Bike Renting Count Prediction

Name: Deepanshu Tambi

All the data, graphs, charts, and results generated in the project are coded either using python jupyter or R Studio.

CONTENTS:

Topics Name Page no.

Chapter 1: Introduction

1.1 PROBLEM STATEMENT:

The objective of this Case is to predict the number of bikes rent count based on the environmental and seasonal settings i.e. to predict total number of people renting the bike on daily basis.

1.2 OVERVIEW OF THE DATASET:

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

Fig 1.1 First 5 rows of the dataset

As we can see, there are 7 categorical variables and 7 numeric variables. The dependent variable or the target variable is cnt, i.e. the count of people renting bikes.

While checking, it is seen that there are no duplicate rows or empty values present throughout the dataset. Also, the numerical variables are scaled by normalizing the data, so no need of feature scaling and missing value analysis.

The details of data attributes in the dataset are as follows -

season: Season (1:springer, 2:summer, 3:fall, 4:winter)

yr: Year (0: 2011, 1:2012)

mnth: Month (1 to 12) (Jan to Dec)

holiday: whether day is holiday or not (extracted from Holiday Schedule)

weekday: Day of the week

workingday: If day is neither weekend nor holiday is 1, otherwise is 0.

weathersit: (extracted from Freemeteo)
1: Clear, Few clouds, Partly cloudy, Partly cloudy
2: Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist
3: Light Snow/Heavy snow, Heavy Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds

temp: Normalized temperature in Celsius.
The values are derived via (t-t_min)/(t_max-t_min), t_min=-8, t_max=+39

atemp: Normalized feeling temperature in Celsius.
The values are derived via (t-t_min)/(t_max- t_min), t_min=-16, t_max=+50

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

Chapter 2: Visualizations

2.1 VISUALIZING AND INTERPRETING THE RAW DATA:

To understand what the raw data is, what it looks like and to understand the parameters, we need to look at various graphical plots.

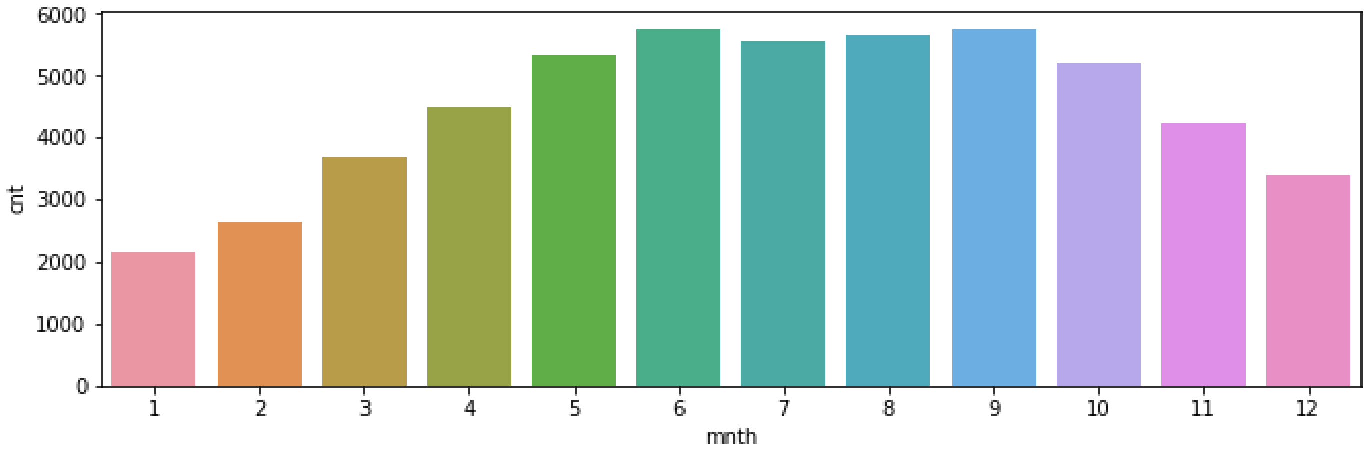

Fig: 2.1 Distribution of count by months.

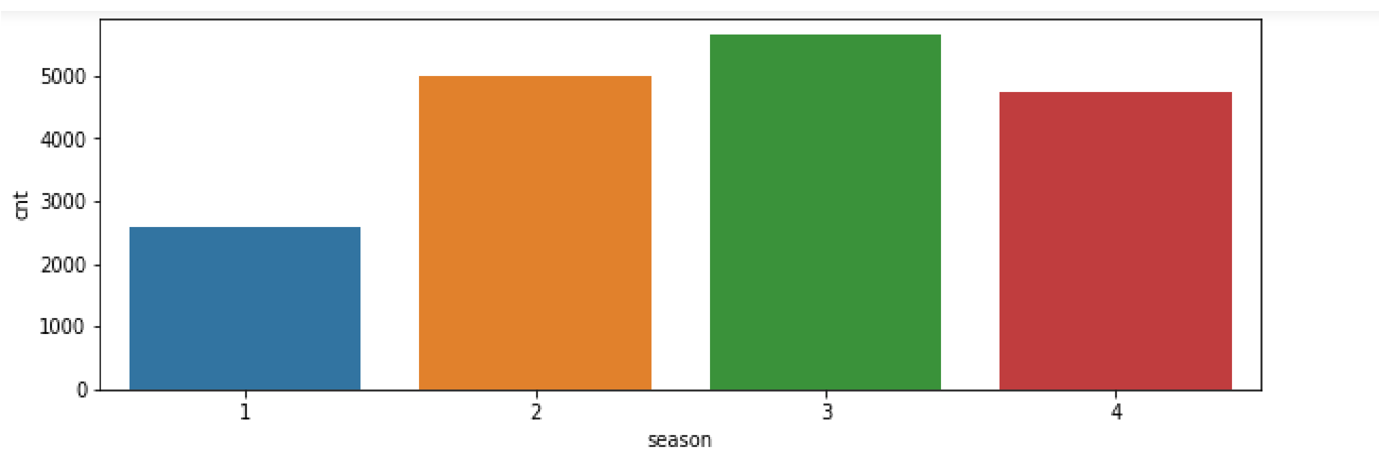

Fig 2.2: Distribution of count by season.

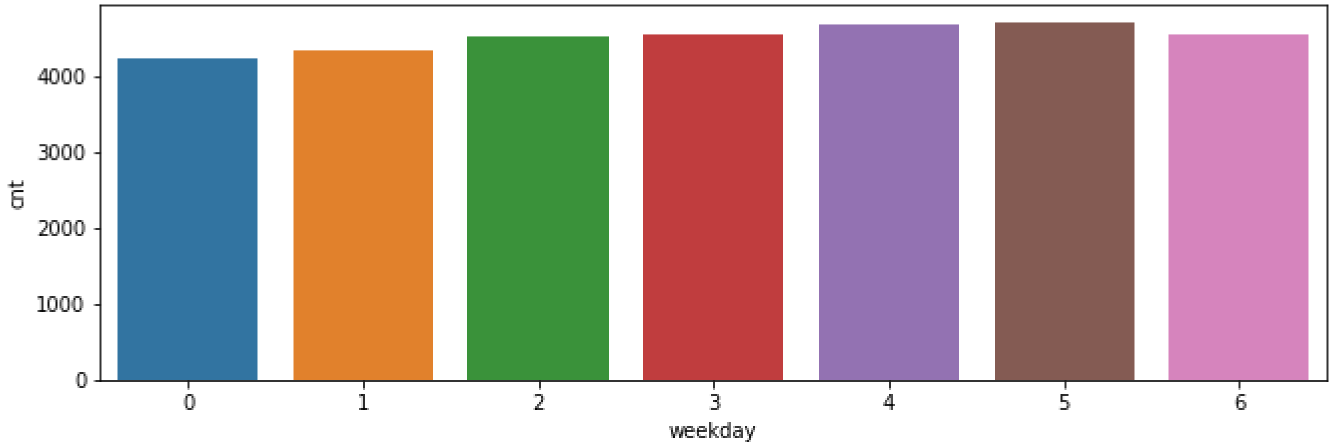

Fig 2.3: Distribution of count by weekday.

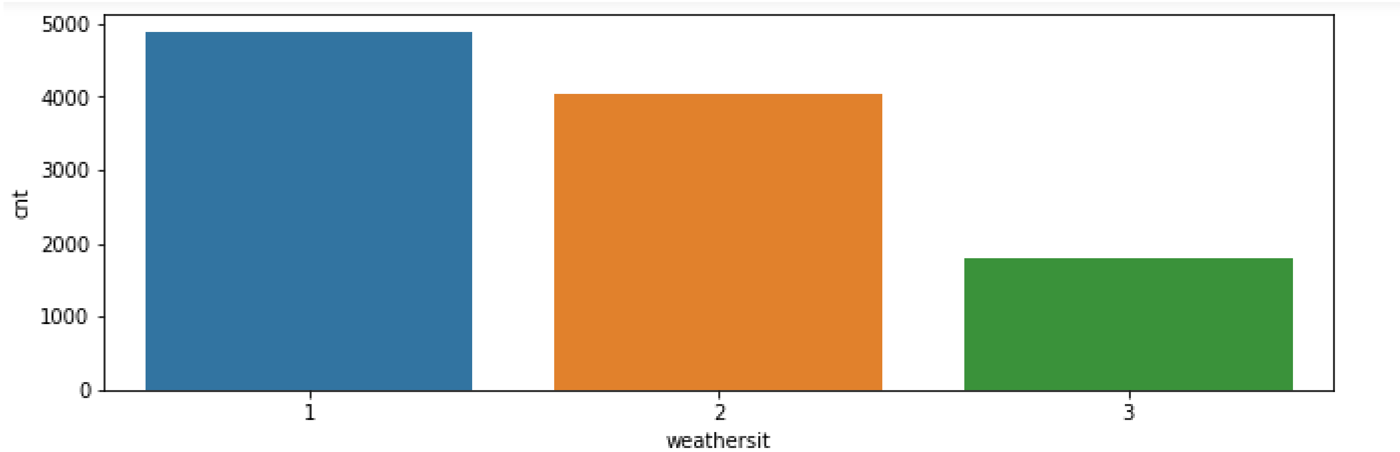

Fig 2.4: Distribution of count by weathersit.

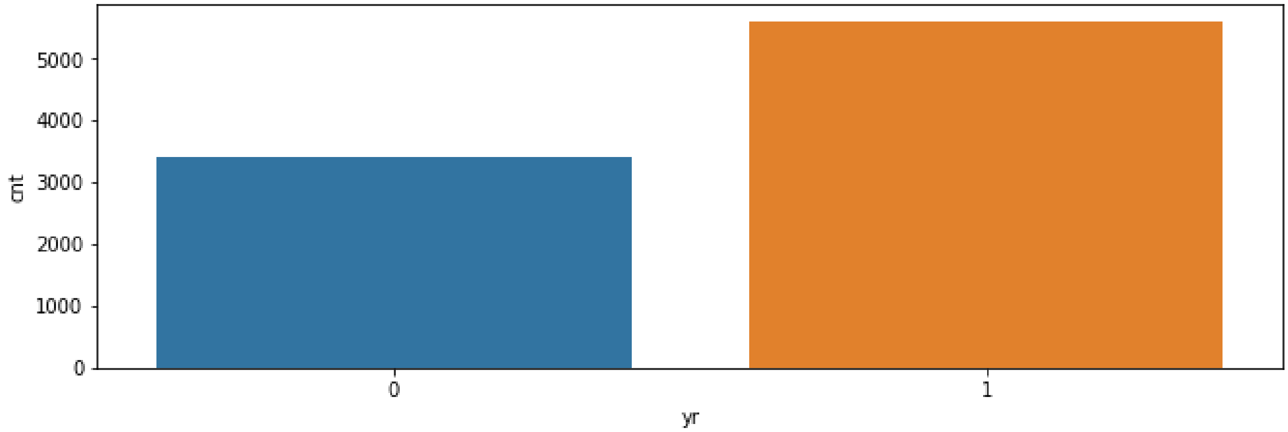

Fig 2.5: Distribution of count by yr.

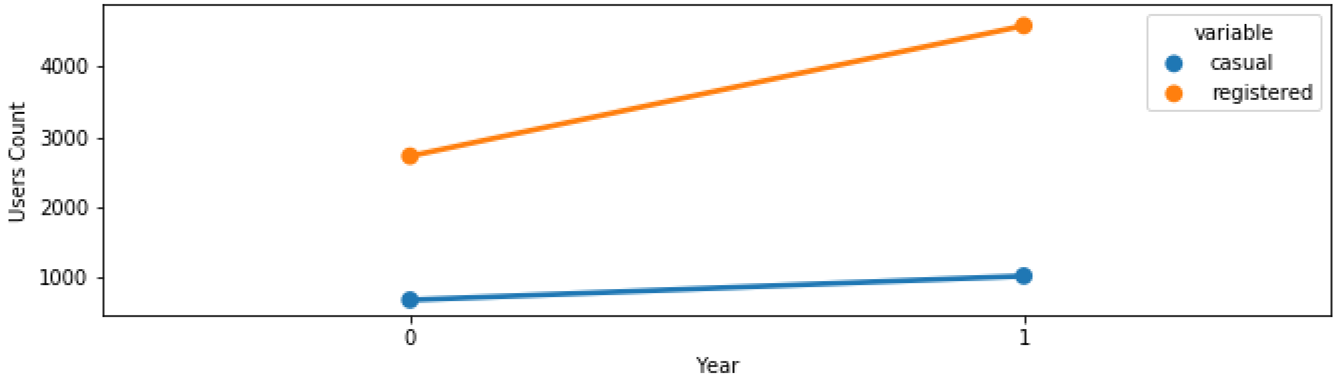

Fig 2.6: Distribution of count by yr divided into casual and registered users.

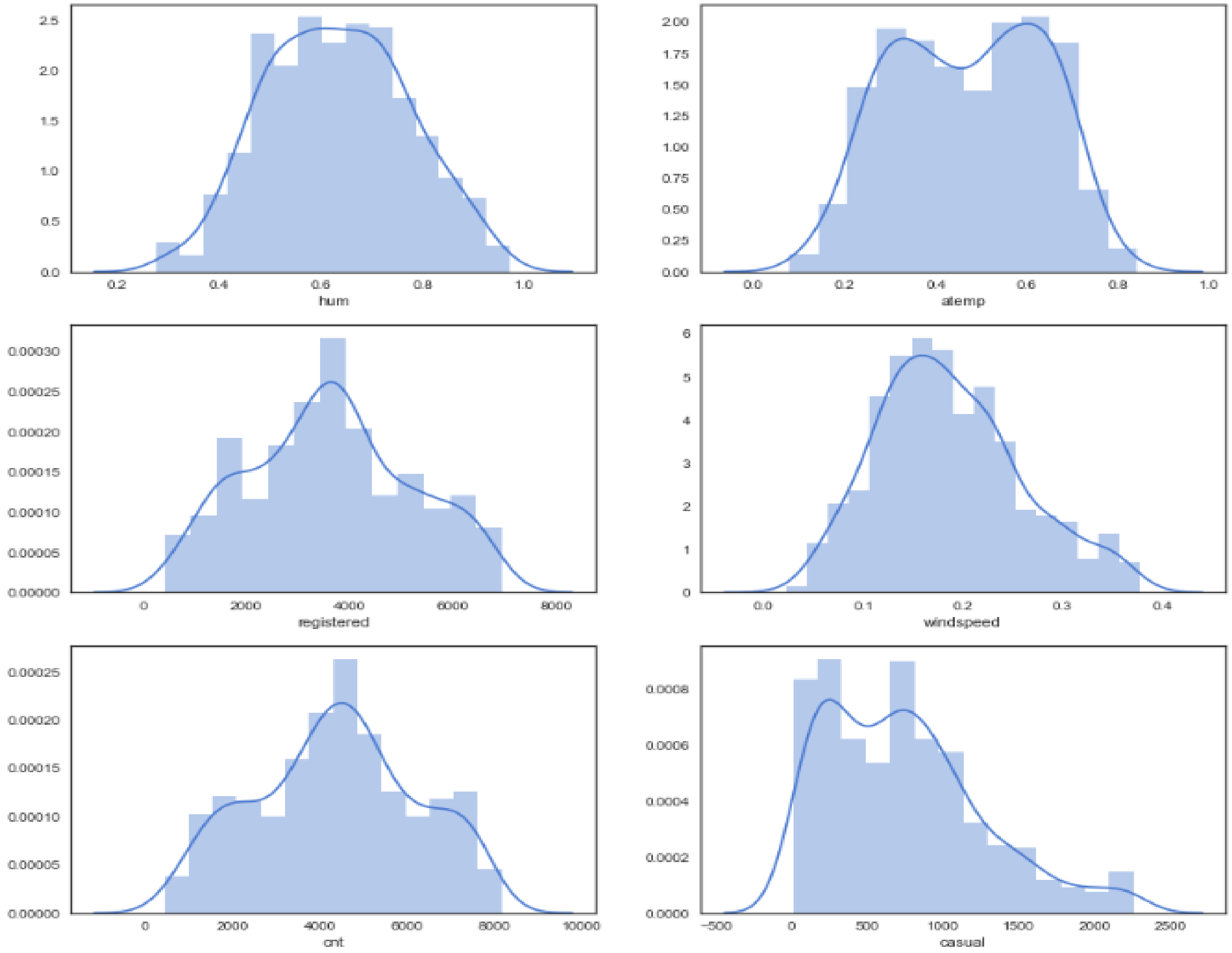

Fig 2.7: Histogram of different numeric data for viewing their distribution.

2.2 SUMMARY OF VISUALIZING THE RAW DATA:

1) Fig 2.2 shows that the rental of bikes is lowest in springer(season 1) and highest in fall(season 3). This can be concluded through Fig 2.1 where months:(1,2,3,12), (3,4,5,6), (6,7,8,9), (9,10,11,12) are seasons 1, 2, 3, 4 respectively.

2) Fig 2.3 shows that weekdays generally don’t affect the rental count much (on overall mean basis) and the mean count is almost the same throughout days.

3) Fig 2.4 suggests that weather situation 1 is totally suitable for bikers, while weather situation 3 disrupts the rental counts.

4) Fig 2.5 shows that there is good amount of increment in the rental bike count from year 2010 to the year 2011.

5) Although the amount of total rental bike count has increased over the year as suggested by Fig. 2.5, the amount of casual users has not increased much whereas the number of registered user have increased sharply over the year as seen in Fig. 2.6.

6) Fig 2.7, i.e. histograms of various numerical variables shows a good normal distribution except temp, and casual which is slightly left skewed

Chapter 3: Preprocessing

3.1 OUTLIER ANALYSIS OF DATA:

To check the outliers present in the data visually

Fig 3.1 Boxplots of various variables with months(since months is the most split categorical variable and captures the entire set nicely)

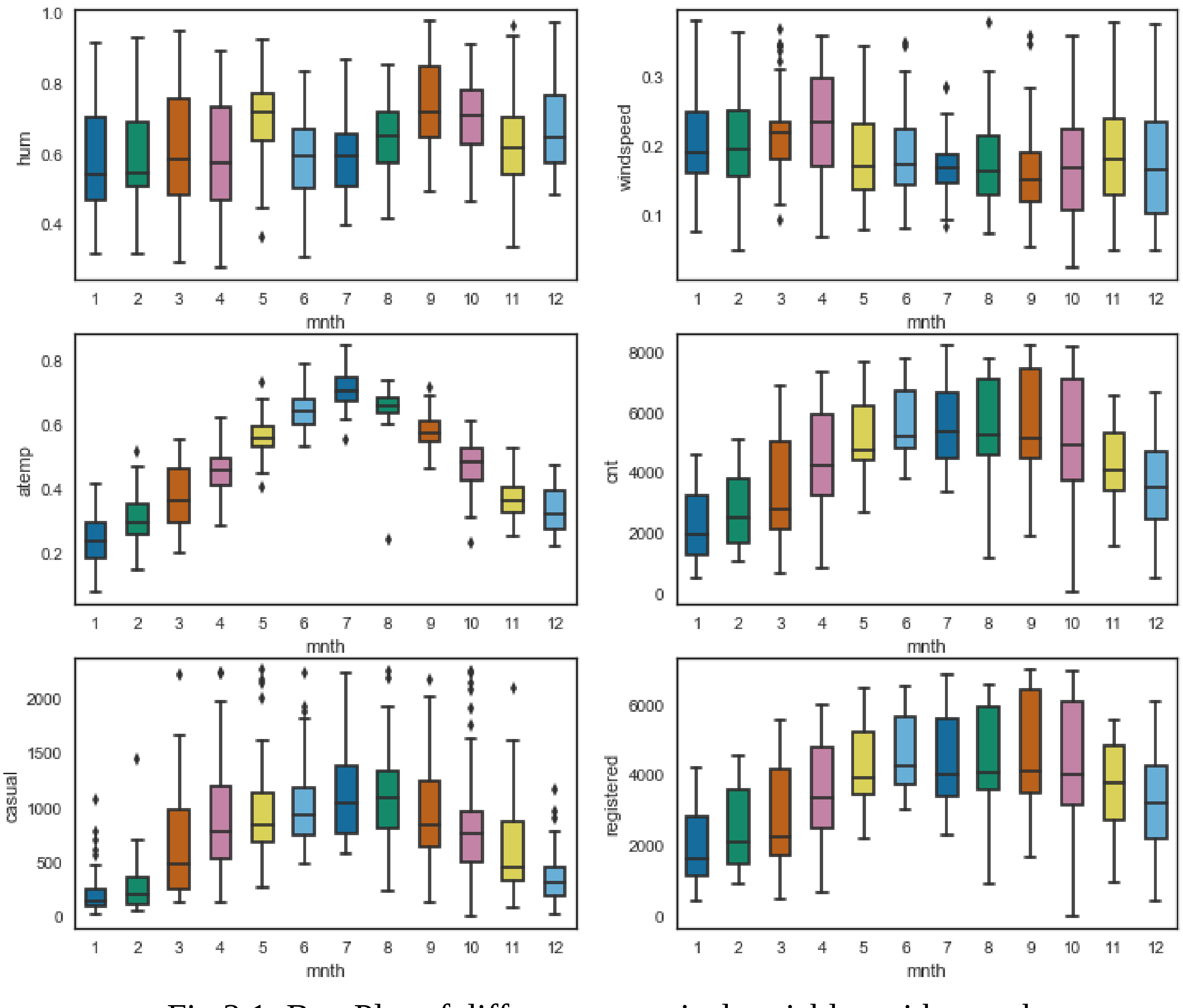

Fig 3.1: Box Plot of different numerical variables with month .

Fig. 3.1 shows the outliers, and it can be seen that there are good amount of outliers in casual. Now, to detect the outliers we will follow the quartile ranges. After detecting the outliers, it was time to decide whether to remove the outliers or to impute them via KNN Imputation or random forest regressor, or other methods. By carefully looking at the outliers, it was seen that 3 variables (casual, hum, windspeed) had 40+, 10+, 3 outliers respectively. Since the outliers were not more than 10% of the total data, and the variable with the most outlier (casual) had only upper limit outlier which were well distributed among different categories except weekday, it was okay to drop the rows. But, getting it imputed by mean value cnt of weekday would have led to even a better model. Let's see how-

Consider the table 3.2b, it shows that all the weekdays are equally distributed except 0 and 6, which were 103 and 102 respectively before removing the outliers. So, it is evident that out of 40+ outlier in casual that got removed, 35+ came from weekday 0 and 6, this is also evident by looking at the table 3.2a and 3.2c that only weekdays 0 and 6 are affected marginally whereas other weekdays didn't change much after outlier removal. Now, while checking and grouping by each category, it was revealed that all the other categories had random distribution of outliers while only weekday 0 and 6 had specific 'casual' outliers. We'll consider this situation at the later stage. Till then, we are removing all outliers.

yr	weekday	
0	0	3274.708333
	1	3452.040816
	2	3500.843137
	3	3253.250000
	4	3394.680000
	5	3500.115385
	6	3376.176471
1	0	4521.948718
	1	5170.215686
	2	5553.288462
	3	5862.000000
	4	5989.431373
	5	5857.060000
	6	4491.925926

Name: cnt, dtype: float64

3.2 a

2	103
5	102
3	102
4	101
1	100
0	87
6	78

3.2 b

yr	weekday	
0	0	3405.269231
	1	3465.788462
	2	3468.038462
	3	3253.250000
	4	3356.769231
	5	3500.115385
	6	3391.377358
1	0	5036.849057
	1	5194.000000
	2	5553.288462
	3	5843.826923
	4	5977.750000
	5	5880.461538
	6	5732.000000

Name: cnt, dtype: float64

3.2 c

Fig 3.2: (a) Mean of cnt grouped by yr and weekday before removing outliers, (b) Total number of observations in weekdays after removing outliers, (c) Mean of cnt grouped by yr and weekday before removing outliers

Deletion of outliers won't pose a significant problem, as we can see from the scatter plot 3.3a and 3.3c, when we deleted outlier from casual, it got deleted from the upper end as the maximum outlier mark for casual was 2300. Now, notice that if the red dots of 'casual' gets removed above 2300(mostly for the instances 400 and above) then out target variable 'cnt' has a good amount of cluster in that range, thus not missing on much important, out-of-the-box or crucial data from our target variable.

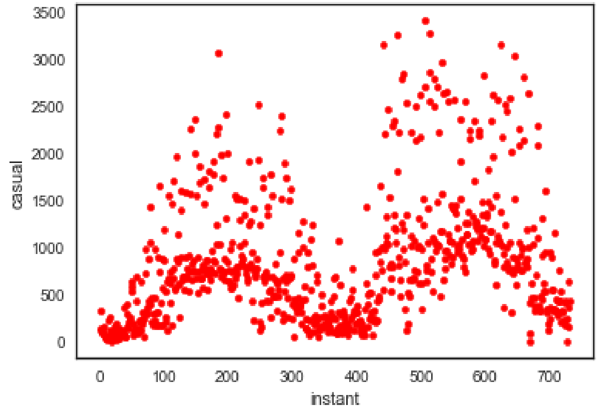

Fig: 3.3a

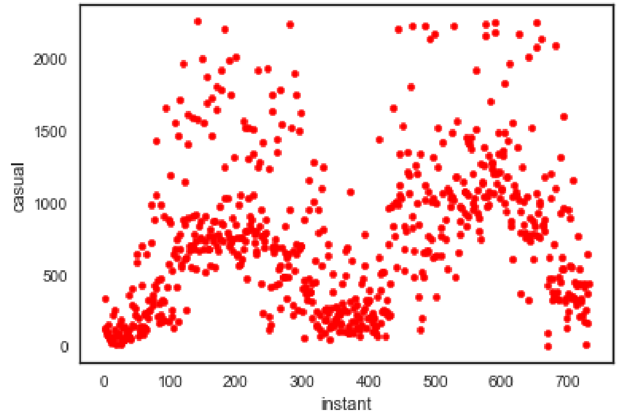

Fig: 3.3b

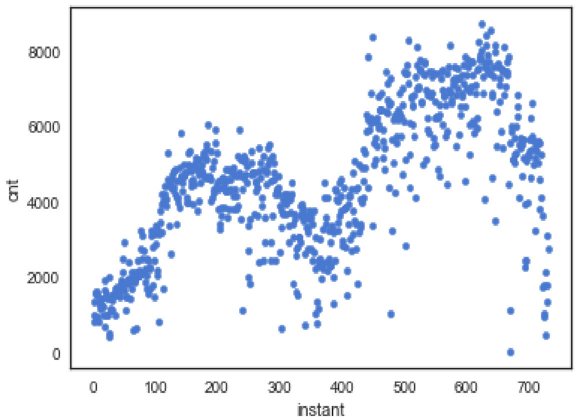

Fig: 3.3c

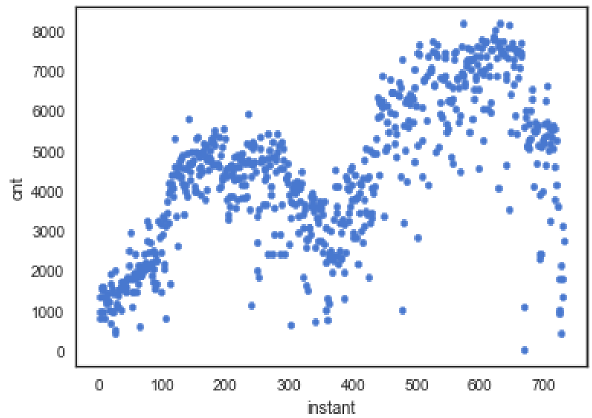

Fig: 3.3d

Fig: 3.3 (a) Scatter plot of casual 365+366 days or instances before outlier removal, (b) Scatter plot of casual 365+366 days or instances after outlier removal, (c) Scatter plot of 'cnt' 365+366 days or instances before outlier removal, (d) Scatter plot of cnt 365+366 days or instances before outlier removal

Fig: 3.4 shows the histogram of variables after outlier removal. Mostly everyone follows normal distribution so no need of transformation is required, although casual is left skewed, we are anyways going to drop it for our modeling as will discuss this in later stage.

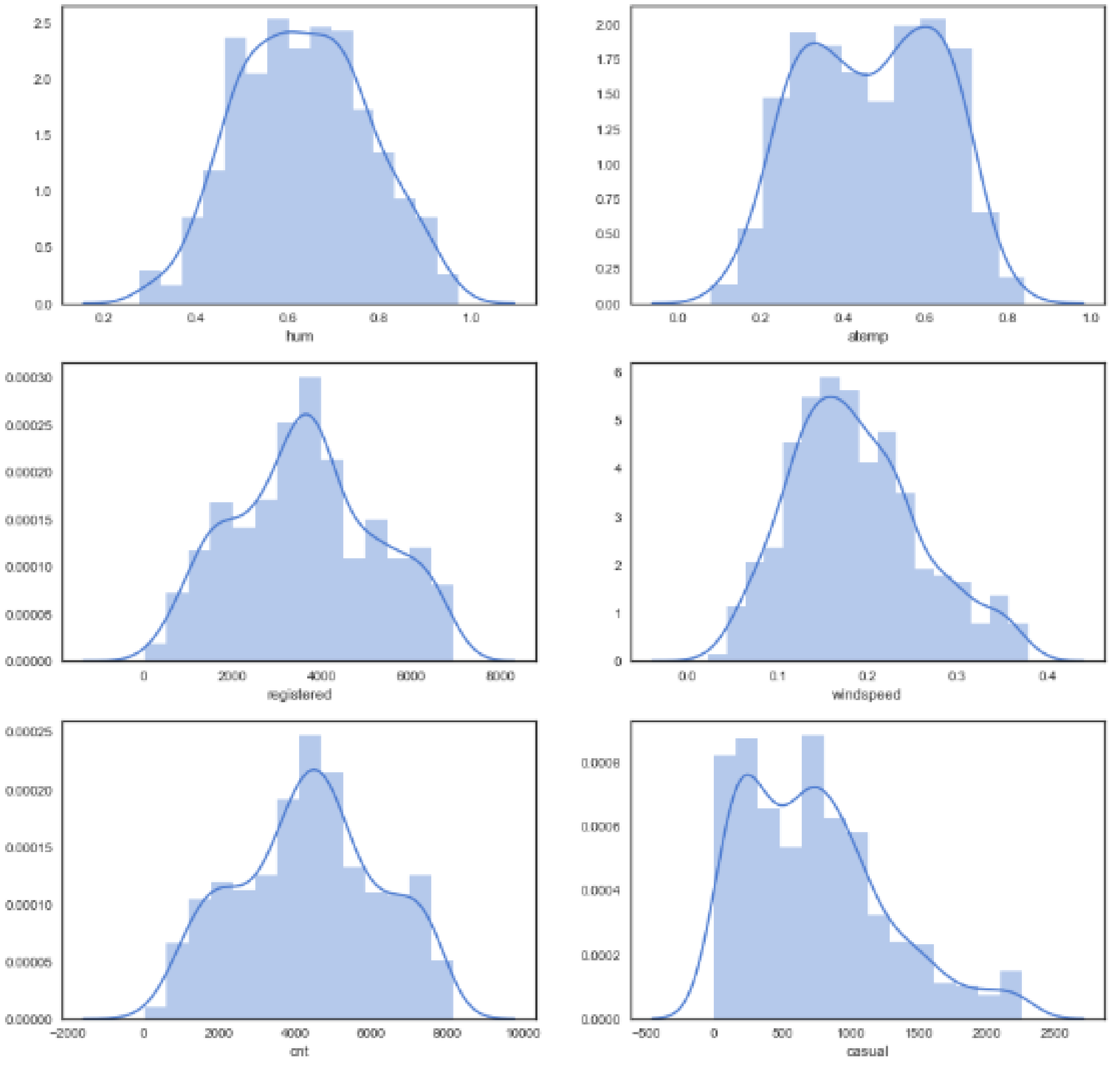

Fig: 3.4 Histograms after removal of outliers

Chapter4: Feature Selection

For selecting the feature and understanding the inter-relationships between the variables, we performed three tests.

4.1 Correlation analysis: Fig: 4.1 shows the correlation analysis table where we can deduce that temp and atemp are highly correlated, also registered and our target variable cnt are highly correlated. And, increase in humidity and windspeed have negative impact on cnt.

	temp	atemp	hum	windspeed	casual	registered	cnt
temp	1	0.991483	0.122486	-0.139599	0.595525	0.54512	0.629031
atemp	0.991483	1	0.135356	-0.167087	0.593962	0.54785	0.630906
hum	0.122486	0.135356	1	-0.206719	-0.0963499	-0.113078	-0.122854
windspeed	-0.139599	-0.167087	-0.206719	1	-0.184026	-0.212375	-0.231596
casual	0.595525	0.593962	-0.0963499	-0.184026	1	0.427474	0.64289
registered	0.54512	0.54785	-0.113078	-0.212375	0.427474	1	0.967266
cnt	0.629031	0.630906	-0.122854	-0.231596	0.64289	0.967266	1

Fig: 4.1 Correlation analysis table heatmap

4.2 Chi-Square test: For performing this test we binned our target variable so that it becomes categorical and binned in such a way that it still remains in normality. This test shows that variable holiday fails to follow null hypothesis with 95% of confidence interval as its p-value is less than 0.05 with our target variable cnt_binned as shown in fig: 4.2. Other variables are well significant wit our target variable.

	season	yr	mnth	holiday	weekday	workingday	weathersit	cnt_binned
season	0.000000e+00	8.610613e-01	0.000000e+00	2.912027e-01	9.998448e-01	1.470297e-01	6.188767e-03	5.925856e-54
yr	8.610613e-01	0.000000e+00	9.996196e-01	9.775989e-01	3.347478e-01	4.579463e-02	2.789441e-01	7.929200e-53
mnth	0.000000e+00	9.996196e-01	0.000000e+00	3.400998e-01	1.000000e+00	6.921360e-01	2.920914e-03	2.147749e-56
holiday	2.912027e-01	9.775989e-01	3.400998e-01	0.000000e+00	5.084896e-09	1.299889e-11	5.867430e-01	1.901768e-01
weekday	9.998448e-01	3.347478e-01	1.000000e+00	5.084896e-09	0.000000e+00	1.973051e-124	2.941414e-01	1.716889e-04
workingday	1.470297e-01	4.579463e-02	6.921360e-01	1.299889e-11	1.973051e-124	0.000000e+00	6.134247e-01	5.480604e-06
weathersit	6.188767e-03	2.789441e-01	2.920914e-03	5.867430e-01	2.941414e-01	6.134247e-01	0.000000e+00	3.533763e-17
cnt_binned	5.925856e-54	7.929200e-53	2.147749e-56	1.901768e-01	1.716889e-04	5.480604e-06	3.533763e-17	0.000000e+00

Fig: 4.2 Chi-Square test table with each variable (p-values)

4.3 ANOVA: ANOVA is performed with numeric and categorical data combined. The PR(>F) value suggests the probability of how much the group means are influencing our target variable. Over here, the variable holiday fails to prove the null hypothesis as this is less than 0.05..

	sum_sq	df	F	PR(>F)
C(yr)	6.822035e+08	1.0	1070.317679	2.372238e-139
C(holiday)	1.086908e+06	1.0	1.705263	1.920645e-01
C(workingday)	2.932195e+07	1.0	46.003572	2.663354e-11
C(mnth)	1.461236e+08	11.0	20.841382	2.328851e-36
C(weekday)	8.026674e+06	6.0	2.098858	5.149788e-02
C(weathersit)	1.789559e+08	2.0	140.383120	2.231628e-51
C(season)	5.509763e+07	3.0	28.814454	1.691807e-17
Residual	4.130249e+08	648.0	NaN	NaN

Fig: 4.3 ANOVA test results

4.4 SUMMARY OF FEATURE SELECTION

Conclusions from the various tests can be drawn are that either temp or atemp has to be dropped because of multi collinearity, registered has to be dropped because of its high correlation with our target variable cnt, ‘casual’ also has to be dropped as it is a leakage variable and for future predictions we have to compute it so can’t take it into account. From both correlation and ANOVA test it can be seen that holiday variable won’t be providing us with much information, apart from that see fig. 4.4, it shows the number of value counts of holiday variable. This can account into class imbalance too, so dropping this variable for our further analysis will be fine.

But, as you can see in fig: 4.5 the mean of count is quite different in different holiday value, just because of this and the shortage of number of observations, we will keep this variable, although dropping it will never be a problem for small test cases analysis.

0	655
1	18

Fig: 4.4 Count of holiday

yr	holiday	
0	0	3412.773256
	1	2664.333333
1	0	5487.257235
	1	3885.222222

Fig: 4.5 mean of count grouped by holiday

CHAPTER5: MODEL DEVELOPMENTAND CONCLUSIONS

When looking at the data and performing certain operations, it was noticed that grouping is the best method which can predict our target variable thus we can almost be assured that random forest will outperform many of the models. In the modeling phase we will be dealing with MAPEs, although these aren't always the correct measure and can lead to big erroneous errors which we'll look as we move forward. But, MAPE provides a visual enticement to the client and is very easy to read and work upon.

5.1 Decision Tree:

While performing the decision tree regressor on the test data, the MAPE turned out to be 150%. The value is just not acceptable, it was believed that there must have been some calculation error or syntax error, but after many checks it was observed that everything was fine, but the MAPE was still not correct. So, to check where the error exactly happened, we had to look at the core formula of MAPE. The equation of MAPE is abs((ytrue –ypred)/(ytrue))*100 and then mean of this. Checking the values of (ytrue –ypred)/(ytrue) and arranging in ascending order we get Fig. 5.1. Clearly we can see the -197.13, while all other values seem normal. This must've been caused only in the case when the true value is way way less than predicted value. So let's check our original data with ascending order of count which might reveal why one value got inflated.
Fig: 5.2 depicts that there's one value 22 which seems to be way less than the other values and explained the inflated MAPE. For e.g. if the predicted value is 400 the MAPE comes out to be (22-400/22)*100 ~ 1700%. So, the question was why didn't this value of casual not get detected in univariate outlier analysis, maybe it was a multi variate one. To check this, maximum and minimum of the outlier was checked for casual, and it turned out to be -800s and +2200s, that means, had the variable casual got negative values it would have not been detected in the outlier analysis, what if the value was zero, it could've caused 'inf' MAPE. Therefore a manual check is always advisable. After seeing this, it didn't mean that the model was erroneous, it just meant that when splitting the dataset into train and test, this value went into the test model(i.e in the denominator of MAPE) which led to such inflation.

NOTE: When performing the same model in R language, this problem was not faced because, while splitting the data for train and test in R, this row of cnt=22, went into the train model or the predicted value (in the numerator of MAPE) which caused no such error.

So, now the row containing the value of cnt=22 was dropped and further remodeling was done.

RE-MODELLING:

The attribute usage in making the decision tree of depth(~100) is shown in Fig 5.3. Here, we see that only 8.27% of variable holiday is contributing in deciding the data. Every other variable has some serious gravity in deciding the tree. A simple visualization of the tree with depth of 3 is shown in fig 5.4 which gets split first on the basis of temp which shows the importance of the variable as the top deciding factor.

The MAPE value comes out to be ~20, or the model is ~80% accurate. This is not that a good accuracy, but will work in worst case scenarios. Although we still are reliable on random forest for this model and heavily expect it to perform well than this.

	subt
24	-197.136364
36	-2.712734
96	-1.643478
86	-1.365867
58	-1.198845
68	-0.983770
12	-0.793528
33	-0.753480

Fig: 5.1 ‘subt’=(ytrue –ypred)/(ytrue)

```
Attribute usage:

100.00% season
100.00% yr
100.00% mnth
100.00% weathersit
100.00% temp
100.00% hum
 96.32% windspeed
 95.22% workingday
 89.52% weekday
  8.27% holiday
```

Fig:5.3 Attribute usage in decison tree making

	season	yr	mnth	holiday	weekday	workingday	weathersit	temp	atemp	hum	windspeed	casual	registered	cnt	cnt_binned
612	4	1	10	0	1	1	3	0.440000	0.439400	0.880000	0.358200	2.0	20	22	1
27	1	0	1	0	4	1	1	0.195000	0.219700	0.687500	0.113837	15.0	416	431	1
668	1	1	12	0	3	1	3	0.243333	0.220333	0.823333	0.316546	9.0	432	441	1
26	1	0	1	0	3	1	3	0.217500	0.203600	0.862500	0.293850	34.0	472	506	1
63	1	0	3	0	0	0	2	0.376522	0.366252	0.948261	0.343287	114.0	491	605	1
290	4	0	10	0	6	0	3	0.254167	0.227913	0.882500	0.351371	57.0	570	627	1

Fig: 5.2 Head of original dataset arranged in ascending order by cnt

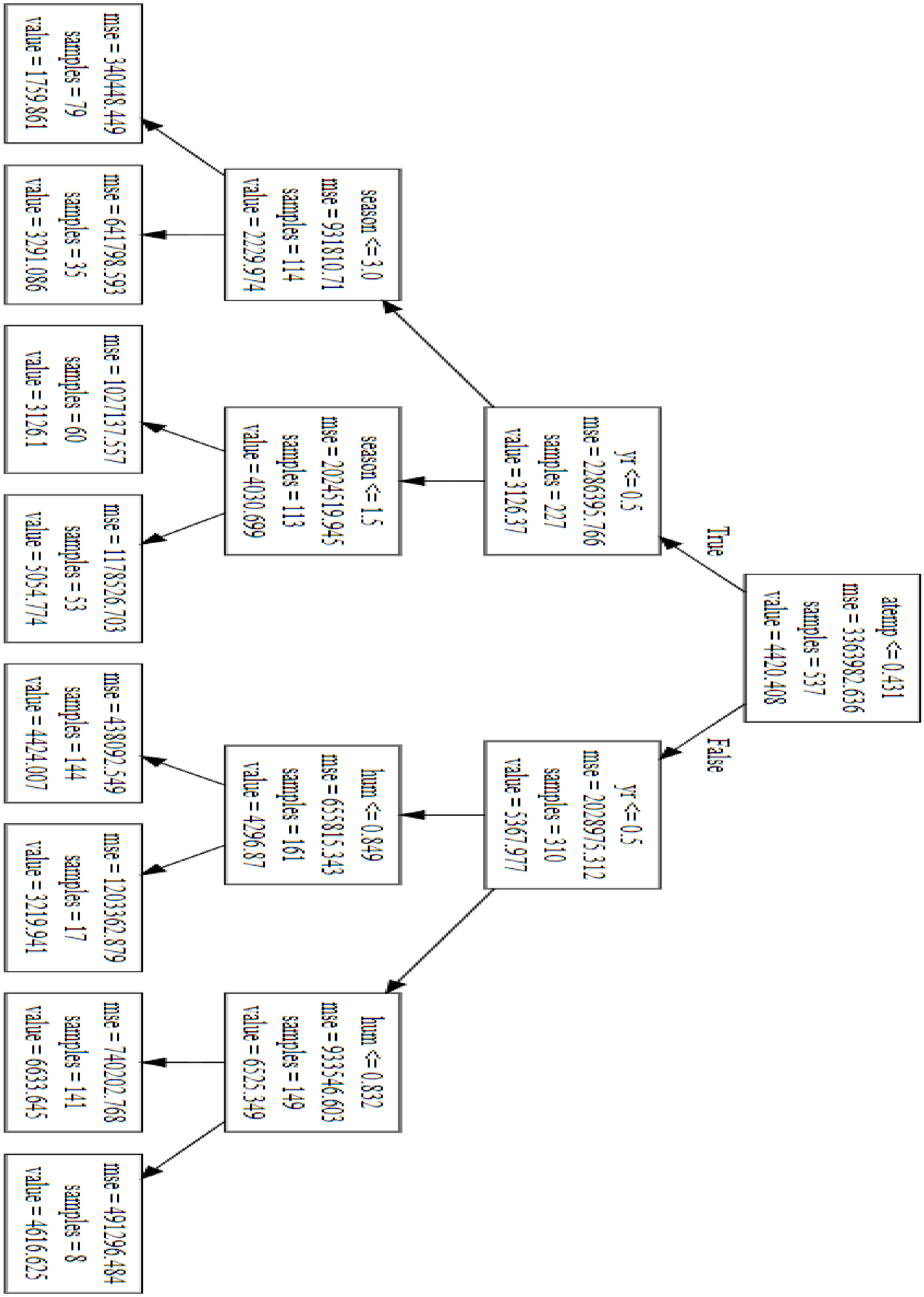

Fig: 5.4 Decision tree with depth = 3

5.2 Linear Regression:

By fig. 5.5 it seems the R-squared value to be good and variation is explained good by the model. Although F-Stats is a bit high causing us to think of over-fitting, but it’s still low to get us into trouble. By looking at the coefficients, it is visible that temp is carrying the highest weight, which was also observed in the decision tree.

Fig. 5.5 shows the test results of linear regression model in Python with no dummy creation of variables. Fig. 5.6 shows the test results of linear regression model in RStudio with dummy creation of variables.

Although the tests were performed by various variations in modeling like, creating dummies of all categorical variables, creating dummies of only ordinal variable, and by creating no dummies at all. The MAPE still came out to be ~20, or ~80% of accuracy, but here test parameters seems to be pretty good.

OLS Regression Results

Dep. Variable:	cnt	R-squared:	0.970
Model:	OLS	Adj. R-squared:	0.970
Method:	Least Squares	F-statistic:	1731.
Date:	Wed, 19 Sep 2018	Prob (F-statistic):	0.00
Time:	17:23:09	Log-Likelihood:	-4378.4
No. Observations:	538	AIC:	8777.
Df Residuals:	528	BIC:	8820.
Df Model:	10		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
season	497.3246	62.460	7.962	0.000	374.624	620.026
yr	2013.1121	71.450	28.175	0.000	1872.752	2153.473
mnth	-21.5636	19.527	-1.104	0.270	-59.923	16.796
holiday	-646.0184	275.099	-2.348	0.019	-1186.441	-105.595
weekday	60.4281	18.382	3.287	0.001	24.317	96.539
workingday	485.4660	82.831	5.861	0.000	322.748	648.184
weathersit	-606.1440	90.770	-6.678	0.000	-784.459	-427.829
atemp	5837.1507	239.186	24.404	0.000	5367.278	6307.023
hum	28.5202	297.402	0.096	0.924	-555.716	612.757
windspeed	-662.8609	429.423	-1.544	0.123	-1506.447	180.726

Omnibus:	67.979	Durbin-Watson:	2.087
Prob(Omnibus):	0.000	Jarque-Bera (JB):	107.285
Skew:	-0.821	Prob(JB):	5.05e-24
Kurtosis:	4.445	Cond. No.	102.

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

Fig: 5.5 Linear regression result from Python Jupyter

```
    Min      1Q  Median      3Q     Max
-3876.9  -344.2    89.6   451.9  1999.5

Coefficients: (1 not defined because of singularities)
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  1673.351    267.334   6.259 8.13e-10 ***
season2       945.426    203.197   4.653 4.17e-06 ***
season3       901.707    233.475   3.862 0.000127 ***
season4      1608.456    192.466   8.357 5.99e-16 ***
yr1          1904.742     64.953  29.325  < 2e-16 ***
mnth2         165.875    153.386   1.081 0.280014
mnth3         482.556    180.084   2.680 0.007606 **
mnth4         369.899    282.810   1.308 0.191476
mnth5         612.459    302.945   2.022 0.043725 *
mnth6         344.146    320.774   1.073 0.283836
mnth7        -111.914    354.783  -0.315 0.752552
mnth8         360.130    337.961   1.066 0.287105
mnth9         807.244    295.076   2.736 0.006439 **
mnth10        509.602    261.759   1.947 0.052097 .
mnth11         -7.369    247.475  -0.030 0.976258
mnth12       -110.533    191.076  -0.578 0.563196
holiday1     -547.815    214.903  -2.549 0.011088 *
weekday1      305.849    123.886   2.469 0.013880 *
weekday2      383.435    120.438   3.184 0.001542 **
weekday3      498.242    122.096   4.081 5.20e-05 ***
weekday4      542.972    121.755   4.460 1.01e-05 ***
weekday5      506.862    119.855   4.229 2.78e-05 ***
weekday6      179.328    133.139   1.347 0.178598
workingday1        NA         NA      NA       NA
weathersit2  -482.989     87.379  -5.528 5.16e-08 ***
weathersit3 -2058.932    229.480  -8.972  < 2e-16 ***
temp         4334.588    460.290   9.417  < 2e-16 ***
hum         -1696.966    342.372  -4.956 9.75e-07 ***
windspeed   -2826.756    472.912  -5.977 4.23e-09 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Fig: 5.6 Linear regression result from Rstudio with dummies of categorical variables

5.3 Random Forest:

The MAPE here comes out to be ~18, and with R it was ~14, the MAPE is fine, though now we are not much concerned about it. But, the scores that were obtained were good which ranged from 83.25 to 85, this can be seen in fig: 5.7 where the scores are plotted on the y-axis and number of trees are plotted on x-axis. This fig. helps us determining optimal number of trees required for a better model.

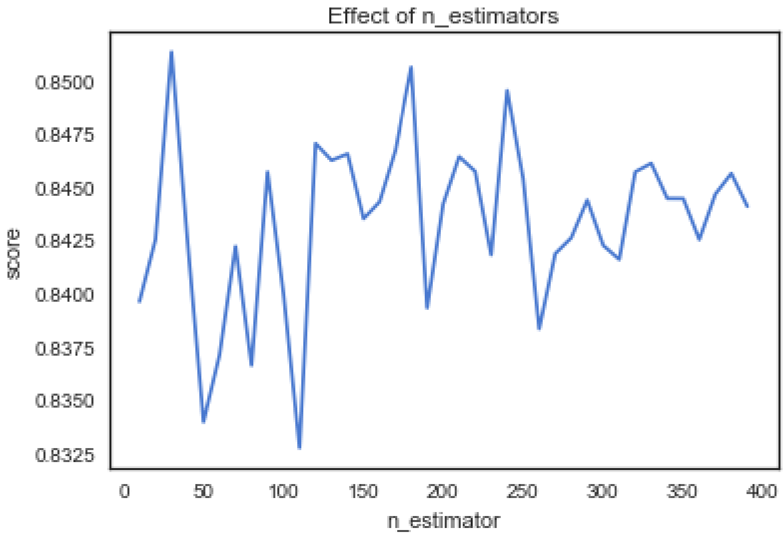

Fig: 5.7 Number of trees by the scores of Random Forest

5.4 SCOPE FOR IMPROVEMENTS AND CONCLUSION:

As we saw in our previous models, we still needed improvements. So, to further enhance the mo del, we will go back to outlier analysis and instead of removing outliers from casual (which had 40+ outliers) we will impute them by grouping means of weekdays, as only the variable 'weekda y' was affected by removing outliers. So, now we will impute the mean of cnt of weekday 6 and 0 to improve our model.

After running the most suitable technique random forest on the new imputed model, the MAPE a lmost went to ~16, our MAPE value got better than previous models, but that was not that impor tant.The score value was now in the range of 86-88% which can be seen in fig: 5.8 unlike in pre vious random forest of 83-85%. Fig 5.9 shows the test versus predicted scatter plot for easy visualization where blue(test), red(predicted) shows the similar shape of both the scatter plots.

With this, we'll conclude that Random Forest is the better approach for building this model as w e were already guessing it to be before initiating model development. One approach could have been building models for determing casuals and registered seperately, and then adding the results

Thus, there are several other ways and mixed approaches we could employ till we can eliminate previous models.

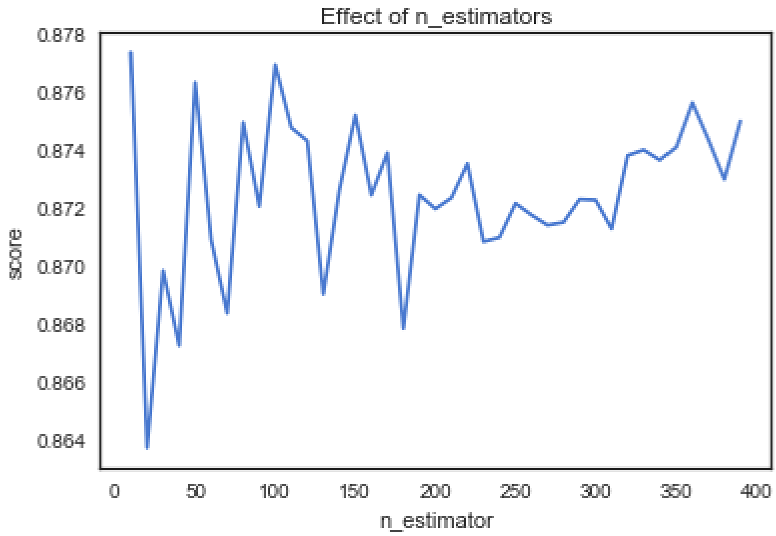

Fig: 5.8 Number of trees by the scores of Random Forest after imputation

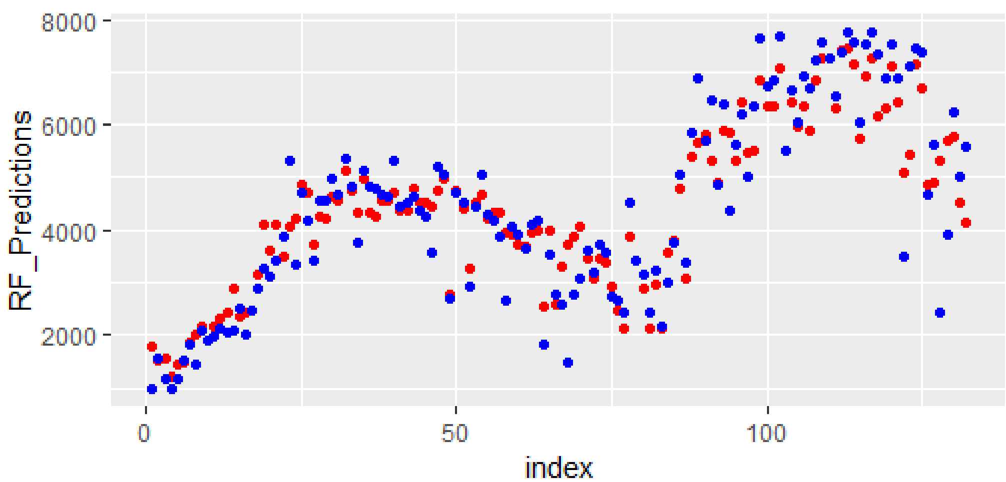

Fig: 5.9 Test versus predicted scatter plot for easy visualization where blue(test), red(predicted)

COMPLETE PYTHON NOTEBOOK CODE

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
import os
import pandas as pd
import numpy as np
from fancyimpute import KNN
import matplotlib.pyplot as plt
from scipy.stats import chi2_contingency
import seaborn as sn
from random import randrange, uniform
import numpy as np
from sklearn.cross_validation import train_test_split
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.neighbors import KNeighborsRegressor
import pylab
from scipy import stats
from sklearn.cross_validation import train_test_split
import statsmodels.api as sm
import warnings
from random import *
from numpy import *
from scipy.stats import chisquare
from sklearn.linear_model import LinearRegression
pd.options.mode.chained_assignment = None
warnings.filterwarnings("ignore", category=DeprecationWarning)
%matplotlib inline
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
#Extracting Data
original_data = pd.read_csv('Data_Project_1.csv', encoding = ' iso-8859-1')
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
original_data = original_data.reset_index(drop=True)
original_data.index += 1
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
working_data=original_data
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
df=working_data.drop(['dteday','instant'],axis=1)
working_data=working_data.drop(['dteday'],1)
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
df_cat=df.copy()
df_num=df.copy()
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
#Converting variables to categoric and numeric
num_names=['temp','atemp','hum','windspeed','casual','registered','cnt']
cat_names=['yr','season','mnth','holiday','weekday','workingday','weathersit']

df_num_sliced = df_num.loc[:,num_names]
df_cat_sliced = df_num.loc[:,cat_names]

for var in cat_names:
    df_cat[var] = df_cat[var].astype("category")
    df_cat_sliced[var] = df_cat_sliced[var].astype("category")
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
#Missing value analysis
missing_val = pd.DataFrame(df.isnull().sum())
missing_val
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
#Visualizing the Raw Data
fig,(ax1,ax2,ax3,ax4, ax5, ax6)= plt.subplots(nrows=6)
fig.set_size_inches(13,25)
cnt_by_mnth = pd.DataFrame(df_cat.groupby("mnth")["cnt"].mean()).reset_index()
sn.barplot(data=cnt_by_mnth,x="mnth",y="cnt",ax=ax1)

cnt_by_season = pd.DataFrame(df_cat.groupby(["season"])["cnt"].mean()).reset_index()
sn.barplot(data=cnt_by_season,x="season",y="cnt",ax=ax2)

cnt_by_weekday = pd.DataFrame(df_cat.groupby("weekday")["cnt"].mean()).reset_index()
sn.barplot(data=cnt_by_weekday,x="weekday",y="cnt",ax=ax3)
```

```
cnt_by_weathersit = pd.DataFrame(df_cat.groupby("weathersit")["cnt"].mean()).reset_index()
sn.barplot(data=cnt_by_weathersit,x="weathersit",y="cnt",ax=ax4)

cnt_by_yr = pd.DataFrame(df_cat.groupby("yr")["cnt"].mean()).reset_index()
sn.barplot(data=cnt_by_yr,x="yr",y="cnt",ax=ax5)

transformed = pd.melt(df_cat[["yr","casual","registered"]], id_vars=['yr'],
value_vars=['casual', 'registered'])
cnt_by_user=
pd.DataFrame(transformed.groupby(["yr","variable"],sort=True)["value"].mean()).reset_index
()
sn.pointplot(x=cnt_by_user["yr"],
y=cnt_by_user["value"],hue=cnt_by_user["variable"],hue_order=["casual","registered"],
data=transformed, join=True,ax=ax6)
ax6.set(xlabel='Year', ylabel='Users Count',label='big')
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
#outlier analysis
#visualizing outliers (boxplot)
fig, axes = plt.subplots(nrows=3,ncols=2)
fig.set_size_inches(12,10)
sn.boxplot(y='hum', x='mnth',data=df_cat, width=.5,palette="colorblind",ax=axes[0][0])
sn.boxplot(y='windspeed', x='mnth',data=df_cat,
width=0.5,palette="colorblind",ax=axes[0][1])
sn.boxplot(y='atemp', x='mnth',data=df_cat, width=0.5,palette="colorblind",ax=axes[1][0])
sn.boxplot(y='cnt', x='mnth',data=df_cat, width=0.5,palette="colorblind",ax=axes[1][1])
sn.boxplot(y='casual', x='mnth',data=df_cat, width=0.5,palette="colorblind",ax=axes[2][0])
sn.boxplot(y='registered', x='mnth',data=df_cat,
width=0.5,palette="colorblind",ax=axes[2][1])
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
#Visualizing distribution
ax8 = original_data.plot(kind='scatter',x='instant', y='registered')
ax9 = original_data.plot(kind='scatter',x='instant', y='casual')
ax10 = original_data.plot(kind='scatter',x='instant', y='cnt')
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
#Histograms before removing outliers
```

```
a4_dims = (14, 14)
fig, axes = plt.subplots(nrows=3,ncols=2,figsize=a4_dims)
sn.set(color_codes=True)
sn.set(style="white", palette="muted")
sn.distplot(df_num['hum'], ax=axes[0][0])
sn.distplot(df_num['atemp'], ax=axes[0][1])
sn.distplot(df_num['registered'], ax=axes[1][0])
sn.distplot(df_num['windspeed'], ax=axes[1][1])
sn.distplot(df_num['cnt'], ax=axes[2][0])
sn.distplot(df_num['casual'], ax=axes[2][1])
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~




Outlier analysis
================

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
#Detecting outlier and replacing it with NA

q75_w, q25_w = np.percentile(df['windspeed'], [75 ,25])


iqr_w = q75_w - q25_w


minimum_w = q25_w - (iqr_w*1.5)
maximum_w = q75_w + (iqr_w*1.5)

q75_h, q25_h = np.percentile(df['hum'], [75 ,25])


iqr_h = q75_h - q25_h


minimum_h = q25_h - (iqr_h*1.5)
maximum_h = q75_h + (iqr_h*1.5)

q75_cnt, q25_cnt = np.percentile(df['casual'], [75 ,25])


iqr_cnt = q75_cnt - q25_cnt
```

```
minimum_cnt = q25_cnt - (iqr_cnt*1.5)
maximum_cnt = q75_cnt + (iqr_cnt*1.5)

q75_atemp, q25_atemp = np.percentile(df_num['atemp'], [75 ,25])


iqr_atemp = q75_atemp - q25_atemp


minimum_atemp = q25_atemp - (iqr_atemp*1.5)
maximum_atemp = q75_atemp + (iqr_atemp*1.5)


df_cat.loc[df_cat['hum'] < minimum_h, 'hum'] = np.nan
df_cat.loc[df_cat['hum'] > maximum_h, 'hum'] = np.nan
df_num.loc[df_num['hum'] < minimum_h, 'hum'] = np.nan
df_num.loc[df_num['hum'] > maximum_h, 'hum'] = np.nan
df_cat.loc[df_cat['windspeed'] < minimum_w, 'windspeed'] = np.nan
df_cat.loc[df_cat['windspeed'] > maximum_w, 'windspeed'] = np.nan
df_num.loc[df_num['windspeed'] < minimum_w, 'windspeed'] = np.nan
df_num.loc[df_num['windspeed'] > maximum_w, 'windspeed'] = np.nan
df_cat.loc[df_cat['casual'] < minimum_cnt, 'casual'] = np.nan
df_cat.loc[df_cat['casual'] > maximum_cnt, 'casual'] = np.nan
df_num.loc[df_num['casual'] < minimum_cnt, 'casual'] = np.nan
df_num.loc[df_num['casual'] > maximum_cnt, 'casual'] = np.nan
df_cat.loc[df_cat['atemp'] < minimum_atemp, 'atemp'] = np.nan
df_cat.loc[df_cat['atemp'] > maximum_atemp, 'atemp'] = np.nan
df_num.loc[df_num['atemp'] < minimum_atemp, 'atemp'] = np.nan
df_num.loc[df_num['atemp'] > maximum_atemp, 'atemp'] = np.nan
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
maximum_cnt
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# #Calculating missing value after outlier analysis
missing_val = pd.DataFrame(df_cat.isnull().sum())
missing_val
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

~~~~~~
df_cat=df_cat.dropna()
df_num=df_num.dropna()
~~~~~~

~~~~~~
#Comparing before and after outlier analysis scatter plot
df_cat_sc= df_cat
df_cat_sc['instant']= original_data['instant']
ax9_new = df_cat_sc.plot(kind='scatter',x='instant', y='casual', color="red")
ax10_new = df_cat_sc.plot(kind='scatter',x='instant', y='cnt')
ax9 = original_data.plot(kind='scatter',x='instant', y='casual',color="red")
ax10 = original_data.plot(kind='scatter',x='instant', y='cnt')
~~~~~~

~~~~~~
df_cat=df_cat.drop(['instant'],1)
~~~~~~

~~~~~~
#Visualizing the data that has been removed by deleting outliers of variable casual
df.loc[df['casual'] > 2266]
~~~~~~

Removal of outliers had only cost major drop to the variable 'weekday'

Seeing the effect on weekday before and after outlier play

~~~~~~
df_cat.groupby(["yr","weekday"])["cnt"].mean()
~~~~~~

~~~~~~
df.groupby(["yr","weekday"])["cnt"].mean()
~~~~~~

~~~~~~
df_cat['weekday'].value_counts()
~~~~~~

~~~~~~
~~~~~~

```
df['weekday'].value_counts()
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
#Histograms after removing outliers
a4_dims = (14, 14)
fig, axes = plt.subplots(nrows=3,ncols=2,figsize=a4_dims)
sn.set(color_codes=True)
sn.set(style="white", palette="muted")
sn.distplot(df_num['hum'], ax=axes[0][0])
sn.distplot(df_num['atemp'], ax=axes[0][1])
sn.distplot(df_num['registered'], ax=axes[1][0])
sn.distplot(df_num['windspeed'], ax=axes[1][1])
sn.distplot(df_num['cnt'], ax=axes[2][0])
sn.distplot(df_num['casual'], ax=axes[2][1])
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
original_data.groupby(["season"])["mnth"].unique().to_frame()
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
#outlier analysis
#visualizing outliers through boxplot after droppping
fig, axes = plt.subplots(nrows=2,ncols=2)
fig.set_size_inches(12, 10)
sn.boxplot(y='hum', x='mnth',data=df_cat, width=0.5,palette="colorblind",ax=axes[0][0])
sn.boxplot(y='windspeed', x='mnth',data=df_cat,
width=0.5,palette="colorblind",ax=axes[0][1])
sn.boxplot(y='atemp', x='mnth',data=df_cat, width=0.5,palette="colorblind",ax=axes[1][0])
sn.boxplot(y='casual', x='mnth',data=df_cat, width=0.5,palette="colorblind",ax=axes[1][1])
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
df_num_sliced = df_num.iloc[:,7:15]
df_cat_sliced = df_cat.iloc[:,0:7]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
df_cat_sliced.dtypes
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
#for var in cat_names:
 #   df_cat[var] = df_cat[var].astype("category")
  #  df_cat_sliced[var] = df_cat_sliced[var].astype("category")
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
df_num = df_num.reset_index(drop=True)
df_num.index += 1
df_cat = df_cat.reset_index(drop=True)
df_cat.index += 1
df_num_sliced = df_num_sliced.reset_index(drop=True)
df_num_sliced.index += 1
df_cat_sliced = df_cat_sliced.reset_index(drop=True)
df_cat_sliced.index += 1
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Converting the target continious variable to categorical variable by binning for various
tests
bins = [0,1000, 2000,3000, 4000,5000, 6000,7000, 8000,9000]
labels = [1,2,3,4,5,6,7,8,9]
df_cat['cnt_binned'] = pd.cut(df_num['cnt'], bins=bins, labels=labels)
df_num['cnt_binned'] = pd.cut(df_num['cnt'], bins=bins, labels=labels)
df_cat_sliced['cnt_binned'] = pd.cut(df_num['cnt'], bins=bins, labels=labels)
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Feature Selection
=================

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
#Feature Selection using Correlation, ANOVA, and Chi-Square test

## 1) Correlation

correlations = df_num_sliced.corr()
correlations
correlations.style.background_gradient()
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

```
## 2) Chi-square test of each variable with other (When dependent variable has been
converted to categorical through binning)


factors_paired_bin = [(i,j) for i in df_cat_sliced.columns.values for j in
df_cat_sliced.columns.values]

chi2_bin, p_values_bin =[], []

for f in factors_paired_bin:
    if f[0] != f[1]:
             chitest_bin = chi2_contingency(pd.crosstab(df_cat_sliced[f[0]],
df_cat_sliced[f[1]]))
             chi2_bin.append(chitest_bin[0])
             p_values_bin.append(chitest_bin[1])

    else:
             chi2_bin.append(0)
             p_values_bin.append(0)

chi2_bin = np.array(chi2_bin).reshape((8,8))
chi2_df_bin = pd.DataFrame(chi2_bin, index=df_cat_sliced.columns.values,
columns=df_cat_sliced.columns.values)
p_values_bin = np.array(p_values_bin).reshape((8,8)) # shaping it as a matrix
p_values_bin = pd.DataFrame(p_values_bin, index=df_cat_sliced.columns.values,
columns=df_cat_sliced.columns.values)
p_values_bin
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
#ANOVA Analysis
import statsmodels.api as sm
from statsmodels.formula.api import ols


cw_lm=ols('cnt ~ C(yr)+C(holiday)+C(workingday)+ C(mnth)+C(weekday)+
C(weathersit)+C(season)', data=df_cat).fit()
print(sm.stats.anova_lm(cw_lm, typ=2))
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
#Checking the amount of data for category holiday
```

```
df_cat['holiday'].value_counts()
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
#checking the importance of variable holiday
df_cat.groupby(["yr","holiday"])["cnt"].mean()
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Visualization of data through bar graphs after removing outliers
fig,(ax1,ax2,ax3,ax4, ax5, ax6)= plt.subplots(nrows=6)
fig.set_size_inches(11,20)
cnt_by_mnth = pd.DataFrame(df_cat.groupby("mnth")["cnt"].mean()).reset_index()
sn.barplot(data=cnt_by_mnth,x="mnth",y="cnt",ax=ax1)

cnt_by_season = pd.DataFrame(df_cat.groupby("season")["cnt"].mean()).reset_index()
sn.barplot(data=cnt_by_season,x="season",y="cnt",ax=ax2)

cnt_by_weekday = pd.DataFrame(df_cat.groupby("weekday")["cnt"].mean()).reset_index()
sn.barplot(data=cnt_by_weekday,x="weekday",y="cnt",ax=ax3)

cnt_by_weathersit = pd.DataFrame(df_cat.groupby("weathersit")["cnt"].mean()).reset_index()
sn.barplot(data=cnt_by_weathersit,x="weathersit",y="cnt",ax=ax4)

cnt_by_yr = pd.DataFrame(df_cat.groupby("yr")["cnt"].mean()).reset_index()
sn.barplot(data=cnt_by_yr,x="yr",y="cnt",ax=ax5)

transformed = pd.melt(df_cat[["yr","casual","registered"]], id_vars=['yr'],
value_vars=['casual', 'registered'])
cnt_by_user=
pd.DataFrame(transformed.groupby(["yr","variable"],sort=True)["value"].mean()).reset_index
()
sn.pointplot(x=cnt_by_user["yr"],
y=cnt_by_user["value"],hue=cnt_by_user["variable"],hue_order=["casual","registered"],
data=transformed, join=True,ax=ax6)
ax6.set(xlabel='Year', ylabel='Users Count',label='big')
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
#mnth_names =
["January","February","March","April","May","June","July","August","September","October","
November","December"]
```

```
#weekday_names = ["Sunday","Monday","Tuesday","Wednesday","Thursday","Friday","Saturday"]
#season_names = ["Spring","Summer","fall", "winter"]
#weather_types = ["lev_a","lev_b","lev_c"]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
#Dropping unnecessary variables according to the model

drop_features_for_lm=["casual","registered","temp"]
drop_features_for_dt_classification_after_chi=["casual","registered","temp"]
drop_features_for_rf_classification_after_chi=["casual","registered","temp"]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Model development
=================

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
#Splitting data into train and test
random.seed(20)
train_num, test_num = train_test_split(df_cat, test_size=0.2)
train_num.dtypes
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
train_num = train_num.drop(['casual','registered','temp'], axis=1)
test_num = test_num.drop(['casual','registered','temp'], axis=1)
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Decision Tree
-------------

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# 1) Decision tree regressor

                 #a)Creating no bins for  categorical variables, and keeping numerical and
other variables as it is

fit_dt = DecisionTreeRegressor(max_depth=10).fit(train_num.iloc[:,0:10],
train_num.iloc[:,10])
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

```
predictions_dt = fit_dt.predict(test_num.iloc[:,0:10])
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

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

    return mape

MAPE(test_num.iloc[:,10], predictions_dt)
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

### Checking the reason for high MAPE

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
dataset = pd.DataFrame({'pred':predictions_dt})
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
col_1 = dataset
col_1
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
col_2=test_num.iloc[:,10].to_frame()
col2=col_2.reset_index().drop('index',1)
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
col_2=col_2.reset_index()
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
col_2=col_2.drop('index',1)
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
col_2
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
new_col= pd.DataFrame(columns=['subt'])
```

~~~~~~

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
new_col['subt']= (col_2['cnt']-col_1['pred'])/(col_2['cnt'])
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
#Values of absolute percentage error
new_col = new_col.sort_values('subt', ascending = True)
new_col
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
minimum_cnt
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
df_num.loc[df_num['casual'] > 2200]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
df_num= df_num.sort_values('cnt', ascending = True)
df_num
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
#Looking for the unnatural data
df_cat.loc[df_cat['casual'] == 2]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
#Removing the row from dataset
df_num=df_num[df_num.casual != 2]
df_cat=df_cat[df_cat.casual != 2]


train_num=train_num[train_num.cnt!=22]

test_num=test_num[test_num.cnt!=22]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

Remodelling
~~~~~~

```
===========

Decision Tree
-------------

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# 1) Decision tree regressor

                 #a)Creating no bins for  categorical variables, and keeping numerical and
other variables as it is

fit_dt = DecisionTreeRegressor(max_depth=10).fit(train_num.iloc[:,0:10],
train_num.iloc[:,10])
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
predictions_dt = fit_dt.predict(test_num.iloc[:,0:10])
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

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

    return mape

MAPE(test_num.iloc[:,10], predictions_dt)
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
#Creating dot file to visualise tree  #http://webgraphviz.com/

dotfile = open("pro.dot", 'w')
dt_export = tree.export_graphviz(fit_dt, out_file=dotfile, feature_names =
train_num_vis.columns)
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Linear Regression
-----------------

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
#REGRESSION MODELS
```

```
# 2) Linear regression:

    #a)Creating dummies for each categorical variable, and keeping numerical variable as
it is

weathersit_dummy = pd.get_dummies(df_num.weathersit)
season_dummy = pd.get_dummies(df_num.season)
mnth_dummy = pd.get_dummies(df_num.mnth)
weekday_dummy = pd.get_dummies(df_num.weekday)
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
mnth_dummy.columns=['mnth_1','mnth_2','mnth_3','mnth_4','mnth_5','mnth_6','mnth_7','mnth_8
','mnth_9','mnth_10','mnth_11','mnth_12']
weekday_dummy.columns=['weekday_1','weekday_2','weekday_3','weekday_4','weekday_5','weekda
y_6','weekday_7']
season_dummy.columns=['season_1','season_2','season_3','season_4']
weathersit_dummy.columns=['weathersit_1','weathersit_1','weathersit_3']


df2tr = pd.merge(train_num, season_dummy ,left_index=True, right_index=True)
df3tr = pd.merge(df2tr, weathersit_dummy ,left_index=True, right_index=True)
df4tr = pd.merge(df3tr, weekday_dummy ,left_index=True, right_index=True)
df5tr = pd.merge(df4tr, mnth_dummy ,left_index=True, right_index=True)

df2te = pd.merge(test_num, season_dummy ,left_index=True, right_index=True)
df3te = pd.merge(df2te, weathersit_dummy ,left_index=True, right_index=True)
df4te = pd.merge(df3te, weekday_dummy ,left_index=True, right_index=True)
df5te = pd.merge(df4te, mnth_dummy ,left_index=True, right_index=True)
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
df5tr = df5tr.drop(['season','mnth','weekday','weathersit','casual','registered','temp'],
axis=1)
df5te = df5te.drop(['season','mnth','weekday','weathersit','casual','registered','temp'],
axis=1)
df5tr=df5tr[df5tr.cnt != 22]
df5te=df5te[df5te.cnt != 22]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
df5tr['count']=df5tr['cnt']
```

```
df5tr['cnt_bin_temp']=df5tr['cnt_binned']
df5tr=df5tr.drop(['cnt','cnt_binned'],1)
df5te['count']=df5te['cnt']
df5te['cnt_bin_temp']=df5te['cnt_binned']
df5te=df5te.drop(['cnt','cnt_binned'],1)
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
df5tr = df5tr.rename(columns={'count': 'cnt', 'cnt_bin_temp': 'cnt_binned'})
df5te = df5te.rename(columns={'count': 'cnt', 'cnt_bin_temp': 'cnt_binned'})
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
df5tr=df5tr.astype('float')
df5te=df5te.astype('float')
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
model_lr_dummies = sm.OLS(df5tr.iloc[:,32], df5tr.iloc[:,0:32].astype('float')).fit()
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
model_lr_dummies.summary()
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
predictions_lr_dummies = model_lr_dummies.predict(df5te.iloc[:,0:32])
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
def MAPE(y_true, y_pred):
    mape_lr_dummies = np.mean(np.abs((y_true - y_pred) / y_true))*100
    return mape_lr_dummies
#Calculate MAPE
MAPE(df5te.iloc[:,32], predictions_lr_dummies)
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# 2) Linear regression:

    #b)Creating dummies for only ordinal variable, and keeping numerical and other
variables as it is
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
weathersit_dummy = pd.get_dummies(df_num.weathersit)
weathersit_dummy.columns=['weathersit_1','weathersit_2','weathersit_3']
df5tror = pd.merge(train_num, weathersit_dummy ,left_index=True, right_index=True)
df5tror=df5tror.drop(['cnt','cnt_binned','weathersit'], 1)
df5tror['cnt']=df5tr['cnt']
df5tror['cnt_binned']=df5tr['cnt_binned']
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
df5teor = pd.merge(test_num, weathersit_dummy ,left_index=True, right_index=True)
df5teor=df5teor.drop(['cnt','cnt_binned','weathersit'], 1)
df5teor['cnt']=df5te['cnt']
df5teor['cnt_binned']=df5te['cnt_binned']
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
#df5tr['yr'] = df5tr['yr'].astype('category')
#df5tr['holiday'] = df5tr['holiday'].astype('category')
#df5tr['workingday'] = df5tr['workingday'].astype('category')
#df5te['yr'] = df5te['yr'].astype('category')
#df5te['holiday'] = df5te['holiday'].astype('category')
#df5te['workingday'] = df5te['workingday'].astype('category')
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
model_lr_or_dummies = sm.OLS(df5tror.iloc[:,12], df5tror.iloc[:,0:12].astype(float)).fit()
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
model_lr_or_dummies.summary()
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
predictions_lr_or_dummies = model_lr_or_dummies.predict(df5teor.iloc[:,0:12])
#df_num.sort_values('cnt',ascending=True)
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
MAPE(df5teor.iloc[:,12], predictions_lr_or_dummies)
```

~~~~~~

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# 2) Linear regression:

    #c)Creating no bins for  categorical variables, and keeping numerical and other
variables as it is
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
model = sm.OLS(train_num.iloc[:,10], train_num.iloc[:,0:10].astype(float)).fit()
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
model.summary()
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
predictions = model.predict(test_num.iloc[:,0:10])
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
def MAPE(y_true, y_pred):
    mape = np.mean(np.abs((y_true - y_pred) / y_true))*100
    return mape
#Calculate MAPE
MAPE(test_num.iloc[:,10], predictions)
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Random Forest
-------------

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    # 3) Random forest regressor

                 #a)Creating bins for only ordinal variable, and keeping numerical and
other variables as it is
rf_model_or = RandomForestRegressor(n_estimators =
20).fit(df5tror.iloc[:,0:12],df5tror.iloc[:,12])
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```
```

```
rf_predictions_or = rf_model_or.predict(df5teor.iloc[:,0:12])
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
MAPE(df5teor.iloc[:,12] , rf_predictions_or)
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# 3) Random forest regressor

                 #b)Creating no bins for  categorical variables, and keeping numerical and
other variables as it is
rf_model = RandomForestRegressor(n_estimators = 80).fit(train_num.iloc[:,0:10],
train_num.iloc[:,10])
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
rf_predictions = rf_model.predict(test_num.iloc[:,0:10])
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
MAPE(test_num.iloc[:,10],rf_predictions)
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
from sklearn import tree
from sklearn.metrics import accuracy_score
from sklearn.cross_validation import train_test_split
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
train_num_vis=train_num.drop(['cnt','cnt_binned'], 1)
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
#Plot of the threshold trees required for random forest
estimators = np.arange(10, 400, 10)
scores = []
for n in estimators:
    rf_model.set_params(n_estimators=n)
    rf_model.fit(train_num.iloc[:,0:10],train_num.iloc[:,10])
    scores.append(rf_model.score(test_num.iloc[:,0:10],test_num.iloc[:,10]))
```

```
plt.title("Effect of n_estimators")
plt.xlabel("n_estimator")
plt.ylabel("score")
plt.plot(estimators, scores)
```

~~~~~~

~~~~~~

Re-Modelling by imputing outliers through random forest
===

Imputation of outliers by categories' means

~~~~~~

```
#Detecting outliers and replacing with NAs

q75_w, q25_w = np.percentile(df['windspeed'], [75 ,25])


iqr_w = q75_w - q25_w


minimum_w = q25_w - (iqr_w*1.5)
maximum_w = q75_w + (iqr_w*1.5)

q75_h, q25_h = np.percentile(df['hum'], [75 ,25])


iqr_h = q75_h - q25_h


minimum_h = q25_h - (iqr_h*1.5)
maximum_h = q75_h + (iqr_h*1.5)

q75_cnt, q25_cnt = np.percentile(df['casual'], [75 ,25])


iqr_cnt = q75_cnt - q25_cnt


minimum_cnt = q25_cnt - (iqr_cnt*1.5)
maximum_cnt = q75_cnt + (iqr_cnt*1.5)
```
~~~~~~

```
df.loc[df['hum'] < minimum_h, 'hum'] = np.nan
df.loc[df['hum'] > maximum_h, 'hum'] = np.nan
df.loc[df['windspeed'] < minimum_w, 'windspeed'] = np.nan
df.loc[df['windspeed'] > maximum_w, 'windspeed'] = np.nan
df.loc[df['casual'] < minimum_cnt, 'casual'] = np.nan
df.loc[df['casual'] > maximum_cnt, 'casual'] = np.nan
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
#Imputing means of weekday category to cnt
df.loc[(df['weekday'] == 6) & pd.isnull(df['casual']) , 'cnt'] = 5732
df.loc[(df['weekday'] == 0) & pd.isnull(df['casual']) , 'cnt'] = 5036
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
df=df.drop('casual', axis=1)
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
#Dropping rest of the NAs from hum and windspeed as they are randomly distributed
df=df.dropna()
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
df=df.drop(['atemp','registered'], 1)
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
random.seed(20)

train_num_imp, test_num_imp = train_test_split(df, test_size=0.2)
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
rf_model_imp = RandomForestRegressor(n_estimators = 51).fit(train_num_imp.iloc[:,0:10],
train_num_imp.iloc[:,10])
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

```
rf_predictions_imp = rf_model_imp.predict(test_num_imp.iloc[:,0:10])
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
MAPE(test_num_imp.iloc[:,10],rf_predictions_imp)
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
##Plot of the threshold trees required for re-modelled random forest
estimators = np.arange(10, 400, 10)
scores = []
for n in estimators:
    rf_model.set_params(n_estimators=n)
    rf_model.fit(train_num_imp.iloc[:,0:10],train_num_imp.iloc[:,10])
    scores.append(rf_model.score(test_num_imp.iloc[:,0:10],test_num_imp.iloc[:,10]))
plt.title("Effect of n_estimators")
plt.xlabel("n_estimator")
plt.ylabel("score")
plt.plot(estimators, scores)
```

Complete RStudio Code:

```
rm(list=ls())
library(rpart)
library(MASS)
library(ggplot2)
library("scales")
library("psych")
library("gplots")
library(corrgram)
x = c("ggplot2", "corrgram", "DMwR", "caret", "randomForest", "unbalanced", "C50",
"dummies", "e1071", "Information",
      "MASS", "rpart", "gbm", "ROSE", 'sampling', 'DataCombine',
'inTrees',"dplyr","plyr","reshape","data.table")
getwd()
#devtools::install_github("hadley/dplyr")
.libPaths()
library(githubinstall)
#githubinstall("dplyr")
```

```
#install.packages(x)
#rm(x)


#Extracting Data
marketing_train=
read.csv("C:\\Users\\Deepanshu\\Desktop\\Edwisor\\Projects\\Problem\\Data_Project_1.cs
v", header = T)

df=
read.csv("C:\\Users\\Deepanshu\\Desktop\\Edwisor\\Projects\\Problem\\Data_Project_1.cs
v", header = T)




#Visualizing the Raw Data
install.packages("dplyr")
library(dplyr)


#Histograms
par(mfrow=c(4,2))
par(mar = rep(2, 4))
hist(df$season)
hist(df$weathersit)
hist(df$hum)
hist(df$holiday)
hist(df$workingday)
hist(df$temp)
hist(df$atemp)
hist(df$windspeed)


#Missing Values Analysis
missing_val = data.frame(apply(marketing_train,2,function(x){sum(is.na(x))}))




##Data Manupulation; converting numeric categories into factor numeric
num_names= c('temp','atemp','hum','windspeed','casual','registered','cnt')
cat_names= c('yr','season','mnth','holiday','weekday','workingday','weathersit')

#Converting variables to categoric and numeric
for(i in c(1,2,3,4,5,6,7,8,9)){
  marketing_train[,i] = as.factor(marketing_train[,i])                          #
Converting multiple variable into data types using loops
}
rm(i)
marketing_train$instant=NULL
marketing_train$dteday=NULL
```

```
# ## BoxPlots - Distribution and Outlier Check
numeric_index = sapply(marketing_train,is.numeric) #selecting only numeric
numeric_data = marketing_train[,numeric_index]
cnames = colnames(numeric_data)
rm(num_names)
boxplot(marketing_train$cnt~marketing_train$mnth,xlab="mnth", ylab="count of users")
boxplot(marketing_train$hum~marketing_train$mnth,xlab="mnth", ylab="count of users")
boxplot(marketing_train$windspeed~marketing_train$mnth,xlab="mnth", ylab="count of
users")
boxplot(marketing_train$casual~marketing_train$mnth,xlab="mnth", ylab="count of
users")



#Bar Plots
cnt.mean <- t(tapply(marketing_train$cnt,
                         list(marketing_train$mnth, marketing_train$yr), mean))
 barplot(cnt.mean, col=c("darkblue","red"), beside=TRUE, legend=rownames(cnt.mean))

cnt1.mean <- t(tapply(marketing_train$cnt,
                      list(marketing_train$weekday, marketing_train$yr), mean))
barplot(cnt1.mean, col=c("darkblue","red"), beside=TRUE, legend=rownames(cnt1.mean))
#install.packages(ggplot2)
#setwd("C:/Users/Deepanshu/Documents/R/win-library")
#getwd()
for (i in 1:length(cnames))
  {
     assign(paste0("gn",i), ggplot(aes_string(y = (cnames[i]), x = "mnth"), data =
subset(marketing_train))+
              stat_boxplot(geom = "errorbar", width = 0.5) +
              geom_boxplot(outlier.colour="red", fill = "white" ,outlier.shape=18,
                           outlier.size=1, notch=FALSE) +
              theme(legend.position="bottom")+
              labs(y=cnames[i],x="months"))
   }

# Plotting plots together
gridExtra::grid.arrange(gn1,gn5,gn2,gn6,gn7,gn3,ncol=3, nrow=2)




#loop to remove outliers from all variables
  for(i in cnames){
    print(i)
   val = marketing_train[,i][marketing_train[,i] %in%
boxplot.stats(marketing_train[,i])$out]
 print(length(val))
    marketing_train = marketing_train[which(!marketing_train[,i] %in% val),]
  }


#Feature Selection####
```

```
# Correlation Plot

corrgram(marketing_train[,numeric_index], order = F,
         upper.panel=panel.pie, text.panel=panel.txt, main = "Correlation Plot")


 ## Chi-squared Test of Independence

#Converting numeric target variable to categorical through binning

marketing_train$cnt_binned=marketing_train$cnt

marketing_train$cnt_binned <- cut(marketing_train$cnt_binned,
                                  breaks = c(-Inf, 1000, 2000, 3000, 4000,
5000,6000,7000,8000, Inf),
                                  labels = c("1", "2", "3", "4", "5",
"6","7","8","9"),
                                  right = FALSE)

factor_index = sapply(marketing_train,is.factor)
factor_data = marketing_train[,factor_index]


for (i in 1:7)
{
  print(names(factor_data)[i])
  print(chisq.test(table(factor_data$cnt_binned,factor_data[,i])))
}


if(!require(car)){install.packages("car")}

#ANOVA Analysis
  anova_multi_way <- aov(cnt~(yr)+(holiday)+(workingday)+ (mnth)+(weekday)+
(weathersit)+(season), data = marketing_train)
 summary(anova_multi_way)


 ## Dimension Reduction
 marketing_train = subset(marketing_train,
                          select = -c(casual,registered,atemp))




 ###Model Development####
library(DataCombine)
  rmExcept("marketing_train")
  original_data=
read.csv("C:\\Users\\Deepanshu\\Desktop\\Edwisor\\Projects\\Problem\\Data_Project_1.cs
v", header = T)
```

```
library(caret)
set.seed(1234)
train.index = createDataPartition(marketing_train$cnt, p = .80, list = FALSE)
train = marketing_train[ train.index,]
test  = marketing_train[-train.index,]
test_cf=test
train_cf=train
train_cf$cnt=NULL
test_cf$cnt=NULL
train$cnt_binned=NULL
test$cnt_binned=NULL

library(C50)

##Decision tree for classification
#Develop Model on training data with binned target variable
C50_model = C5.0(cnt_binned ~., train_cf, trials = 100, rules = TRUE)

#Summary of DT model
summary(C50_model)

#Lets predict for test cases
C50_Predictions = predict(C50_model, test_cf[,-11], type = "class")

  ConfMatrix_C50 = table(test_cf$cnt_binned, C50_Predictions)
confusionMatrix(ConfMatrix_C50)



#Decision tree regression
library(rpart)
library(MASS)
library(DMwR)
d=train[which(train$cnt == 22),]

test[which(test$cnt == 22),]
train1=train[!d,]
rm(train1)
train1 <- train[!(train$cnt==22),]
fit = rpart(cnt ~ ., data = train1, method = "anova")


#Predict for new test cases
predictions_DT = predict(fit, test[,-11])

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

MAPE(test[,11], predictions_DT)
#0.18
```

```
 #Using packages
regr.eval(test[,11], predictions_DT, stats=c('mae','rmse','mape','mse'))


#Linear Regression
#check multicollearity
#install.packages(usdm)
library(usdm)


#running regression model
lm_model = lm(cnt ~., data = train)

#Summary of the model
summary(lm_model)

#Predict
predictions_LR = predict(lm_model, test[,1:10])

#Calculate MAPE
MAPE(test[,11], predictions_LR)
#0.157

library(randomForest)
###Random Forest
RF_model = randomForest(cnt ~ ., train, importance = TRUE, ntree = 500)

library(inTrees)
#Extract rules fromn random forest
#transform rf object to an inTrees' format
 treeList = RF2List(RF_model)
#
# #Extract rules
 exec = extractRules(treeList, train[,-11])  # R-executable conditions
#
# #Visualize some rules
 exec[1:2,]
#
# #Make rules more readable:
 readableRules = presentRules(exec, colnames(train))
 readableRules[1:2,]
#
# #Get rule metrics
 ruleMetric = getRuleMetric(exec, train[,-11], train$cnt)  # get rule metrics
#
# #evaulate few rules
 ruleMetric[1:2,]

#Predict test data using random forest model
RF_Predictions = predict(RF_model, test[,-11])

plot_pred = as.data.frame(RF_Predictions)
vec1=c(1:132)
vec1['index']=as.data.frame(vec1)
```

```
plot_pred['index']=original_data['instant']
plot_test['index']=original_data['instant']

plot_test = as.data.frame(test[,11])

#Calculate MAPE
MAPE(test[,11], RF_Predictions)
#0.132

#Looing at the scatter plot of test and predicted values of random forest model
require(ggplot2)

ggplot() +
  geom_point(data=plot_pred, aes(index, RF_Predictions), color="red") +
  geom_point(data=plot_test, aes(index, test[,11]), color='blue')
```