Bike Renting

Ujwal Kumar

November, 2019

Contents

Chapter 1

1. Introduction

1.1 Problem Statement

The objective of this Case is Predication of bike rental count on the basis of data provided to us, The data contains various environmental factors as well as other variables which might be able to impact the daily rental count of bikes.

1.2 Data

Before we proceed with the different techniques and methods we can have a brief look at our data. Our data set contains 16 variables. Out of these 16 variables there are 15 independent variables and there is 1 dependent variable (cnt) whose value we have to predict using the various variables. Here is a brief description of the different variables in our data set.

- instant: Record index
- dteday: Date
- season: Season (1:spring, 2:summer, 3:fall, 4:winter)
- yr: Year (0: 2011, 1:2012)
- mnth: Month (1 to 12) hr: Hour (0 to 23)
- holiday: weather day is holiday or not (extracted fromHoliday Schedule)
- weekday: Day of the week workingday: If day is neither weekend nor holiday is 1, otherwise is 0. weathersit: (extracted fromFreemeteo) 1: Clear, Few clouds, Partly cloudy, Partly cloudy 2: Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist 3: Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds 4: Heavy Rain + Ice Pallets + Thunderstorm + Mist, Snow + Fog
- temp: Normalized temperature in Celsius. The values are derived via (t-t_min)/(t_max-t_min), t_min=-8, t_max=+39 (only in hourly scale)
- atemp: Normalized feeling temperature in Celsius. The values are derived via (t-t_min)/(t_maxt_min), t_min=-16, t_max=+50 (only in hourly scale)
- hum: Normalized humidity. The values are divided to 100 (max)
- windspeed: Normalized wind speed. The values are divided to 67 (max)
- casual: count of casual users
- registered: count of registered users
- cnt: count of total rental bikes including both casual and registered

I

```
os.chdir("D:/Python")
df = pd.read_csv("day.csv", index_col = 0)
df.head()
```

instant	dteday	season	yr	mnth	holiday	weekday	workingday	weathersit	temp	atemp	hum	windspeed	casual	registered	cnt
1	2011-01-01	1	0	1	0	6	0	2	0.344167	0.363625	0.805833	0.160446	331	654	985
2	2011-01-02	1	0	1	0	0	0	2	0.363478	0.353739	0.696087	0.248539	131	670	801
3	2011-01-03	1	0	1	0	1	1	1	0.196364	0.189405	0.437273	0.248309	120	1229	1349
4	2011-01-04	1	0	1	0	2	1	1	0.200000	0.212122	0.590435	0.160296	108	1454	1562
5	2011-01-05	1	0	1	0	3	1	1	0.226957	0.229270	0.436957	0.186900	82	1518	1600

Data set for Bike Renting Problem

```
df.dtypes
```

```
dteday         object
season          int64
yr              int64
mnth            int64
holiday         int64
weekday         int64
workingday      int64
weathersit      int64
temp          float64
atemp         float64
hum           float64
windspeed     float64
casual          int64
registered      int64
cnt             int64
dtype: object
```

Data Types for various variables in the data set

From the figure we can see the data types that python detects when we load this data. We can see this data is combination of categorical and continuous variables. Our target variable is cnt which is the sum of casual and registered. So this is a regression problem. So right from the start we can say that casual and registered should not be part of our model. Also the instant variable is like index. So we can remove these three variables right from the start before we even go for pre-processing step.

Chapter 2

Methodology

2.1 Pre Processing

Now we will start analysing our data to get more insights on the various variables in our data set and also find out which variable might be useful for our model. We will perform various pre-processing methods on our data set so that we can make better predictions. This part is also called exploratory data analysis. Let us go through each one of them step by step.

2.1.1 Missing Value Analysis

We check our data set for missing values. If we encounter missing values we can either impute the missing values or we can choose to delete these records. We should also check if there is less than 30% missing values in a column. If there are more than 30% missing values in the column that means that column is not useful as we should not impute so many missing values. Luckily our data set has no missing values so we do need to deal with this problem.

```
missing_val = pd.DataFrame(df.isnull().sum())
missing_val
```

	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

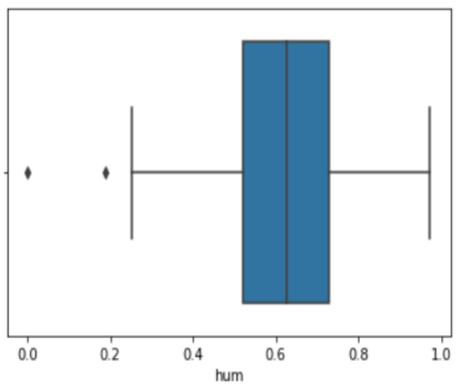
Missing Value Analysis

2.1.2 Outlier Analysis

This is the second method for pre-processing our data set. We analyse the continuous variables in our data set to check if outliers are present in them or not. Outliers are nothing but data which might be different from the trend i.e. it is either too low or too high as compared to rest of the values for that variable. For e.g. Let us say we have 10 values in a column. Out of these 10 values, 8 of them are between 1-10 and rest of the two are above 1000. Then we can say that these 2 values are outliers and we need to deal with them. Outliers can occur due to error in recording or they might simply be unique data which might be correct but will have a strange impact on our model. It's difficult to tell whether the outlier was a mistake or it was genuine data which was just different from the rest of the data. I have used boxplot method to analyse the outliers in my data set. I have also used imputation methods for the outliers instead of simply deleting the record as deleting them might result in loss of data.

```
sns.boxplot(df['hum'])
```

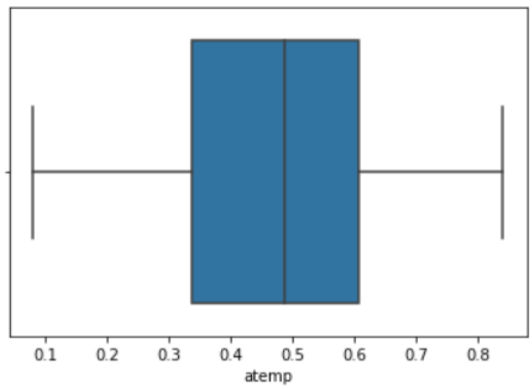
<matplotlib.axes._subplots.AxesSubplot at 0x612fbd0>



Hum Boxplot

```
sns.boxplot(df['atemp'])
```

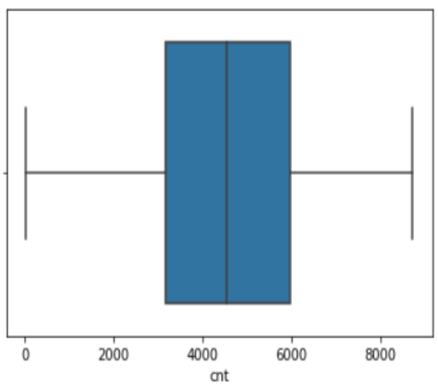
<matplotlib.axes._subplots.AxesSubplot at 0x6103ef0>



Atemp Boxplot

```
sns.boxplot(df['cnt'])
```

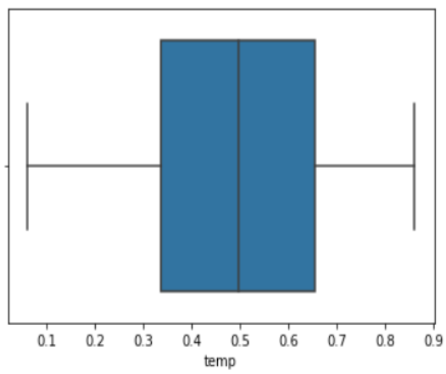
<matplotlib.axes._subplots.AxesSubplot at 0x9f24930>



Cnt Boxplot

```
sns.boxplot(df['temp'])
```

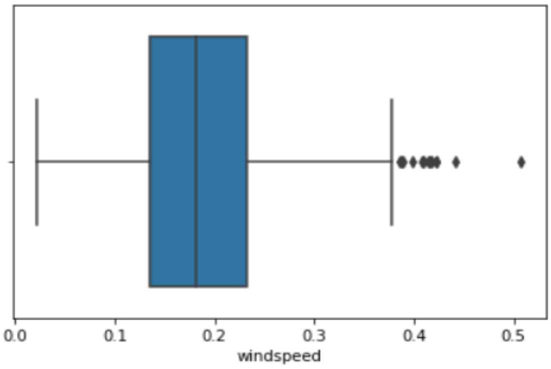
<matplotlib.axes._subplots.AxesSubplot at 0x60cad50>



Temp Boxplot

```
sns.boxplot(df['windspeed'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x6168710>
```



Windspeed Boxplot

2.1.3 Feature Engineering

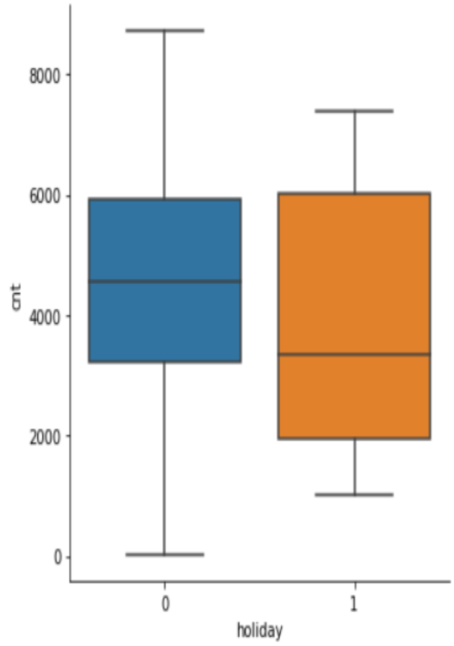
This is probably the part of pre-processing which requires a lot of time and usually impacts your model the most. So in this step we will analyse the various variables in our data set and try to figure out if these variables are actually important for our model or not. I have plotted some graphs to get a feel about the data. I have plotted categorical box plots for categorical variables and I have plotted regression plots and histograms for continuous variables.

Although these plots do not give much useful information but still they can be useful in other scenarios we should always do some visualization before. From the categorical plots it is clear that demand of bike is really low in season 1 or the spring season. In year 2011 the demand of bikes is less as compared to that in 2012. Similarly the demand of bikes is more In months 5, 6 and 7 (May, June and July) as compared to rest of the months.

From the regression plots we can see temp and atemp are directly proportional to the cnt variable and have a high value of slope that is they have a high impact on target variable. Windspeed and hum have a negative relationship with the target variable but the relationship is quite weak. We can also see that our continuous variables are mostly normally distributed with only slight level of skewness. So linear regression will work fine for our data set.

```
sns.catplot(x="holiday", y="cnt", kind="box", data=df)
```

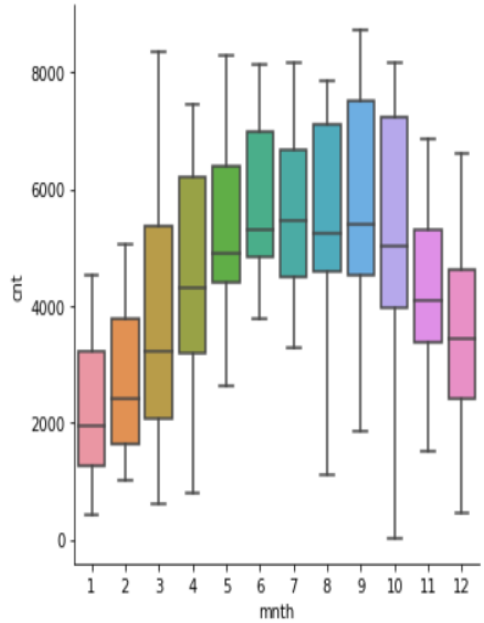
```
<seaborn.axisgrid.FacetGrid at 0xffbcd30>
```



Holiday Categorical Plot

```
sns.catplot(x="mnth", y="cnt", kind="box", data=df)
```

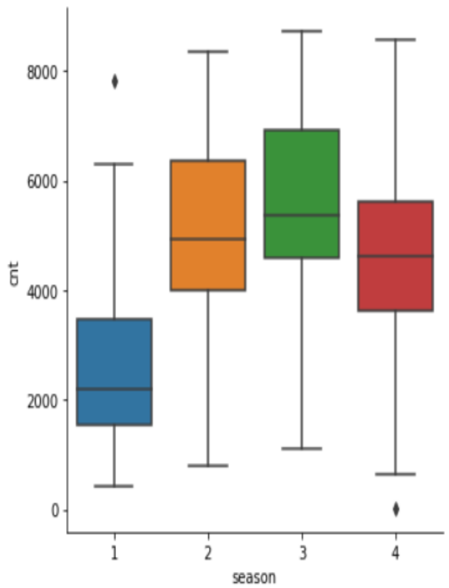
```
<seaborn.axisgrid.FacetGrid at 0x1004c4f0>
```



Mnth Categorical Plot

```
sns.catplot(x="season", y="cnt", kind="box", data=df)
```

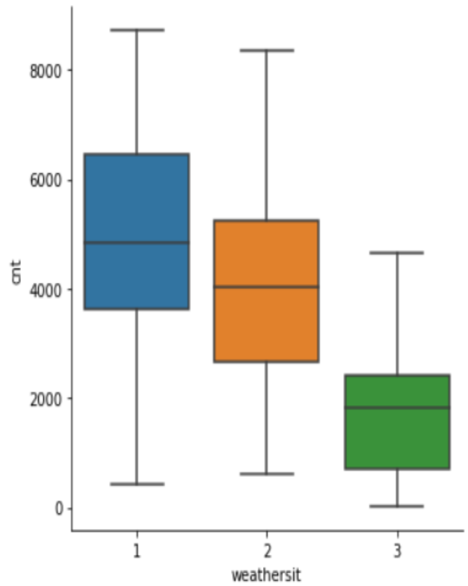
```
<seaborn.axisgrid.FacetGrid at 0x100456b0>
```



Season Categorical Plot

```
sns.catplot(x="weathersit", y="cnt", kind="box", data=df)
```

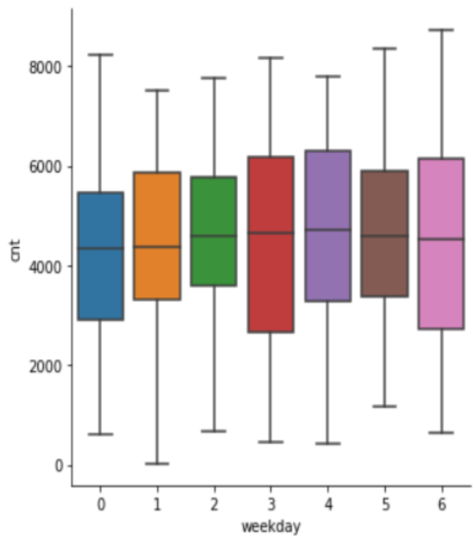
```
<seaborn.axisgrid.FacetGrid at 0x101754f0>
```



Weathersit Categorical Plot

```
sns.catplot(x="weekday", y="cnt", kind="box", data=df)
```

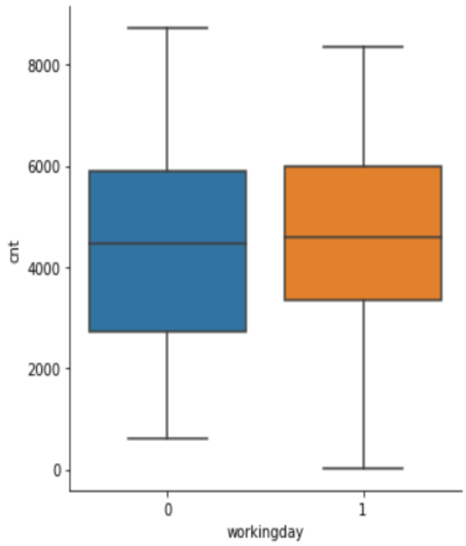
<seaborn.axisgrid.FacetGrid at 0x101012b0>



Weekday Categorical Plot

```
sns.catplot(x="workingday", y="cnt", kind="box", data=d
```

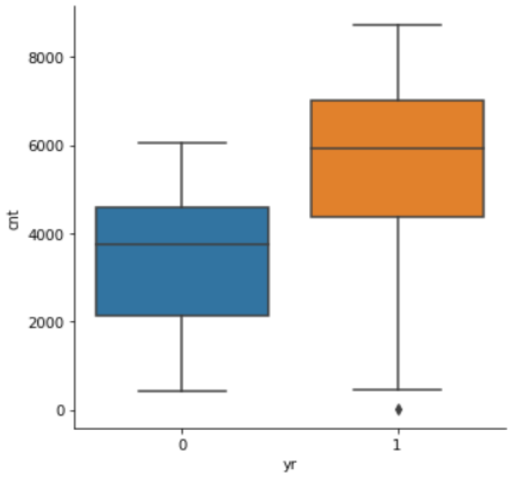
<seaborn.axisgrid.FacetGrid at 0x101c5fb0>



Workingday Categorical Plot

```
sns.catplot(x="yr", y="cnt", kind="box", data=df)
```

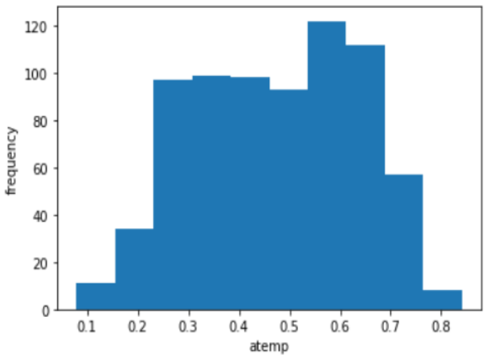
<seaborn.axisgrid.FacetGrid at 0x14874610>



Yr Categorical Plot

```
plt.hist(df['atemp'])
plt.xlabel('atemp')
plt.ylabel('frequency')
```

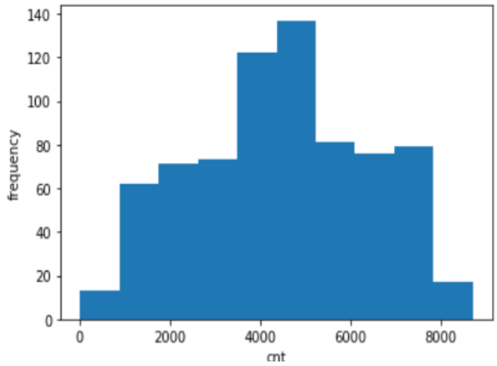
```
Text(0, 0.5, 'frequency')
```



Atemp Histogram

```
plt.hist(df['cnt'])
plt.xlabel('cnt')
plt.ylabel('frequency')
```

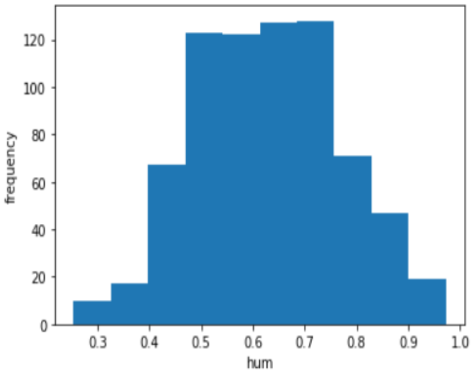
```
Text(0, 0.5, 'frequency')
```



Cnt Histogram

```
plt.hist(df['hum'])
plt.xlabel('hum')
plt.ylabel('frequency')
```

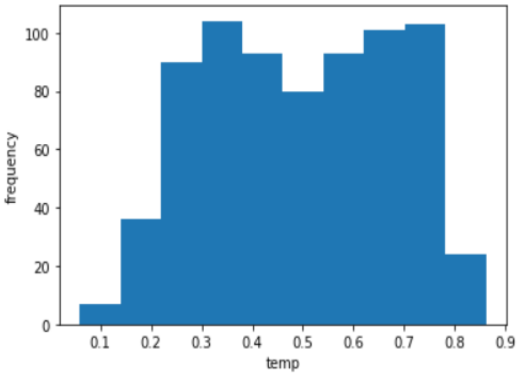
```
Text(0, 0.5, 'frequency')
```



Hum Histogram

```
plt.hist(df['temp'])
plt.xlabel('temp')
plt.ylabel('frequency')
```

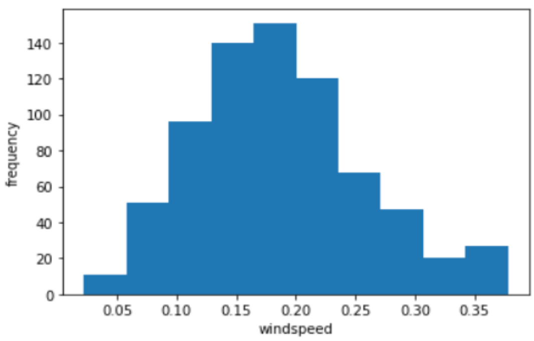
```
Text(0, 0.5, 'frequency')
```



Temp Histogram

```
plt.hist(df['windspeed'])
plt.xlabel('windspeed')
plt.ylabel('frequency')
```

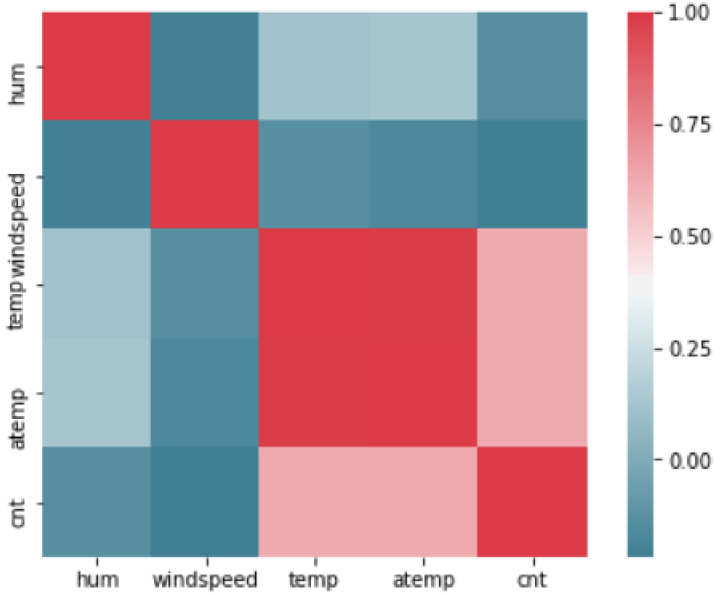
```
Text(0, 0.5, 'frequency')
```



Windspeed Histogram

2.1.3.1 Multicollinearity Test

In this test we will analyse the correlation between the continuous variables in our data set. We will check if a pair of independent variables conveys the same information in our data set. From the correlation matrix it is clear that temp and atemp are highly correlated so we have to drop one of these variables. Also we can see that the target variable cnt has a pretty weak correlation with the windspeed variable which might mean that this variable will not have an impact on our model so we might need to drop it but we will check it further.



Correlation Matrix

2.1.3.2 Anova Test

In the anova test we will check if our categorical variables have any significant impact on our model. From the figure it is clear that the p – value for holiday, weekday and workingday is greater than 0.05 which we usually take as the threshold value. So we drop these three variables. Also the f-statistic is pretty low for these variables. On a higher level the p - value is the probability that the null hypothesis is true (meaning our independent variable does not have a relationship with the target variable). So the lower it is, the better. The f - value helps in determining whether there is significant variation between the various categorical levels of a categorical variable with respect to the target variable.

	Variable	FValue	PValue
0	season	143.967653	2.133997e-30
1	yr	344.890586	2.483540e-63
2	mnth	62.004625	1.243112e-14
3	holiday	3.421441	6.475936e-02
4	weekday	3.331091	6.839081e-02
5	workingday	2.736742	9.849496e-02
6	weathersit	70.729298	2.150976e-16

Anova Test Results

2.1.3.3 Dummy Coding

In our dataset we have categorical variables which have more than 2 levels of categories like mnth, season and weathersit. We will perform dummy coding for these variables. So basically what we are going to do is create new columns so that all our categorical variables will only have 2 levels i.e. 0 or 1. We do not want multiple levels in our categorical variables as they tend to have a bad effect on our regression model as the levels are only categories and they do not have significance other than that, But in linear regression model a value of 12 in the mnth colum will have a higher impact than a value of 1. For e.g. let us say we have a column called fruits. The three fruits are apple, mango and orange. First of all we can convert these strings to numerical values i.e. 0, 1 and 2. Then we create three new columns as apple, mango and orange. We set value as 1 if in the respective column for that fruit and 0 in other two columns and we drop the original fruit column. Also we drop one dummy coded column to avoid the dummy coding trap as 2 columns are enough to describe the categorical variable with 3 levels.

```
#Drop data variable as it won't have impact on our predictions
df = df.drop(['dteday'], axis=1)
#Drop atemp as it is collinear to temp
df = df.drop(['atemp'], axis=1)
#Drop the below three variables as they did not pass the anova test due to p-value greater than 0.05
df = df.drop(['holiday', 'weekday','workingday'], axis=1)

#Compute dummy values for categorical variables
df_mnth = pd.get_dummies(df['mnth'])
df_mnth.rename(columns={1: "mnth_1", 2: "mnth_2", 3: "mnth_3",4: "mnth_4", 5: "mnth_5", 6: "mnth_6",7: "mnth_7",8: "mnth_8",9: '

df_season = pd.get_dummies(df['season'])
df_season.rename(columns={1: "season_1", 2: "season_2", 3: "season_3",4: "season_4"} ,inplace = True)

df_weathersit = pd.get_dummies(df['weathersit'])
df_weathersit.rename(columns={1: "weathersit_1", 2: "weathersit_2", 3: "weathersit_3"} ,inplace = True)

df = pd.concat([df_mnth,df_season,df_weathersit,df] ,axis=1)
df= df.drop(['mnth','mnth_1'],axis=1)
df= df.drop(['season','season_1'],axis=1)
df= df.drop(['weathersit','weathersit_1'],axis=1)
```

Dummy coding for Categorical Variables

2.1.3.4 Feature Selection

In this step we analyse the remaining variables in our data set using the p - value concept. We do a simple linear regression method on our variables and check the significance of all those variables. Most of the assumptions for linear regression are satisfied for our data set. The variables have some sort of linear relationship with the target and the continuous variables are also normally distributed. We have also coded dummy variables for categorical variables. For elimination of the non-significant features we use a technique called as backward elimination. We remove the variable with the highest p - value and then use our linear regression model to check the p - value of the variable that remains. By this way we remove variables one by one and check the p - value of the remaining variables on our linear regression model. We repeat this process until all the variables have p – value less than 0.05 which is our threshold value.

	coef	std err	t	P>|t|	[0.025	0.975]
mnth_2	428.8043	148.523	2.887	0.004	137.209	720.400
mnth_3	775.7085	174.599	4.443	0.000	432.916	1118.501
mnth_4	578.7488	264.457	2.188	0.029	59.539	1097.958
mnth_5	715.6853	286.207	2.501	0.013	153.773	1277.598
mnth_6	590.6324	300.753	1.964	0.050	0.162	1181.103
mnth_7	24.0040	334.654	0.072	0.943	-633.024	681.032
mnth_8	423.1793	322.108	1.314	0.189	-209.218	1055.577
mnth_9	953.3917	283.064	3.368	0.001	397.651	1509.133
mnth_10	541.0414	258.487	2.093	0.037	33.552	1048.531
mnth_11	34.6760	245.414	0.141	0.888	-447.147	516.499
mnth_12	114.6119	192.998	0.594	0.553	-264.302	493.526
season_2	883.6985	192.371	4.594	0.000	506.016	1261.381
season_3	834.9055	228.466	3.654	0.000	386.358	1283.453
season_4	1656.6764	193.286	8.571	0.000	1277.197	2036.156
weathersit_2	-613.1610	79.245	-7.738	0.000	-768.744	-457.578
weathersit_3	-2375.7742	203.285	-11.687	0.000	-2774.885	-1976.664
yr	2098.4407	61.397	34.178	0.000	1977.900	2218.981
temp	5180.1852	431.040	12.018	0.000	4333.922	6026.448
hum	8.8496	234.827	0.038	0.970	-452.188	469.887
windspeed	-758.7923	389.521	-1.948	0.052	-1523.542	5.957

Step 1

	coef	std err	t	P>|t|	[0.025	0.975]
mnth_2	429.8251	145.930	2.945	0.003	143.321	716.329
mnth_3	775.8084	174.457	4.447	0.000	433.297	1118.320
mnth_4	577.8684	263.238	2.195	0.028	61.052	1094.685
mnth_5	714.6401	284.660	2.511	0.012	155.766	1273.514
mnth_6	588.2211	293.662	2.003	0.046	11.675	1164.768
mnth_7	21.6715	328.650	0.066	0.947	-623.567	666.910
mnth_8	421.4290	318.519	1.323	0.186	-203.920	1046.778
mnth_9	952.8229	282.463	3.373	0.001	398.263	1507.383
mnth_10	541.1486	258.290	2.095	0.037	34.047	1048.250
mnth_11	35.5844	244.056	0.146	0.884	-443.571	514.740
mnth_12	116.0351	189.134	0.614	0.540	-255.293	487.363
season_2	883.8985	192.163	4.600	0.000	506.626	1261.171
season_3	834.4559	227.994	3.660	0.000	386.835	1282.076
season_4	1656.6481	193.149	8.577	0.000	1277.439	2035.857
weathersit_2	-611.4686	65.246	-9.372	0.000	-739.566	-483.371
weathersit_3	-2372.9879	189.230	-12.540	0.000	-2744.503	-2001.472
yr	2098.3247	61.276	34.244	0.000	1978.021	2218.629
temp	5189.8272	346.647	14.972	0.000	4509.255	5870.399
windspeed	-755.6075	379.976	-1.989	0.047	-1501.615	-9.600

Step 2

	coef	std err	t	P>|t|	[0.025	0.975]
mnth_2	426.2174	135.191	3.153	0.002	160.797	691.638
mnth_3	769.1834	142.523	5.397	0.000	489.368	1048.998
mnth_4	566.9478	204.472	2.773	0.006	165.509	968.387
mnth_5	701.8614	208.366	3.368	0.001	292.777	1110.946
mnth_6	572.7072	175.620	3.261	0.001	227.914	917.501
mnth_8	402.6610	142.893	2.818	0.005	122.120	683.203
mnth_9	936.7762	143.310	6.537	0.000	655.417	1218.136
mnth_10	530.8451	205.522	2.583	0.010	127.344	934.346
mnth_11	26.9269	205.589	0.131	0.896	-376.705	430.559
mnth_12	109.4511	160.520	0.682	0.496	-205.697	424.599
season_2	888.8515	176.751	5.029	0.000	541.838	1235.865
season_3	843.7482	179.104	4.711	0.000	492.115	1195.382
season_4	1660.6646	183.165	9.066	0.000	1301.057	2020.272
weathersit_2	-611.6274	65.156	-9.387	0.000	-739.548	-483.707
weathersit_3	-2373.1332	189.085	-12.551	0.000	-2744.363	-2001.903
yr	2097.9741	61.003	34.392	0.000	1978.208	2217.740
temp	5204.0132	271.624	19.159	0.000	4670.734	5737.292
windspeed	-757.6446	378.454	-2.002	0.046	-1500.661	-14.628

Step 3

	coef	std err	t	P>|t|	[0.025	0.975]
mnth_2	424.2384	134.252	3.160	0.002	160.663	687.814
mnth_3	766.8471	141.305	5.427	0.000	489.424	1044.270
mnth_4	563.9502	203.047	2.777	0.006	165.310	962.591
mnth_5	699.5874	207.499	3.372	0.001	292.207	1106.968
mnth_6	570.2492	174.494	3.268	0.001	227.667	912.831
mnth_8	399.7073	141.005	2.835	0.005	122.873	676.542
mnth_9	929.4140	131.735	7.055	0.000	670.780	1188.048
mnth_10	511.4570	142.473	3.590	0.000	231.740	791.174
mnth_12	96.1447	124.192	0.774	0.439	-147.681	339.970
season_2	889.8128	176.477	5.042	0.000	543.338	1236.288
season_3	845.7049	178.357	4.742	0.000	495.539	1195.871
season_4	1678.6544	121.091	13.863	0.000	1440.917	1916.392
weathersit_2	-611.4755	65.101	-9.393	0.000	-739.287	-483.664
weathersit_3	-2372.2855	188.844	-12.562	0.000	-2743.042	-2001.529
yr	2098.4136	60.868	34.475	0.000	1978.911	2217.916
temp	5202.3461	271.139	19.187	0.000	4670.021	5734.671
windspeed	-746.5879	368.664	-2.025	0.043	-1470.383	-22.793

Step 4

	coef	std err	t	P>|t|	[0.025	0.975]
mnth_2	409.4725	132.853	3.082	0.002	148.645	670.300
mnth_3	751.9955	139.957	5.373	0.000	477.219	1026.772
mnth_4	551.5745	202.360	2.726	0.007	154.284	948.865
mnth_5	686.5994	206.761	3.321	0.001	280.668	1092.531
mnth_6	558.8946	173.828	3.215	0.001	217.621	900.168
mnth_8	392.1871	140.631	2.789	0.005	116.089	668.286
mnth_9	914.0773	130.200	7.021	0.000	658.457	1169.698
mnth_10	474.1450	134.036	3.537	0.000	210.994	737.296
season_2	883.4844	176.238	5.013	0.000	537.479	1229.490
season_3	833.0028	177.550	4.692	0.000	484.420	1181.585
season_4	1698.4271	118.334	14.353	0.000	1466.104	1930.751
weathersit_2	-605.4175	64.611	-9.370	0.000	-732.267	-478.569
weathersit_3	-2366.1098	188.623	-12.544	0.000	-2736.430	-1995.789
yr	2100.6340	60.784	34.559	0.000	1981.298	2219.970
temp	5218.4523	270.264	19.309	0.000	4687.846	5749.058
windspeed	-712.7964	365.968	-1.948	0.052	-1431.298	5.705

Step 5

	coef	std err	t	P>|t|	[0.025	0.975]
mnth_2	350.5939	129.620	2.705	0.007	96.114	605.074
mnth_3	697.0728	137.354	5.075	0.000	427.407	966.738
mnth_4	486.0847	199.936	2.431	0.015	93.554	878.615
mnth_5	680.9114	207.144	3.287	0.001	274.230	1087.593
mnth_6	561.6652	174.160	3.225	0.001	219.739	903.591
mnth_8	394.7567	140.899	2.802	0.005	118.133	671.380
mnth_9	923.0529	130.372	7.080	0.000	667.095	1179.010
mnth_10	502.8443	133.483	3.767	0.000	240.779	764.910
season_2	910.1045	176.049	5.170	0.000	564.470	1255.739
season_3	883.6665	175.977	5.021	0.000	538.174	1229.159
season_4	1681.0749	118.228	14.219	0.000	1448.960	1913.190
weathersit_2	-621.5001	64.206	-9.680	0.000	-747.554	-495.446
weathersit_3	-2433.8671	185.748	-13.103	0.000	-2798.543	-2069.191
yr	2089.8016	60.647	34.459	0.000	1970.735	2208.868
temp	4983.9783	242.443	20.557	0.000	4507.995	5459.962

Step 6

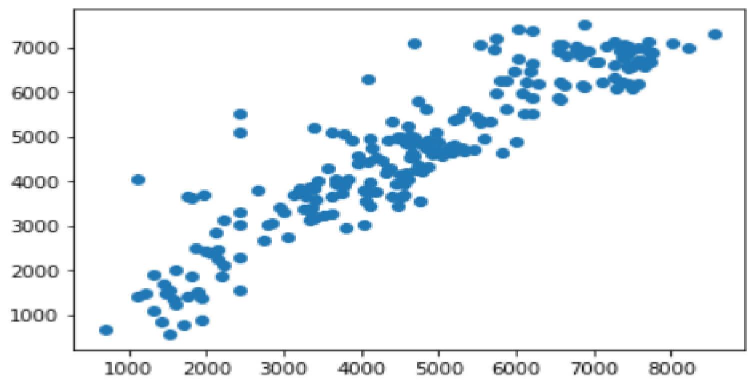
Backward Elimination Technique for Feature Selection

2.2 Modelling and Validation

For our final model I have chosen to test my model using three different algorithms – Multiple linear regression, Decision Tree Regression and Random Forest Regression. Before using the three algorithms I have also divided my data into training and test subset using a division of 70% to 30% respectively. I have used the data which was left after all the pre-processing techniques. I have also used certain validation parameters which might help me to find which model is the best for our scenario.

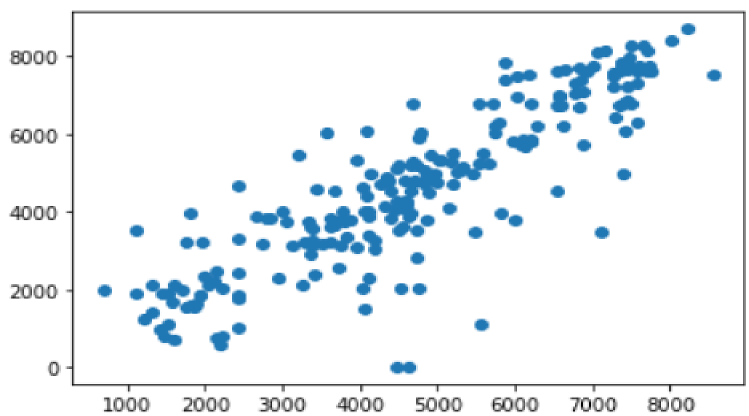
To check which model is working the best I have chosen three parameters, they are MAPE or mean absolute percentage error, r square value and k cross validation test. MAPE is basically the percentage error between the predicted and actual values. So lower it is the better it is. R square tells us that how much variance of the predicted values is explained by our independent variables. In k cross validation we divide the whole data in k folds and train our model on k-1 folds and use the remaining fold for testing. I have used the cross validated score in k cross validation and basically the higher the cross validation score the better the model.

```
R Square: 0.8338277307477933
MAPE: 16.019987918767892
Cross-validated score: 0.6238033179206159
```



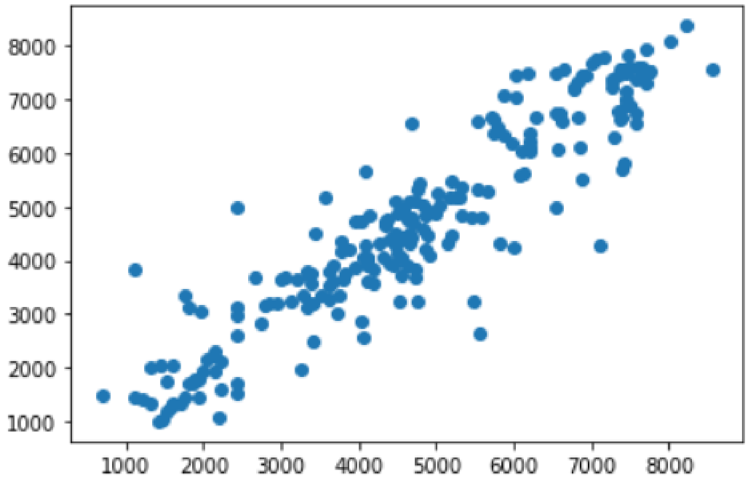
Linear Regression Model

```
R Square: 0.6827050729054907
MAPE: 20.04088089643881
Cross-validated score: 0.028097372628669487
```



Decision Tree Regression

```
R Square: 0.8441756143223421
MAPE: 14.56206284734203
Cross-validated score: 0.34296144145471164
```



Random Forest Regression

Chapter 3

Conclusion

After training our model using the different kind of algorithm we come to find out that multiple linear regression is probably the best algorithm for this problem. For a particular combination of train and test we get R square to be approx.0.83, MAPE as 16.1 and cross validation score as 0.62. These are decent values for our model. If we compare it with decision tree, the values of validation parameters were bad in all the three tests. In random forest though we got slightly less MAPE and slightly better R square than the linear regression model but it gave a significantly lower value of cross validation. Therefore we choose the linear regression model as it performs better on this particular data set.