BIKE RENTAL PREDICTION

Predication of bike rental count on daily based on the environmental and seasonal settings

Submitted by: Ritu Khurana

Date: 24th March 2020

Contents

Chapter 1

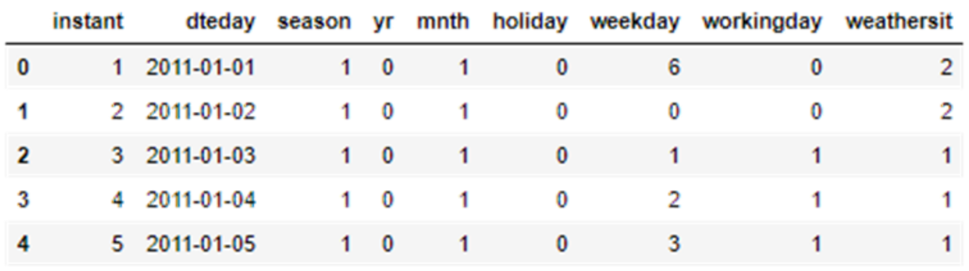
Introduction

1.1 Problem Statement

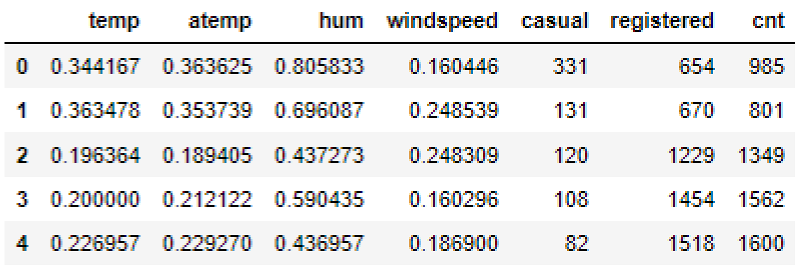
The objective of this case is to make Prediction of bike rental count based on the environmental and seasonal settings.

1.2 Data View

The main objective deals with constructing a model that predicts the rental count by understanding the user behaviour and usage pattern. The data given to us is a historical data for the year 2011 and 2012. A quick sneak-peak into the data we have is shown below:



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

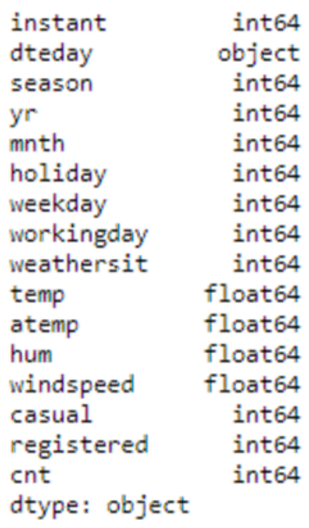


	temp	atemp	hum	windspeed	casual	registered	cnt
0	0.344167	0.363625	0.805833	0.160446	331	654	985
1	0.363478	0.353739	0.696087	0.248539	131	670	801
2	0.196364	0.189405	0.437273	0.248309	120	1229	1349
3	0.200000	0.212122	0.590435	0.160296	108	1454	1562
4	0.226957	0.229270	0.436957	0.186900	82	1518	1600

1.3 Data Shape & Type

The data provided has a shape (731, 16). The data has total 16 attributes and 731 observations with no missing values in the data. The variables qualifying as numerical variables from the dataset are normalized.

The overall variables received have the following data-type when loaded.



```
instant         int64
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
```

Chapter 2

Exploratory Data Analysis

As stated by Wikipedia –

"In statistics, exploratory data analysis (EDA) is an approach to analysing data sets to summarize their main characteristics, often with visual methods"

Exploratory data analysis is for seeing what the data can tell us beyond the formal modelling or hypothesis testing task. It deals with performing initial investigations on data so as to discover patterns, to spot anomalies.

2.1 The initial insights that were seen could be easily figured out:

1. The feature *dteday* is providing us the information related to time dimension, however the crux of the variable is already we can get from other variables like *year, month, weekday* so *dteday* is not required.
2. The feature *instant* provides us sequencing only and not providing any other information. So, need to remove this variable too.
3. Though we have observed *casual* and *registered* variables are leaky variables and their sum is the *cnt* variable so we will remove this variable also but little later after some visualization and **correlation analysis**.

2.2 Data Pre-processing

The realization of missing values is important to understand, when we are trying to successfully manage the data. These can lead to inaccurate inference if not handled properly. Handling missing values often includes checking for the relative percentage of the missing values with the total observations. Any variable possessing missing percentage more than a threshold percentage is dropped as it shall not be providing any deterministic inference to the model else includes imputing the missing values through statistical or any other suitable technique.

The missing value analysis for the data provided gives us the following:

```
season        0
year          0
month         0
holiday       0
weekday       0
workingday    0
weather       0
temp          0
atemp         0
humdity       0
windspeed     0
casual        0
registered    0
count         0
dtype: int64
```

As observed from the data above, our **dataset isn't missing any values**, we're good in this handling.

2.3 Data type Handling

It is an important pre-processing step to transform the data into their correct data-type. This typically requires domain knowledge and understanding of the problem statement. Data type handling helps in setting relationships between measures and dimensions. Also this helps in better training of the models as helps in deciding which features can be taken as running data and which can be taken as categories.

Below are the data-types after processing.

```
season       731 non-null category
year         731 non-null category
month        731 non-null category
holiday      731 non-null category
weekday      731 non-null category
workingday731 non-null category
weather      731 non-null category
temp         731 non-null float64
atemp        731 non-null float64
humdity      731 non-null float64
windspeed  731 non-null float64
casual       731 non-null int64
registered   731 non-null int64
count        731 non-null int64
```

2.4 Organizing, Plotting and Summarizing the data.

By visualising the data we can know the data is distributed and the we can detect the pattern in the data. According to the data, Fig. 2.1 is bike rental count in each season most bike rentals are in fall whereas the least in the spring. Fig. 2.2 is the bike rental counts in every month of a year.

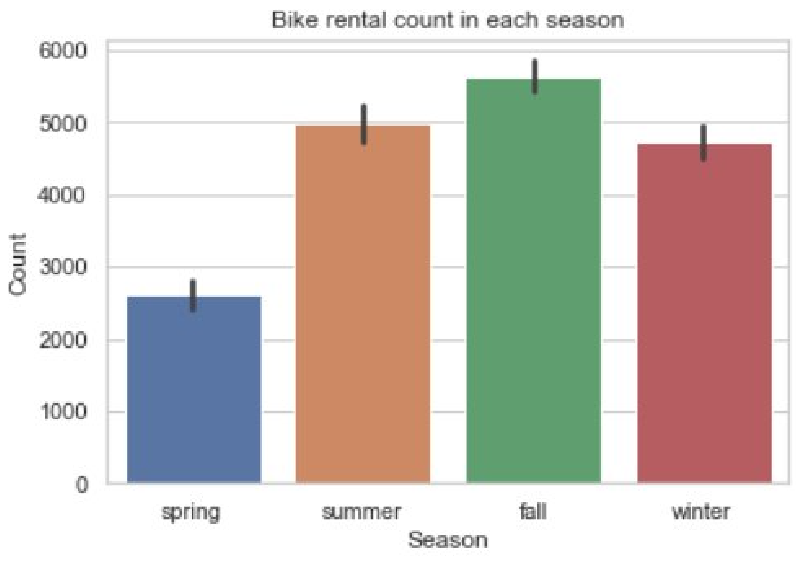


Fig. 2.1

As we can see most of the bike rentals are in the months of June and September and the least are in the beginning of the year i.e., January & February.

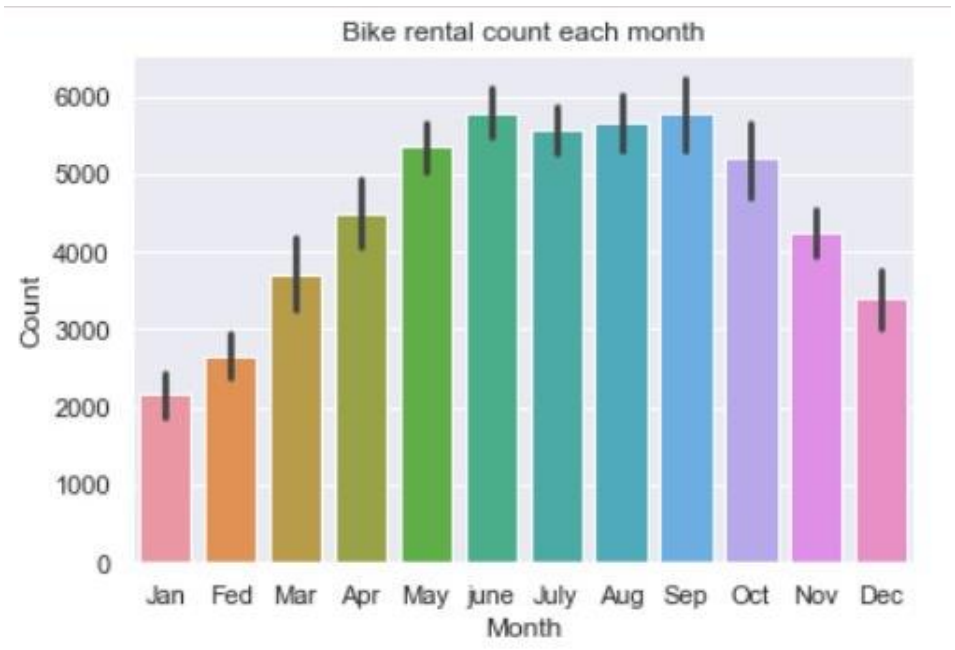


Fig. 2.2

As the given data is for years 2011 and 2012, in the Fig. 2.3 we can see the bike rental counts in the both the years.

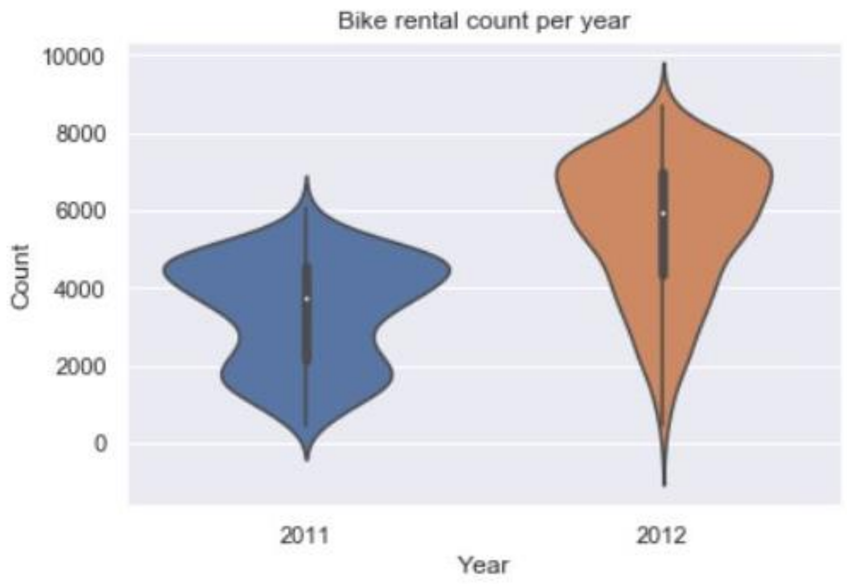


Fig. 2.3

And then the plot of bike rentals on the holidays and the working days in the Fig. 2.4

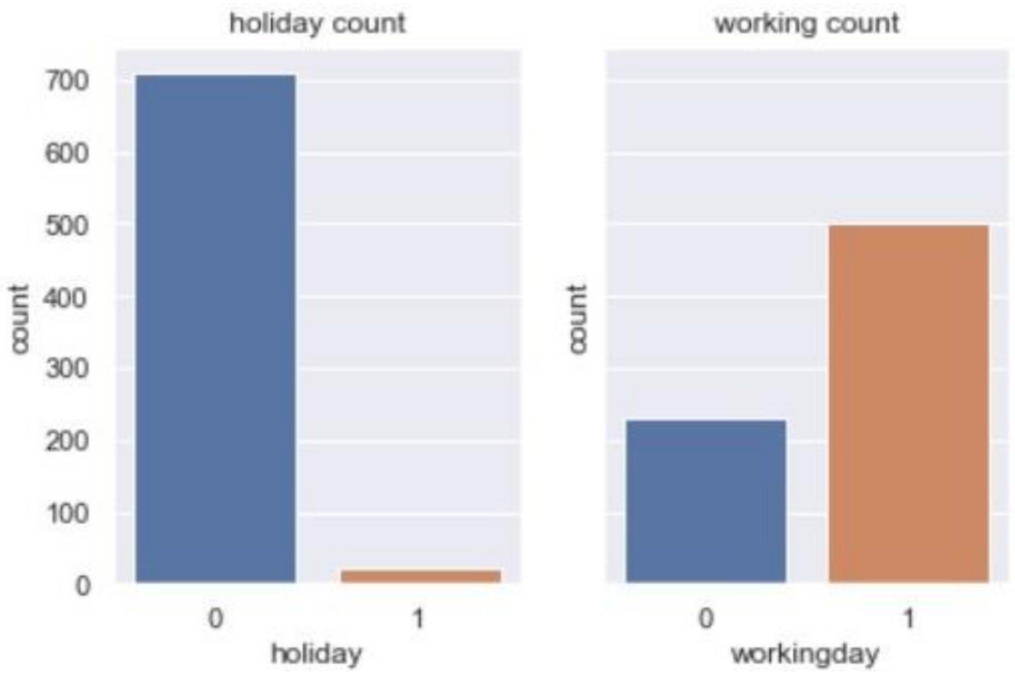


Fig. 2.4 Bike rental count weather the day is holiday or working day.

As we have the rental counts based on the day of the week in Fig. 2.5 we can see bike rentals on each day of the week.

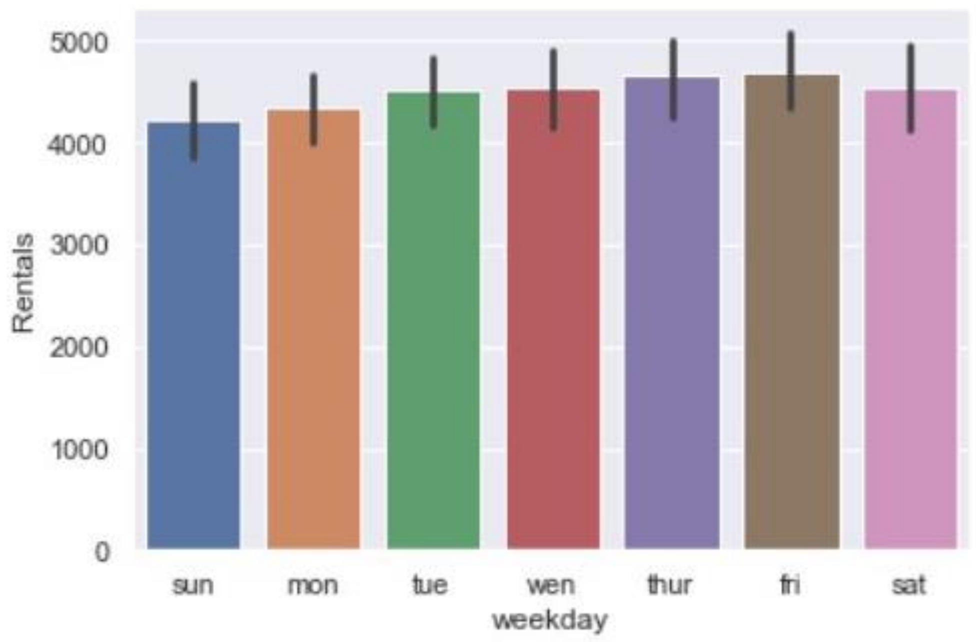


Fig. 2.5 Bike rentals count on each day of a week.

2.5 Missing Values Analysis:

In predictive modelling we have to look into data weather the data is clean, does it have any missing values. If it does have any missing then those values have to be replaced by mathematical methods if the number is insignificant they can be removed.

```
b_rent.isnull().sum()
```

```
instant       0
date          0
season        0
year          0
month         0
holiday       0
weekday       0
workingday    0
weather       0
temp          0
atemp         0
humdity       0
windspeed     0
casual        0
registered    0
count         0
```

We don’t have any missing values so, we are good to go.

2.6 Outlier Analysis:

Outliers in the data may occur due to poor measurement quality or some external reasons. In a simple way we can detect outliers by plotting box plots of the different variables in the data set. In Fig. 2.6 we can see the distributions of the four main predictors.

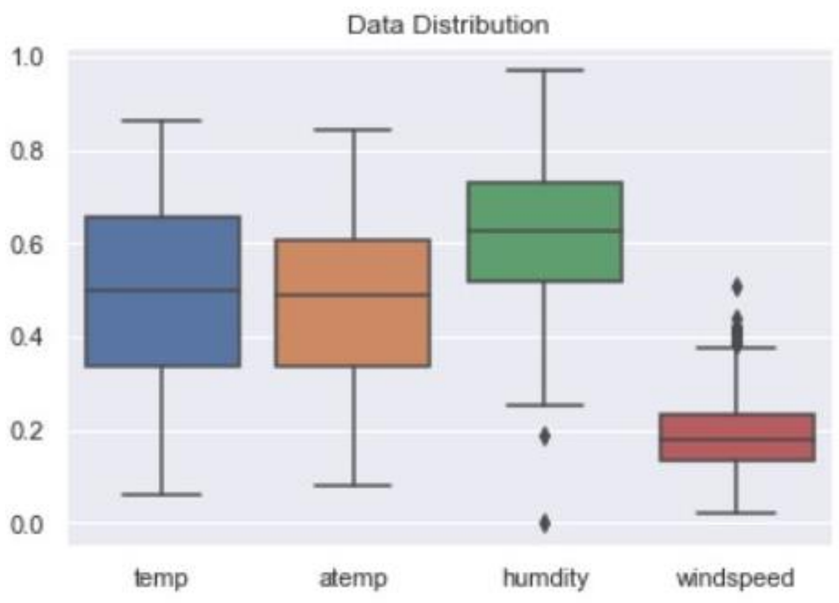


Fig. 2.6

By observing the plots we can say that both the humidity and windspeed have the outliers may due to the adverse weather conditions whatever may be the reason having outliers in the data set while predictive modelling may result in wrong predictions so, these outliers should be removed. In Fig. 2.7 we can see that the casual column has outliers but here it does not matter because we have to predict these columns, they are the target variables.

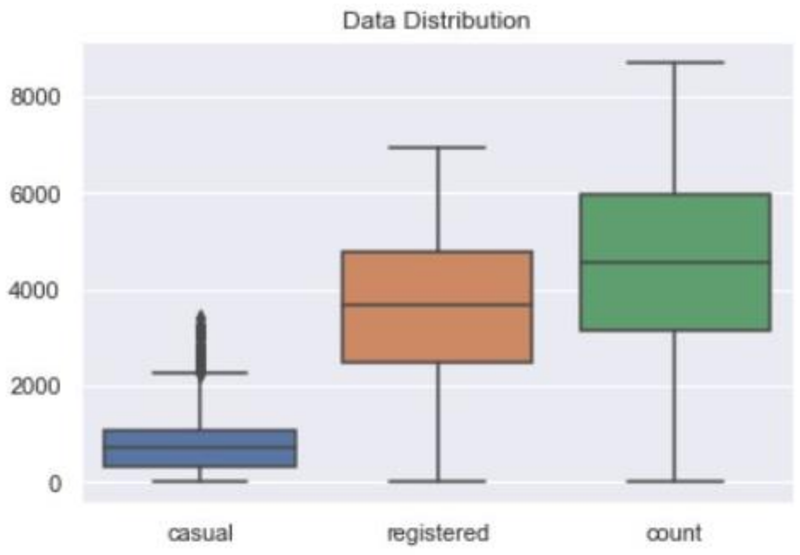
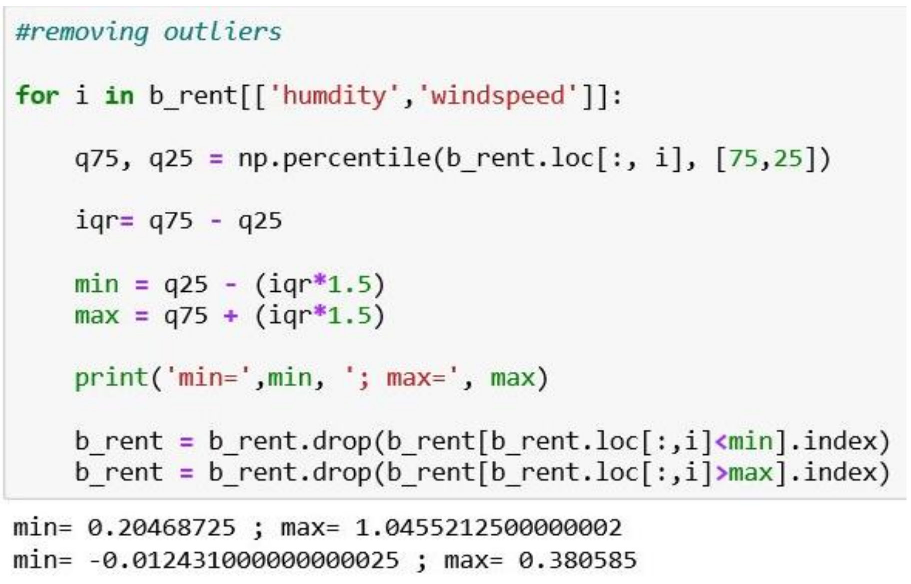


Fig. 2.7

Here is the method to remove the outliers from the data with the 25th and 75th percentile calculating the MIN and MAX then removing all the data points less than MIN and greater than MAX.

<u>Code in Python:</u>



```
#removing outliers

for i in b_rent[['humdity','windspeed']]:

    q75, q25 = np.percentile(b_rent.loc[:, i], [75,25])

    iqr= q75 - q25

    min = q25 - (iqr*1.5)
    max = q75 + (iqr*1.5)

    print('min=',min, '; max=', max)

    b_rent = b_rent.drop(b_rent[b_rent.loc[:,i]<min].index)
    b_rent = b_rent.drop(b_rent[b_rent.loc[:,i]>max].index)
```

```
min= 0.20468725 ; max= 1.0455212500000002
min= -0.012431000000000025 ; max= 0.380585
```

2.7 Feature Selection:

In the predictive modelling feature selection is about selecting the independent variables which will be helpful in predicting the target variable. It is also know as Dimensionality Reduction. For numerical data we can use correlation plot for categorical data we use chi-squared test.

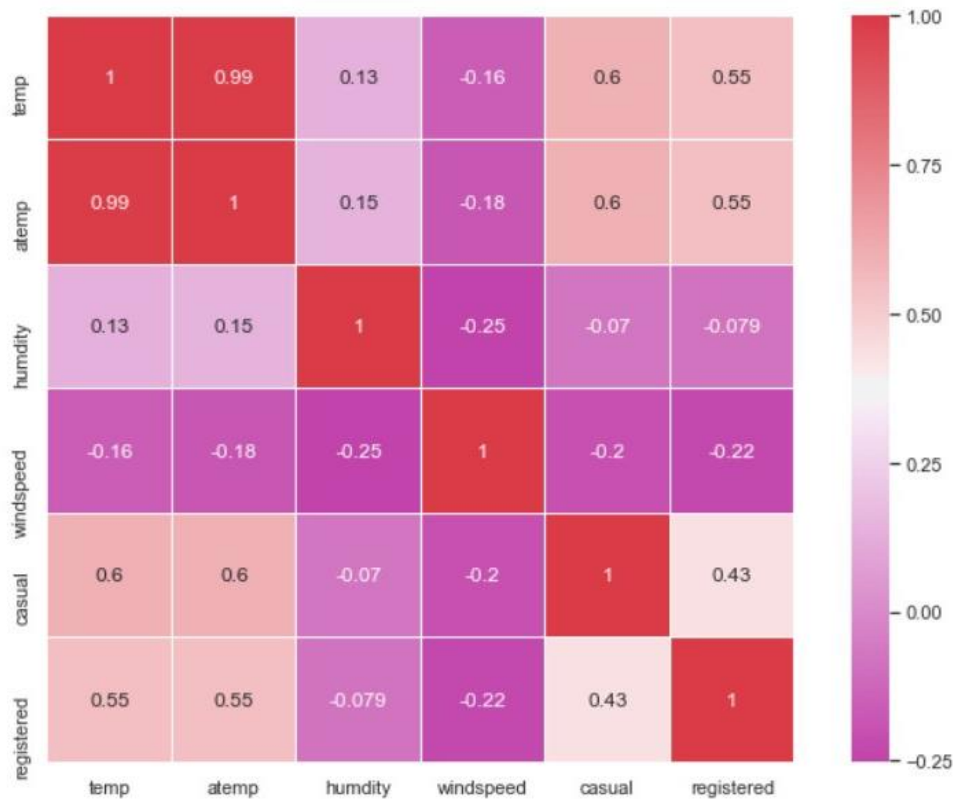
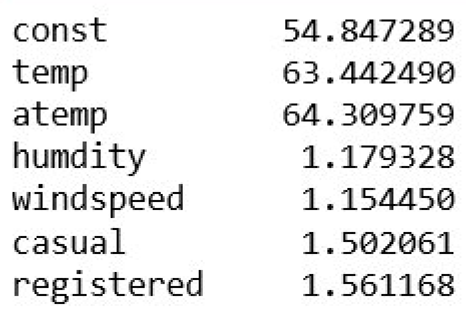


Fig. 2.8

And the multi-collinearity is checked by Variance inflation factor:



```
const         54.847289
temp          63.442490
atemp         64.309759
humdity        1.179328
windspeed      1.154450
casual         1.502061
registered     1.561168
```

- From the correlation plot (Fig. 2.8) and VIF temp & atemp are highly correlated to each other so, having both of them is useless so, we have to remove atemp.
- Removing casual and registered because they are what we have to predict. (casual + Registered = Count)

For the categorical data we use chi-squared test in which we can compare variables with each other or with the target variable this is a hypothesis test we get p-value. If the p-value is greater than critical value then we can reject the null hypothesis.

```
season

0.5132912794522374

year

0.38524412679198045

month

0.4854115917709501

holiday

0.7845587226117702

weekday

0.4779487865516262

workingday

0.49471021239790564

weather

0.6747466999358342
```

After removing all the unnecessary columns this is how the clean data set look like:

	season	year	month	weekday	workingday	weather	temp	humdity	windspeed	count
0	1	0	1	6	0	2	0.344167	0.805833	0.160446	985
1	1	0	1	0	0	2	0.363478	0.696087	0.248539	801
2	1	0	1	1	1	1	0.196364	0.437273	0.248309	1349
3	1	0	1	2	1	1	0.200000	0.590435	0.160296	1562
4	1	0	1	3	1	1	0.226957	0.436957	0.186900	1600

MODEL DEVLOPMENT

3.1 Making Dummies Variables:

In predictive modelling we will first try and look at all the probability distributions of the variables. Most analysis like regression, require the data to be normally distributed. We can visualize that in a glance by looking at the probability distributions or probability density functions of the variable.

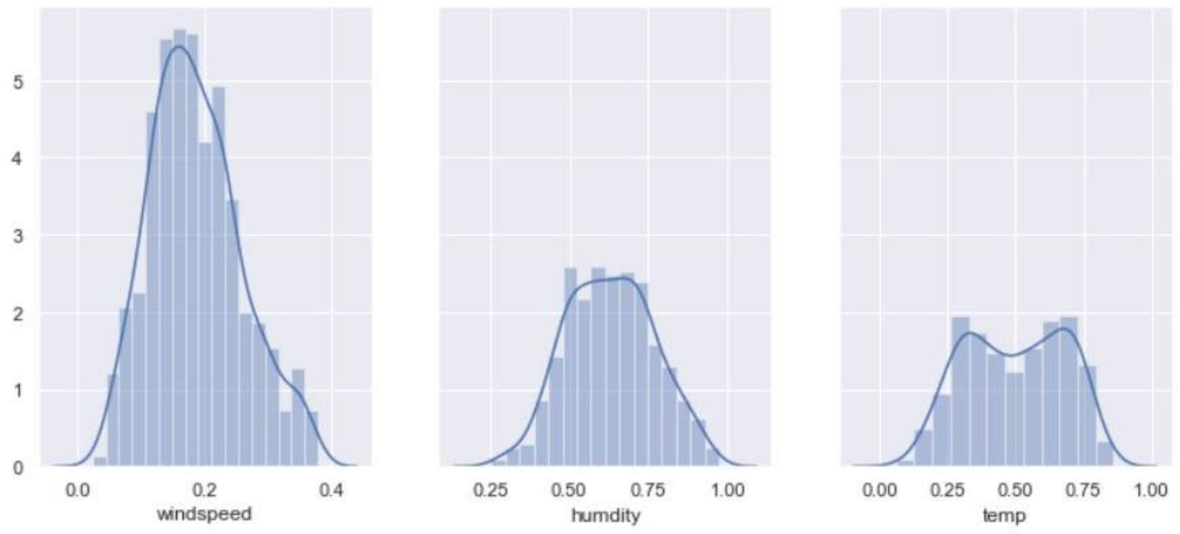


Fig. 2.10 Probability distribution

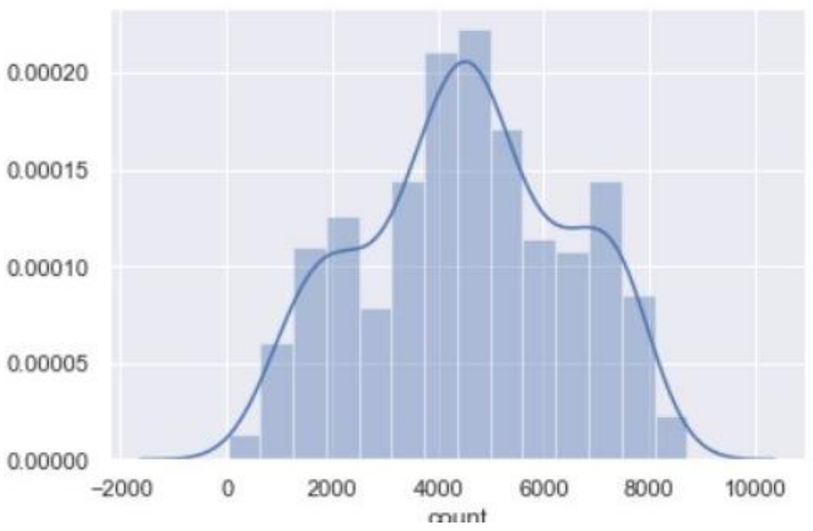
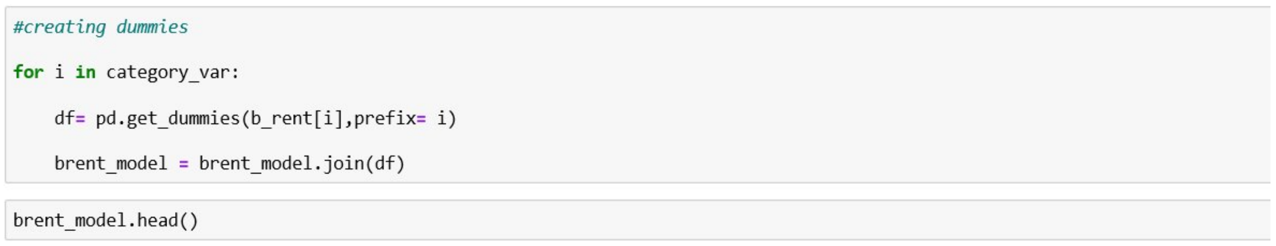


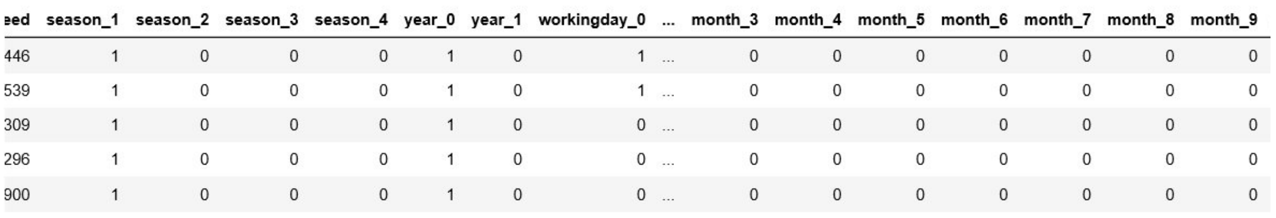
Fig. 2.11

And the rest all categorical variables should be stored as dummies so that they can be good in prediction model.



```
#creating dummies

for i in category_var:

    df= pd.get_dummies(b_rent[i],prefix= i)

    brent_model = brent_model.join(df)
```

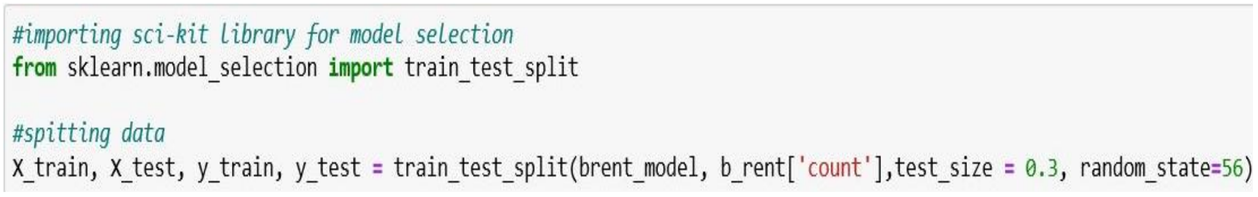
```
brent_model.head()
```



eed	season_1	season_2	season_3	season_4	year_0	year_1	workingday_0	...	month_3	month_4	month_5	month_6	month_7	month_8	month_9
446	1	0	0	0	1	0	1	...	0	0	0	0	0	0	0
539	1	0	0	0	1	0	1	...	0	0	0	0	0	0	0
309	1	0	0	0	1	0	0	...	0	0	0	0	0	0	0
296	1	0	0	0	1	0	0	...	0	0	0	0	0	0	0
900	1	0	0	0	1	0	0	...	0	0	0	0	0	0	0

3.2 Modelling- Train/test Split:

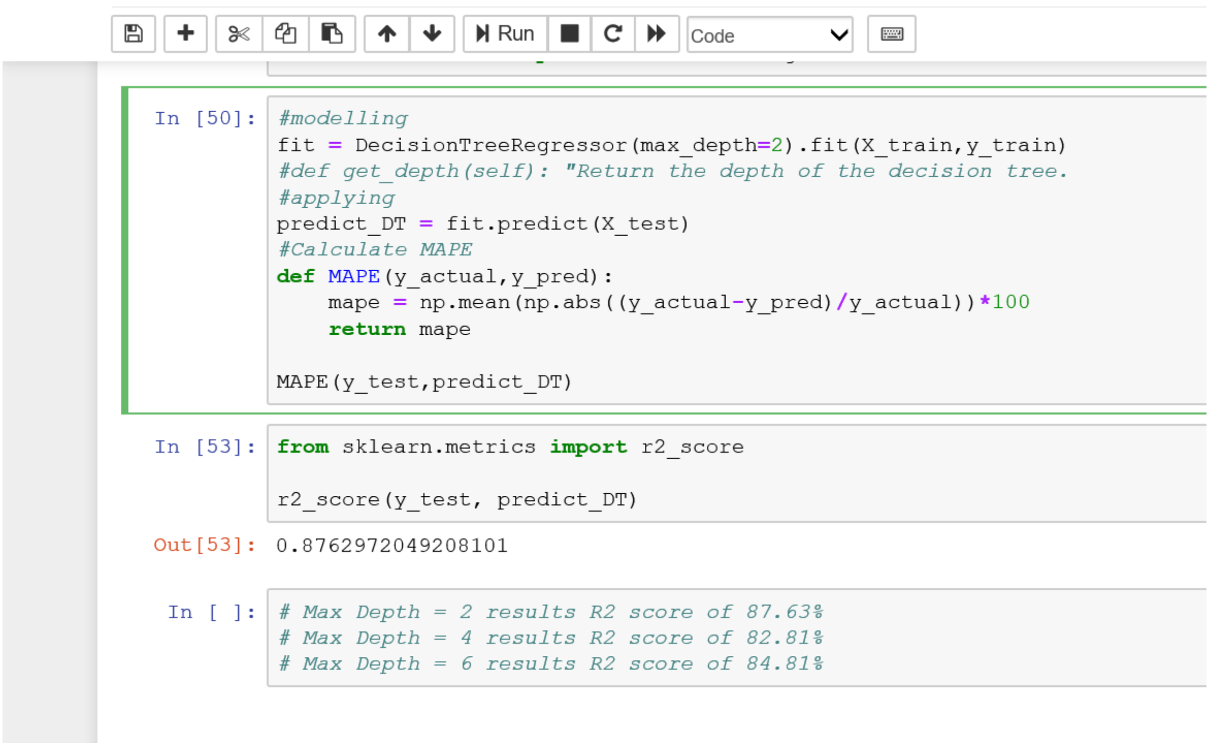
In modelling we first have to split the clean dataset to train-set and test-set and then develop different models and evaluate them by metrics.



```
#importing sci-kit library for model selection
from sklearn.model_selection import train_test_split

#spitting data
X_train, X_test, y_train, y_test = train_test_split(brent_model, b_rent['count'],test_size = 0.3, random_state=56)
```

3.3 Decision Tree:

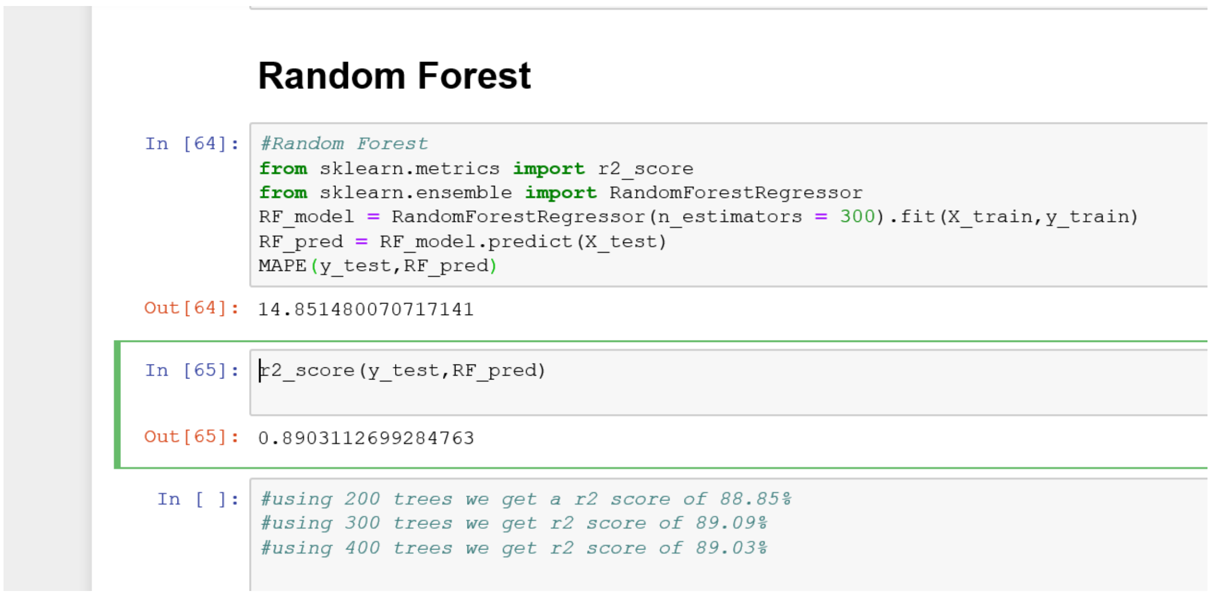
A predictive model based on a branching series of Boolean tests. It Can be used for classification and regression. Decision tree is a rule. Each branch connects nodes with “and” and multiple branches are connected by “or”. A decision tree is drawn upside down with its root at the top. The Following code I executed for Decision tree regressor in Python:



```
In [50]: #modelling
         fit = DecisionTreeRegressor(max_depth=2).fit(X_train,y_train)
         #def get_depth(self): "Return the depth of the decision tree.
         #applying
         predict_DT = fit.predict(X_test)
         #Calculate MAPE
         def MAPE(y_actual,y_pred):
             mape = np.mean(np.abs((y_actual-y_pred)/y_actual))*100
             return mape

         MAPE(y_test,predict_DT)

In [53]: from sklearn.metrics import r2_score

         r2_score(y_test, predict_DT)

Out[53]: 0.8762972049208101

In [ ]: # Max Depth = 2 results R2 score of 87.63%
        # Max Depth = 4 results R2 score of 82.81%
        # Max Depth = 6 results R2 score of 84.81%
```

3.4 Random Forest:

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes or mean prediction of the individual trees. The Following code I Executed in Python for Random Forest Regressor:

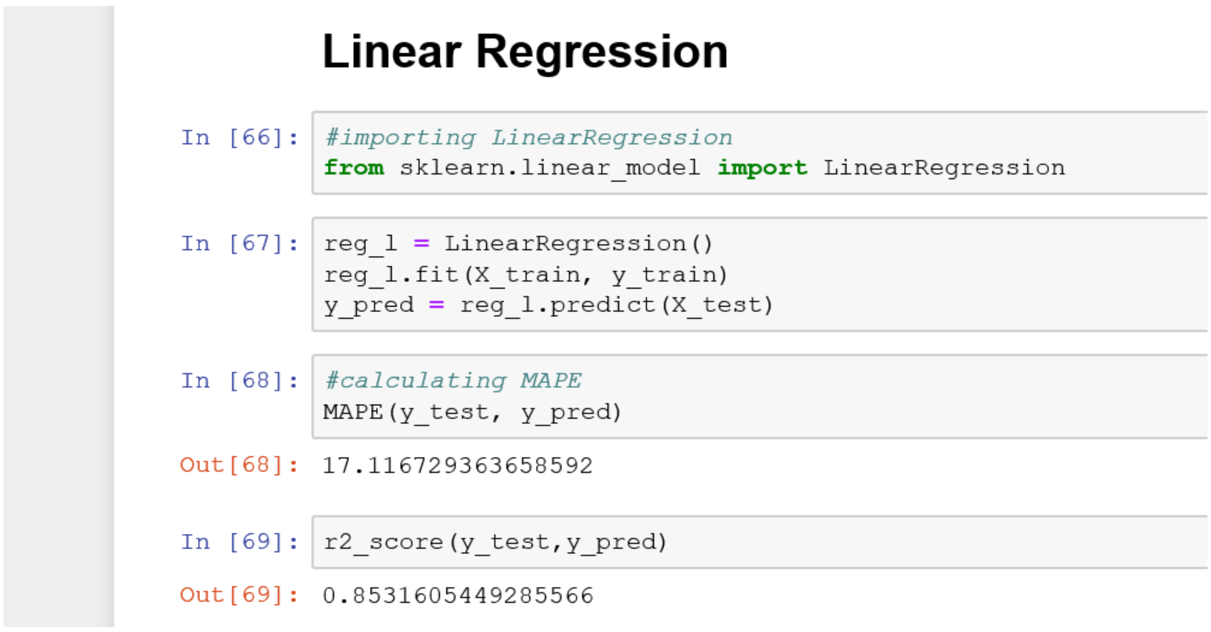


Random Forest

```
In [64]: #Random Forest
         from sklearn.metrics import r2_score
         from sklearn.ensemble import RandomForestRegressor
         RF_model = RandomForestRegressor(n_estimators = 300).fit(X_train,y_train)
         RF_pred = RF_model.predict(X_test)
         MAPE(y_test,RF_pred)

Out[64]: 14.851480070717141

In [65]: r2_score(y_test,RF_pred)

Out[65]: 0.8903112699284763

In [ ]: #using 200 trees we get a r2 score of 88.85%
        #using 300 trees we get r2 score of 89.09%
        #using 400 trees we get r2 score of 89.03%
```

3.5 Linear Regression:

Linear Regression is a machine learning algorithm based on supervised learning. It performs a regression task. Regression models a target prediction value based on independent variables. It is mostly used for finding out the relationship between variables and forecasting. Different regression models differ based on – the kind of relationship between dependent and independent variables, they are considering, and the number of independent variables being used. Linear regression performs the task to predict a dependent variable value (y) based on a given independent variable (x). So, this regression technique finds out a linear relationship between x (input) and y(output). Hence, the name is Linear Regression. The code:

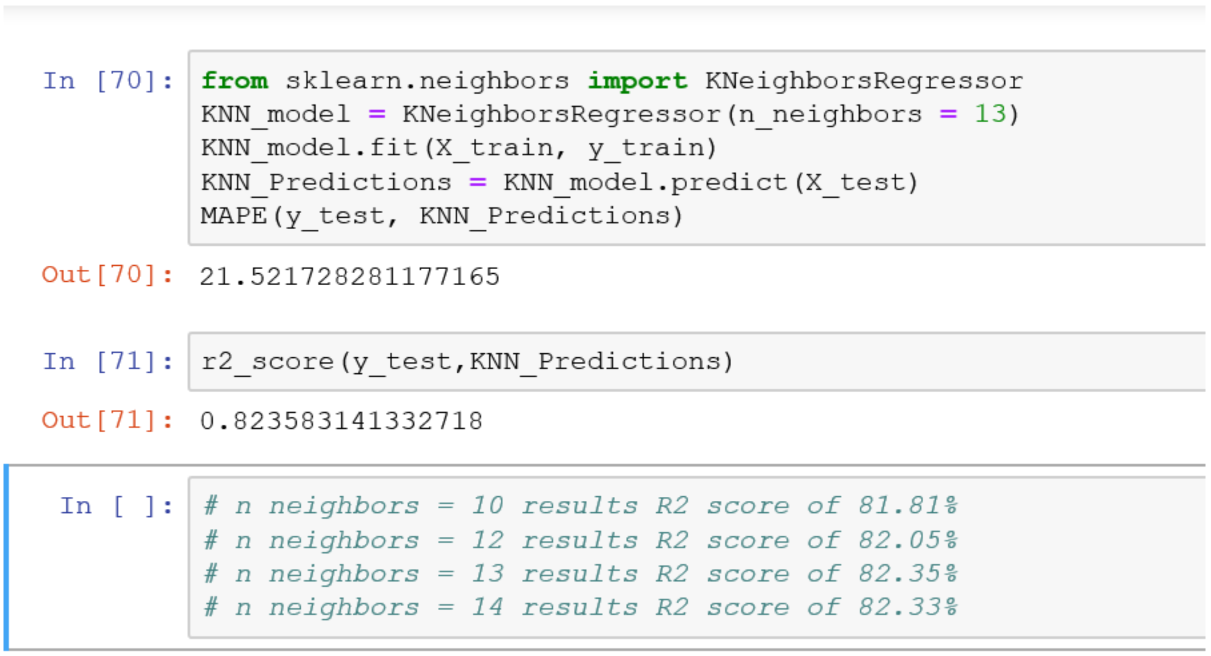


Linear Regression

```
In [66]: #importing LinearRegression
         from sklearn.linear_model import LinearRegression

In [67]: reg_l = LinearRegression()
         reg_l.fit(X_train, y_train)
         y_pred = reg_l.predict(X_test)

In [68]: #calculating MAPE
         MAPE(y_test, y_pred)

Out[68]: 17.116729363658592

In [69]: r2_score(y_test,y_pred)

Out[69]: 0.8531605449285566
```

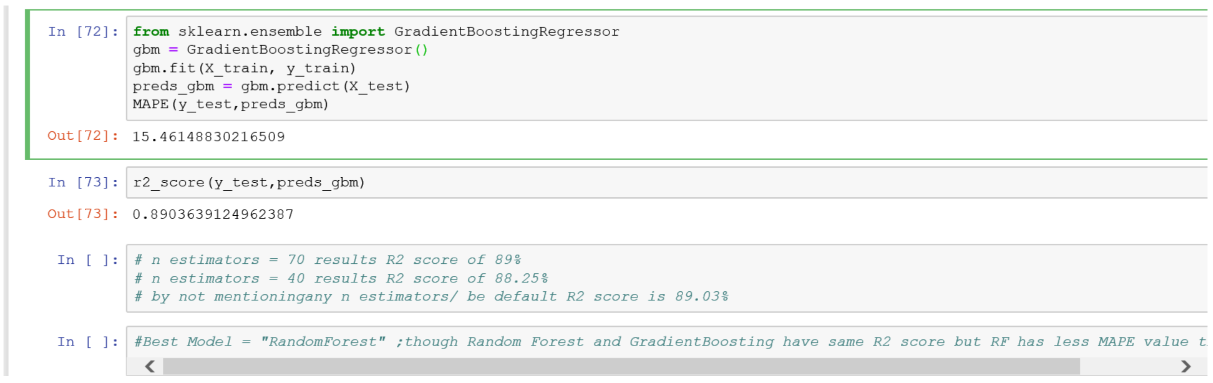
3.6 KNN:

Regression based on k-nearest neighbors. The target is predicted by local interpolation of the targets associated of the nearest neighbors in the training set. The code for KNN regressor in python:



```
In [70]: from sklearn.neighbors import KNeighborsRegressor
         KNN_model = KNeighborsRegressor(n_neighbors = 13)
         KNN_model.fit(X_train, y_train)
         KNN_Predictions = KNN_model.predict(X_test)
         MAPE(y_test, KNN_Predictions)

Out[70]: 21.521728281177165

In [71]: r2_score(y_test,KNN_Predictions)

Out[71]: 0.823583141332718

In [ ]: # n neighbors = 10 results R2 score of 81.81%
        # n neighbors = 12 results R2 score of 82.05%
        # n neighbors = 13 results R2 score of 82.35%
        # n neighbors = 14 results R2 score of 82.33%
```

3.7 Gradient Boosting: Gradient boosting is a machine learning technique for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees. Code of Gradient boosting regressor in Python:



```
In [72]: from sklearn.ensemble import GradientBoostingRegressor
         gbm = GradientBoostingRegressor()
         gbm.fit(X_train, y_train)
         preds_gbm = gbm.predict(X_test)
         MAPE(y_test,preds_gbm)
Out[72]: 15.46148830216509

In [73]: r2_score(y_test,preds_gbm)
Out[73]: 0.8903639124962387

In [ ]: # n estimators = 70 results R2 score of 89%
        # n estimators = 40 results R2 score of 88.25%
        # by not mentioningany n estimators/ be default R2 score is 89.03%

In [ ]: #Best Model = "RandomForest" ;though Random Forest and GradientBoosting have same R2 score but RF has less MAPE value t
```

4.1 Model Evaluation:

Our problem the target variable is continuous in nature we will go for Regression Matrix evaluation.

Regression Evaluation measure of error:

1. Mean Absolute Error:The mean absolute error(MAE) has the same unit as the original data, and it can only be compared between models whose errors are measured in the same units. It is similar in magnitude to RMSE, but slightly smaller.
2. Mean Absolute Percentage Error: It measures accuracy as a percentage of error. It is obtained by multiplying MAE by 100.
3. Root Mean Square Error: It is the square root of the square of difference of the given value and true value.

✓ Which Error method to choose among these three methods:

If our dataset contains a transition data or time-based which is also called as time series analysis or time series data , then it is better to go for RMSE

If we want to convert our error number into percentage then MAPE is best method. In our analysis of error, it is better to go for the percentage error method. **So MAPE error method is used.**

- ✓ **Accuracy**: Is obtained as: 100-MAPE

- ✓ **R Square**: Another parameter which is often used than MAPE for regression models evaluations is using R2. Maximizing R2 is equivalent to minimizing the sum of squared errors or the Mean Squared Error.

Python executed Results->	MAPE	Accuracy	R2 score(With Right Parameter)	Right Parameter name & Value	Other Parameter value, which we not selected	Other Parameter R2 score, which we not selected
Decision tree	28.18%	71.82	87.63%	Max debth= 2	4, 6	82.81%, 84.81%
Random Forest	14.85%	85.15	89.03%	n_estimators = 300	200, 400	88.85%,89.03%
Linear Regression	17.11%	82.89	85.31%			
KNN	21.52%	78.48	82.35%	n_neighbors = 13	12, 14	82.05%, 82.33%
Gradient Boosting	15.46%	84.54	89.03%	n estimators= bydefault	70, 40	89.00%, 88.25%

4.2 Model Selection:

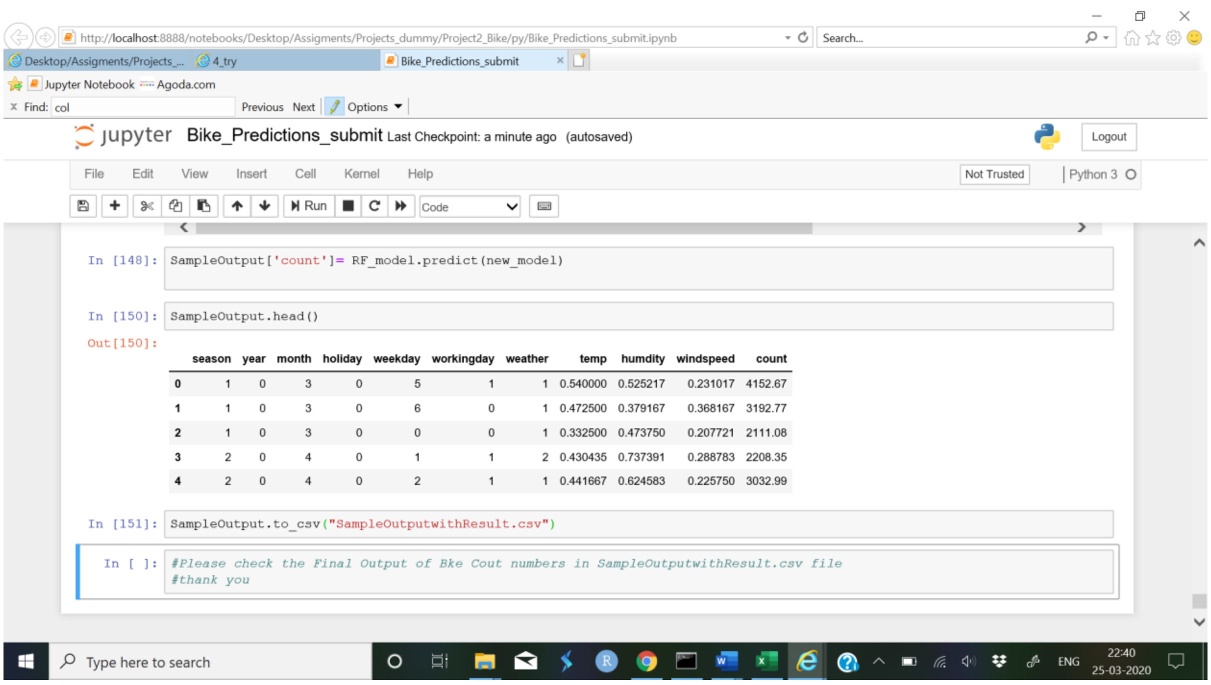
We got the same R2 values for Gradient boosting & Random Forest, but the MAPE value is less for Random Forest, Hence Accuracy is highest seen in **Random Forest Regressor**, so we choose the same.

4.3 Example of output with a sample input:

Please check the Input sample file	: SampleInput.csv
Output sample file	: SampleOutputwithResult.csv
Model used here	: Random Forest.
Python file name	: Bike_Predictions_submit.ipynb

Note: Dummies were again created on the SampleInput categorical variables before applying the prediction

Screenshot:



```
In [148]: SampleOutput['count']= RF_model.predict(new_model)

In [150]: SampleOutput.head()

Out[150]:
```

	season	year	month	holiday	weekday	workingday	weather	temp	humdity	windspeed	count
0	1	0	3	0	5	1	1	0.540000	0.525217	0.231017	4152.67
1	1	0	3	0	6	0	1	0.472500	0.379167	0.368167	3192.77
2	1	0	3	0	0	0	1	0.332500	0.473750	0.207721	2111.08
3	2	0	4	0	1	1	2	0.430435	0.737391	0.288783	2208.35
4	2	0	4	0	2	1	1	0.441667	0.624583	0.225750	3032.99

```
In [151]: SampleOutput.to_csv("SampleOutputwithResult.csv")

In [ ]: #Please check the Final Output of Bke Cout numbers in SampleOutputwithResult.csv file
        #thank you
```

5.1 End Notes:

- I have used not only classic algorithms like Random forest or Linear Regression but also Gradient boosting algos also.
- I have tried the hyperpameter values with my models and evaluated the good output out of it. Thanks for sharing it in the previous project remarks.
- Please find attached :
    - python code file name: Bike_Predictions_submit.ipynb
    - R code file name: Bike_rental_R_File.R
    - Sample Input file name: SampleInput.csv
    - Sample Output file name: SampleOutputwithResult.csv
- I have done the sampleInput and sampleout code in Python file, Though screenshot is attached with this report with Appendix 4.3.
- Thanks for your time, Looking forward to get a mock call soon.

5.2 References:

For calculating r2 score in R: https://stackoverflow.com/questions/40901445/function-to-calculate-r2-r-squared-in-r

Various types of plots in seaborn: https://seaborn.pydata.org/tutorial/categorical.html

For dummies in R: https://www.r-bloggers.com/r-the-dummies-package/

**********************End of the Report *********************