Bike Renting Predictions

Sanket Mote

11 January 2020

Table of Contents

Sr No. Title Pg No.

Chapter 1

Introduction

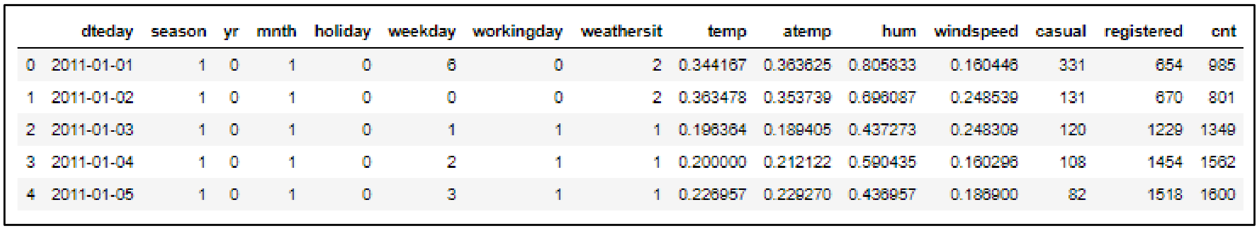
1.1 Problem Statement

In order to provide better accessibility to the customer a Bike Renting firm wants to know number of bikes rented depending on the environmental and seasonal settings so that approximate number of bikes can be made available. This will improve its operational efficiency which eventually lead in customer satisfaction and increase in sales over period.

1.2 Data

Our task is to build a model which will predict the number of bikes rented on day to day basis based on environmental and seasonal changes. Given below is a sample data set which we will be using for the model development and prediction of number of bikes rented:

Table 1.1: Bike Rental Data (Observations: 1-5)



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

As you can see in the data, we have 7 categorical variables (season, year, month, holiday, weekday, working day, weathersit), 6 predictor variables (temp, atemp, hum, windspeed, casual, registered) and 1 Output or Dependent variable (cnt).

Chapter 2

Methodology

2.1 Pre Processing

Before starting predictive modelling, we need to check the data and understand various data definitions. Also, we need to check whether data is in proper format before developing model and feeding the data, if data is not in proper format, we need to clean the data in order to increase the efficiency of the model. To start the process, we have basic steps which needs to be completed such as data cleaning, visualization using graph or plots, standardizing if data is not uniformly distributed these steps are a part of Exploratory Data Analysis.

We have plotted various graphs in order to understand the uniformity and visualize data for better understanding. We also compare the independent variables with dependent variable in order to understand its contribution in the prediction of target or dependent variable. If we find two or more variables with similar data, we keep only one and remove other with similar data in order to lower the complexity of model.

In fig. 1 we have plotted various graphs wherein we analyze skewness of data which will help understand if data is uniformly distributed or not. We can also make a guess whether outliers are present in the specific variable.

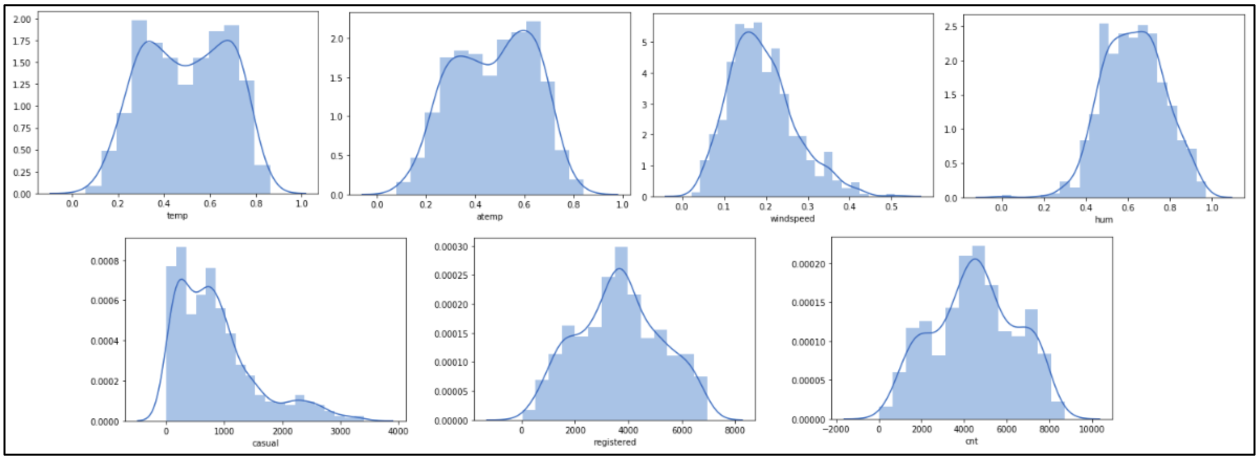


Fig. 2.1 Univariate Analysis of Continuous Dependent Variable

2.1.1 Missing Value Analysis

We always check whether dataset contains any missing value. If it contains missing value, we need to impute missing observations in a variable if it has at least 70% of data. We impute the missing data with the help of mean, median in case of continuous variable and mode in case of categorical variable. We can also use K Nearest Neighbor (KNN) in order to impute missing values. In our dataset we do not have any missing value therefore we can proceed to the next step.

2.1.2 Outlier Analysis

We can see in Fig.1 that hum, windspeed and casual variables are skewed. These variables have a high chance of containing which we need to remove so that our model is not biased and is as accurate as possible at prediction. Fig.2 shows the boxplot showing outliers below:

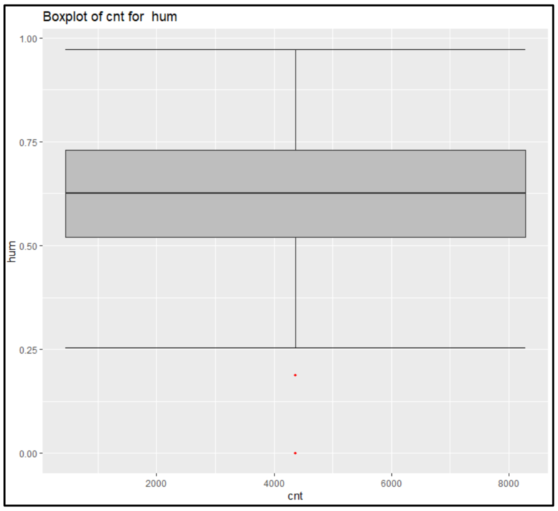


Fig 2.2 Boxplot of Hum

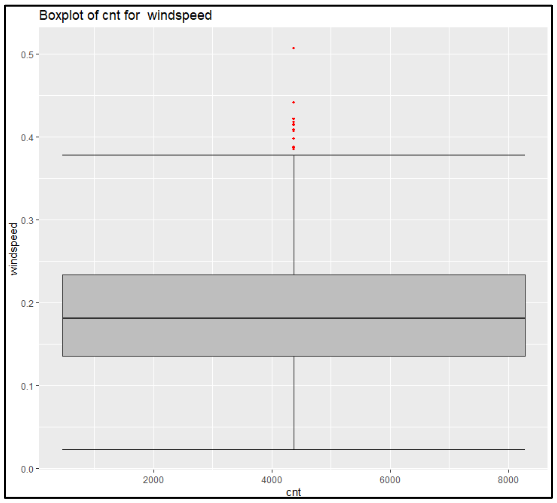


Fig 2.3 Boxplot of Windspeed

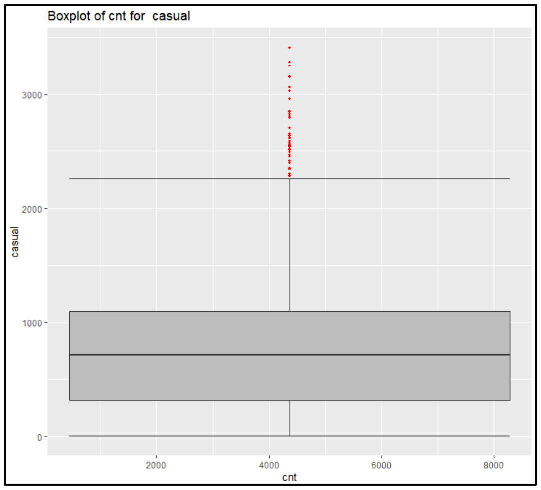
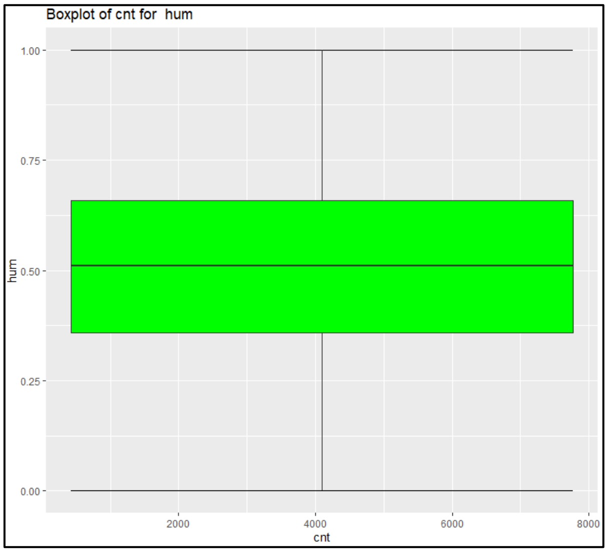


Fig 2.4 Boxplot of Casual

As we have successfully detected outliers in above variables in the dataset, we will now remove them from the data and re plot the graphs as below:



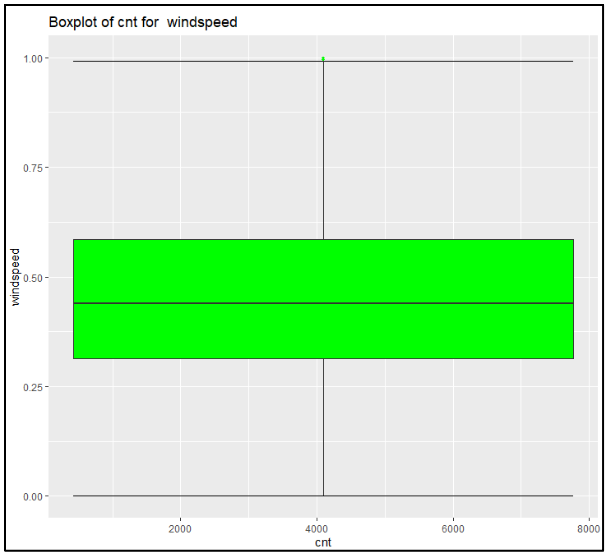


Fig 2.5 Boxplot of Hum

Fig 2.6 Boxplot of Windspeed

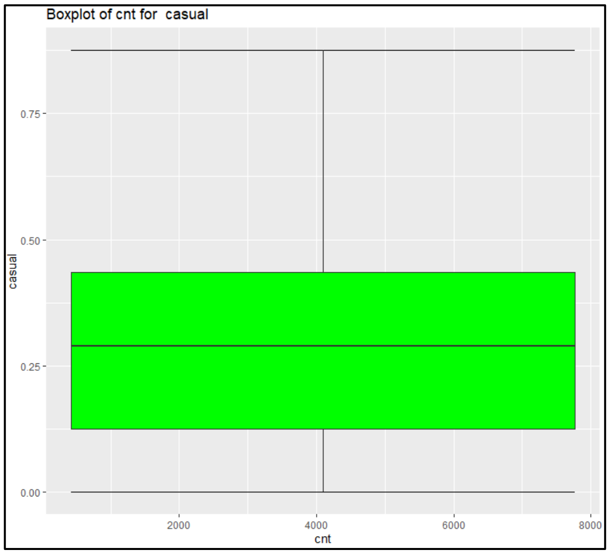


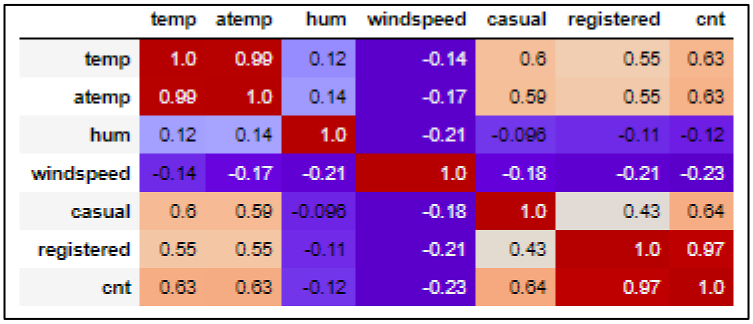
Fig 2.7 Boxplot of Casual

Our data is now free from outliers and we can proceed to next step that is feature selection.

2.1.3 Feature Selection

Before running the any machine learning model, we need to check the variables which are used for the prediction. It happens that sometimes most of the variables in the dataset do not contribute much in the prediction of target or dependent data. In some cases, it can also happen two or more data may pass the similar information to the model which also may increase in the complexity of the model. In order to avoid these issues, we do feature selection and select variables which are contributing in the prediction of dependent variable.

We will plot the correlation matrix between all the numeric variables to understand the relation between them as shown below:



	temp	atemp	hum	windspeed	casual	registered	cnt
temp	1.0	0.99	0.12	-0.14	0.6	0.55	0.63
atemp	0.99	1.0	0.14	-0.17	0.59	0.55	0.63
hum	0.12	0.14	1.0	-0.21	-0.096	-0.11	-0.12
windspeed	-0.14	-0.17	-0.21	1.0	-0.18	-0.21	-0.23
casual	0.6	0.59	-0.096	-0.18	1.0	0.43	0.64
registered	0.55	0.55	-0.11	-0.21	0.43	1.0	0.97
cnt	0.63	0.63	-0.12	-0.23	0.64	0.97	1.0

Fig 2.8 Correlation Matrix (Numerical Variables)

From the above matrix plotted amongst the numerical variables in the dataset we can see that independent variables temp and atemp are positively correlated and carry similar information so we will proceed with only one variable amongst temp and atemp.

Another way of checking feature is using Seaborn library which is well known visualization library. Seaborn has a defined function which is known as *pairplot()* using which we can plot multiple variables plot together and identify their trend. Below is the pairplot for bike rental dataset. As also seen in below figure we can look at the temp feature and atemp feature they are containing almost similar data as we checked earlier from the correlation matrix (Fig 2.8). Therefore, we can exclude any one feature from temp and atemp. In our prediction we will be considering the temp variable.

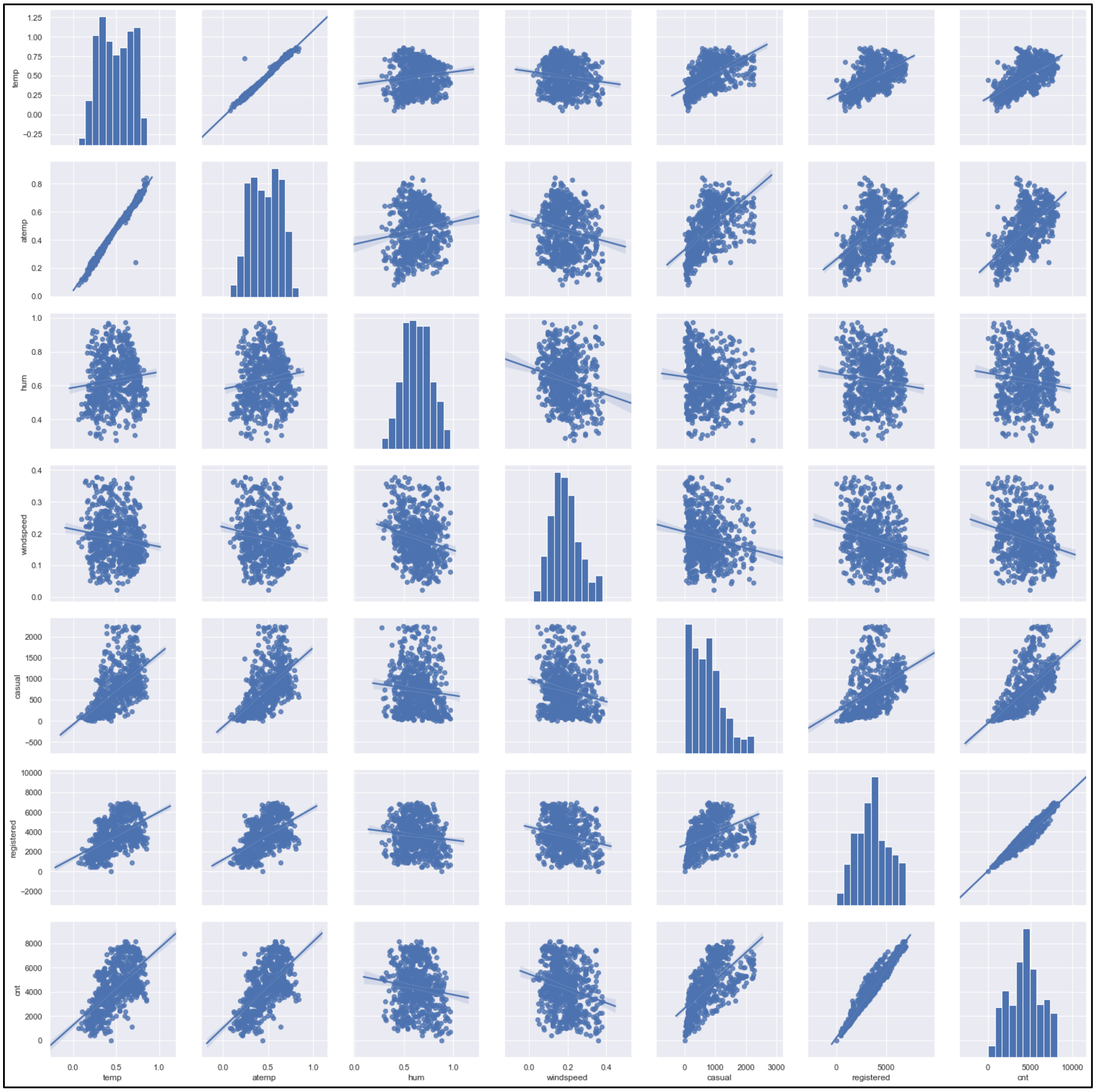
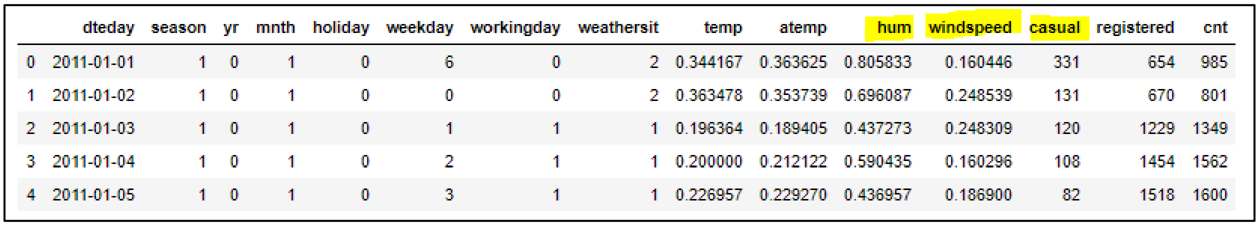


Fig 2.9 Continuous features comparison

2.1.4 Feature Scaling

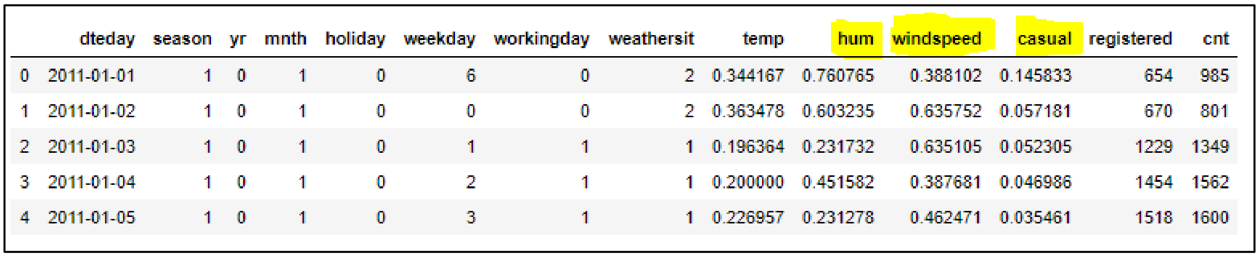
Feature scaling is done in order to remove the skewness of the data and make it normalized so that it will prevent biasness in the predicted model. Feature scaling is of two types normalization and standardization. Normalization is the process of reducing unwanted variation either within or between variables and it has a range between 0 to 1. Standardization technique is also known as Z-score. Z represents the data between raw score and population mean in the units of standard deviation. Z is negative when the raw score is below the mean and z is positive when above mean.

In our bike rental dataset, we found outliers in 3 variables and skewness as well so we will normalize the data since our data is not uniformly distributed.



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

Fig 2.10 Data before feature scaling



	dteday	season	yr	mnth	holiday	weekday	workingday	weathersit	temp	hum	windspeed	casual	registered	cnt
0	2011-01-01	1	0	1	0	6	0	2	0.344167	0.760765	0.388102	0.145833	654	985
1	2011-01-02	1	0	1	0	0	0	2	0.363478	0.603235	0.635752	0.057181	670	801
2	2011-01-03	1	0	1	0	1	1	1	0.196364	0.231732	0.635105	0.052305	1229	1349
3	2011-01-04	1	0	1	0	2	1	1	0.200000	0.451582	0.387681	0.046986	1454	1562
4	2011-01-05	1	0	1	0	3	1	1	0.226957	0.231278	0.462471	0.035461	1518	1600

Fig 2.11 Data After feature scaling

As you can see in Fig. 2.10 and 2.11 the features hum, windspeed and casual are normalized, and updated data has values ranging from 0 to 1.

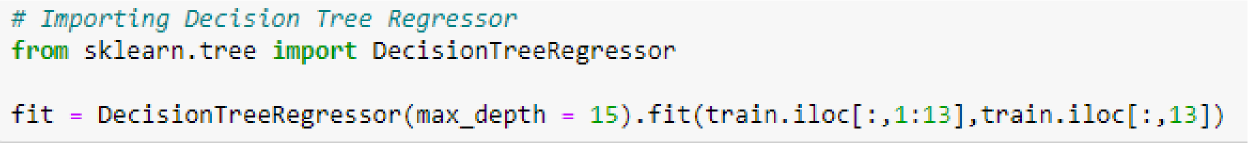
2.2 Modelling

Once we clean the data in the data exploration phase, we can proceed with the model development. After data exploration initial phase, we came to know few points about our dataset such as categorical variables and continuous variables, dependent and independent variables. We also know our target or dependent variable is continuous since we need to predict the number of daily bikes rented. Therefore, our problem is a regression problem. We will be using Decision tree regression, Random forest and Linear Regression algorithm.

2.2.1 Decision Tree Regression Algorithm

Decision tree is a predictive model based on branching series based on the Boolean tests. It can be used for both classification and regression problems. There are number of different types of decision trees that can be used in the machine learning algorithms. In machine learning algorithms in case of data mining we always must train the model using our data and post that we need to check accuracy of the model to predict the data. Therefore, we need to split our cleaned dataset into two parts train and test. From sk.learn.model_selection we can use the defined function train_test_split to do the required task. It is always better to divide the data into 80:20 which is train and test data respectively.

Below code in python is used to run the Decision tree algorithm on our bike renting dataset. Sklearn.tree is a package which has DecisionTreeRegressor function which we can use to implement the model. In next line of code we specify maximum depth as 15 wherein we inform the model that do not proceed beyond 15th level followed with the fit function which has independent variables and the dependent variable.



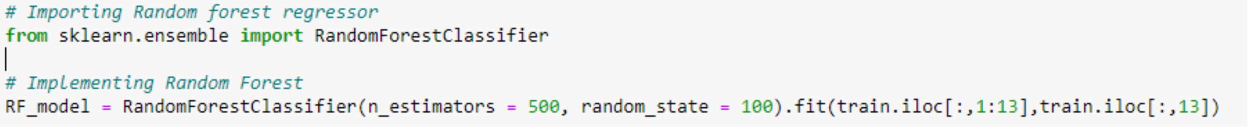
```
# Importing Decision Tree Regressor
from sklearn.tree import DecisionTreeRegressor

fit = DecisionTreeRegressor(max_depth = 15).fit(train.iloc[:,1:13],train.iloc[:,13])
```

Once the model is trained, we run it on the test data to check the predictions and compare the predicted data with the actual dependent variable.

2.2.2 Random Forest Algorithm

Algorithm uses the CART decision tree algorithm to generate the random forest. It is an ensemble which consists of many decision trees. Random forest uses the mean for regression problems, and it can be used for both classification and regression problems. Below is the Random forest model implementation python code. We can use the RandomForestClassifier function defined in the sklearn.ensemble library. In the function n_estimators is the number of trees to be used in the forest. A sub optimal greedy algorithm is repeated several times using random selections of features and samples. The random_state parameter allows controlling these random choices. Followed with these we need to provide the independent and dependent variable in the fit function to train the model.



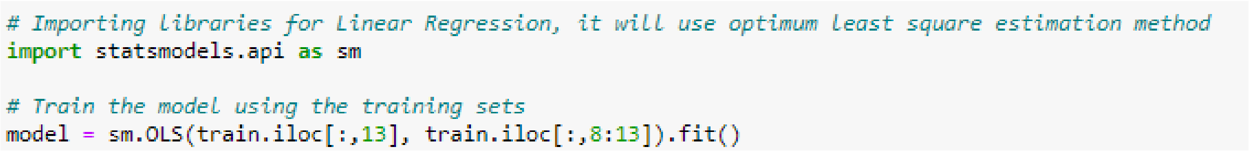
```
# Importing Random forest regressor
from sklearn.ensemble import RandomForestClassifier

# Implementing Random Forest
RF_model = RandomForestClassifier(n_estimators = 500, random_state = 100).fit(train.iloc[:,1:13],train.iloc[:,13])
```

Similarly, as in the decision tree algorithm we predict the data using fit.predict() function by passing the dependent variables. Finally, these predicted values are compared with actual target variable to check the accuracy of the model.

2.2.3 Linear Regression

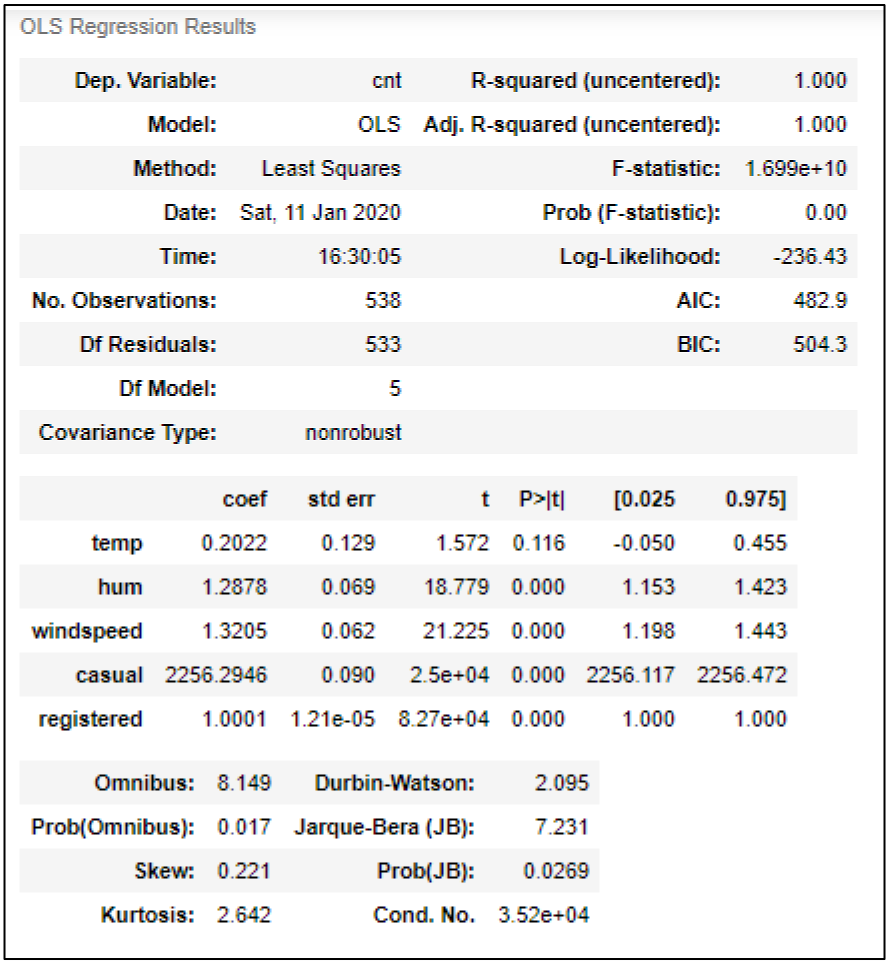
Linear regression is one of the statistical models and it can only be used for regression problems. In linear regression we have two types of model’s simple linear regression model and multiple linear regression model. We can use the statsmodels.api library which has the linear regression function OLS defined. Below is the code used in python for implementation of Linear Regression.



```
# Importing libraries for Linear Regression, it will use optimum least square estimation method
import statsmodels.api as sm

# Train the model using the training sets
model = sm.OLS(train.iloc[:,13], train.iloc[:,8:13]).fit()
```

Once the model is trained on train data we check the summary of the model using summary() function. We can test the model by predict() and passing independent continuous variables.



```
OLS Regression Results

     Dep. Variable:             cnt       R-squared (uncentered):      1.000
             Model:             OLS  Adj. R-squared (uncentered):      1.000
            Method:   Least Squares                  F-statistic:  1.699e+10
              Date: Sat, 11 Jan 2020           Prob (F-statistic):      0.00
              Time:        16:30:05               Log-Likelihood:    -236.43
 No. Observations:             538                          AIC:      482.9
      Df Residuals:             533                          BIC:      504.3
          Df Model:               5
  Covariance Type:       nonrobust

                 coef    std err          t   P>|t|    [0.025    0.975]
      temp     0.2022      0.129      1.572   0.116    -0.050     0.455
       hum     1.2878      0.069     18.779   0.000     1.153     1.423
 windspeed     1.3205      0.062     21.225   0.000     1.198     1.443
    casual  2256.2946      0.090    2.5e+04   0.000  2256.117  2256.472
registered     1.0001   1.21e-05   8.27e+04   0.000     1.000     1.000

       Omnibus:  8.149     Durbin-Watson:      2.095
Prob(Omnibus):  0.017  Jarque-Bera (JB):      7.231
          Skew:  0.221          Prob(JB):     0.0269
      Kurtosis:  2.642         Cond. No.   3.52e+04
```

Fig 2.10 Summary of Linear Regression Model

Chapter 3

Conclusion

3.1 Model Evaluation

After generating predictions from few models as above we need to finalize and select the best performing and efficient model. Mostly models are compared based on below criteria:

- Predictive Performance
- Interpretability
- Computational Efficiency

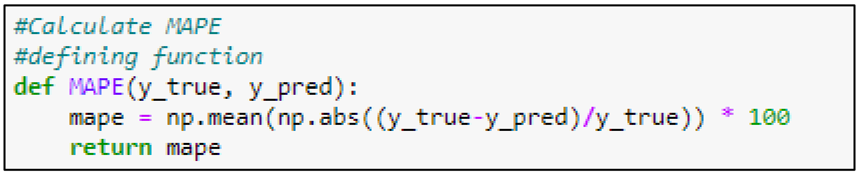
In our case of bike renting data since we have predicted the target variable, we will judge the models based on their predictive performance. Predictive performance can be measured using calculate average measures such as Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE), and Root Mean Squared Error (RMSE).

3.1.1 Mean Absolute Error (MAE)

MAE is the average of the absolute errors. We will not calculate this since people are more prone to understand data in terms of percentages instead of numbers.

3.1.2 Mean Absolute Percentage Error (MAPE)

MAPE is the percentage version of MAE and it measures accuracy as a percentage of error. Below is the code snippet from implementation using python.

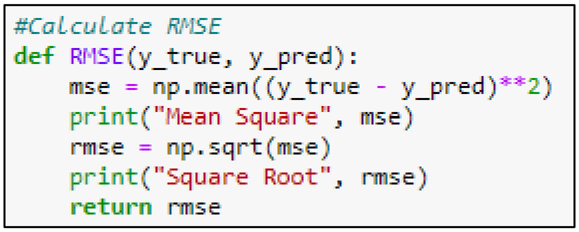


```
#Calculate MAPE
#defining function
def MAPE(y_true, y_pred):
    mape = np.mean(np.abs((y_true-y_pred)/y_true)) * 100
    return mape
```

Fig 3.1 MAPE function

3.1.2 Root Mean Square Error (RMSE)

RMSE is the squaring of errors, finding their averages and taking the square root. It is a time based measure. Below is the code snippet from implementation using python.

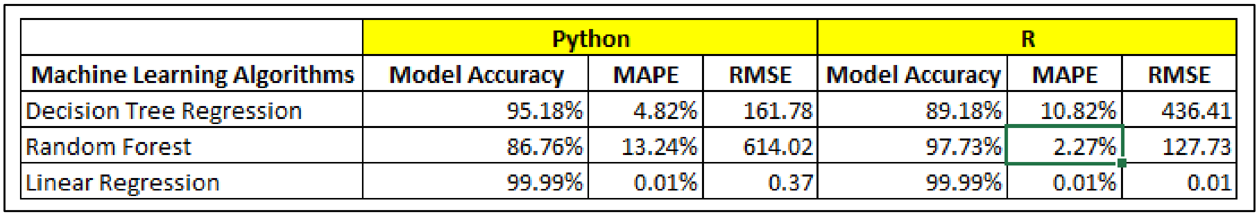


```
#Calculate RMSE
def RMSE(y_true, y_pred):
    mse = np.mean((y_true - y_pred)**2)
    print("Mean Square", mse)
    rmse = np.sqrt(mse)
    print("Square Root", rmse)
    return rmse
```

Fig 3.2 RMSE function

3.2 Model Selection

After running various model, we got some percentage errors on our data set. As we can see in the code implementation, below is the table of machine learning algorithm implemented with their MAPE and RMSE values. Percentages will be different in both Python and R language because we made a data split into train and test data randomly.



	Python			R		
Machine Learning Algorithms	Model Accuracy	MAPE	RMSE	Model Accuracy	MAPE	RMSE
Decision Tree Regression	95.18%	4.82%	161.78	89.18%	10.82%	436.41
Random Forest	86.76%	13.24%	614.02	97.73%	2.27%	127.73
Linear Regression	99.99%	0.01%	0.37	99.99%	0.01%	0.01

Fig 3.3 Output Values Comparison

It is clearly visible that **Linear Regression Model** is the best fit to predict the target variable in our data set since it can predict almost 99.99% data accurately.

References

Edwisor study materials

Machine Learning (in Python & R) dummies by John Paul Mueller & Luca Massaron

R and Python official documents