Bike Renting

By Srikrishna Ghandikota

June 2019

Declaration

This report is in reference to the Python Code of the Project.

Contents

Chapter 1

Introduction

1.1 Problem Statement

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

1.2 Data

Our task is to build a regression model which predicts the bike rental count. Given below is a sample of the data set that we are using to predict the bike rental count:

	instant	dteday	season	yr	mnth	holiday	weekday	workingday
0	1	2011-01-01	1	0	1	0	6	0
1	2	2011-01-02	1	0	1	0	0	0
2	3	2011-01-03	1	0	1	0	1	1
3	4	2011-01-04	1	0	1	0	2	1
4	5	2011-01-05	1	0	1	0	3	1

Table 1.1: Bike Rental Prediction Sample Data Columns(1-8)

	weathersit	temp	atemp	hum	windspeed	casual	registered	cnt
0	2	0.344167	0.363625	0.805833	0.160446	331	654	985
1	2	0.363478	0.353739	0.696087	0.248539	131	670	801
2	1	0.196364	0.189405	0.437273	0.248309	120	1229	1349
3	1	0.200000	0.212122	0.590435	0.160296	108	1454	1562
4	1	0.226957	0.229270	0.436957	0.186900	82	1518	1600

Table 1.2: Bike Rental Prediction Sample Data Columns(9-16)

Given in the table below are the variables, using which we can use to predict the bike rental count:

S.No	Predictor
1	instant
2	dteday
3	season
4	yr
5	mnth
6	holiday
7	weekday
8	workingday
9	weathersit
10	temp
11	atemp
12	hum
13	windspeed
14	casual
15	registered

Chapter 2

Methodology

2.1 Exploratory Data Analysis

Before building our model it is important to understand how the data is varying. This can be done by exploring the trends in the predictor variables. Exploring the data involves cleaning the data, understanding the distribution of the data, visualizing the data etc., This process is called Exploratory Data Analysis.

2.1.1 Missing Value Analysis

Missing Values in a data set occur due to human error or some measurement errors. These values provide no weightage to the model. If there are any missing values there is a need for imputation. Imputation is a process of impute values in the place of missing values. This imputation can be based on central statistics like mean or based on Euclidean Distance or so on. Given below is the total distribution of the missing values in the dataset :

Variables	Missing Value Count
instant	0
dteday	0
season	0
yr	0
mnth	0
holiday	0
weekday	0
workingday	0
weathersit	0
temp	0
atemp	0
hum	0
windspeed	0
casual	0
registered	0
cnt	0

Table 2.1: Missing Value Analysis

As there are no missing values there is no need for imputation.

2.1.2 Outlier Analysis

Outlier is a data point which is inconsistent with the rest of the dataset. It is significantly different from other observations thereby effecting the outcome of the model. Outliers can be detected using graphical plots called Boxplots.
Below are the boxplots of all the numeric data in dataset :

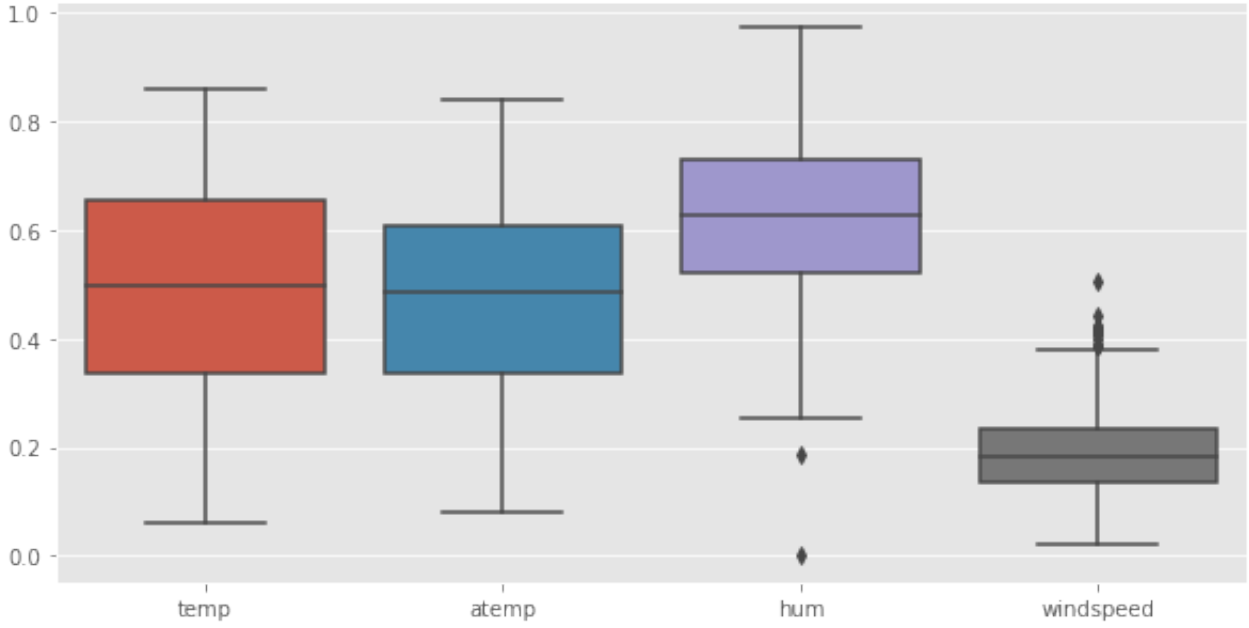

Figure 2.1: Outlier Analysis

From the above plot we can observe some outliers in Windspeed and Humidity. The histogram distribution of each predictor variable can give us a better idea of the variables. Check the next page for Histogram of Predictor Variables

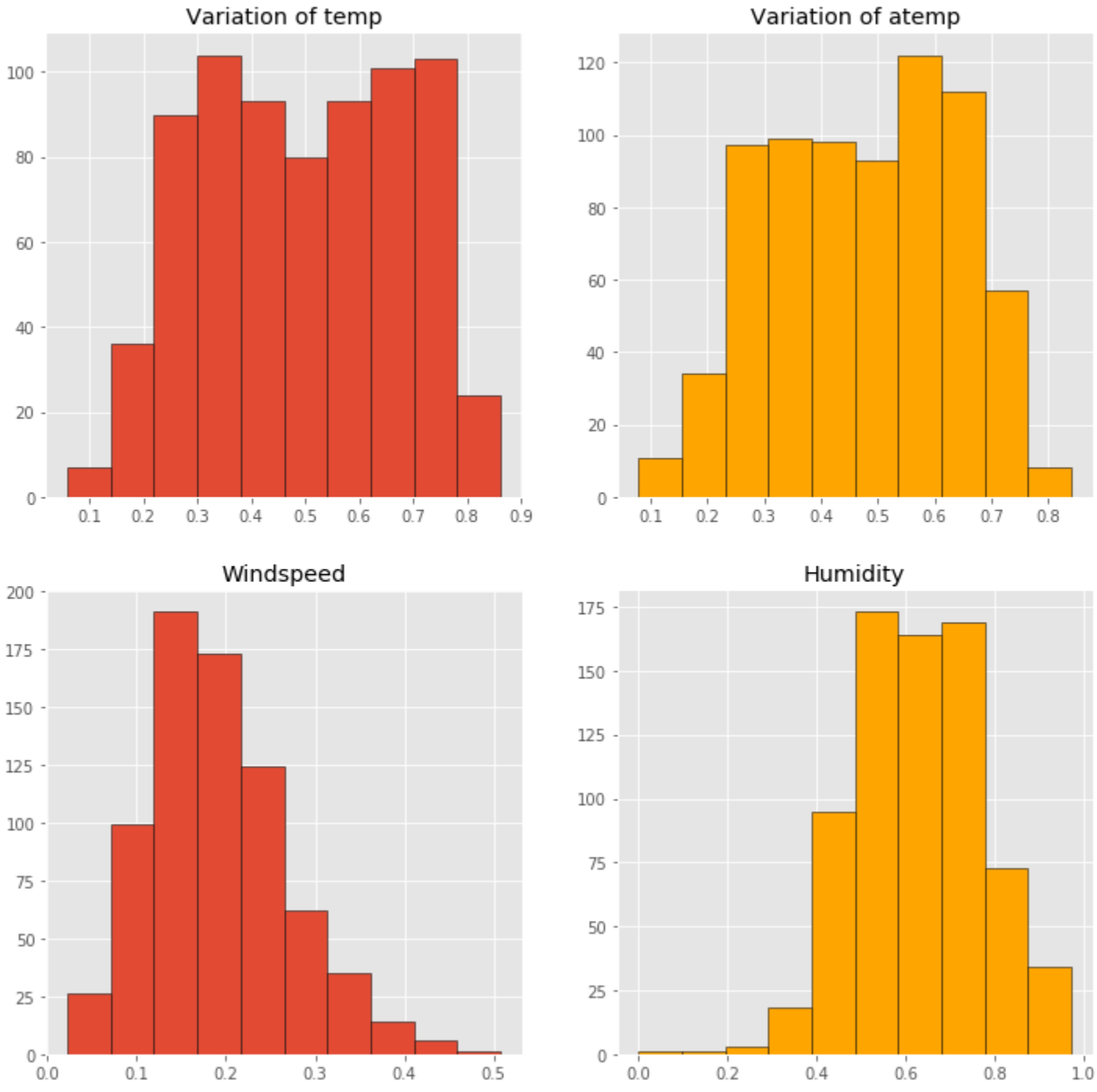

Figure 2.2: Histogram of Predictor Variables

These variables are already normalized and we can observe some skewness in the data. This can be remove my removing the outliers in the data.

Following is the python code to remove outliers in the data

Listing 2.1: Code for removing ouliers in python

```
for i in df.columns:
    if(i == 'dteday' or i == 'cnt'):
        continue
    print(i)
    q75, q25 = np.percentile(df.loc[:,i], [75 ,25])
    iqr = q75 - q25
    min = q25 - (iqr*1.5)
    max = q75 + (iqr*1.5)
    print(min)
    print(max)
    df =df.drop(df[df.loc[:,i] < min].index)
    df =df.drop(df[df.loc[:,i] > max].index)
```

The boxplots of the predictor variables after removing the outliers:

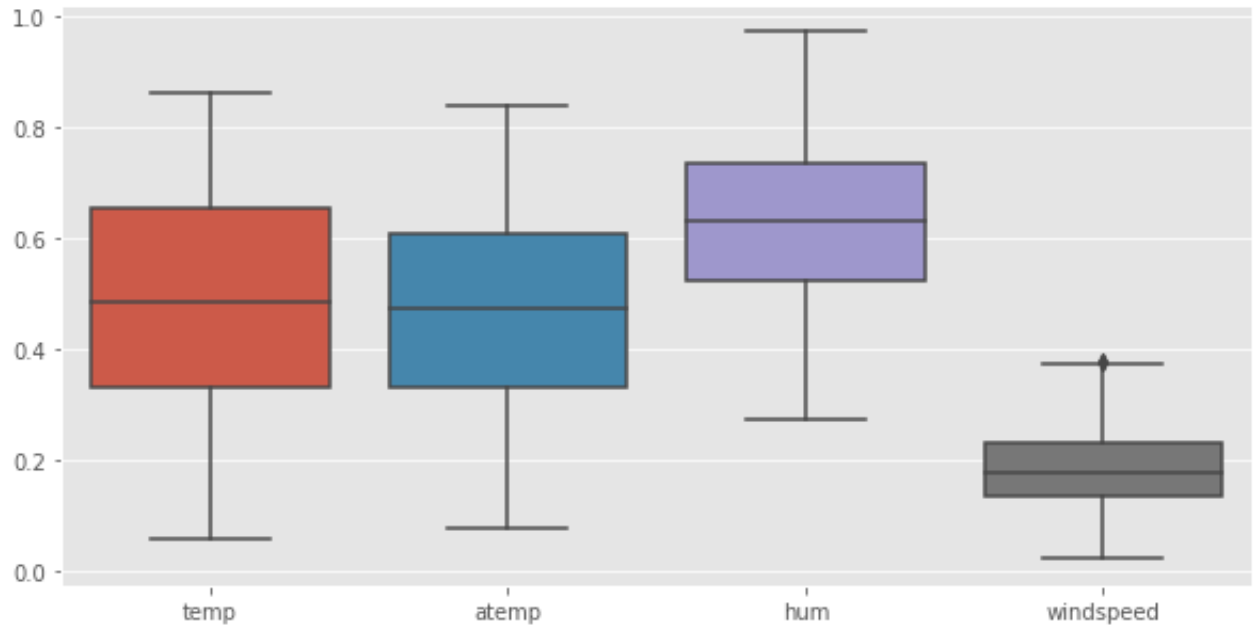

Figure 2.3: Boxplots after outlier removal

2.2 Feature Selection

Correlation Analysis can be used in selection of features for prediction. The feature ***dteday*** can be dropped since its neither a numeric variable or a categorical variable. Correlation Heatmap can be used to understand the relationship between predictor variables. The plot is as follows:

Figure 2.4: Correlation Analysis -1

From the plot we can see that the variables ***temp*** and ***atemp*** are highly related. One of them can be dropped. ***atemp*** is more correlated to the variable ***cnt*** than ***temp***, therefore ***temp*** can be dropped. The variable ***holiday*** has no significance. ***instant*** is a index variable which can also be dropped. ***cnt*** is the sum of ***casual*** and ***registered***. Both of them can be dropped.

Correlation plot with the final variables is as follows :

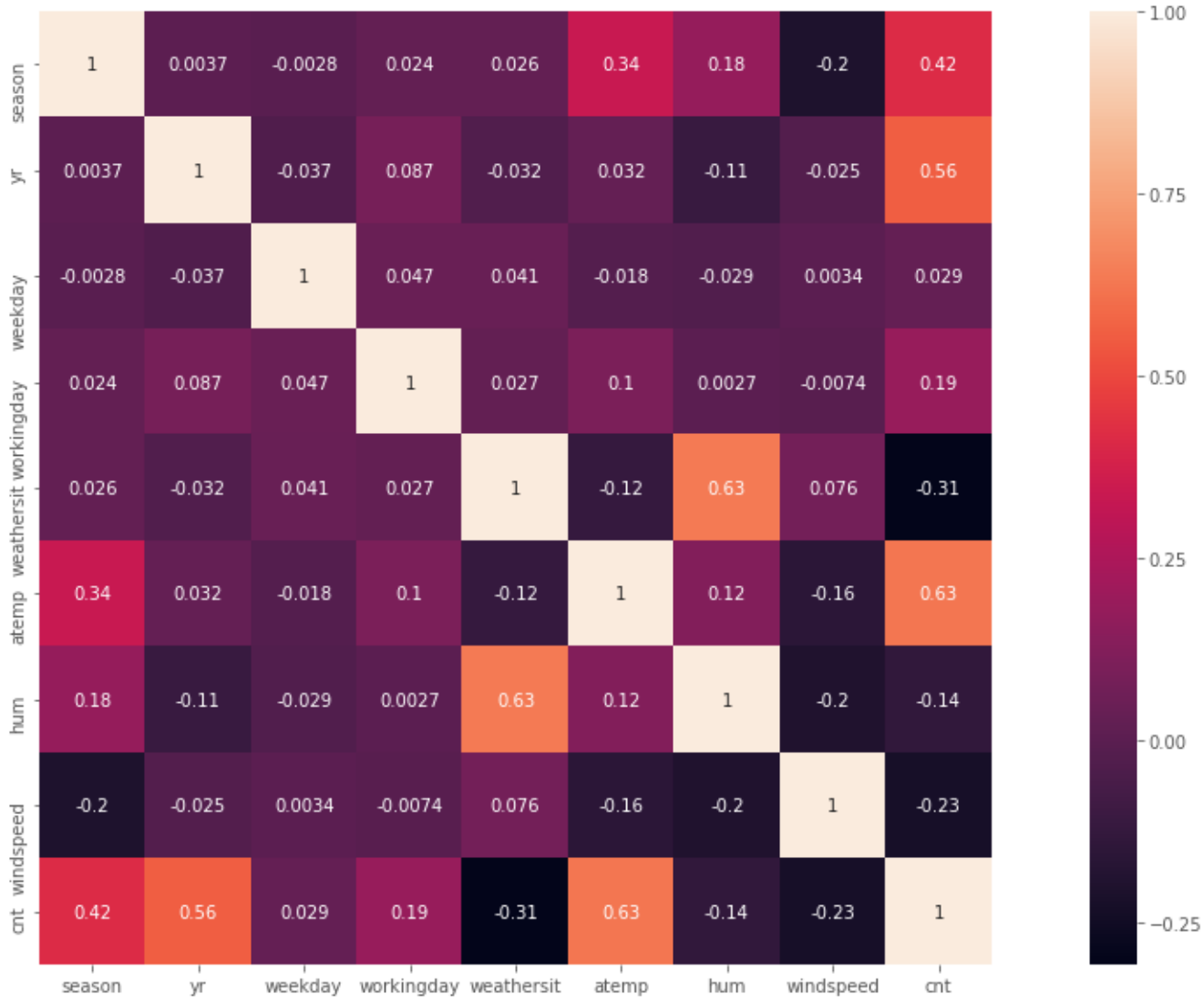

Figure 2.5: Correlation Analysis -2

Chapter 3

Modeling

3.1 Model Selection

From the Exploratory Data Analysis it is understood that the dependent variable ***cnt*** is a numeric. Therefore we need a **Regression Model** for fitting the data.

We can start modeling by using the simplest regression model. Linear Regression.

3.1.1 Linear Regression

Listing 3.1: Code for Linear Regression

```
model = lm.LinearRegression().fit(x_train,y_train)

predictions = model.predict(x_test.drop('dteday',axis=1))

print('r2_score:',r2_score(y_test,predictions),'\nMAPE:',
      mape(predictions,y_test),
       '\nScore:',model.score(x_train,y_train),
       '\nRMSE:',rmse(y_test,predictions))

sns.regplot(x = predictions,y = y_test,scatter_kws={"s": 40})
```

Output is as follows:

r2_score: 0.7297211343210271

MAPE: 16.282254139103873

Score: 0.8260884517179874

RMSE: 912.9030618723432

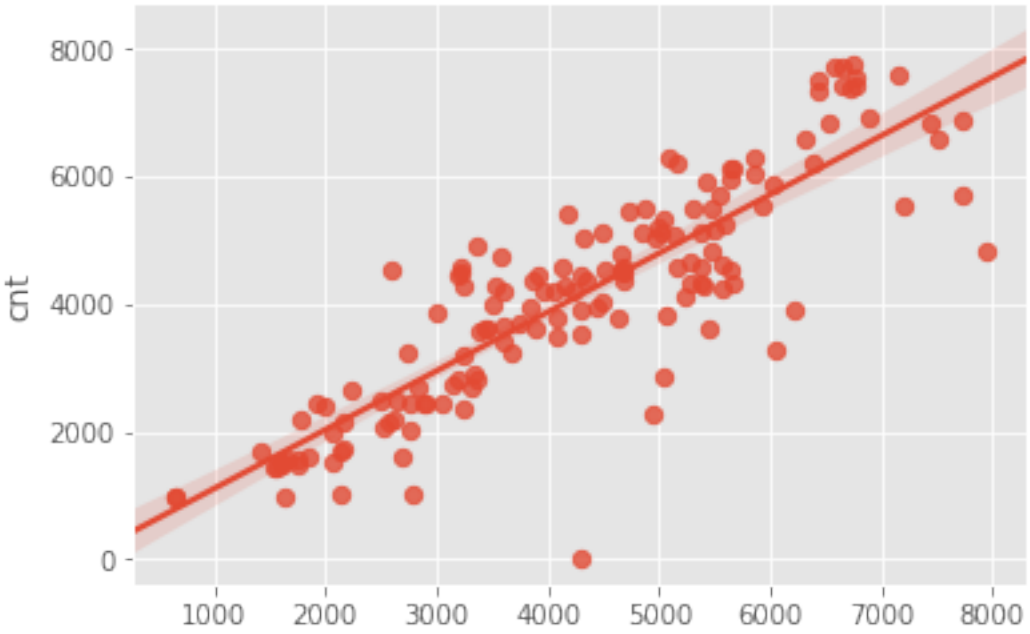

Figure 3.1: Linear Regression

3.1.2 Random Forest Regressor

Listing 3.2: Code for Random Forest Regressor

```
model = RandomForestRegressor(max_depth=5,n_estimators=10).fit(x_train,y_train)

predictions = model.predict(x_test.drop('dteday',axis=1))

print('r2_score:',r2_score(y_test,predictions),
      '\nMAPE:',mape(predictions,y_test),
      '\nScore:',model.score(x_train,y_train),
      '\nRMSE:',rmse(y_test,predictions))

sns.regplot(x = predictions,y = y_test,scatter_kws={"s": 40},color = 'orange')
```

Output is as follows:

r2_score: 0.800436204175282

MAPE: 13.228957057122456

Score: 0.9145630055610217

RMSE: 784.4395788583342

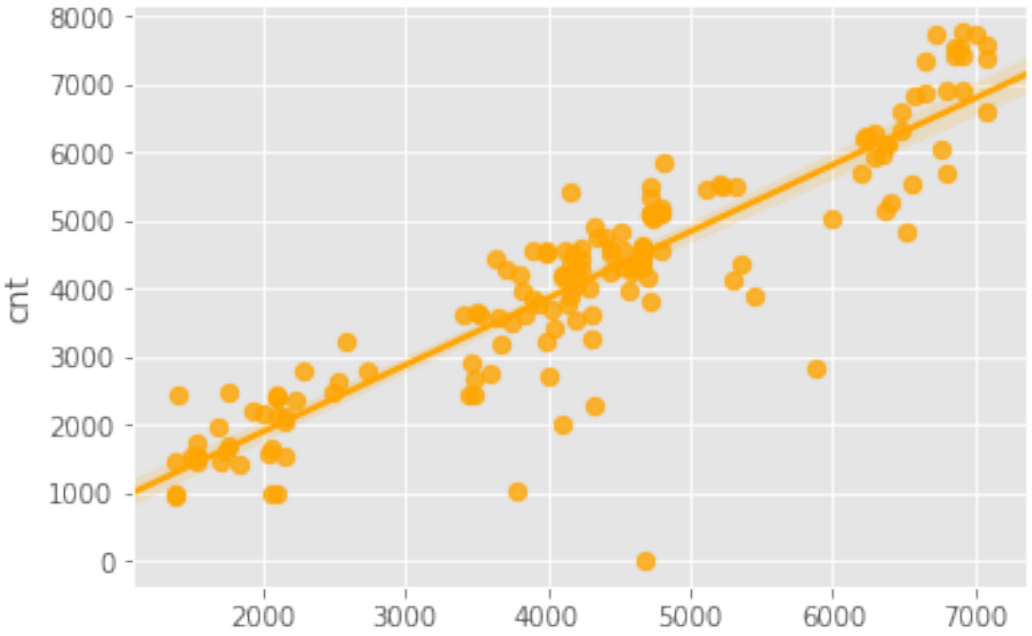

Figure 3.2: Random Forest Regressor

3.1.3 Bagging Regressor

Listing 3.3: Code for Bagging Regressor

```
model = BaggingRegressor().fit(x_train,y_train)

predictions = model.predict(x_test.drop('dteday',axis=1))

print('r2_score:',r2_score(y_test,predictions),
      '\nMAPE:',mape(predictions,y_test),'\nScore:',
      model.score(x_train,y_train),'\nRMSE:',
      rmse(y_test,predictions))

sns.regplot(x = predictions,y = y_test,scatter_kws={"s": 40},color = '#528deb')
```

Output is as follows:

r2_score: 0.8834834030590797

MAPE: 11.162928409782937

Score: 0.9732954438733665

RMSE: 650.7388817224851

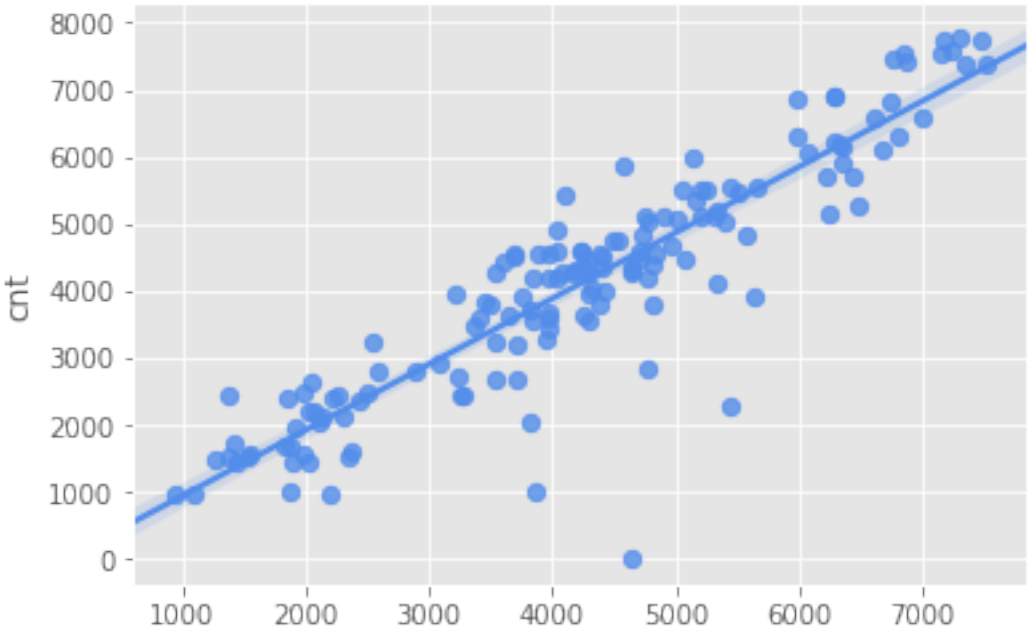

Figure 3.3: Bagging Regressor

A Bagging regressor is an ensemble meta-estimator that fits base regressors each on random subsets of the original dataset and then aggregate their individual predictions (either by voting or by averaging) to form a final prediction.

Chapter 4

Conclusion

4.1 Model Evaluation

Now that we have three models to select from, we can compare the metrics of each model and select the best from all of them. Clearly Bagging Regressor performs the best overall with low error rate and better scores.

4.1.1 Mean Squared Error

The dependent variable ***cnt*** is in the range of 1000 to 8000+ and cannot be normalised whereas the predictor variables are normalised. Therefore the error of 750 can be negligible since the MAPE error is low and the score of the model is high.

Chapter 5

References

1. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.BaggingRegressor.html
2. https://scikit-learn.org/stable/modules/ensemble.htmlbagging