

*for i in range(len(df)):*

*if(df['hum'].iloc[i] < humminimum):*

*df['hum'].iloc[i]=humminimum*

***#Again checking if outliers remaining***

*f, axes = plt.subplots(4, 2, figsize=(10, 17))*

*sns.boxplot( y=df["temp"] , color="olive", ax=axes[0, 0])*

*sns.boxplot( df["atemp"] , color="green", ax=axes[0, 1])*

*sns.boxplot( df["hum"] , color="red", ax=axes[1, 0])*

*sns.boxplot( y=df["windspeed"] , color="black", ax=axes[1, 1])*

*sns.boxplot( y=df["casual"] , color="orange", ax=axes[2, 0])*

*sns.boxplot( y=df["registered"] , color="blue", ax=axes[2, 1])*

*sns.boxplot( y=df["cnt"] , color="brown", ax=axes[3, 0])*

***#Correlation Analysis***

*cnames = ["temp", "atemp", "hum", "windspeed", "casual", "registered", "cnt"]*

***#Correlation plot***

*df\_corr = df.loc[:,cnames]*

***#Set the width and hieght of the plot***

*f, ax = plt.subplots(figsize=(7, 5))*

***#Generate correlation matrix***

*corr = df\_corr.corr()*

***#Plot using seaborn library***

*sns.heatmap(corr, mask=np.zeros\_like(corr, dtype=np.bool), cmap=sns.diverging\_palette(220, 10, as\_cmap=True),*

*square=True, ax=ax)*

***#drop temp,casual,registered***

*df.drop(['temp','casual','registered'], axis = 1, inplace = True)*

***#create dummy variables with all weathersit values***

*weathersit = pd.get\_dummies(df['weathersit'],prefix='weathersit')*

*df=pd.concat([df,weathersit],axis=1)*

***#create dummy variables with all season values***

*season = pd.get\_dummies(df['season'],prefix='season')*

*df=pd.concat([df,season],axis=1)*

***#drop not required variables***

*df.drop(['season','dteday','weathersit','instant'],axis=1,inplace=True)*

***#checking count plot for all categorical variables***

*f, axes = plt.subplots(6, 2, figsize=(15, 22))*

*sns.factorplot(x="yr",y="cnt",data=df,kind='bar', ax=axes[0, 0])*

*sns.factorplot(x="mnth",y="cnt",data=df,kind='bar' , ax=axes[0, 1])*

*sns.factorplot(x="holiday",y="cnt",data=df,kind='bar', ax=axes[1, 0])*

*sns.factorplot(x="weekday",y="cnt",data=df,kind='bar', ax=axes[1, 1])*

*sns.factorplot(x="workingday",y="cnt",data=df,kind='bar', ax=axes[2, 0])*

*sns.factorplot(x="weathersit\_1",y="cnt",data=df,kind='bar', ax=axes[2, 1])*

*sns.factorplot(x="weathersit\_2",y="cnt",data=df,kind='bar', ax=axes[3, 0])*

*sns.factorplot(x="weathersit\_3",y="cnt",data=df,kind='bar', ax=axes[3, 1])*

*sns.factorplot(x="season\_1",y="cnt",data=df,kind='bar', ax=axes[4, 0])*

*sns.factorplot(x="season\_2",y="cnt",data=df,kind='bar', ax=axes[4, 1])*

*sns.factorplot(x="season\_3",y="cnt",data=df,kind='bar', ax=axes[5, 0])*

*sns.factorplot(x="season\_4",y="cnt",data=df,kind='bar', ax=axes[5, 1])*

***#Divide data into train and test with 80% train data***

*X = df.values[:, df.columns!=8]*

*Y = df.values[:,8]*

*X\_train, X\_test, y\_train, y\_test = train\_test\_split( X, Y, test\_size = 0.2)*

***#Random Forest***

*RF\_model = RandomForestRegressor(n\_estimators = 20).fit(X\_train, y\_train)*

*RF\_Predictions = RF\_model.predict(X\_test)*

*np.sqrt(metrics.mean\_squared\_log\_error(y\_test,RF\_Predictions))*

*metrics.r2\_score(y\_test,RF\_Predictions)*

***#XGBoost***

*model = XGBRegressor(n\_estimators = 100).fit(X\_train, y\_train)*

*Predictions = model.predict(X\_test)*

*np.sqrt(metrics.mean\_squared\_log\_error(y\_test,Predictions))*

*metrics.r2\_score(y\_test,Predictions)*

***#KNeighborRegressor***

*model = KNeighborsRegressor(n\_neighbors = 7).fit(X\_train, y\_train)*

*Predictions = model.predict(X\_test)*

*np.sqrt(metrics.mean\_squared\_log\_error(y\_test,Predictions))*

*metrics.r2\_score(y\_test,Predictions)*

***#Linear Regression***

*linreg=LinearRegression()*

*linreg.fit(X\_train, y\_train)*

*Linear\_Predictions=linreg.predict(X\_test)*

*np.sqrt(metrics.mean\_squared\_log\_error(y\_test,Linear\_Predictions))*

*metrics.r2\_score(y\_test,Linear\_Predictions)*

## References

*Jiao Yuntai, Li Wenquan, Feng Peiyu, Ding Ran, "A Scheduling Demand Model for Public Bicycle Rental Station [J]",* Transportation and information security*, vol. 32, no. 4, pp. 8-13, 2014.*