Bike Rental Count Prediction-Project Report

Navaneeth Kumar O M

13 March 2019

Contents

Chapter 1

Introduction

1.1 Problem statement

The objective of this case study is the prediction of daily bike rental count, based on the environmental and seasonal settings using analytics concepts.

1.2 Problem description

The objective of this case is to predict daily bike rental count. We are provided with a dataset Containing 731 observations, 15 predictor variables and 1 target variable. The predictors are describing the various environment factors like season, weather situation, temperature, humidity and windspeed. We must develop a machine learning model to predict the estimated count of the bikes being rented out on a particular day based on the environmental factors.

1.3 Data

The data set consists of 731 observations recorded over a period of 2 years, between 2011 and 2012. It has 15 predictors or independent variables and 1 target variable 'cnt'.
Below table 1.1 describes all the variables.

Table – 1.1 Description of the variables

Variable Names	Description
Instant	Record index
dteday	Date
season	Season (1:springer, 2:summer, 3:fall, 4:winter)
yr	Year (0: 2011, 1:2012)
mnth	Month (1 to 12)
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	1: Clear, Few clouds, Partly cloudy, Partly cloudy 2: Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist 3: Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds 4: Heavy Rain + Ice Pallets + Thunderstorm + Mist, Snow + Fog
temp	Normalized temperature in Celsius. The values are derived via (t-t_min)/(t_max-t_min), t_min=-8, t_max=+39 (only in hourly scale)

atemp	Normalized feeling temperature in Celsius. The values are derived via (t-t_min)/(t_maxt- t_min), t_min=-16, t_max=+50 (only in hourly scale)
hum	Normalized humidity. The values are divided to 100 (max)
windspeed	Normalized wind speed. The values are divided to 67 (max)
casual	Count of casual users
registered	Count of registered users
cnt	Count of total rental bikes including both casual and registered

The below tables 1.2 & 1.3 shows the first 5 rows of all the variables.

Table 1.2: Bike renting sample data (Columns: 1-8)

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

Table 1.3: Bike renting sample data (Columns: 9-16)

weathersit	temp	atemp	hum	windspeed	casual	registered	cnt
2	0.344167	0.363625	0.805833	0.160446	331	654	985
2	0.363478	0.353739	0.696087	0.248539	131	670	801
1	0.196364	0.189405	0.437273	0.248309	120	1229	1349
1	0.2	0.212122	0.590435	0.160296	108	1454	1562
1	0.226957	0.22927	0.436957	0.1869	82	1518	1600

Chapter 2

Methodology

2.1 Pre-Processing

Any predictive modelling requires that we look at the data before we start modelling. However, in data mining terms looking at data refers to so much more than just looking. Looking at data refers to exploring the data, cleaning the data as well as visualizing the data through graphs and plots. This is often called as **Exploratory Data Analysis**. To start this process, 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.

A Density Plot or kernel density plot visualises the distribution of data over a continuous interval or time period. This chart is a variation of a Histogram that uses kernel smoothing to plot values, allowing for smoother distributions by smoothing out the noise. The peaks of a density plot help display where values are concentrated over the interval.

An advantage that density plots have over Histograms is that they're better at determining the shape because they're not affected by the number of bins used (each bar used in a typical histogram). A Histogram comprising of only 4 bins wouldn't produce a distinguishable enough shape of distribution as a 20-bin Histogram would. However, with density plots, this isn't an issue.

In Figure 2.1 we have plotted the probability density functions of all the numeric variables which shows density of distribution of the respective variable. We see that most of the variables are similar to the normal distribution curve, and certain variables are skewed due to the presence of outliers which we will address in the coming sections.

The structure of the bike rental prediction data is as shown in Figure-2.2, The given data as a whole has 731 observations with 15 independent/predictor variables, and 1 target/dependent variable.

As we see that the some of the categorical independent variables have been given in terms of their level codes, we convert them to the factor datatype from int, the numeric variables ‘temp’, ‘atemp’, ‘hum’ and ‘windspeed’ are given in their normalized forms.

Figure 2.1: Probability Density Functions of the numeric variables

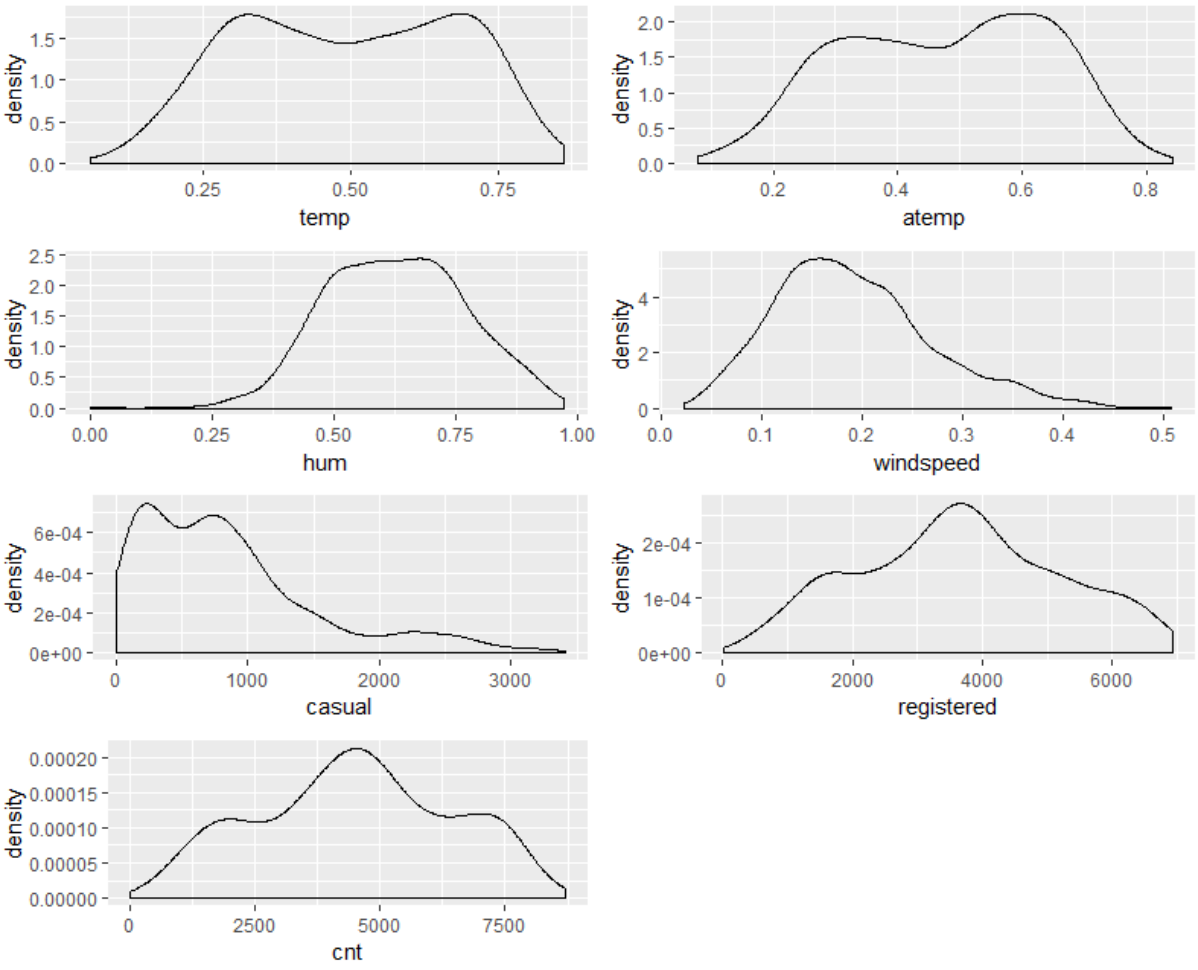
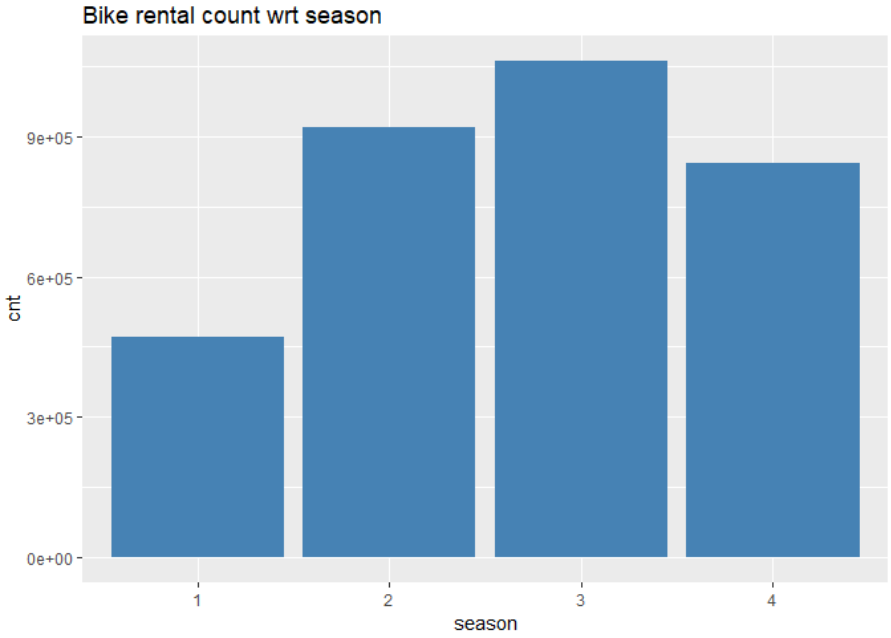


Figure 2.2: Structure overview of the churn reduction dataset.

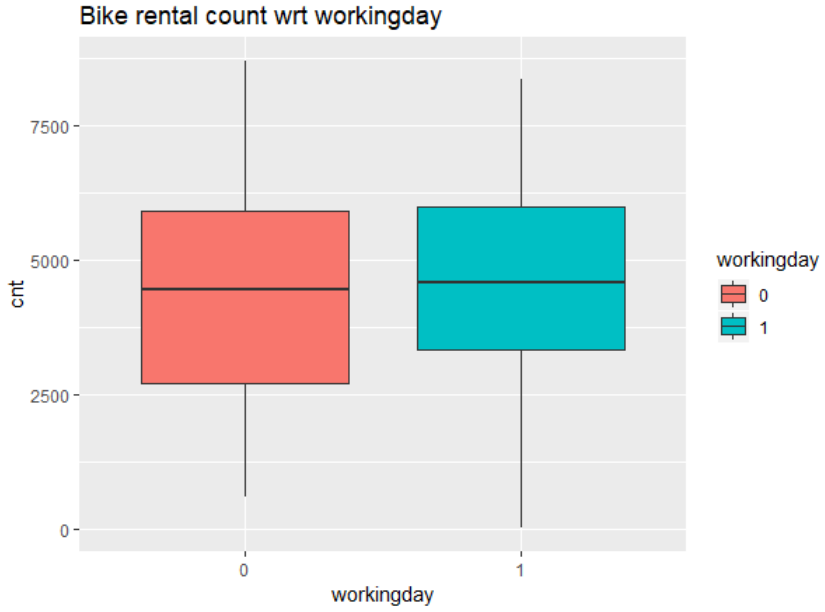
```
> str(data)
'data.frame':   731 obs. of  14 variables:
 $ season    : Factor w/ 4 levels "1","2","3","4": 1 1 1 1 1 1 1 1 1 1 ...
 $ yr        : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
 $ mnth      : Factor w/ 12 levels "1","2","3","4",..: 1 1 1 1 1 1 1 1 1 1 ...
 $ holiday   : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
 $ weekday   : Factor w/ 7 levels "0","1","2","3",..: 7 1 2 3 4 5 6 7 1 2 ...
 $ workingday: Factor w/ 2 levels "0","1": 1 1 2 2 2 2 2 1 1 2 ...
 $ weathersit: Factor w/ 3 levels "1","2","3": 2 2 1 1 1 1 2 2 1 1 ...
 $ temp      : num  0.344 0.363 0.196 0.2 0.227 ...
 $ atemp     : num  0.364 0.354 0.189 0.212 0.229 ...
 $ hum       : num  0.806 0.696 0.437 0.59 0.437 ...
 $ windspeed : num  0.16 0.249 0.248 0.16 0.187 ...
 $ casual    : int  331 131 120 108 82 88 148 68 54 41 ...
 $ registered: int  654 670 1229 1454 1518 1518 1362 891 768 1280 ...
 $ cnt       : int  985 801 1349 1562 1600 1606 1510 959 822 1321 ...
```

Below are the visualizations performed on the predictors variables v/s bike rental prediction :

- **Bike rental count based on season :** We see that most of the bike rentals happen during the fall season and least rental period is during the spring season.

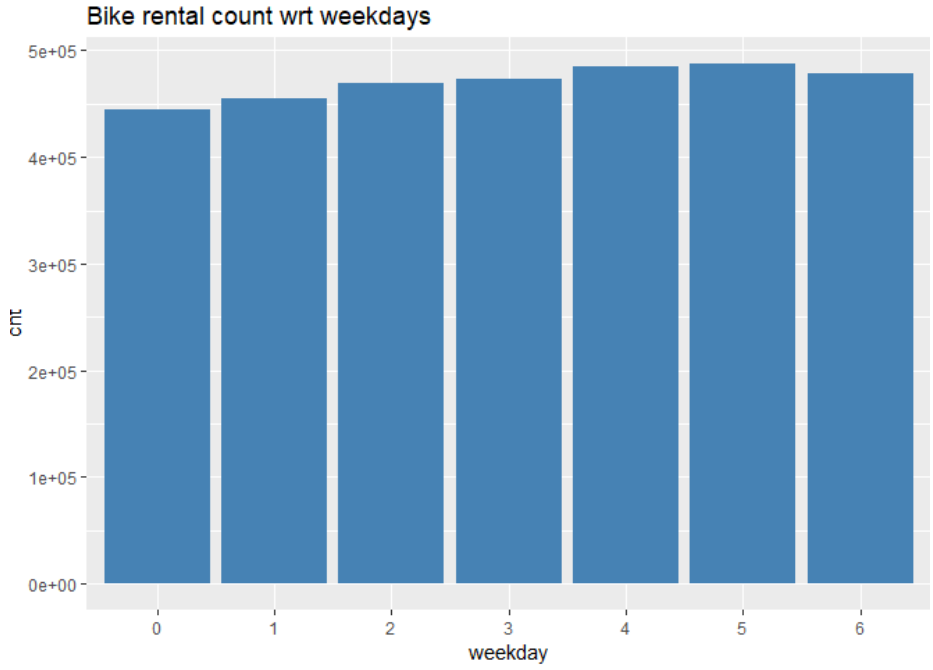


- **Bike rental count based on the working day :** We observe that most of the bike rentals occur on the non-working days than working days.



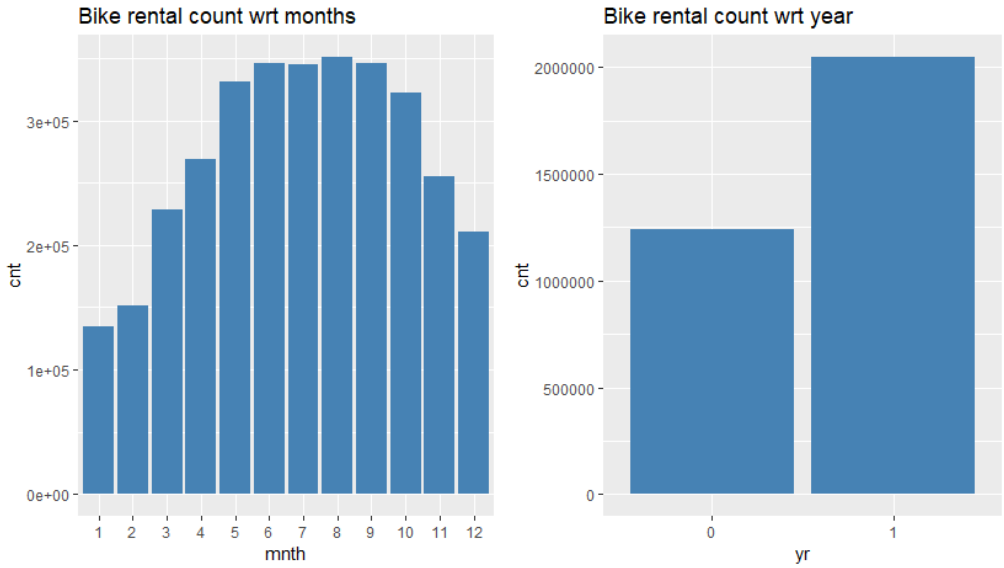
- **Bike rental count based on holidays :** We observe that most of the bike rentals happen during holidays which indicates the interest in the people opting to rent out the bikes for outing.

- **Bike rental count based on the weekdays :** We observe that bike rentals are high during the weekends, usually on Friday, Saturday and Sundays.

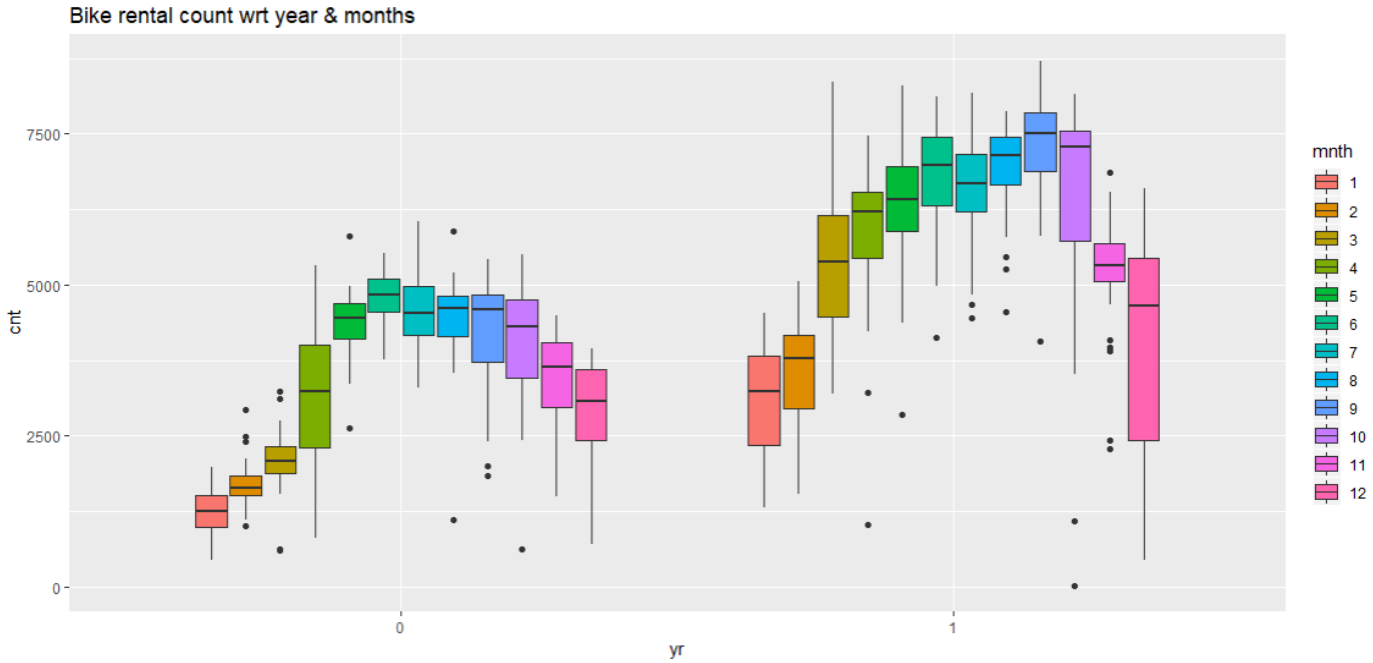


- **Bike rental count based on the months and year :** We observe that the bike rentals are high during the months of May, June, July, August and September(around fall season), with August month being the highest.

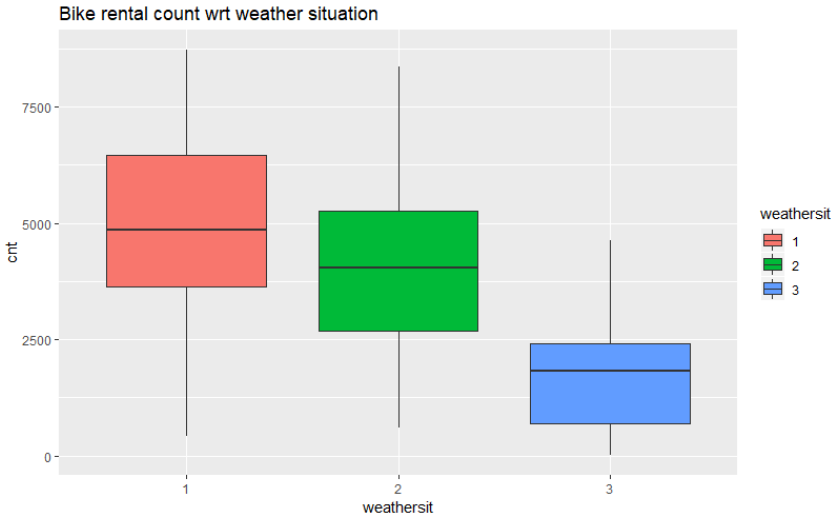
  Also, with respect to the year, we observe that the year 2012 has more rentals than the previous year 2011, which shows that there is an improvement in the business and quality in the service being provided.



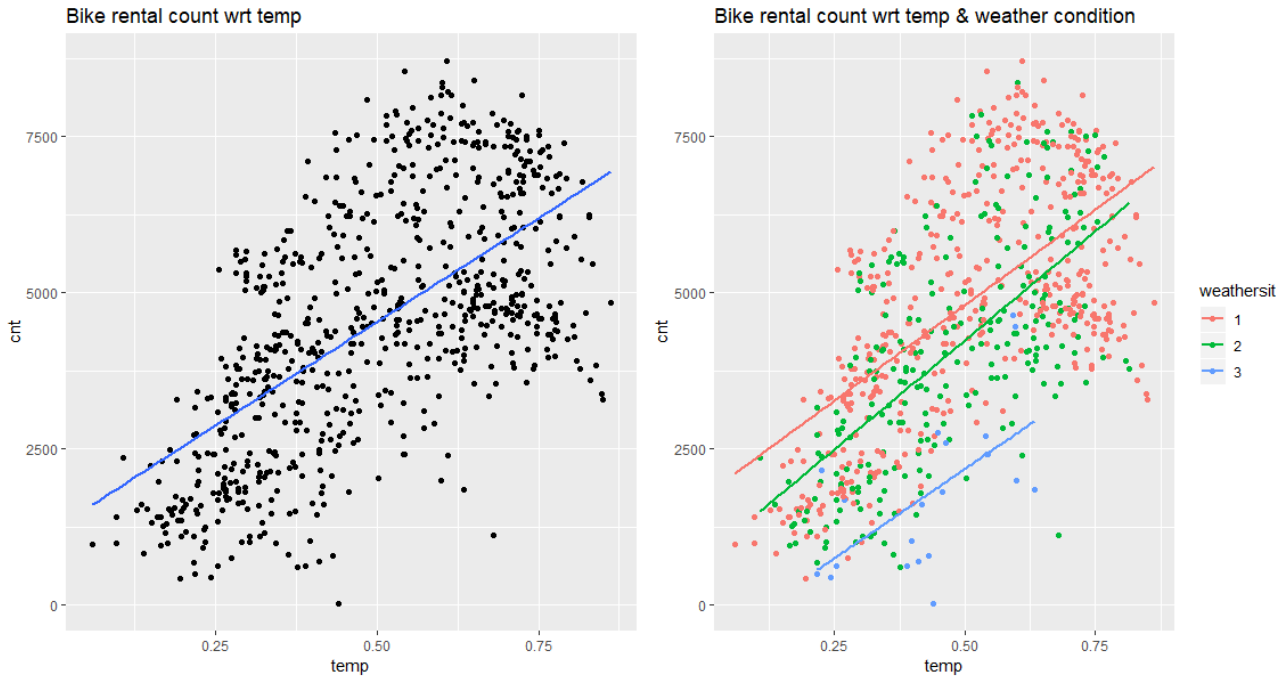
- **Bike rental count with respect to the year and its months :** We can observe that in the year 2011 the highest rental month was June with lowest rental month being January, and in the year 2012 the highest rental month was September with the lowest rental month being January.



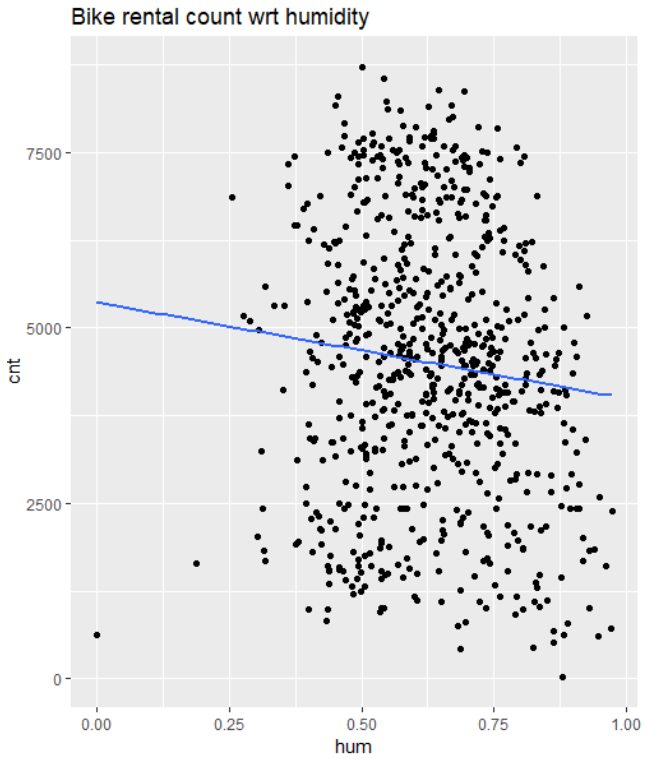
- **Bike rental count based on the weather situation :** We observe that the highest bike rentals happen when the weather is Clear, with few clouds, and partly cloudy. The least bike rental is when the weather is light snow, light rain, thunderstorm and scattered clouds.

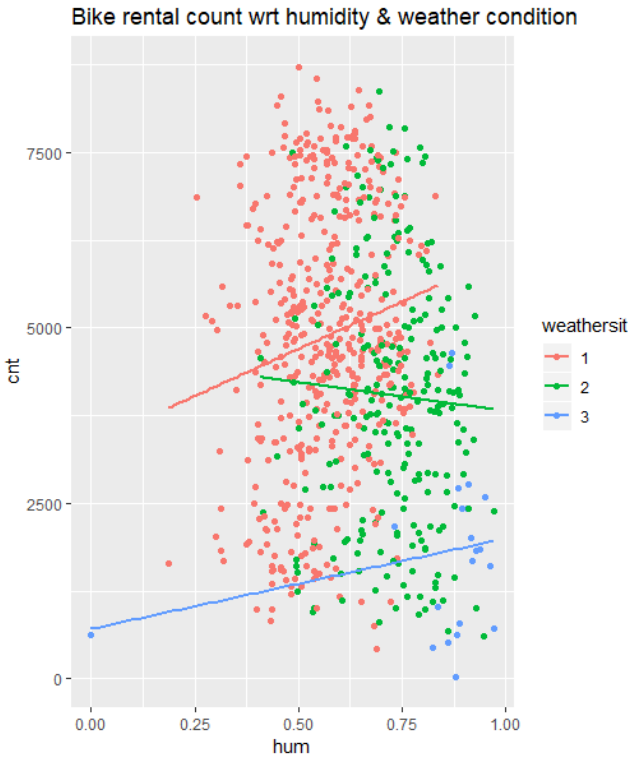


- **Bike rental count with respect to temperature** : We observe that the bike rental count is high when the temperature increases and they both show the linear relationship with one another.

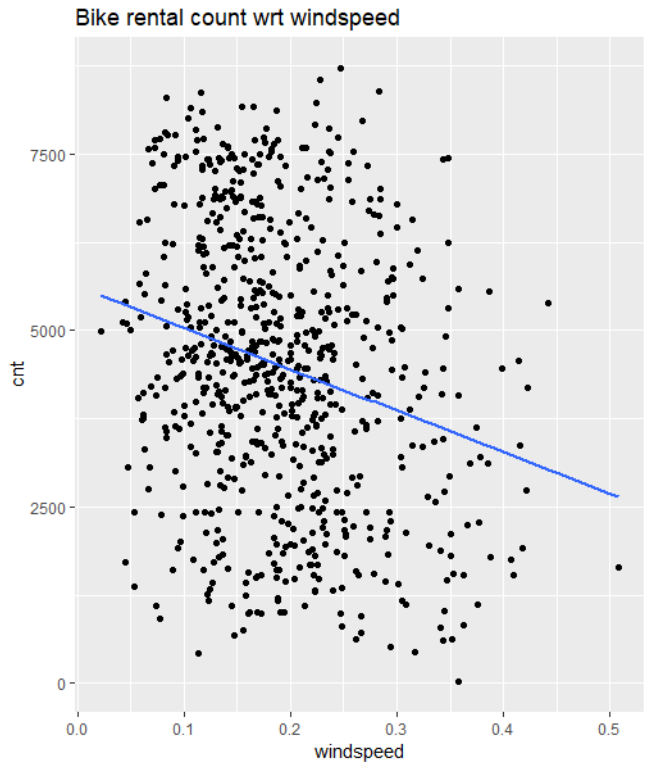


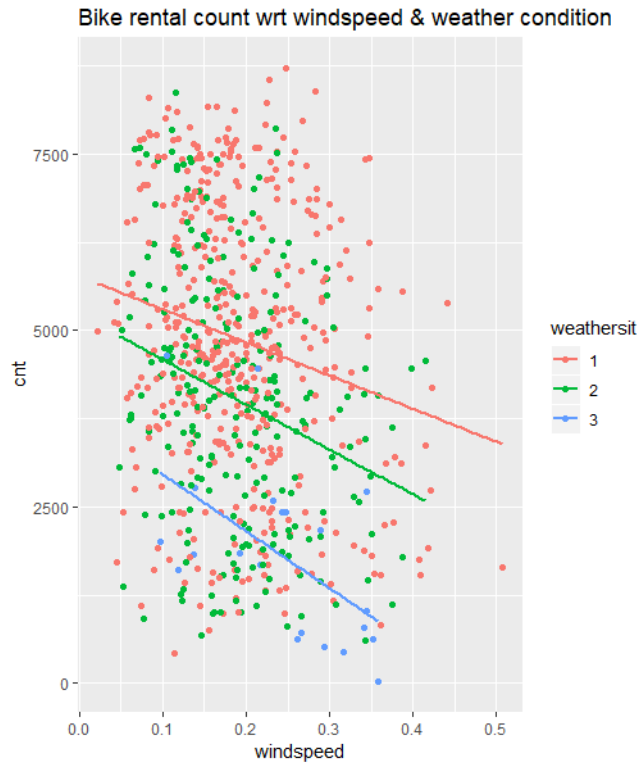
- **Bike rental count with respect to humidity :** We observe that the bike rental count decreases when the humidity increases and they both show negative linear relationship with each other. Also, when the weather condition is misty, cloudy with broken clouds, the humidity is high, and the bike rentals decrease.





- **Bike rental count with respect to windspeed :** We observe that the bike rental count decreases when the windspeed increases and they both show negative linear relationship with each other. Also, in any of the weather conditions when the windspeed is high, the bike rentals decrease.





2.1.1 Missing Value Analysis

Missing values are which, where the values are missing in an observation in the dataset. It can occur due to human errors, individuals refusing to answer while surveying, optional box in questionnaire.

Missing data mechanism is divided into 3 categories as below :

- **Missing Completely at Random (MCAR)**, means there is no relationship between the missingness of the data and any values, observed or missing. Those missing data points are a random subset of the data. There is nothing systematic going on that makes some data more likely to be missing than others.

- **Missing at Random (MAR)**, means there is a systematic relationship between the propensity of missing values and the observed data, but not the missing data. Whether an observation is missing has nothing to do with the missing values, but it does have to do with the values of an individual's observed variables. So, for example, if men are more likely to tell you their weight than women, weight is MAR.

- **Missing Not at Random** (**MNAR)**, means there is a relationship between the propensity of a value to be missing and its values. This is a case where the people with the lowest education are missing on education or the sickest people are most likely to drop out of the study. MNAR is called "non-ignorable" because the missing data mechanism itself must be modelled as we deal with the missing data.

Usually we only consider those variables for missing value imputation whose missing values is less than 30%, if it above this we will drop that variable in our analysis as imputing missing values which are more than 30% doesn't make any sense and the information would also be insensible to consider.

From the exploratory data analysis, we see that our bike rental prediction dataset doesn't have any missing values and hence we are not proceeding with this step to impute any missing values.

The below figure 2.3 shows the variables with percentage of missing values in them.

Figure 2.3: Percentage of missing values present in the dataset

	Columns	Missing_percentage
1	dteday	0
2	season	0
3	yr	0
4	mnth	0
5	holiday	0
6	weekday	0
7	workingday	0
8	weathersit	0
9	temp	0
10	atemp	0
11	hum	0
12	windspeed	0
13	casual	0
14	registered	0
15	cnt	0

2.1.2 Outlier Analysis

An Outlier is an observation which is inconsistent(or distant) with rest of the observations. The presence of outliers in the data adds to the skewness and this needs to be addressed. We have various methods to detect the outliers but for this dataset we are going to detect the outliers using Tukey's Boxplot method.

Tukey's Boxplot method or simply called boxplot method is a standardized way of displaying the distribution of the data based on the five-number summary, they are minimum, first quartile, median, third quartile and maximum. A segment inside the rectangle shows the median and "whiskers" above and below the box show the locations of the minimum and maximum.

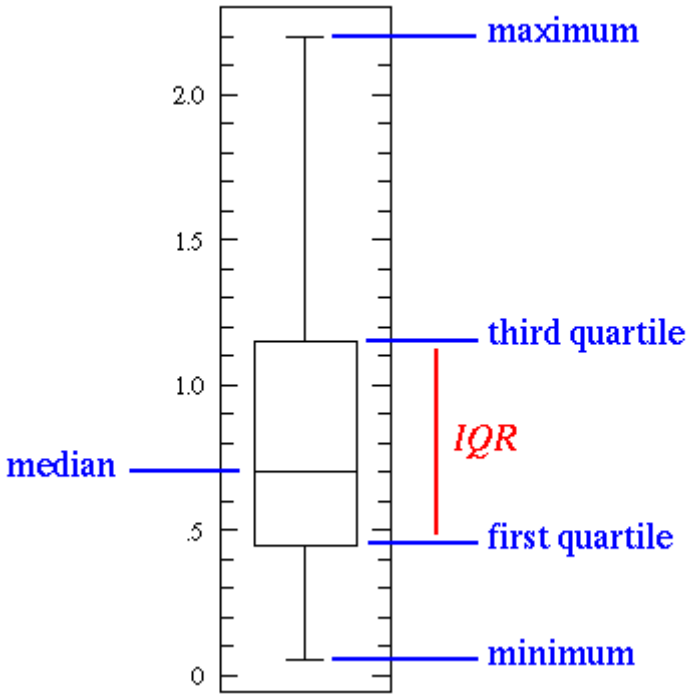


Figure 2.4 : Structure of a typical boxplot

The above Figure 2.4 shows the typical box plot. The interquartile range(IQR) is an area where the bulk of the majority values lie, in other words it is the difference between the third quartile(Q3) and first quartile(Q1). The maximum value of a box plot is 1.5 times the IQR beyond the third quartile, minimum value of a box plot is 1.5 times the IQR below the first quartile. Mathematically it is given as below :

Maximum value = Q3 + (1.5*IQR)
Minimum value = Q1 – (1.5*IQR)

The values which fall beyond the minimum and maximum values are considered to be as outliers. Hence, further we replace these outlier values with NA's and impute the same with KNN Imputation which estimates the NAs considering the mean K amount of surrounding values.

As the percentage of Outliers are very less, we are following the above method else the other way of imputing the outliers would be replacing the outliers falling beyond the minimum value with the minimum value and replacing the outliers falling beyond the maximum with the maximum value (doing so makes the data less vulnerable to the insensible information rather imputing with NAs)

In figure 2.5 we have plotted the boxplots for the 6 continuous variables. The red dots indicate the outliers which are the extreme values after minimum/maximum points and the figure 2.6 shows the boxplot after the outlier treatment with KNN imputation, we could see that the outliers have been reduced very much and our data is now free from the outliers.

We are fixing outliers only for 'hum' and 'windspeed' variables as 'casual' variable is just the count of casual bikes rent and this cannot be categorised as an outlier.

Figure 2.5 : Boxplot of continuous variables with outliers

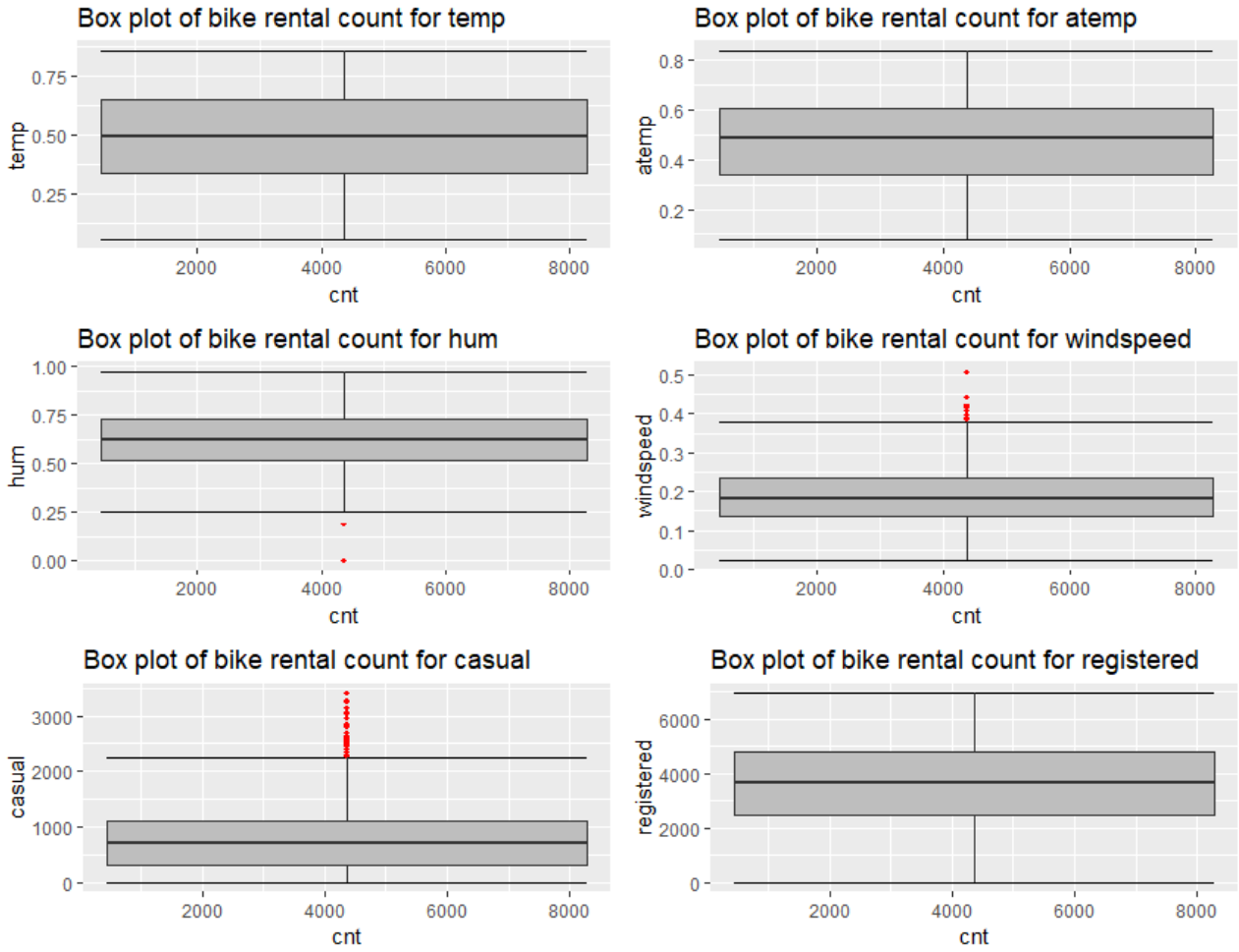
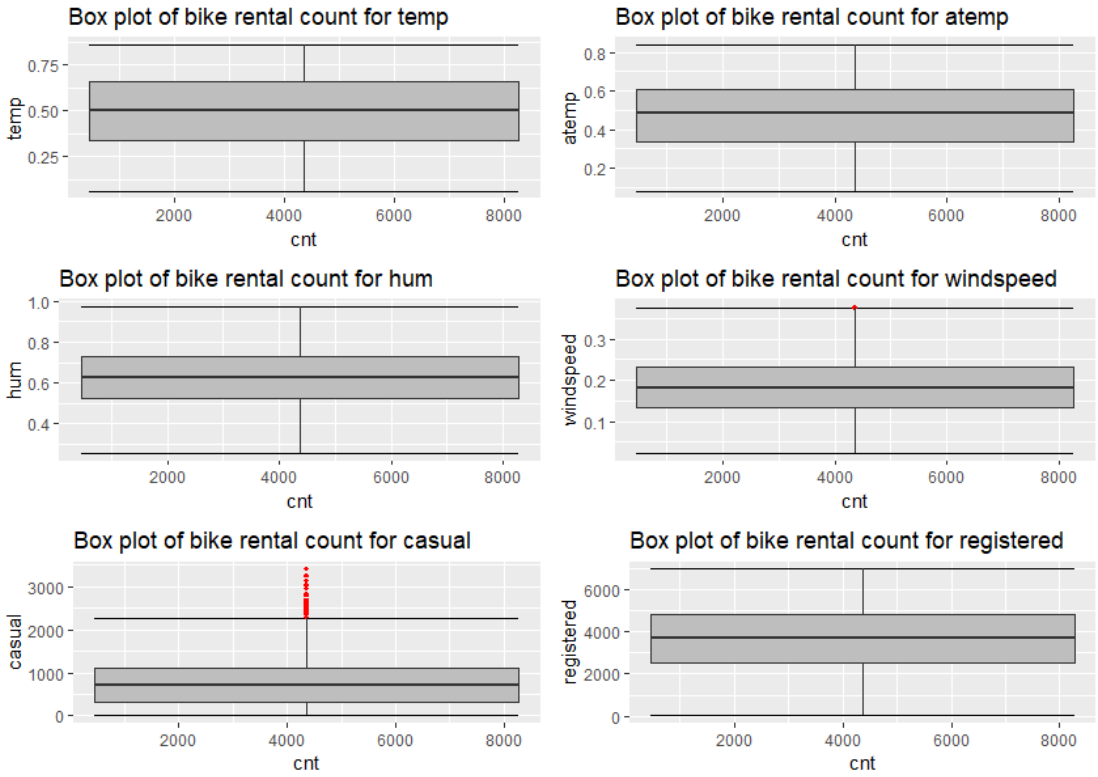


Figure 2.5 : Boxplot of continuous variables after outlier treatment for ‘hum’ & ‘windspeed’



2.1.3 Feature Selection

Before performing any type of modelling we need to assess the importance of each predictor variable in our analysis. There is a possibility that many variables in our analysis are not important at all to the problem of class prediction. This process of selecting a subset of relevant features/variables is known as feature selection. There are several methods of doing feature selection. We have used correlation analysis for continuous variables and Chi-square test/ANOVA for categorical variables.

Correlation Analysis :

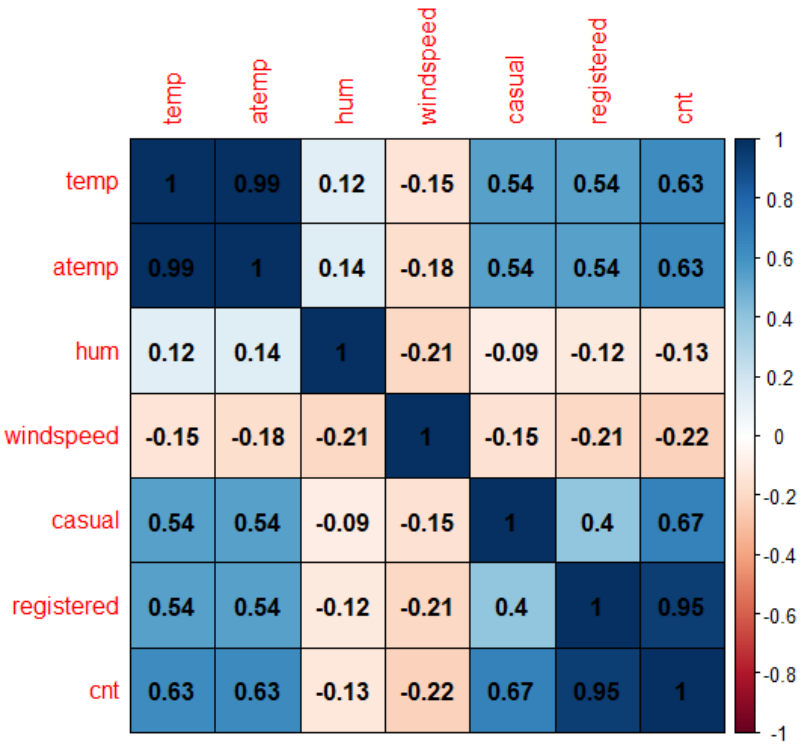


Figure 2.6 : Correlation plot before feature selection

The above Figure 2.6 is a correlation plot for continuous variables in bike rental dataset, it shows the correlation between two variables. The blue shade colour implies that two variables are positively correlated with each other and pinkish red colour implies that the variables are negatively correlated with each other.
By observing the heap map/correlation plot pattern we can find the variables that are highly correlated with any other variables and remove such variables which may lead to multicollinearity . Below are the observations made with respect to the highly correlated variables:

- Variable 'temp' is highly positively correlated with variable 'atemp' and we are dropping atemp variable.
- Variable 'casual' and 'registered' also are positively correlated and they are not considered for model building as we have the total count of these two variables as 'cnt'.
- Variable 'windspeed' has negative correlation with 'cnt' variable.

The below figure 2.7 shows the correlation plot after removing the highly correlated variables and also, we have removed the casual & registered variables.

We could now observe from the below correlation plot that our dataset is free from multicollinearity effect.

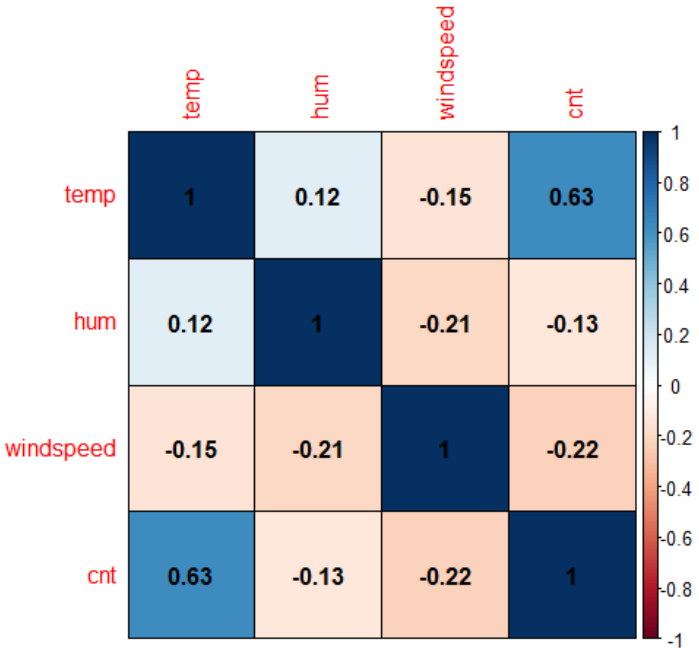


Figure 2.7 : Correlation plot after feature selection

Chi-Square Analysis or Chi-Square test of Independence :

Chi-square test of independence is used to check the dependency between the two categorical variables. The dependency between 2 independent variables must be low and dependency between independent & dependent variables must be high.

The Chi-square test uses contingency table to establish the relation between 2 categorical variables and is interpreted based on the p-value(probability value).

The null and alternate hypothesis in Chi-square test is given as :

- Null hypothesis : The 2 variables are independent.
- Alternate hypothesis : The 2 variables are not independent.

The Chi-square test is calculated by the formula :

$$c^2 = \sum_{i=1}^{k} \left[\frac{(O_i - E_i)^2}{E_i} \right]$$

Where Oi = Observed value & Ei = Expected value

In our Chi-Square test we are comparing each of the independent categorical variables with the other independent categorical variables, in other words we take each independent variable as reference and analyse the Chi-square test with other independent variables.

For each test performed, the p-value is observed and,

- If the obtained p-value < 0.05, then we reject the null hypothesis concluding that there is a dependency between the 2 independent variables.
- If the obtained p-value > 0.05, then we accept the null hypothesis concluding that 2 independent variables have no dependency on each other, and they are independent.

The below table 2.1 shows the p-values noted for each of the independent variables when compared with other independent variables during the Chi-Square test analysis.

	season	yr	mnth	holiday	weekday	workingday	weathersit
season	2.20E-16	0.9999	2.20E-16	0.6832	1	0.8866	0.02118
yr	0.9999	2.20E-16	1	1	1	1	0.1274
mnth	2.20E-16	1	2.20E-16	0.5593	1	0.9933	0.01464
holiday	0.6832	1	0.5593	2.20E-16	8.57E-11	4.03E-11	0.6009
weekday	1	1	1	8.57E-11	2.20E-16	2.20E-16	0.2785
workingday	0.8866	1	0.9933	4.03E-11	2.20E-16	2.20E-16	0.2538
weathersit	0.02118	0.1274	0.01464	0.6009	0.2785	0.2538	2.20E-16

Table 2.1 : p-values of Chi -Square test

From the above table we see that the variables holiday & weekday have more p-values < 0.05, and hence further we are performing ANOVA test to decide which of these 2 variables to drop.

Anova Test :

Analysis of variance (ANOVA) is a statistical technique that is used to check if the means of two or more groups in a categorical variable are significantly different from each other or not. In other words, anova is used to test the significance of categorical variables with respect to the continuous target variable.

The null and alternate hypothesis are :

- Null hypothesis (H0) : The means between the groups are equal/similar (p-value > 0.05)
- Alternate hypothesis (H1) : The means between the groups are different (p-value < 0.05)

The below table 2.2 shows the p-values for all the categorical variable when compared with continuous target variable.

Table-2.2 : Output of ANOVA test in terms of p-value

Variable names	p-value
season	2e-16 ***
yr	2e-16 ***
mnth	2e-16 ***
holiday	0.0648 .
weekday	**0.58**
workingday	0.0985 .
weathersit	2e-16 ***

From the above table we see that the variable 'weekday' has the highest p-value of 0.5 which is very much greater than the threshold p-value of 0.05. Hence, we drop the weekday variable as it has no significance to explain the target variable.

2.1.4 Feature Scaling

Data scaling or feature scaling is a method used to standardize the range of variables or features present in the dataset so that they can be compared on a common ground.

Since the range of values for some variables in the raw data vary highly in magnitudes, units and range, we need feature scaling to bring all the features/variables to the same level of magnitudes, else the whole output of our analysis may get biased to one of the variables.

Most of the machine learning algorithms which use distance-based calculation might go wrong in their calculations if we do not scale our variables in the dataset before feeding into the model.

Another reason why feature scaling is applied is that gradient descent converges much faster with feature scaling than without it.

Normalization is a scaling method in which all the variables is brought into proportion with one another with values ranging from 0 to 1.

Normalization is given by:

$$Value_{Norm} = (Value - MinValue)/ (MaxValue - MinValue)$$

Where Minvalue and MaxValue are the minimum & maximum values of a given variable respectively.

As observed in the exploratory data analysis stage that our dataset is already normalized, we are good with the dataset and moving ahead with the modelling steps.

2.2 Modelling

2.2.1 Preparing the data for modelling

In order for the machine learning algorithms to understand the data, it is necessary that we modify our data accordingly in a format which the machine learning algorithms can interpret. Hence, we have to change the data of categorical variables to numerical data by assigning levels to the corresponding categories in a categorical variable.

From the exploratory data analysis, we observed that the categorical variables have been assigned with the respective numeric levels.

We are splitting 70% of the given dataset into train and the remaining 30% into test. Further, we are validating the model using 10-fold cross validation.

2.2.2 Model Selection

In our dataset, the target variable to be predicted is 'cnt', and we need to predict the count of bike rentals. Hence, this is a regression problem and we will build the following regression models and later finalize a particular model which would best-fit our criteria.

- Decision Tree model
- Random Forest model
- Logistic Regression

2.2.3 Decision Tree Model

Decision tree is a supervised predictive model based on the branching series of Boolean tests. Decision tree generates a series of rules which it makes use for its prediction and in the tree, each branch connects the nodes with "and" and multiple branches are connected by "or".

The model that we are building makes use of information gain to identify the root node and proceed further to build the rules.

The root node of this decision tree would be that independent variable for which the information gain is high, or the entropy is low. The node of a tree corresponds to an attribute/variable and each leaf node corresponds to a class label/predicted value.

The below figure-2.8 shows the validation RMSE, R-square and MAE values for 10-fold cross validation. The decision tree is developed for the different values of complexity parameter. The complexity parameter (cp) is used to control the size of the decision tree and to select the optimal tree size. If the cost of adding another variable to the decision tree from the current node is above the value of cp, then tree building does not continue. We could also say that tree construction does not continue unless it would decrease the overall lack of fit by a factor of cp. Setting cp to zero will build a tree to its maximum depth. We see that for our dataset, the lowest RMSE was produced when cp=0.

Figure-2.8 : 10-fold cross validation results of decision tree model

```
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 463, 464, 463, 463, 464, 463, ...
Resampling results across tuning parameters:

  cp            RMSE      Rsquared   MAE
  0.0000000000  876.2096  0.7996646  626.7108
  0.0003593751  878.5102  0.7987033  632.0277
  0.0004168285  880.1828  0.7980018  632.7116
  0.0004812352  882.6942  0.7971267  633.7119
  0.0011145360  882.9795  0.7963766  636.7127
  0.0056360034  911.5757  0.7828073  660.0251
  0.0085121890  951.7229  0.7654318  702.6840
  0.0128596757  985.7272  0.7454655  726.7120
  0.0134436256  991.7765  0.7430943  734.4426

RMSE was used to select the optimal model using the smallest value.
The final value used for the model was cp = 0.
```

2.2.4 Random Forest Model

Random forest algorithm builds N number of decision trees like a forest by selecting a random number of initial variables and observations. It is an ensemble model that consists of multiple decision trees.

This ensemble technique improves the accuracy and reduces the misclassification error due to the weak learners by combining multiple decision tress back to back which then prepares a strong classifier/regressor. In other words, Random forest is a combination of weak learners to produce a strong learner from its model.

The Random forest algorithm uses CART(classification & regression tree) which in turn makes use of Gini index to build all the decision trees internally. To classify a particular class the mode of all the classes generated by all the individual trees are considered for classification and mean of all the individual trees are considered for regression.

The below figure-2.9 shows the validation RMSE, R-square and MAE values for 10-fold cross validation performed on different values of mtry parameter. mtry refers to how many variables we should select at a node split, the default value is p/3 for regression and sqrt(p) for classification. We should always try to avoid using smaller values of mtry to avoid overfitting.

Figure-2.9 : 10-fold cross validation results of Random Forest model

```
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 463, 463, 465, 464, 464, 464, ...
Resampling results across tuning parameters:

  mtry  RMSE       Rsquared   MAE
   1    1452.1937  0.7555062  1188.4385
   3     856.6152  0.8535612   677.0590
   9     708.1844  0.8684675   509.9892
  10     709.1425  0.8667232   509.9663
  12     715.0011  0.8634422   513.9678
  13     715.8533  0.8628463   511.8213
  16     723.4910  0.8593387   512.6729

RMSE was used to select the optimal model using the smallest value.
The final value used for the model was mtry = 9.
```

2.2.5 Linear Regression

The linear regression model calculates the prediction value using a gradient descent algorithm in the background. This algorithm estimates the optimum co-efficient & intercept for different predictor variables such that the sum of the squared errors or cost function is minimum.

The sign of each coefficient indicates the direction of the relationship between a predictor variable and the dependent/response variable. A positive sign indicates that, as the predictor variable increases, the response variable also increases. A negative sign indicates that as the as the predictor variable increases, the response variable decreases.

The regression coefficient value represents the mean change in the response variable (Y) given a one-unit change for the corresponding predictor variable when theother predictor variables are held constant.

The below figure-3.0 shows the validation RMSE, R-square and MAE values for 10-fold cross validation performed for the linear regression model.

Figure-3.0 : 10-fold cross validation results of Linear Regression model

```
> print(lm_model)
Linear Regression

515 samples
  9 predictor

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 464, 463, 463, 463, 464, 463, ...
Resampling results:

  RMSE      Rsquared   MAE
  835.0623  0.8209737  617.5492

Tuning parameter 'intercept' was held constant at a value of TRUE
```

Chapter 3

Conclusion

3.1 Model Evaluation

Now that we have built a few models for predicting the target variable, we need to decide which one of the models to be chosen. There are several criteria that exists for evaluating and comparing the models. We can compare the models using any of the following criteria :

1.) Predictive performance
2.) Interpretability
3.) Computational efficiency

In our case, we are using predictive performance to decide and compare the models as the other two criteria do not hold much significance.

Predictive performance can be measured by comparing the predictions of the models with the real values of the target test variable, and by analysing various metrics.

Predictive regression performance is usually measured by MAPE and RMSE metrics.

3.1.1 Mean absolute percentage error (MAPE)

Mean absolute percentage error is the percentage equivalent of mean absolute error and it says how far the model's predictions are off from their corresponding outputs on average.

MAPE is given by,

$$\text{MAPE} = \frac{1}{N}\sum_{t=1}^{N}\frac{ABS(Actual_t - Forecast_t)}{Actual_t} * 100\%$$

The drawback of MAPE is, if the actual value is zero then the division is undefined. So, this metric can't help in such cases.

3.1.2 Root mean square error (RMSE)

Root mean square error is another metric to estimate the regression models, it is nothing but the root taken from the averaged square residuals. In other words, RMSE gives the average error of the model's prediction.

RMSE is given by,

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} (Predicted_i - Actual_i)^2}{N}}$$

3.2 Model Selection

As it can be seen from the below table 3.1 and table 3.2 that Random Forest model has the lowest RMSE score in both R & Python compared to other models, and Linear Regression model falls behind the random forest model with respect to the RMSE score in R. Hence, we finalize with Random Forest model as the best choice of our prediction.

- **R Implementation Output:**

Table-3.1 : R models and its related metrics

Model	Cross validated R^2	Cross validated RMSE	Test MAPE	Test RMSE
Decision Tree	0.7996	876.2096	23.0673	885.047
Random Forest	**0.8684**	**708.1844**	**18.2325**	**684.0097**
Linear Regression	0.8209	835.0623	16.1991	727.8427

- **Python Implementation Output :**

Table-3.2 : Python models and its related metrics

Model	Cross validated R^2	Test MAPE	Test RMSE
Decision Tree	0.7939	20.5027	880.3916
Random Forest	**0.8720**	**18.7033**	**759.8633**
Linear Regression	0.7937	20.9786	969.2466

Appendix – A

Example of output with a sample input :

Below are the R code output screenshots captured for the input data of Bike rental count prediction project.

The graphs are same as attached in the Chapter-1 and hence I'm not attaching them here.

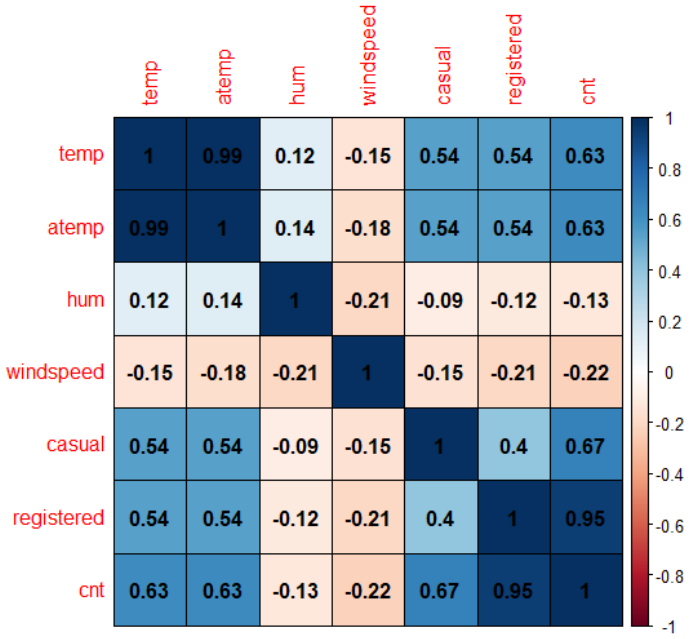
- Structure of whole dataset after changing the required parameters to factor

```
> str(data)
'data.frame':   731 obs. of  16 variables:
 $ instant   : int  1 2 3 4 5 6 7 8 9 10 ...
 $ dteday    : Factor w/ 731 levels "2011-01-01","2011-01-02",..: 1 2 3 4 5 6 7 8 9 10 ..
 $ season    : Factor w/ 4 levels "1","2","3","4": 1 1 1 1 1 1 1 1 1 1 ...
 $ yr        : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
 $ mnth      : Factor w/ 12 levels "1","2","3","4",..: 1 1 1 1 1 1 1 1 1 1 ...
 $ holiday   : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
 $ weekday   : Factor w/ 7 levels "0","1","2","3",..: 7 1 2 3 4 5 6 7 1 2 ...
 $ workingday: Factor w/ 2 levels "0","1": 1 1 2 2 2 2 2 1 1 2 ...
 $ weathersit: Factor w/ 3 levels "1","2","3": 2 2 1 1 1 1 2 2 1 1 ...
 $ temp      : num  0.344 0.363 0.196 0.2 0.227 ...
 $ atemp     : num  0.364 0.354 0.189 0.212 0.229 ...
 $ hum       : num  0.806 0.696 0.437 0.59 0.437 ...
 $ windspeed : num  0.16 0.249 0.248 0.16 0.187 ...
 $ casual    : int  331 131 120 108 82 88 148 68 54 41 ...
 $ registered: int  654 670 1229 1454 1518 1518 1362 891 768 1280 ...
 $ cnt       : int  985 801 1349 1562 1600 1606 1510 959 822 1321 ...
```

- Missing_val dataframe info :

	Columns	Missing_percentage
1	season	0
2	yr	0
3	mnth	0
4	holiday	0
5	weekday	0
6	workingday	0
7	weathersit	0
8	temp	0
9	atemp	0
10	hum	0
11	windspeed	0
12	casual	0
13	registered	0
14	cnt	0

- Correlation plot :



- **ANOVA test analysis output :**

```
> summary(anova_season)
                  Df    Sum Sq   Mean Sq F value Pr(>F)
factor_data[, i]   3 9.506e+08 316865289   128.8 <2e-16 ***
Residuals        727 1.789e+09   2460715
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
> summary(anova_yr)
                  Df    Sum Sq   Mean Sq F value Pr(>F)
factor_data[, i]   1 8.798e+08 879828893   344.9 <2e-16 ***
Residuals        729 1.860e+09   2551038
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
> summary(anova_mnth)
                  Df    Sum Sq  Mean Sq F value Pr(>F)
factor_data[, i]  11 1.070e+09 97290206    41.9 <2e-16 ***
Residuals        719 1.669e+09  2321757
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
> summary(anova_holiday)
                  Df    Sum Sq  Mean Sq F value Pr(>F)
factor_data[, i]   1 1.280e+07 12797494   3.421 0.0648 .
Residuals        729 2.727e+09  3740381
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
> summary(anova_weekday)
                  Df    Sum Sq Mean Sq F value Pr(>F)
factor_data[, i]   6 1.766e+07 2943170   0.783  0.583
Residuals        724 2.722e+09 3759498
> summary(anova_workingday)
                  Df    Sum Sq  Mean Sq F value Pr(>F)
factor_data[, i]   1 1.025e+07 10246038   2.737 0.0985 .
Residuals        729 2.729e+09  3743881
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
> summary(anova_weathersit)
                  Df    Sum Sq   Mean Sq F value Pr(>F)
factor_data[, i]   2 2.716e+08 135822286   40.07 <2e-16 ***
Residuals        728 2.468e+09   3389960
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Decision tree model's cross validation & prediction :

- **10-fold cross validation results :**

```
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 463, 464, 463, 463, 464, 463, ...
Resampling results across tuning parameters:

  cp            RMSE      Rsquared   MAE
  0.0000000000  876.2096  0.7996646  626.7108
  0.0003593751  878.5102  0.7987033  632.0277
  0.0004168285  880.1828  0.7980018  632.7116
  0.0004812352  882.6942  0.7971267  633.7119
  0.0011145360  882.9795  0.7963766  636.7127
  0.0056360034  911.5757  0.7828073  660.0251
  0.0085121890  951.7229  0.7654318  702.6840
  0.0128596757  985.7272  0.7454655  726.7120
  0.0134436256  991.7765  0.7430943  734.4426

RMSE was used to select the optimal model using the smallest value.
The final value used for the model was cp = 0.
```

- **MAPE & RMSE scores on prediction dataset :**

```
> predictions= predict(DT_model,test[,-10])
> mape(test[,10],predictions)
[1] 23.06735
> #RMSE(predictions,test[,10]) #using caret
> rmse(predictions,test[,10])
[1] 885.047
```

Random Forest model's cross validation & prediction :

- **10-fold cross validation results :**

```
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 463, 463, 465, 464, 464, 464, ...
Resampling results across tuning parameters:

  mtry  RMSE       Rsquared   MAE
   1    1452.1937  0.7555062  1188.4385
   3     856.6152  0.8535612   677.0590
   9     708.1844  0.8684675   509.9892
  10     709.1425  0.8667232   509.9663
  12     715.0011  0.8634422   513.9678
  13     715.8533  0.8628463   511.8213
  16     723.4910  0.8593387   512.6729

RMSE was used to select the optimal model using the smallest value.
The final value used for the model was mtry = 9.
```

- **MAPE & RMSE scores on prediction dataset :**

```
> rmse(rf_predictions,test[,10])
[1] 684.0097
> mape(test[,10],rf_predictions)
[1] 18.23251
>
```

Linear Regression model's cross validation & prediction :

- **10-fold cross validation results :**

```
> print(lm_model)
Linear Regression

515 samples
  9 predictor

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 464, 463, 463, 463, 464, 463, ...
Resampling results:

  RMSE      Rsquared   MAE
  835.0623  0.8209737  617.5492

Tuning parameter 'intercept' was held constant at a value of TRUE
```

- **MAPE & RMSE scores on prediction dataset :**

```
> lm_predictions= predict(lm_model,test[,-10])
> mape(test[,10],lm_predictions)
[1] 16.19918
> rmse(lm_predictions,test[,10])
[1] 727.8427
```

```
> summary(lm_model)

Call:
lm(formula = .outcome ~ ., data = dat)

Residuals:
     Min       1Q   Median       3Q      Max
-3912.0  -392.3    49.6   479.0  3084.1

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  1766.31     286.08   6.174 1.39e-09 ***
season2       678.22     227.97   2.975  0.00307 **
season3       637.17     271.54   2.347  0.01935 *
season4      1658.11     229.03   7.240 1.74e-12 ***
yr1          1977.30      73.34  26.960  < 2e-16 ***
mnth2         124.50     186.07   0.669  0.50374
mnth3         594.90     213.04   2.792  0.00543 **
```

```
mnth4          649.49     323.41   2.008  0.04516 *
mnth5          959.19     345.35   2.777  0.00569 **
mnth6          608.94     360.28   1.690  0.09163 .
mnth7          236.39     397.15   0.595  0.55197
mnth8          527.24     385.63   1.367  0.17218
mnth9         1076.42     337.18   3.192  0.00150 **
mnth10         370.32     301.55   1.228  0.22001
mnth11        -198.61     287.13  -0.692  0.48944
mnth12         -69.76     222.57  -0.313  0.75409
holiday1      -410.56     204.71  -2.006  0.04545 *
workingday1     94.07      82.23   1.144  0.25318
weathersit2   -461.29      95.36  -4.838 1.76e-06 ***
weathersit3  -1903.08     266.67  -7.137 3.45e-12 ***
temp          4493.67     504.37   8.909  < 2e-16 ***
hum          -1488.52     370.43  -4.018 6.78e-05 ***
windspeed    -2807.16     512.79  -5.474 7.01e-08 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 806.8 on 492 degrees of freedom
Multiple R-squared:  0.8368,    Adjusted R-squared:  0.8295
F-statistic: 114.7 on 22 and 492 DF,  p-value: < 2.2e-16
```

Appendix – B

Full R code :

```
#Clean the environment
rm(list = ls())

#set the working directory
setwd("C:/Users/Navaneeth/Desktop/Edwisor/Project/Bike rental")


#load the required libraries
x = c("ggplot2","gridExtra","DMwR","caret","corrplot")
lapply(x, require, character.only = TRUE)
rm(x)


#read the data csv file by replacing space,comma & NA values with NA.
data=read.csv("day.csv",header=T,na.strings = c("","  ","NA"))

#*********************  Exploratory data Analysis  ***************************

#To have a look at the structure of the dataset
str(data)

#Convert the reqired variables to factor
data$season= as.factor(data$season)
data$yr= as.factor(data$yr)
data$mnth= as.factor(data$mnth)
data$holiday= as.factor(data$holiday)
data$weekday= as.factor(data$weekday)
data$workingday= as.factor(data$workingday)
data$weathersit= as.factor(data$weathersit)

#dropping the instant variable as it's just the row numbers
data$instant=NULL
#dropping the dteday variable as we have separate variables for workingdays and hol
idays
data$dteday=NULL


# get indexes of numerical variables
numeric_index = sapply(data,is.numeric) #selecting only numeric
# store the numeric data
numeric_data = data[,numeric_index]
# store the column names of numerical variables
cnames = colnames(numeric_data)

#plotting density plot for numeric variables
p1= ggplot(numeric_data, aes(x= temp)) + geom_density()
p2= ggplot(numeric_data, aes(x= atemp)) + geom_density()
p3= ggplot(numeric_data, aes(x= hum)) + geom_density()
p4= ggplot(numeric_data, aes(x= windspeed)) + geom_density()
p5= ggplot(numeric_data, aes(x= casual)) + geom_density()
p6= ggplot(numeric_data, aes(x= registered)) + geom_density()
p7= ggplot(numeric_data, aes(x= cnt)) + geom_density()

gridExtra::grid.arrange(p1, p2, p3, p4, p5, p6,p7, ncol = 2)


#*************************   Visualizations    **********************

```

```
#Bike rental count with respect to season
ggplot(data, aes(x = season,y=cnt)) + geom_bar(stat = 'identity',fill='steelblue') + ggtitle("Bike rental count wrt season")


#Bike rental count with respect to workingday
ggplot(data, aes(x = workingday,y=cnt,fill=workingday)) + geom_boxplot() + ggtitle("Bike rental count wrt workingday")


#Bike rental count with respect to weekdays(Monday is taken as zero)
ggplot(data, aes(x = weekday,y=cnt)) + geom_bar(stat = 'identity',fill='steelblue') + ggtitle("Bike rental count wrt weekdays")


#Bike rental count with respect to holiday
ggplot(data, aes(x = holiday,y=cnt,fill=holiday)) + geom_boxplot() + ggtitle("Bike rental count wrt holidays")

#Bike rental count with respect to months
g1=ggplot(data, aes(x = mnth,y=cnt)) + geom_bar(stat = 'identity',fill='steelblue') + ggtitle("Bike rental count wrt months")

#Bike rental count with respect to year
g2=ggplot(data, aes(x = yr,y=cnt)) + geom_bar(stat = 'identity',fill='steelblue') + ggtitle("Bike rental count wrt year")
gridExtra::grid.arrange(g1,g2, ncol = 2)

#Bike rental count with respect to year & months
ggplot(data,aes(x=yr,y=cnt,fill=mnth))+geom_boxplot() + ggtitle("Bike rental count wrt year & months")


#Bike rental count with respect to weather situation
ggplot(data,aes(x=weathersit,y=cnt,fill=weathersit))+geom_boxplot() + ggtitle("Bike rental count wrt weather situation")

#Bike rental count with respect to temperature
g3=ggplot(data,aes(x=temp,y=cnt))+geom_point()+geom_smooth(method='lm',se=F)+ggtitle("Bike rental count wrt temp")
#Bike rental count with respect to temperature & weather situation
g4=ggplot(data,aes(x=temp,y=cnt,col=weathersit))+geom_point()+geom_smooth(method='lm',se=F)+ggtitle("Bike rental count wrt temp & weather condition")
gridExtra::grid.arrange(g3,g4, ncol = 2)

#Bike rental count with respect to adjusted temperature
ggplot(data,aes(x=atemp,y=cnt))+geom_point()+geom_smooth(method='lm',se=F)+ggtitle("Bike rental count wrt atemp")

#Bike rental count with respect to humidity
g5=ggplot(data,aes(x=hum,y=cnt))+geom_point()+geom_smooth(method='lm',se=F)+ggtitle("Bike rental count wrt humidity")
#Bike rental count with respect to humidity & weather situation
g6=ggplot(data,aes(x=hum,y=cnt,col=weathersit))+geom_point()+geom_smooth(method='lm',se=F)+ggtitle("Bike rental count wrt humidity & weather condition")
        gridExtra::grid.arrange(g5,g6, ncol = 2)

        #Bike rental count with respect to windspeed
        g7=ggplot(data,aes(x=windspeed,y=cnt))+geom_point()+geom_smooth(method='lm',se=F)+ggtitle("Bike rental count wrt windspeed")
        #Bike rental count with respect to windspeed & weather situation
        g8=ggplot(data,aes(x=windspeed,y=cnt,col=weathersit))+geom_point()+geom_smooth(method='lm',se=F)+ggtitle("Bike rental count wrt windspeed & weather condition")

        gridExtra::grid.arrange(g7,g8, ncol = 2)
```

```
107.
108.
109.
110.       #**********************   Missing Value Analysis   ***********************
    ********
111.
112.       #calcualte the sum of NAs in each colum and store it as a dataframe in missi
    ng_val
113.       missing_val = data.frame(apply(data,2,function(x) {sum(is.na(x))}))
114.       #Storing the row names
115.       missing_val$Columns = row.names(missing_val)
116.       # rename the 1st column name
117.       names(missing_val)[1] =  "Missing_percentage"
118.       # calculate the percentage of NAs
119.       missing_val$Missing_percentage = (missing_val$Missing_percentage/nrow(data))
     * 100
120.       # get percentage of NAs in decreasing order
121.       missing_val = missing_val[order(-missing_val$Missing_percentage),]
122.       #drop the row names
123.       row.names(missing_val) = NULL
124.       #re-arrange the columns
125.       missing_val = missing_val[,c(2,1)]
126.
127.
128.       #************************     Outlier Analysis    ***************************
    *****
129.
130.
131.       # get indexes of numerical variables
132.       numeric_index = sapply(data,is.numeric) #selecting only numeric
133.       # store the numeric data
134.       numeric_data = data[,numeric_index]
135.       #dropping the target variable
136.       numeric_data$cnt=NULL
137.       # store the column names of numerical variables
138.       cnames = colnames(numeric_data)
139.
140.       # loop for plotting the box plot for all the numerical variables
141.       for (i in 1:length(cnames))
142.       {
143.         assign(paste0("gn",i), ggplot(aes_string(y = (cnames[i]), x = "cnt"), data
     = subset(data))+
144.                  stat_boxplot(geom = "errorbar", width = 0.5) +
145.                  geom_boxplot(outlier.colour="red", fill = "grey" ,outlier.shape=1
    8,
146.                               outlier.size=1, notch=FALSE) +
147.                  theme(legend.position="bottom")+
148.                  labs(y=cnames[i],x="cnt")+
149.                  ggtitle(paste("Box plot of bike rental count for",cnames[i])))
150.       }
151.
152.       # Plotting the boxplots together
153.       gridExtra::grid.arrange(gn1,gn2,gn3,gn4,gn5,gn6,ncol=2)
154.
155.       #*********************    Outlier Treatment    ******************
156.
157.       #We are treating outliers on hum and windspeed
158.       num=c('hum','windspeed')
159.
160.       # Replace Outlier values with NAs
161.       for (i in num)
162.       {
163.         outlier_values=data[,i][data[,i] %in% boxplot.stats(data[,i])$out]
164.         data[,i][data[,i] %in% outlier_values]= NA
165.       }
```

```
166.
167.
168.        # Impute NAs with KNN Imputation
169.        data= knnImputation(data,k=3)
170.
171.        #**********************    Feature Selection    ***************************

172.
173.        # Correlation Plot
174.        numeric_data = data[,numeric_index]
175.        cnames = colnames(numeric_data)
176.        corr= cor(data[,cnames])
177.        corrplot(corr, method = "color",outline = TRUE,addCoef.col = "black")
178.
179.        #From the above corrplot we will remove atemp as it is positively correlated
    with temp.
180.        # Also, the variables 'casual' & 'registered' will be dropped as we have sum
    of both these variables as 'cnt'
181.
182.
183.        #Chi square test to check the dependency/correlation between the categorical
    variables
184.        factor_index=sapply(data,is.factor)
185.        factor_data = data[,factor_index]
186.        cat_names = colnames(factor_data)
187.
188.        for (i in 1:length(cat_names))
189.        {
190.          for (j in 2:length(cat_names))
191.          {
192.            print(names(factor_data)[i])
193.            print(names(factor_data)[j])
194.            print(chisq.test(table(factor_data[,i], factor_data[,j])))
195.          }
196.        }
197.
198.        #Finding relationship between categorical variable and target variable(Conti
    nuous) using ANOVA
199.        #For loop for ANOVA
200.        for(i in cat_names)
201.        {
202.          assign(paste0("anova_", i), aov(data$cnt ~ factor_data[,i]))
203.        }
204.
205.        summary(anova_season)# season is statistically significant to explain the ta
    rget variable
206.        summary(anova_yr) #yr is statistically significant to explain the target var
    iable
207.        summary(anova_mnth) # mnth is statistically significant to explain the targe
    t variable
208.        summary(anova_holiday) #holiday is statistically significant to explain the
    target variable
209.        summary(anova_weekday) #weekday is not statistically significant to explain
    the target variable
210.        summary(anova_workingday) #workignday is statistically significant to explai
    n the target variable
211.        summary(anova_weathersit) #weathersit is statistically significant to explai
    n the target variable
212.
213.
214.        # Dimension Reduction
215.
216.        data= subset(data,select = -c(atemp,casual,registered,weekday))
217.
218.        #**********************   Train-Test Split  *****************
```

```
219.
220.       #Split 70% of the data as train and 30% as test
221.       set.seed(1234)
222.       train.index = createDataPartition(data$cnt, p = .70, list = FALSE)
223.       train = data[ train.index,]
224.       test  = data[-train.index,]
225.
226.       #Function for mape
227.       mape = function(act,pre)
228.       {
229.         mean(abs((act-pre)/act))*100
230.       }
231.
232.       #Function for RMSE calculation
233.       rmse = function(pred,act)
234.       {
235.         difference = (pred - act)
236.         root_mean_square_error = sqrt(mean(difference^2))
237.         return(root_mean_square_error)
238.       }
239.
240.       #*********************   Decision Tree model using 10 fold cross validation
     *************
241.
242.       #set the traincontrol parameters
243.       control = trainControl(method="cv",number = 10,search = "random")
244.       #Train the decision tree model
245.       set.seed(12)
246.       DT_model= train(cnt ~., data=train,method="rpart",trControl=control,tuneLeng
    th=10)
247.       #Print the CV results
248.       print(DT_model)
249.       #Predict on the test data
250.       predictions= predict(DT_model,test[,-10])
251.
252.       #Calculate MAPE
253.       mape(test[,10],predictions)
254.       #Calculate RMSE
255.       rmse(predictions,test[,10])
256.
257.       #Save the DT Model
258.       saveRDS(DT_model,"DecisionTree_Rmodel.rds")
259.
260.
261.       # MAPE = 23.06735
262.       # RMSE = 885.047
263.
264.
265.       #****************    Random Forest model using 10 fold cross validation   **
    ********
266.
267.       #set the traincontrol parameters
268.       control = trainControl(method="cv",number = 10,search = "random")
269.       #Train the random forest model
270.       set.seed(111)
271.       rf_model= train(cnt ~., data=train,method="rf",trControl=control,tuneLength=
    10)
272.       #Print the CV results
273.       print(rf_model)
274.       #Predict on the test data
275.       rf_predictions= predict(rf_model,test[,-10])
276.
277.       #Calculate MAPE
278.       mape(test[,10],rf_predictions)  #18.23251
279.       #Calculate RMSE
```

```
        rmse(rf_predictions,test[,10])

        #Save the RF Model
        saveRDS(rf_model,"rf_Rmodel.rds")


        # MAPE = 18.23251
        # RMSE = 684.0097

        #***************    Linear regression using 10 fold cross validation    ***********

        #set the traincontrol parameters
        control = trainControl(method="cv",number = 10,search = "random")
        #Train the LM model
        set.seed(112)
        lm_model= train(cnt ~., data=train,method="lm",trControl=control,tuneLength=5)
        #Print the CV results
        print(lm_model)
        #Predict on the test data
        lm_predictions= predict(lm_model,test[,-10])

        #Calculate MAPE
        mape(test[,10],lm_predictions)
        ##Calculate RMSE
        rmse(lm_predictions,test[,10])

        #Save the RF Model
        saveRDS(lm_model,"lm_Rmodel.rds")


        # MAPE = 16.19918
        # RMSE = 727.8427


        #*********************    END OF THE FILE        *********************#
```

Appendix – C

Full Python code :

```
# Load the pre-required libraries
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import chi2_contingency
import seaborn as sns
from fancyimpute import KNN
from sklearn.model_selection import train_test_split

#set the working directory
os.chdir("C:/Users/Navaneeth/Desktop/Edwisor/Project/Bike rental")
# Load the given data
data=pd.read_csv("day.csv")
print('The shape of given dataset is : ',data.shape)
#set display option to display all the columns in a dataframe
pd.set_option('display.max_columns', 30)
#display first 10 rows of whole data
data.head(10)

#get the info about the dataset
data.info()

#Convert the required variables to object datatype
names=['season','yr','mnth','holiday','weekday','workingday','weathersit']
for i in names :
    data[i] = data[i].astype('object')

#drop the instant variable as it's just the row numbers
data.drop('instant',axis=1,inplace=True)
#drop the dteday variable as we have separate variables for workingdays and holiday
s
data.drop('dteday',axis=1,inplace=True)
#Check the data types
data.dtypes

# Density plots 0f numeric variables
cnames= ['temp','atemp','hum','windspeed','casual','registered','cnt']
for i in cnames:
    pd.DataFrame(data[i]).plot(kind='density')

# Bike rental count with respect to season
%matplotlib inline
sns.barplot(x='season',y='cnt',data=data).set_title("Bike rental count wrt season")


# Bike count with respect to workingday
sns.boxplot(x='workingday',y='cnt',data=data).set_title('Bike count wrt workingday'
)

#Bike count with respect to holiday
sns.boxplot(x='holiday',y='cnt',data=data).set_title('Bike count wrt holiday')

# Bike count with respect to weekdays
sns.barplot(x='weekday',y='cnt',data=data).set_title('Bike count wrt weekdays')

# Bike count with respect to months
sns.barplot(x='mnth',y='cnt',data=data).set_title('Bike count wrt months')
```

```
56.
57. # Bike count with respect to year
58. sns.barplot(x='yr',y='cnt',data=data).set_title('Bike count wrt year')
59.
60. # Bike count with respect to year & months
61. sns.boxplot(x='yr',y='cnt',hue='mnth',data=data).set_title('Bike count wrt year & m
    onths')
62.
63. # Bike count with respect to weather situation
64. sns.boxplot(x='weathersit',y='cnt',data=data).set_title('Bike count wrt weather con
    dition')
65.
66. # Bike count with respect to temperature & weather situation
67. sns.lmplot(x='temp',y='cnt',hue='weathersit',data=data)
68.
69. # Bike count with respect to humidity & weather situation
70. sns.lmplot(x='hum',y='cnt',hue='weathersit',data=data)
71.
72. # Bike count with respect to windspeed & weather situation
73. sns.lmplot(x='windspeed',y='cnt',hue='weathersit',data=data)
74.
75. #*************** Missing value Analysis *****************
76.
77. #Store the sum of null values as a dataframe in a new variable
78. missing_val = pd.DataFrame(data.isnull().sum())
79. #Reset index
80. missing_val = missing_val.reset_index()
81. #Rename variables
82. missing_val = missing_val.rename(columns = {'index': 'Variables', 0: 'Missing_perce
    ntage'})
83. #Calculate percentage
84. missing_val['Missing_percentage'] = (missing_val['Missing_percentage']/len(data))*1
    00
85. #descending order
86. missing_val = missing_val.sort_values('Missing_percentage', ascending = False).rese
    t_index(drop = True)
87. print(missing_val)
88.
89. # Get the names of numeric and categorical variables
90. cnames=[]
91. cat_names=[]
92.
93. for i in range(0,data.shape[1]):
94.     if (data.iloc[:,i].dtype=='object'):
95.         cat_names.append(data.columns[i])
96.
97.     else:
98.         cnames.append(data.columns[i])
99.
100.        print("The numeric variables are :",cnames)
101.        print("The categorical variables are :",cat_names)
102.
103.        # Plotting boxplot to visulaize outliers for all the numeric variables
104.        %matplotlib inline
105.        for i in cnames :
106.            plt.figure()
107.            plt.clf()
108.            plt.boxplot(data[i])
109.            plt.title(i)
110.            #plt.savefig(i)
111.            plt.show()
112.
113.        # We are imputing outliers present in hum & windspeed variables with NA
114.        names=['hum','windspeed']
115.
```

```
        for i in names:
            print(i)
            q75,q25=np.percentile(data.loc[:,i],[75,25])
            iqr= q75-q25

            minimum = q25-(1.5*iqr)
            maximum = q75+(1.5*iqr)
            print('Minimum value is',minimum)
            print('Maximum value is',maximum)

            data.loc[:,i][data.loc[:,i] < minimum]= np.nan
            data.loc[:,i][data.loc[:,i] > maximum]= np.nan

        # Impute the Outlier NAs with KNN imputation
        data= pd.DataFrame(KNN(k=3).fit_transform(data),columns=data.columns)

        # Verify if there is any NA's in the dataset
        data.isnull().sum()

        #After KNN imputation all the variables will be in 'float' datatype, hence I 'm converting all the object datatypes as it were before.
        cat_names=['season','yr','mnth','holiday','weekday','workingday','weathersit']

        for i in cat_names:
            print(i)
            data[i] = data[i].astype('object')
        data.dtypes


        # Correlation Analysis
        #Extract numeric variables dataset for correlation analysis
        n_names=['temp','atemp','hum','windspeed','casual','registered','cnt']
        numeric_data= data.loc[:,n_names]
        #Set the width and hieght of the plot
        f, ax = plt.subplots(figsize=(7, 5))
        #Generate correlation matrix
        corr = numeric_data.corr()
        #Plot using seaborn library
        plt.title('Pearson Correlation Plot')
        sns.heatmap(corr, mask=np.zeros_like(corr, dtype=np.bool), cmap=sns.diverging_palette(220, 10,as_cmap=True),square=True, ax=ax,annot=True)

        #Finding correlation between categorical variables
        factor_data = data.loc[:,cat_names]

        #Comparing the columns
        factors_paired = [(i,j) for i in factor_data.columns.values for j in factor_data.columns.values]
        chi2, p_values =[], []

        for k in factors_paired:
            #print(f[0], f[1])
            if k[0] != k[1]:
                chitest = chi2_contingency(pd.crosstab(factor_data[k[0]], factor_data[k[1]]))
                chi2.append(chitest[0])
                p_values.append(chitest[1])
            else:      #for same factor pair
                chi2.append(0)
                p_values.append(0)
        chi2 = np.array(p_values).reshape((7,7)) # shape it as a matrix
        chi2 = pd.DataFrame(chi2, index = factor_data.columns.values, columns=factor_data.columns.values)
        print(chi2)
```

```
175.
176.
177.        #ANOVA Test performed for all the categorical variables and weekday has high
     p-value
178.        import statsmodels.api as sm
179.        from statsmodels.formula.api import ols
180.        mod = ols('cnt ~ weekday',data=data).fit()
181.        aov_table = sm.stats.anova_lm(mod, typ=1)
182.        print (aov_table)
183.
184.        #Features to be dropped
185.        data_new=data.drop(['atemp','weekday','casual','registered'],axis=1)
186.        print("The shape of the data after dropping the unnecessary variables are :"
    ,data_new.shape)
187.        data_new.head()
188.
189.
190.        #Import Libraries for decision tree
191.        from sklearn.model_selection import train_test_split
192.        #Separate the independent & target variables
193.        X = data_new.iloc[:,0:9]
194.        y = data_new.iloc[:,9]
195.        #Splitting into train and test, with 70:30 ratio respectively
196.        X_train, X_test, y_train, y_test= train_test_split(X, y, test_size = 0.3, ra
    ndom_state = 4)
197.
198.        # import MSE function
199.        from sklearn.metrics import mean_squared_error
200.        # Function for calculating MAPE
201.        def mape(act,pred):
202.            m=np.mean(np.abs((act - pred)/act))*100
203.            return m
204.
205.
206.        # Decision Tree Model with 10-fold cross validation
207.
208.        # import the gridserachcv function & decision tree function
209.        from sklearn.model_selection import GridSearchCV
210.        from sklearn.tree import DecisionTreeRegressor
211.        # define a list of range of depths for the decision tree model to be built
212.        depth=list(range(1,9))
213.        param_grid= dict(max_depth=depth)
214.        print(param_grid)
215.        # instantiate the gridsearch using decisiontree model
216.        DTmod=DecisionTreeRegressor(random_state=3)
217.        grid = GridSearchCV(DTmod, param_grid, cv=10,scoring='r2', return_train_scor
    e=False)
218.        #Fit the grid on the trian data
219.        grid.fit(X_train,y_train)
220.        #Print the best estimated model
221.        print("The best cross validated estimator of decision tree is :",grid.best_e
    stimator_)
222.        # Print the best score of r-square
223.        print("The best cross validated r-
    square of decision tree is :",grid.best_score_)
224.        #Print the given parameters & corresponding r-square value from each 10-
    fold cross validation
225.        pd.DataFrame(grid.cv_results_)[['param_max_depth', 'mean_test_score', 'param
    s']]
226.
227.        # Predict on the test dataset
228.        DT_predicted=grid.predict(X_test)
229.        # Print the MAPE & RMSE scores
230.        print("The MAPE of DT prediction is : %.4f" % mape(y_test,DT_predicted))
```

```
231.        print("The RMSE of DT prediction is : %.4f" % np.sqrt(mean_squared_error(DT_
    predicted,y_test)))
232.
233.        # Save the Decision Tree model for future use
234.        from sklearn.externals import joblib
235.        joblib.dump(grid, "DecisionTree_Python_Final.sav")
236.
237.
238.        #Random Forest Model with 10-fold cross validation
239.
240.        # Import the random forest module
241.        from sklearn.ensemble import RandomForestRegressor
242.        r=RandomForestRegressor(random_state=1)
243.        # Define the list of estimators for the Random forest to be built
244.        est=list([100,200,300,400,500,600])
245.        param_grid= dict(n_estimators=est)
246.        print(param_grid)
247.        # instantiate the gridsearch using random forest model
248.        grid_rf = GridSearchCV(r, param_grid, cv=10,scoring='r2', return_train_score
    =False)
249.        # Fit the random forest grid
250.        grid_rf.fit(X_train,y_train)
251.        #Print the best estimated model
252.        print("The best cross validated estimator of Random Forest is :",grid_rf.bes
    t_estimator_)
253.        # Print the best score of r-square
254.        print("The best cross validated r-
    square of Random Forest is :",grid_rf.best_score_)
255.        #Print the given parameters & corresponding r-square value from each 10-
    fold cross validation
256.        pd.DataFrame(grid_rf.cv_results_)[[ 'params','mean_test_score']]
257.
258.        # Predict on the test dataset
259.        predicted_rf=grid_rf.predict(X_test)
260.        # Print the MAPE & RMSE scores
261.        print("The MAPE of Random Forest prediction is : %.4f" %mape(y_test,predicte
    d_rf))
262.        print("The RMSE of Random Forest prediction is : %.4f" %np.sqrt(mean_squared
    _error(predicted_rf,y_test)))
263.
264.        # Save the Random forest model for future use
265.        joblib.dump(grid_rf, "RF_model_Python_Final.sav")
266.
267.
268.        #Linear Regression model with 10-fold cross validation
269.
270.        # Import the Linear Regression module
271.        from sklearn.linear_model import LinearRegression
272.        lm= LinearRegression()
273.        # Define the intercept value for the Linear Regression
274.        intr=list(['True'])
275.        param_grid= dict(fit_intercept=intr)
276.        print(param_grid)
277.        #instantiate the gridsearch using Linear Regression model
278.        grid_lm = GridSearchCV(lm,param_grid, cv=10,scoring='r2', return_train_score
    =False)
279.        # Fit the Linear Regression grid
280.        grid_lm.fit(X_train,y_train)
281.        #Print the best estimated model
282.        print("The best cross validated estimator of Linear Regression is :",grid_lm
    .best_estimator_)
283.        # Print the best score of r-square
284.        print("The best cross validated r-
    square of Linear Regression is :",grid_lm.best_score_)
```

```
285.        #Print the given parameters & corresponding r-square value from each 10-
    fold cross validation
286.        pd.DataFrame(grid_lm.cv_results_)[[ 'params','mean_test_score']]
287.
288.        # Predict on the test dataset
289.        predicted_lm=grid_lm.predict(X_test)
290.        # Print the MAPE & RMSE scores
291.        print("The MAPE of Linear Regression prediction is : %.4f" %mape(y_test,pred
    icted_lm))
292.        print("The RMSE of Linear Regression prediction is : %.4f" %np.sqrt(mean_squ
    ared_error(predicted_lm,y_test)))
293.
294.        # Save the Linear Regression model for future use
295.        joblib.dump(grid_rf, "Linear_Regression_Python.sav")
```

References :

https://www.theanalysisfactor.com/missing-data-mechanism/

https://datavizcatalogue.com/methods/density_plot.html

https://www.sharpsightlabs.com/blog/density-plot-in-r/

http://www.physics.csbsju.edu/stats/box2.html

https://nelsontouchconsulting.wordpress.com/2011/01/17/deeper-into-box-plots/

https://stackoverflow.com/questions/26553526/how-to-add-frequency-count-labels-to-the-bars-in-a-bar-graph-using-ggplot2

https://www.dataquest.io/blog/understanding-regression-error-metrics/

https://scikit-learn.org/stable/modules/cross_validation.html

https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html

https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html

https://scikit-learn.org/stable/modules/model_evaluation.html#scoring-parameter

https://machinelearningmastery.com/how-to-tune-algorithm-parameters-with-scikit-learn/

https://scikit-learn.org/stable/modules/cross_validation.html#computing-cross-validated-metrics