Bike Renting - Report

Imran Syed

13 November 2018

Contents

1 Introduction 1

1.1 Problem Statement 1

1.2 Data 1

2 Methodology 3

2.1 Pre Processing 3

2.1.1 Exploratory Data Analysis 3

2.1.2 Feature Selection 6

2.2 Modelling 8

2.2.1 Model Selection 8

2.2.2 Multiple Linear regression 8

2.2.3 Decision Tree Regressor 9

2.2.4 Random Forest Regressor 9

3 Conclusion 11

3.1 Model Evaluation 11

3.2 Model Selection 11

A Python Code 13

1 Exploratory data Analysis¶ 24

2 Regression Models 35
3 Multiple linear Regression 35
4 Decision Tree Regressor 36
5 Random Forest Regressor 38
6 Result 39
7 Output using Selected Model i.e. Random Forest Regressor 41

1 Introduction

1.1 Problem Statement

The objective of this case is to Prediction of bike rental count on daily based on the environmental and seasonal settings. Bike Rent is a program which is running around the world to get membership and renting a bike. People used to rent a bike from a place for some time and return to the same place or other place of the bike renting branch. There are two types of people who rent the bike, one is the registered user who rent the bike for almost daily purpose like service men, students, employees on the other hand there are casuals peoples who rent the bike on need, like people in the vacation. This is the problem in which we have to predict the count of people who rent the bike on summing registered and casual problem. It is a **regression problem** in which the target variable is continuous one.

1.2 Data

The datasets shows hourly rental data for two years (2011-2012). Here, we have to predict the total count of the bikes rented during each hour.

In the data, they have separately given bike demand by registered, casual's users and sum of both is given as count.

The data has 16 variables in which 15 are independent variables and 1 are dependent variables.

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

Table 1: Training Data (Columns 1-9)

temp	atemp	hum	windspeed	casua	registered	cnt
0.344167	**0.363625**	**0.805833**	**0.160446**	**331**	**654**	**985**
0.363478	**0.353739**	**0.696087**	**0.248539**	**131**	**670**	**801**
0.196364	**0.189405**	**0.437273**	**0.248309**	**120**	**1229**	**1349**
0.2	**0.212122**	**0.590435**	**0.160296**	**108**	**1454**	**1562**
0.226957	**0.22927**	**0.436957**	**0.1869**	**82**	**1518**	**1600**

Table 2: Training Data (Columns 10-16)

The features available to predict the count of rental bikes on a particular day are:

S.No.	Features
0	Instant
1	Dteday
2	Season
3	Yr
4	mnth
5	holiday
6	weekday
7	workingday
8	weathersit
9	temp
10	atemp
11	hum
12	windspeed
13	casual
14	registered
15	cnt

Table 3: Available Features

2 Methodology

2.1 Pre Processing

Data Pre-Processing means the processing the data before modelling the data. It is includes data exploration, data manipulation, data cleaning and visualizing the data. Pre-Processing is done to make the data in a well-structured way because the data we get from the client could contain the missing values, outliers etc. So, to convert this unstructured data in a well-structured way data Pre-Processing is done. This phenomenon is called **Exploratory Data Analysis**. It includes many techniques to convert the data in a structured format like, Missing Value Analysis, Outlier Analysis, and Feature selection and Feature scaling.

Before Pre-Processing we have to convert the variables into categorical or numerical based on model selection process.

Since, season has only four values so better to convert it to categorical variable from numeric variable. Similarly, yr, mnth, holiday, weekday, workingday and weathersit have only some repeated values so better to convert them to the categorical variable from numerical variable.

2.1.1 Exploratory Data Analysis

In Fig. 1, 2, 3, 4 we have plotted the distribution plots of the continuous variables we have in the data and we can clearly see that all of them are pretty much uniformly distributed.

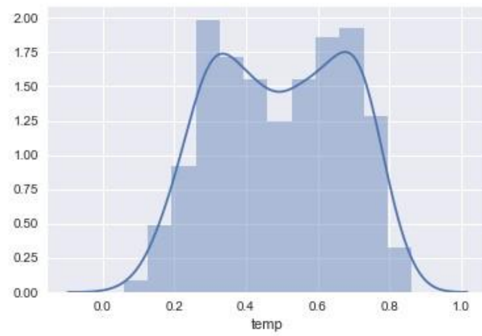


Figure 1: Temperature Distribution Plot

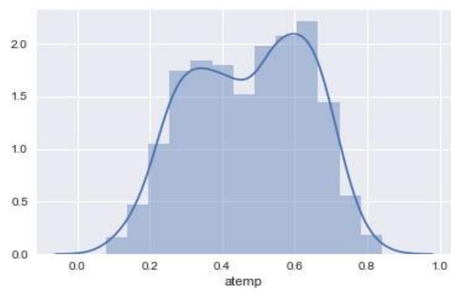


Figure 2: aTemperature Distribution Plot

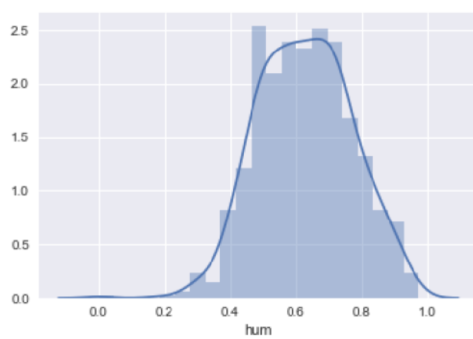


Figure 3: Humidity Distribution Plot

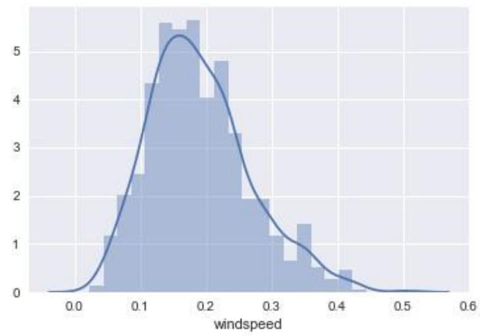


Figure 4: Windspeed Distribution Plot We also look at the target variable

i.e. **cnt** in Fig. 5

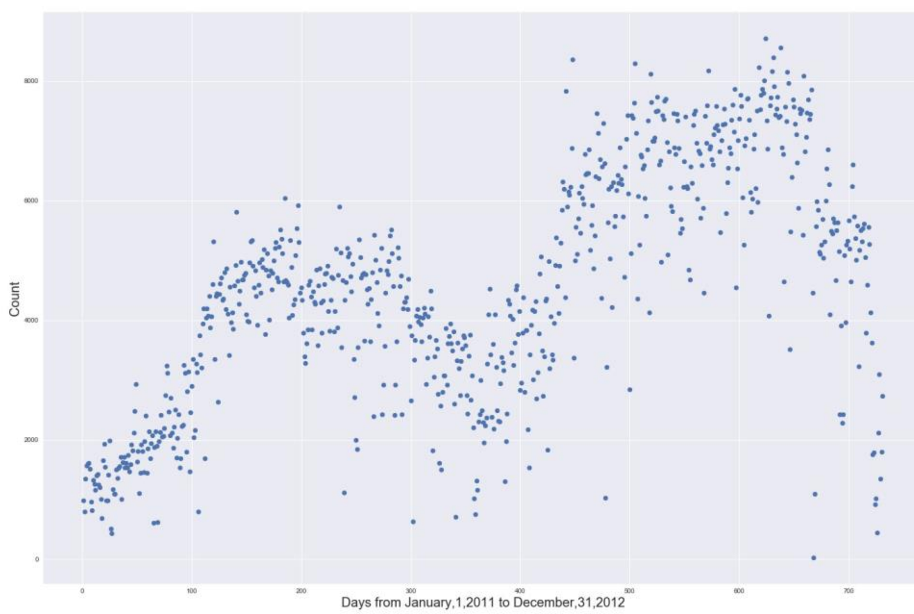


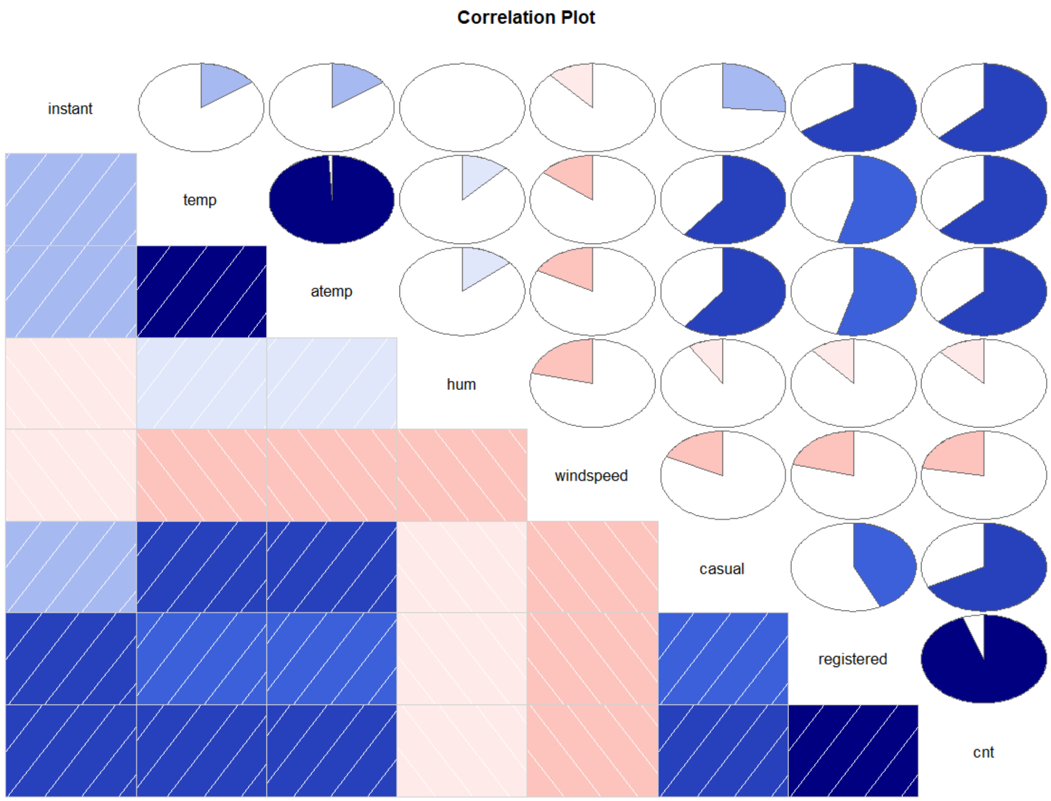
Figure 5: Count scatter Plot

2.1.2 Feature Selection

Now we use the data to see the releationship between the viariables.Feature selection is the meth od of selecting a subset of relevant features(variables,predictors) for use in model construction. It r educes the dimensions of the data. There are different variables in the data which are not playing t he useful role to predict the target variable on the other hand there are some variables which stron gly play the role to pedict the target variable. There are also some independent variable which are highly correlated with each other. So, these all the features are taken into consideration to and the variables which are needed are only taken into consideration to predict the target variable.
For the numerical variable we go for correlation technique to see the correlation between the varia bles. This is done by plotting the correlation plot. The extreme blue color shows the variables are st rongly positively correleated and the extreme red color shows the variables are strongly negatively correlated with each other.

	temp	atemp	hum	windspeed
Temp	1	0.991702	0.126963	-0.15794
atemp	0.991702	1	0.139988	-0.18364
Hum	0.126963	0.139988	1	-0.24849
windspeed	-0.15794	-0.18364	-0.24849	1

Table 4: Correlation Matrix



In the given data, there are 8 continuous variable as instant, temp, atemp, hum, windspeed, casual, registered and cnt. The above diagram shows the dependency between the different variables. Let us considered each variable one by one:-

a. Instant:- this variable is the index of the data. It shows what is the index of first and last observation present in the data. So, we can ignore this variable in the feature selection.
b. Temp:- it is normalized temperature in celsius. The normalized value is calculated by the for mula,

   Value= (t-t_min)/(t_max-t_min)

   The temp shows a high dependency with the target variable cnt. So, temp variable is includ ed in the feature selection to predict the target variable.
c. Atemp:- it is also the noramlized temperature in celsius. The noramlized value is calculated by the formula,

   Value=(t-t_min)/(t_max-t_min)

   The atemp also shows a high dependency with the target variable cnt. But atemp and temp are almost strongly correlated with each other. So, while feature selection we take only one variable for prediction. So here we take only temp variable for prediction and ignore the atemp.

d. Hum:- it is also the normalized value. It is calculated by dividing the value with the max humidity which is 100. In the diagram we see that the hum variable is very less correleated with the target variable. So better to ignore this variable while feature selection.
e. Windpeed:- it is also the normalized value. It is calculated by dividing the value with the max value which is 67. The correlation between the hum and cnt is average. So this variable is included in the feature selection.
f. Casual:- it is the user which rent the bike casually. It highly correlated with the target variable, as it adds to the target variable. The casual variable is count on the daily survey of the data.
g. Registered:- it is the daily user wo rent the bikes. It is stronly correlated with each other. As the number of registered users increases by day. It is count of on the basis of daily uses. It aads to the target variable.

For the categorical variable we use the chi-square test for to see the relation between the categorical variables. The relation is based on the p-value. If the p-value is less than 0.05 the we reject the null hypothesis and say that the variables are independent else the variables are dependent.

2.2 Modelling

2.2.1 Model Selection

The dependent variable, in our case *cnt* is continuous. So the only predictive analytics we can use is **Regression**. Now we will try building various regressors to predict our target variable and then select whichever will work best.

2.2.2 Multiple Linear regression

It is the most common form of linear regression analysis. As a predictive analysis, the multiple linear regression is used to explain the relationship between one continuous dependent variable and two or more independent variables. The independent variables can be continuous or categorical (dummy coded as appropriate).

We can tune the hyper parameters of Linear Regression like *copy-X, normalize, fit-intercept*. For finding the best parameters for our data we run a grid search over various values of these parameters based on which we found that the *copy-X* of **True**, *normalize* of **True** and *fit-intercept* of **True** works best.

2.2.3 Decision Tree Regressor

Decision Tree(D.T) is a rule

A predictive model based on branching series of Boolean test

Can be used for classification and regression

Extremely easy to understand by the business users

Two most popular D.T algorithms

1. C5.O
    - Multi Split
    - Information Gain
    - Rule Base Pruning
2. CART
    - Binary Split
    - Gini Index
    - Tree Based Pruning

    I.G=Entropy of the system before split – Entropy of the system after split

    Entropy=Uncertainty in the data

    Select the variable whose I.G is high

We can also tune the hyper parameters of Decision Tree like *max-depth, min-samples-leaf, maxfeatures*. For finding the best parameters for our data we run a grid search over various values of these parameters based on which we found that the *max-depth* of **12**, *min-samples-leaf* of **10** and *max-features* of **auto** works best.

2.2.4 Random Forest Regressor

Random Forest is an ensemble that consist of many D.T.

The method combines Breimen's "bagging" idea and the random selection of feature.

Can be used for Classification and Regression.

- Gini index

  Gini index says, if we select two items from a population at random then they must be of same class and probability for this is 1 if population is pure.

1. It works with categorical target variable "Success" or "Failure".
2. It performs only Binary splits
3. Higher the value of Gini higher the homogeneity.
4. CART (Classification and Regression Tree) uses Gini method to create binary splits.

$$Gini = 1 - \sum_{i=1}^{C} (p_i)^2$$

The algorithm works as 1 – (P(class1)^2 + P(class2)^2 + ... + P(classN)^2

```
Actual=1 predicted 1
1 0 , 0,1, 0 0
P(Target=1).P(Target=1) + P(Target=1).P(Target=0) +
P(Target=0).P(Target=1) + P(Target=0).P(Target=0) = 1
P(Target=1).P(Target=0) + P(Target=0).P(Target=1) = 1 —
P^2(Target=0) — P^2(Target=1)
```

We can also tune the hyper parameters of Random Forest like *max-depth, min-samples-leaf, maxfeatures, n-estimators, oob-score*. For finding the best parameters for our data we run a grid search over various values of these parameters based on which we found that the *max-depth* of **15**, *minsamples-leaf* of **2**, *max-features* of **auto**, *n-estimators* of **600** and *oob-score* of **True** works best.

3 Conclusion

3.1 Model Evaluation

Model evaluation is defined as how the predicted value are correct with the actual value.
For this we have different techniques like:-

a. MAE or MAD(Mean Absolute Error/Deviation)
   It Averages the Absolute Error

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n}|f_i - y_i|$$

b. MAPE(Mean Absolute Percentage Error)
   It measures accuracy as a percentage of error.

$$\mathrm{M} = \frac{1}{n}\sum_{t=1}^{n}\left|\frac{A_t - F_t}{A_t}\right|$$

c. RMSE/RMSD(Root Mean Square Error/Deviation)
   It squares the error, find their average and take the square root.

$$\mathrm{RMSD} = \sqrt{\frac{\sum_{t=1}^{n}(\hat{y}_t - y_t)^2}{n}}.$$

   Here, for the evaluation purpose we take the MAPE technique to check the accuracy of the model.

3.2 Model Selection

After analyzing all the models and checking the evaluation result the best model is the Multiple Regression which gives the accuracy of 96.8%. so, for the given data we choose the multiple Regression Model for predictions.

The casual and registered users can be calculated independently, but the data is very less and it does not contains the hour variable, which shows the bike rented in certain interval of time. So calculating the data indecently causes more error due to less data.

And the casual and registered user are counted on the daily basis not the hourly basis, so to get the better result the model is developed on combining the casual and registered data. As both the variable combine to give the total count users who rent the bike. So the cnt is highly dependent on the casual and registered users.

```
#calculate MAPE
MAPE = function(y, yhat){
  mean(abs((y - yhat)/y))*100
}
```

a. Decision Tree:-

```
[1] 12.61215
```

Here, we get a error of 12.61% which indicates that our model is giving the accuracy of a 87.69%, which is a good result.

b. Random Forest:-

```
[1] 5.995044
```

Here, we get a error of approximate 6% which indicates that our model is 94% accurate which shows that the result is better than the Decision Tree.

c. Multiple Regression:-

```
[1] 3.216991
```

Here, the error we get a error of 3.2% which indicates that our result is 96.8% accurate which is the best one.

#Decision Tree

Accuracy=87.69%

Error=12.61%

MAPE(test[,8], predictions_DT)=0.698272

#Random Forest

Accuracy=94%

Error=6%

MAPE(test[,8], RF_Predictions)=0.1794407

#Multiple Regression

Accuracy=96.8%

Error=3.2%

MAPE(test[,8], predictions_LR)=0.2019088

	Model	Training MAPE	Training RMSE	Test MAPE	Test RMSE
0	Multiple Linear Regressor	46.161484	765.895325	19.081074	794.448747
1	Decision Tree Regressor	51.809725	706.577452	25.133274	892.361957
2	Random Forest Regressor	27.097224	368.149119	20.023176	708.951493

A Python Code

```
# Importing Libraries

import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.cross_validation import train_test_split
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import mean_squared_error
from math import sqrt
from scipy.stats import chi2_contingency
```

```
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
```

```
%matplotlib inline
```

```
# Setting Working Directory
```

```
os.chdir("C:\\Users\\DELL\\Desktop\\project 2")
```

```
# Loading Data
```

```
data = pd.read_csv('project_2.csv')
```

```
data.shape
```

```
(731, 32)
```

```
type(data)
```

```
pandas.core.frame.DataFrame
```

```
data.columns
```

```
Index(['yr', 'holiday', 'workingday', 'temp', 'atemp', 'hum', 'windspeed'
,
       'casual', 'registered', 'cnt', 'season_2', 'season_3', 'season_4',
       'mnth_2', 'mnth_3', 'mnth_4', 'mnth_5', 'mnth_6', 'mnth_7', 'mnth_
8',
       'mnth_9', 'mnth_10', 'mnth_11', 'mnth_12', 'weekday_1', 'weekday_2
',
       'weekday_3', 'weekday_4', 'weekday_5', 'weekday_6', 'weathersit_2'
,
       'weathersit_3'],
      dtype='object')
```

```
data.describe
```

```
<bound method NDFrame.describe of      yr  holiday  workingday      temp
    atemp       hum  windspeed  casual  \
0     0        0           0  0.344167  0.363625  0.805833   0.160446
```

331

1	0	0	0	0.363478	0.353739	0.696087	0.248539	131
2	0	0	1	0.196364	0.189405	0.437273	0.248309	120
3	0	0	1	0.200000	0.212122	0.590435	0.160296	108
4	0	0	1	0.226957	0.229270	0.436957	0.186900	82
5	0	0	1	0.204348	0.233209	0.518261	0.089565	88
6	0	0	1	0.196522	0.208839	0.498696	0.168726	148
7	0	0	0	0.165000	0.162254	0.535833	0.266804	68
8	0	0	0	0.138333	0.116175	0.434167	0.361950	54
9	0	0	1	0.150833	0.150888	0.482917	0.223267	41
10	0	0	1	0.169091	0.191464	0.686364	0.122132	43
11	0	0	1	0.172727	0.160473	0.599545	0.304627	25
12	0	0	1	0.165000	0.150883	0.470417	0.301000	38
13	0	0	1	0.160870	0.188413	0.537826	0.126548	54
14	0	0	0	0.233333	0.248112	0.498750	0.157963	222
15	0	0	0	0.231667	0.234217	0.483750	0.188433	251
16	0	1	0	0.175833	0.176771	0.537500	0.194017	117
17	0	0	1	0.216667	0.232333	0.861667	0.146775	9
18	0	0	1	0.292174	0.298422	0.741739	0.208317	78
19	0	0	1	0.261667	0.255050	0.538333	0.195904	83

20	0	0	1	0.177500	0.157833	0.457083	0.353242	75
21	0	0	0	0.059130	0.079070	0.400000	0.171970	93
22	0	0	0	0.096522	0.098839	0.436522	0.246600	150
23	0	0	1	0.097391	0.117930	0.491739	0.158330	86
24	0	0	1	0.223478	0.234526	0.616957	0.129796	186
25	0	0	1	0.217500	0.203600	0.862500	0.293850	34
26	0	0	1	0.195000	0.219700	0.687500	0.113837	15
27	0	0	1	0.203478	0.223317	0.793043	0.123300	38
28	0	0	0	0.196522	0.212126	0.651739	0.145365	123
29	0	0	0	0.216522	0.250322	0.722174	0.073983	140
..	..	...	...	...	...	...	...	...
701	1	0	0	0.347500	0.359208	0.823333	0.124379	892
702	1	0	1	0.452500	0.455796	0.767500	0.082721	555
703	1	0	1	0.475833	0.469054	0.733750	0.174129	551
704	1	0	1	0.438333	0.428012	0.485000	0.324021	331
705	1	0	1	0.255833	0.258204	0.508750	0.174754	340
706	1	0	1	0.320833	0.321958	0.764167	0.130600	349
707	1	0	0	0.381667	0.389508	0.911250	0.101379	1153
708	1	0	0	0.384167	0.390146	0.905417	0.157975	441
709	1	0	1	0.435833	0.435575	0.925000	0.190308	

329							
710	1	0	1	0.353333	0.338363	0.596667	0.296037
282							
711	1	0	1	0.297500	0.297338	0.538333	0.162937
310							
712	1	0	1	0.295833	0.294188	0.485833	0.174129
425							
713	1	0	1	0.281667	0.294192	0.642917	0.131229
429							
714	1	0	0	0.324167	0.338383	0.650417	0.106350
767							
715	1	0	0	0.362500	0.369938	0.838750	0.100742
538							
716	1	0	1	0.393333	0.401500	0.907083	0.098258
212							
717	1	0	1	0.410833	0.409708	0.666250	0.221404
433							
718	1	0	1	0.332500	0.342162	0.625417	0.184092
333							
719	1	0	1	0.330000	0.335217	0.667917	0.132463
314							
720	1	0	1	0.326667	0.301767	0.556667	0.374383
221							
721	1	0	0	0.265833	0.236113	0.441250	0.407346
205							
722	1	0	0	0.245833	0.259471	0.515417	0.133083
408							
723	1	0	1	0.231304	0.258900	0.791304	0.077230
174							
724	1	1	0	0.291304	0.294465	0.734783	0.168726
440							
725	1	0	1	0.243333	0.220333	0.823333	0.316546
9							
726	1	0	1	0.254167	0.226642	0.652917	0.350133
247							
727	1	0	1	0.253333	0.255046	0.590000	0.155471
644							
728	1	0	0	0.253333	0.242400	0.752917	0.124383
159							

```
729   1        0          0  0.255833  0.231700  0.483333   0.350754
 364
730   1        0          1  0.215833  0.223487  0.577500   0.154846
 439

     registered   cnt      ...       mnth_11  mnth_12  weekday_1  weekday
_2  \
0           654   985      ...             0        0          0
 0
1           670   801      ...             0        0          0
 0
2          1229  1349      ...             0        0          1
 0
3          1454  1562      ...             0        0          0
 1
4          1518  1600      ...             0        0          0
 0
5          1518  1606      ...             0        0          0
 0
6          1362  1510      ...             0        0          0
 0
7           891   959      ...             0        0          0
 0
8           768   822      ...             0        0          0
 0
9          1280  1321      ...             0        0          1
 0
10         1220  1263      ...             0        0          0
 1
11         1137  1162      ...             0        0          0
 0
12         1368  1406      ...             0        0          0
 0
13         1367  1421      ...             0        0          0
 0
14         1026  1248      ...             0        0          0
 0
15          953  1204      ...             0        0          0
 0
```

```
16          883  1000     ...          0        0          1
 0
17          674   683     ...          0        0          0
 1
18         1572  1650     ...          0        0          0
 0
19         1844  1927     ...          0        0          0
 0
20         1468  1543     ...          0        0          0
 0
21          888   981     ...          0        0          0
 0
22          836   986     ...          0        0          0
 0
23         1330  1416     ...          0        0          1
 0
24         1799  1985     ...          0        0          0
 1
25          472   506     ...          0        0          0
 0
26          416   431     ...          0        0          0
 0
27         1129  1167     ...          0        0          0
 0
28          975  1098     ...          0        0          0
 0
29          956  1096     ...          0        0          0
 0
..          ...   ...     ...        ...      ...        ...        .
..
701        3757  4649     ...          0        1          0
 0
702        5679  6234     ...          0        1          1
 0
703        6055  6606     ...          0        1          0
 1
704        5398  5729     ...          0        1          0
 0
705        5035  5375     ...          0        1          0
```

							0
706	4659	5008	...	0	1	0	0
707	4429	5582	...	0	1	0	0
708	2787	3228	...	0	1	0	0
709	4841	5170	...	0	1	1	0
710	5219	5501	...	0	1	0	1
711	5009	5319	...	0	1	0	0
712	5107	5532	...	0	1	0	0
713	5182	5611	...	0	1	0	0
714	4280	5047	...	0	1	0	0
715	3248	3786	...	0	1	0	0
716	4373	4585	...	0	1	1	0
717	5124	5557	...	0	1	0	1
718	4934	5267	...	0	1	0	0
719	3814	4128	...	0	1	0	0
720	3402	3623	...	0	1	0	0
721	1544	1749	...	0	1	0	0
722	1379	1787	...	0	1	0	0
723	746	920	...	0	1	1	0
724	573	1013	...	0	1	0	1

```
725          432   441      ...            0        1          0
 0
726         1867  2114      ...            0        1          0
 0
727         2451  3095      ...            0        1          0
 0
728         1182  1341      ...            0        1          0
 0
729         1432  1796      ...            0        1          0
 0
730         2290  2729      ...            0        1          1
 0

     weekday_3  weekday_4  weekday_5  weekday_6  weathersit_2  weathersit
_3
0            0          0          0          1             1
 0
1            0          0          0          0             1
 0
2            0          0          0          0             0
 0
3            0          0          0          0             0
 0
4            1          0          0          0             0
 0
5            0          1          0          0             0
 0
6            0          0          1          0             1
 0
7            0          0          0          1             1
 0
8            0          0          0          0             0
 0
9            0          0          0          0             0
 0
10           0          0          0          0             1
 0
11           1          0          0          0             0
 0
```

12	0	1	0	0	0	0
13	0	0	1	0	0	0
14	0	0	0	1	1	0
15	0	0	0	0	0	0
16	0	0	0	0	1	0
17	0	0	0	0	1	0
18	1	0	0	0	1	0
19	0	1	0	0	1	0
20	0	0	1	0	0	0
21	0	0	0	1	0	0
22	0	0	0	0	0	0
23	0	0	0	0	0	0
24	0	0	0	0	1	0
25	1	0	0	0	0	1
26	0	1	0	0	0	0
27	0	0	1	0	1	0
28	0	0	0	1	0	0
29	0	0	0	0	0	0
..	...	...	...	...	...	. ..
701	0	0	0	0	1	

0					
702 0	0	0	0	0	0
703 0	0	0	0	0	0
704 0	1	0	0	0	0
705 0	0	1	0	0	0
706 0	0	0	1	0	1
707 0	0	0	0	1	1
708 0	0	0	0	0	1
709 0	0	0	0	0	1
710 0	0	0	0	0	1
711 0	1	0	0	0	1
712 0	0	1	0	0	0
713 0	0	0	1	0	0
714 0	0	0	0	1	0
715 0	0	0	0	0	1
716 0	0	0	0	0	1
717 0	0	0	0	0	0
718 0	1	0	0	0	0
719 0	0	1	0	0	1
720 0	0	0	1	0	1

```
721         0         0         0         1            0
 0
722         0         0         0         0            0
 0
723         0         0         0         0            1
 0
724         0         0         0         0            1
 0
725         1         0         0         0            0
 1
726         0         1         0         0            1
 0
727         0         0         1         0            1
 0
728         0         0         0         1            1
 0
729         0         0         0         0            0
 0
730         0         0         0         0            1
 0

[731 rows x 32 columns]>
```

1 Exploratory data Analysis¶

```
data.head()
```

instant

dteday

season

yr

mnth

holiday

weekday

workingday

weathersit

temp

atemp

hum
windspeed
casual
registered
cnt
0
1
01/01/2011
1
0
1
0
6
0
2
0.344167
0.363625
0.805833
0.160446
331
654
985
1
2
02/01/2011
1
0
1
0
0
0
2
0.363478

0.353739
0.696087
0.248539
131
670
801
2
3
03/01/2011
1
0
1
0
1
1
1
0.196364
0.189405
0.437273
0.248309
120
1229
1349
3
4
04/01/2011
1
0
1
0
2
1
1

0.200000
0.212122
0.590435
0.160296
108
1454
1562
4
5
05/01/2011
1
0
1
0
3
1
1
0.226957
0.229270
0.436957
0.186900
82
1518
1600

```
features = pd.DataFrame(data.columns)
#features.to_csv('features.csv')

# Continous variables

cnames = ['temp','atemp','hum','windspeed']
cat_names = ['season', 'mnth', 'holiday', 'weekday', 'workingday', 'weath
ersit']

# Corelation between continous variables
```

```
corr = data[cnames].corr()
corr
#corr.to_csv('Correlations.csv')
```

temp

atemp

hum

windspeed

temp

1.000000

0.991702

0.126963

-0.157944

atemp

0.991702

1.000000

0.139988

-0.183643

hum

0.126963

0.139988

1.000000

-0.248489

windspeed

-0.157944

-0.183643

-0.248489

1.000000

```
sns.distplot(data['temp'])
#plt.savefig('temp.png')
```

```
C:\Users\DELL\Anaconda4\lib\site-packages\matplotlib\axes\_axes.py:6462:
UserWarning: The 'normed' kwarg is deprecated, and has been replaced by t
he 'density' kwarg.
  warnings.warn("The 'normed' kwarg is deprecated, and has been "
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x234a6a30198>
```

png

png

```
sns.distplot(data['atemp'])
#plt.savefig('atemp.png')
```

```
C:\Users\DELL\Anaconda4\lib\site-packages\matplotlib\axes\_axes.py:6462:
UserWarning: The 'normed' kwarg is deprecated, and has been replaced by t
he 'density' kwarg.
  warnings.warn("The 'normed' kwarg is deprecated, and has been "
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x234a6e2ab00>
```

png

png

```
sns.distplot(data['hum'])
#plt.savefig('hum.png')
```

```
C:\Users\DELL\Anaconda4\lib\site-packages\matplotlib\axes\_axes.py:6462:
UserWarning: The 'normed' kwarg is deprecated, and has been replaced by t
he 'density' kwarg.
  warnings.warn("The 'normed' kwarg is deprecated, and has been "
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x234a6e983c8>
```

png

png

```
sns.distplot(data['windspeed'])
#plt.savefig('windspeed.png')
```

```
C:\Users\DELL\Anaconda4\lib\site-packages\matplotlib\axes\_axes.py:6462:
UserWarning: The 'normed' kwarg is deprecated, and has been replaced by t
he 'density' kwarg.
  warnings.warn("The 'normed' kwarg is deprecated, and has been "
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x234a6f07b00>
```

png

png

```
plt.figure(figsize=(24,16))
plt.scatter(data['instant'], data['cnt'])
plt.xlabel('Days from January,1,2011 to December,31,2012', fontsize = 20)
plt.ylabel('Count', fontsize =20)
#plt.savefig('RentCount.png')
```

```
Text(0,0.5,'Count')
```

png

png

```
# Creating Dummy Variables for non-binary categorical variables

for i in ['season','mnth','weekday','weathersit']:
    temp = pd.get_dummies(data[i], prefix = i)
    data = data.join(temp)
    data.drop(i, axis =1,inplace = True)
```

```
data.drop(['instant','dteday','season_1','mnth_1','weekday_0','weathersit
_1'],axis=1,inplace = True)
```

```
data.head()
```

yr
holiday
workingday
temp
atemp
hum
windspeed
casual
registered
cnt
...
mnth_11
mnth_12
weekday_1
weekday_2
weekday_3
weekday_4
weekday_5
weekday_6
weathersit_2
weathersit_3
0
0
0
0
0.344167
0.363625
0.805833
0.160446
331
654
985
...

0
0
0
0
0
0
0
1
1
0
1
0
0
0
0.363478
0.353739
0.696087
0.248539
131
670
801
...
0
0
0
0
0
0
0
0
1
0
2

0
0
1
0.196364
0.189405
0.437273
0.248309
120
1229
1349
...
0
0
1
0
0
0
0
0
0
0
3
0
0
1
0.200000
0.212122
0.590435
0.160296
108
1454
1562
...

0
0
0
1
0
0
0
0
0
0
4
0
0
1
0.226957
0.229270
0.436957
0.186900
82
1518
1600
...
0
0
0
0
1
0
0
0
0
0
5 rows × 32 columns

```
# Splitting the data into train and test sets

train,test = train_test_split(data,test_size =0.2, random_state =0)
```

```
# Preparing Data for modelling

X_train = train.drop(['casual','registered','cnt','temp'],axis=1)
X_test = test.drop(['casual','registered','cnt','temp'],axis=1)
y_casual = train['casual']
y_registered = train['registered']
y_cnt = train['cnt']
```

```
# Evaluation Functions

def MAPE(y_true, y_pred):
    mape = np.mean(np.abs((y_true - y_pred) / y_true))*100
    return mape
#Calculate MAPE

def RMSE(y_true, y_pred):
    rms = sqrt(mean_squared_error(y_true, y_pred))
    return rms
#Calculate RMSE
```

2 Regression Models

3 Multiple linear Regression

```
from sklearn.linear_model import LinearRegression
```

```
# Grid Search for best Parameters

reg_lm = LinearRegression()
params_lm = [{'copy_X':[True, False],
              'fit_intercept':[True,False],
              'normalize':[True, False]}]
grid_search_lm = GridSearchCV(reg_lm, param_grid = params_lm, cv =10, n_j
obs =-1)
grid_search_lm = grid_search_lm.fit(X_train,y_cnt)
```

```
grid_search_lm.best_score_
```

```
0.8103662854156968
```

```
grid_search_lm.best_params_
```

```
{'copy_X': True, 'fit_intercept': True, 'normalize': True}
```

```
# Training with best paramaeters

reg_lm_best = LinearRegression(copy_X=True, fit_intercept=True, normalize
=True)
reg_lm_best.fit(X_train,y_cnt)
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=Tru
e)
```

```
# Evaluating on training set

y_pred_lm = reg_lm_best.predict(X_train)
mape1_lm = MAPE(y_cnt, y_pred_lm)
rmse1_lm = RMSE(y_cnt, y_pred_lm)
print('MAPE : {:.2f}'.format(mape1_lm))
print('RMSE : {:.2f}'.format(rmse1_lm))
```

```
MAPE : 46.16
RMSE : 765.90
```

```
# Evaluating on Test Set

y_pred_lm = reg_lm_best.predict(X_test)
mape2_lm = MAPE(test['cnt'], y_pred_lm)
rmse2_lm = RMSE(test['cnt'], y_pred_lm)
print('MAPE : {:.2f}'.format(mape2_lm))
print('RMSE : {:.2f}'.format(rmse2_lm))
```

```
MAPE : 19.08
RMSE : 794.45
```

4 Decision Tree Regressor

```
from sklearn.tree import DecisionTreeRegressor
```

```
# Grid Search for best Parameters
```

```
reg_dt = DecisionTreeRegressor(random_state = 0)
params = [{'max_depth':[2,4,6,8,10,12,15],
           'max_features':['auto','sqrt'],
           'min_samples_leaf':[2,4,6,8,10]}]
grid_search_dt = GridSearchCV(reg_dt, param_grid = params, cv =10, n_jobs
 =-1)
grid_search_dt = grid_search_dt.fit(X_train,y_cnt)
```

```
grid_search_dt.best_score_
```

```
0.7923245462173745
```

```
grid_search_dt.best_params_
```

```
{'max_depth': 12, 'max_features': 'auto', 'min_samples_leaf': 10}
```

```
# Training with best parameters

reg_dt_best = DecisionTreeRegressor(random_state = 0, max_depth = 12,
                                    min_samples_leaf = 10, max_features =
 'auto')
reg_dt_best.fit(X_train,y_cnt)
```

```
DecisionTreeRegressor(criterion='mse', max_depth=12, max_features='auto',
           max_leaf_nodes=None, min_impurity_decrease=0.0,
           min_impurity_split=None, min_samples_leaf=10,
           min_samples_split=2, min_weight_fraction_leaf=0.0,
           presort=False, random_state=0, splitter='best')
```

```
# Evaluating on training set

b = reg_dt_best.predict(X_train)
mape1_dt = MAPE(y_cnt,b)
rmse1_dt = RMSE(y_cnt,b)
print('MAPE : {:.2f}'.format(mape1_dt))
print('RMSE : {:.2f}'.format(rmse1_dt))
```

```
MAPE : 51.81
RMSE : 706.58
```

```
# Evaluating on test set

y_pred_dt = reg_dt_best.predict(X_test)
```

```
mape2_dt = MAPE(test['cnt'],y_pred_dt)
rmse2_dt = RMSE(test['cnt'],y_pred_dt)
print('MAPE : {:.2f}'.format(mape2_dt))
print('RMSE : {:.2f}'.format(rmse2_dt))
```

```
MAPE : 25.13
RMSE : 892.36
```

5 Random Forest Regressor

```
from sklearn.ensemble import RandomForestRegressor
```

```
# Grid Search for best Parameters

reg_rf = RandomForestRegressor(random_state = 0)
params_rf = [{'max_depth':[8,10,12,15],
              'max_features':['auto','sqrt'],
              'min_samples_leaf':[2,4,6,8,10],
              'n_estimators': [200, 500, 600],
              'oob_score':[True, False]}]
grid_search_rf = GridSearchCV(reg_rf, param_grid = params_rf, cv =10, n_j
obs =-1)
grid_search_rf = grid_search_rf.fit(X_train,y_cnt)
```

```
grid_search_rf.best_score_
```

```
0.8467041706322699
```

```
grid_search_rf.best_params_
```

```
{'max_depth': 12,
 'max_features': 'auto',
 'min_samples_leaf': 2,
 'n_estimators': 500,
 'oob_score': True}
```

```
reg_rf_best = RandomForestRegressor(random_state = 0, max_depth = 15,
                                    max_features = 'auto', min_samples_le
af = 2,
                                    n_estimators = 600, oob_score = True)
reg_rf_best.fit(X_train,y_cnt)
```

```
RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=15,
           max_features='auto', max_leaf_nodes=None,
           min_impurity_decrease=0.0, min_impurity_split=None,
           min_samples_leaf=2, min_samples_split=2,
           min_weight_fraction_leaf=0.0, n_estimators=600, n_jobs=1,
           oob_score=True, random_state=0, verbose=0, warm_start=False)
```

```
# Evaluating on training set

c = reg_rf_best.predict(X_train)
mape1_rf = MAPE(y_cnt,c)
rmse1_rf = RMSE(y_cnt,c)
print('MAPE : {:.2f}'.format(mape1_rf))
print('RMSE : {:.2f}'.format(rmse1_rf))
```

```
MAPE : 27.10
RMSE : 368.15
```

```
# Evaluating on test set

y_pred_rf = reg_rf_best.predict(X_test)
mape2_rf = MAPE(test['cnt'],y_pred_rf)
rmse2_rf = RMSE(test['cnt'],y_pred_rf)
print('MAPE : {:.2f}'.format(mape2_rf))
print('RMSE : {:.2f}'.format(rmse2_rf))
```

```
MAPE : 20.02
RMSE : 708.95
```

6 Result

```
result = pd.DataFrame()
result['Model'] = ['Multiple Linear Regressor',
                   'Decision Tree Regressor', 'Random Forest Regressor']
result['Training MAPE'] = [mape1_lm, mape1_dt, mape1_rf]
result['Training RMSE'] = [rmse1_lm, rmse1_dt, rmse1_rf]
result['Test MAPE'] = [mape2_lm, mape2_dt, mape2_rf]
result['Test RMSE'] = [rmse2_lm,  rmse2_dt, rmse2_rf]
#result.to_csv('result.csv')
```

```python
# Evaluating on Test Set

y_pred = reg_svr_best.predict(X_test)
mape2_svr = MAPE(test['cnt'],y_pred)
rmse2_svr = RMSE(test['cnt'],y_pred)
print('MAPE : {:.2f}'.format(mape2_svr))
print('RMSE : {:.2f}'.format(rmse2_svr))
```

```
MAPE : 18.01
RMSE : 765.09
```

```python
result
```

Model

Training MAPE

Training RMSE

Test MAPE

Test RMSE

0

Multiple Linear Regressor

46.161484

765.895325

19.081074

794.448747

1

Decision Tree Regressor

51.809725

706.577452

25.133274

892.361957

2

Random Forest Regressor

27.097224

368.149119

20.023176

708.951493

```python
# Evaluating on Test Set

y_pred = reg_svr_best.predict(X_test)
mape2_svr = MAPE(test['cnt'],y_pred)
rmse2_svr = RMSE(test['cnt'],y_pred)
print('MAPE : {:.2f}'.format(mape2_svr))
print('RMSE : {:.2f}'.format(rmse2_svr))
```

```
MAPE : 18.01
RMSE : 765.09
```

7 Output using Selected Model i.e. Random Forest Regressor

```python
#pd.DataFrame(y_pred_rf).to_csv('Output.csv')
```