Bike Rental Prediction

Priya Karode

Contents

Chapter 1

Introduction

1.1 Problem Statement

Bike sharing is an innovative approach to urban mobility, combining the convenience and flexibility of a bicycle with the accessibility of public transportation. A bike sharing system is a service in which users can rent/use bicycles available for shared use on a short term basis for a price. Such system usually aim to reduce congestion, noise and air pollution by providing affordable access to bikes for short distance trips as opposed to motorised vehicles. The number of users on any given day can vary greatly for such systems depending upon various factors. Our aim is to use and optimise Machine Learning Models that effectively predict the number of ride-sharing bikes that will be used in any given day, using available information about that particular day.

1.2 Data

The dataset shows daily rental data for two years historical log corresponding to years 2011 and 2012. The dataset is also joined by the weather statistics for the corresponding date as bike sharing rental process is highly correlated to the environmental and seasonal settings. Let's start by importing the data and doing some small checks to better understand it. We see that we have 16 variables or features and 731 registers. Given below is the sample of the data set that we are using:

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
6	06-01-2011	1	0	1	0	4	1
7	07-01-2011	1	0	1	0	5	1
8	08-01-2011	1	0	1	0	6	0
9	09-01-2011	1	0	1	0	0	0
10	10-01-2011	1	0	1	0	1	1

Table 1.1: Bike Sharing Sample Data (Columns: 1-8)

weathersit	temp	atem	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
1	0.204348	0.233209	0.518261	0.0895652	88	1518	1606
2	0.196522	0.208839	0.498696	0.168726	148	1362	1510
2	0.165	0.162254	0.535833	0.266804	68	891	959
1	0.138333	0.116175	0.434167	0.36195	54	768	822
1	0.150833	0.150888	0.482917	0.223267	41	1280	1321

Table 1.2: Bike Sharing Sample Data (Columns: 9-16)

The description of the variables are:

Sr. No	Variable Name	Description
1.	Instant	Record index
2.	Dteday	Date
3.	Season	Seasons: 1: Springer 2: Summer 3: Fall 4: Winter
4.	Yr	Year 0: 2011 1: 2012
5.	Mnth	Months (1 to 12)
6.	Holiday	Whether day is holiday or not. 0: No holiday 1: Holiday
7.	weekday	Day of the Week (0-6) Starting from Sunday
8.	workingday	Whether day working or not 0: Non-Working Day 1: If day is neither weekend nor holiday.
9.	weathersit	Weather Situation (extracted from Freemeteo) 1: Clear, Few clouds, 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
10.	temp:	Normalized temperature in Celsius. The values are derived via $\frac{t - t\_min}{t\_max - t\_min}$ Where, t_min=-8, t_max=+39 (only in hourly scale)
11.	Atemp	Normalized feeling temperature in Celsius. The values are derived via $\frac{t - t\_min}{t\_max - t\_min}$ Where, t_min=-16, t_max=+50 (only in hourly scale)
12.	hum	Normalized Humidity. The values are divided to 100 (max)
13.	windspeed	Normalized Wind Speed. The values are divided to 67 (max)
14.	Casual	count of Casual Users
15.	registered	Count of Registered Users
16.	Cnt	Count of total rental bikes including both Casual and Registered Users.

As we can see in the table above, we have count as response variable and all other as predictor variables. We also see that some of our variable are categorical but they are interpreted as numerical which we need to handle by converting them into proper datatypes. Lastly, it would be a great idea to change the variable names for something nicer to read.

So, the variables with updated datatypes and names look like:

```
day.dtypes
```

```
sr.no                          int64
date                 datetime64[ns]
season                       category
year                         category
month                        category
holiday                      category
weekday                      category
workingday                   category
weather_situation            category
normalised_temp               float64
apparent_temp                 float64
humidity                      float64
windspeed                     float64
casual                          int64
registered                      int64
count                           int64
dtype: object
```

Chapter 2

Methodology

2.1 Hypothesis Generation

Before exploring the data to understand the relationship between variables, it is recommended to focus on hypothesis generation first. A Hypothesis is an explanation for something. It is a provisional idea, an educated guess that requires some evaluation.

Here are some of the hypothesis which I thought could influence the demand of bikes:

Daily Trend: Registered users demand more bike on weekdays as compared to weekend or holiday.

Rain: The demand of bikes will be lower on a rainy day as compared to a sunny day. Similarly, higher humidity will cause to lower the demand and vice versa.

Temperature: The demand of the bikes will be lower for extremely low and extremely high temperatures.

Time: With the increasing time, the demand will increase as the popularity will increase.

Weather: The demand will increase on a good, pleasant day but at the same time weather will not affect the registered users.

2.2 Exploratory Data Analysis

Exploratory Data Analysis refers to the critical process of performing initial investigations on data so as to discover patterns, to spot anomalies, to test hypothesis and to check assumptions with the help of summary statistics and graphical representations. It's a good practice to understand the data first and try to gather as many insights from it. Here, I am going to explore data visually through the process of Data Visualisation.

2.2.1 Data Visualisation

Before starting to process a data set with algorithms, it's always a good idea to explore it visually. Data visualisation is the graphical representation of information and data. As the age of Big Data kicks into high gear, visualisation is an increasingly key tool to make sense of the trillions of rows of data generated every day. Using the ggplot2 and ggExtra packages in R, and Seaborn library in python, we can quickly make some plots to investigate how the bike usage count is affected by the continuous and categorical variables available. Now let's look at univariate and bivariate analysis of the predictor variables through some graphs.

2.2.1.1 Distribution of continuous variables

As seen from the scatter plots below, there is a positive correlation between both Normalised temperature to Usage and Apparent temperature to Usage for most of the temperature range, and a linear fit is not very far from the best fit curve. Also, for the maximum temperatures, there is a dip in the curve. This should intuitively make sense as people as people are not likely to step outside in extreme cold and hot weather which supports our hypothesis for temperature vs count.

There seems to be a negative correlation between the humidity and the usage rate. The plot is negatively skewed. This may be due to the presence of outliers or higher value in the data. The humidity plot shows that generally, the higher the relative humidity, the lower the bike rental demand.

Looking at the wind speed data, however, doesn't give a clear idea of how it affects usage. The correlation between the wind speed and the count is very week. This can be due to the presence of outliers and extreme data in this variable.

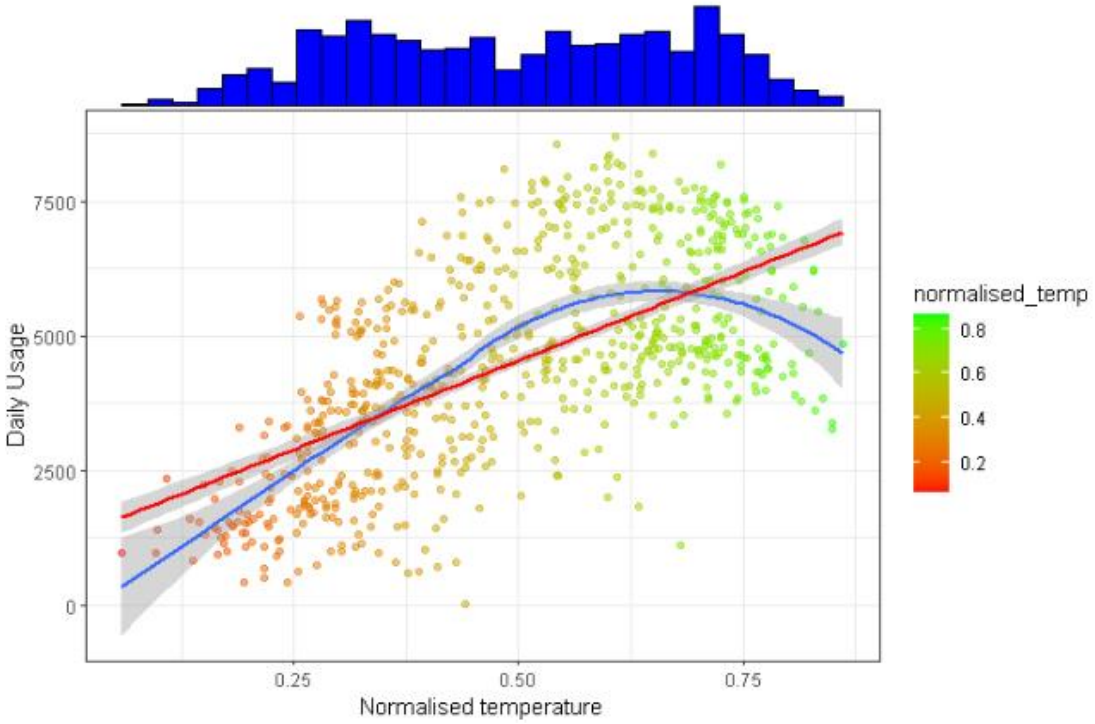

Figure 2.1 : Scatter Plot- Normalised Temperature vs Usage

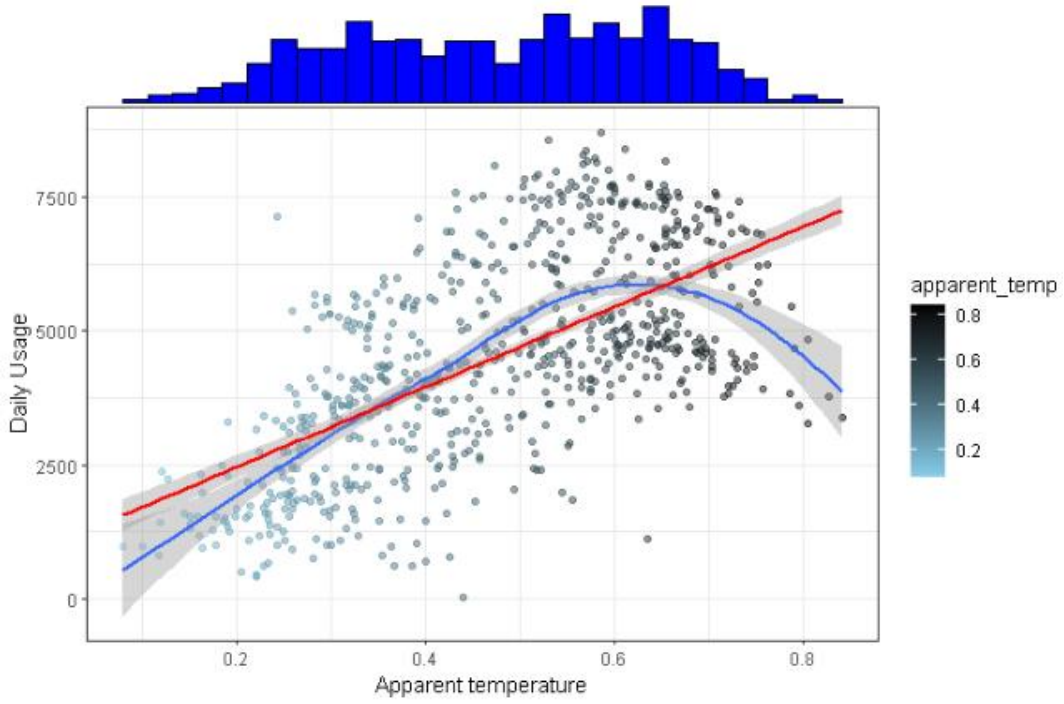

Figure 2.2: Scatter Plot- Apparent Temperature vs Usage

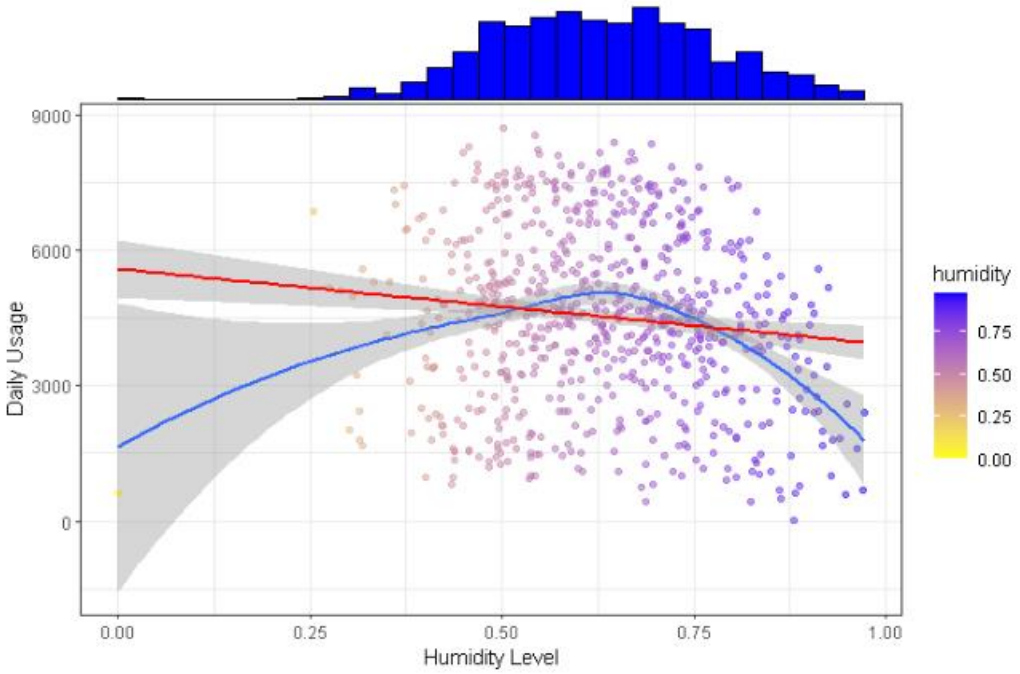

Figure 2.3: Scatter Plot- Humidity vs Usage

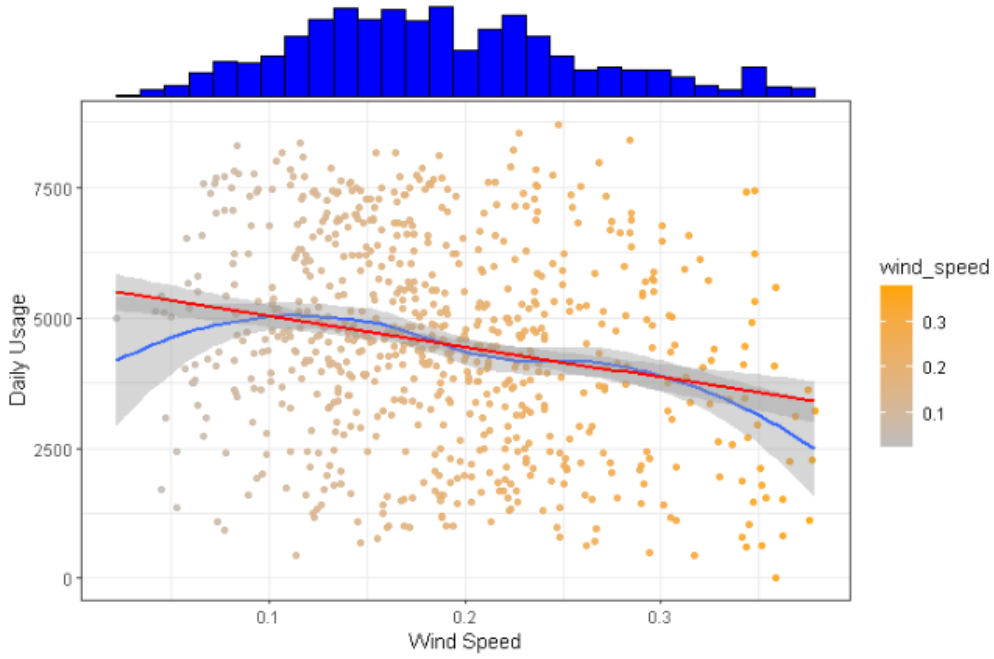

Figure 2.4: Scatter Plot: WindSpeed vs Usage

2.2.1.2 Distribution of Categorical Variables

In the Month vs Usage plot based on Season, we see that there is an evidently higher usage rate during the summer and fall season and the lowest usage in the months of January and February in Spring season.

From working day and usage plot, we see that there is more usage rate when there is working day as compared to non-working days or holidays which supports are hypothesis foe daily trend.

From the next plot of weather situation vs usage, we can clearly interpret that weather situation will affect bike usage, with rainfall deterring usage. Hence, this plot supports our hypothesis for rain affecting usage rate of bikes.

Finally, looking at the year variable, it can be seen that the usage rate goes up from year 2011 to 2012, which suggests that system grew in popularity, which supports our hypothesis for time affecting usage rate.

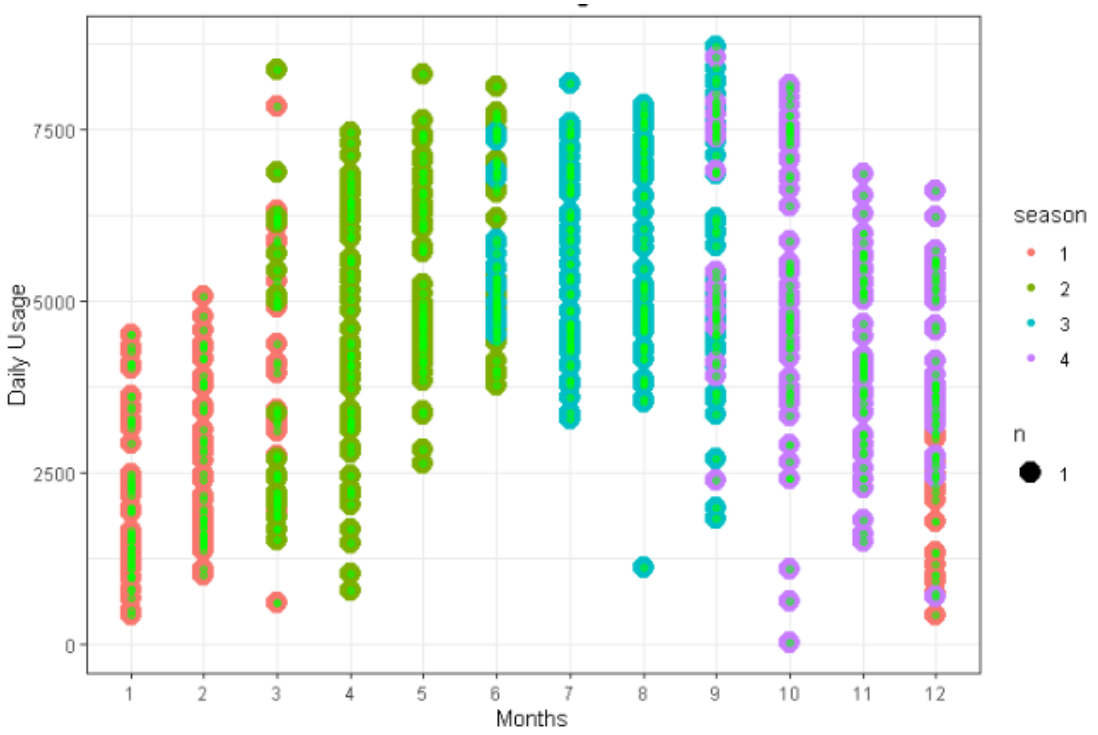

Figure 2.5: Scatter Plot- Months vs Usage based on Seasons

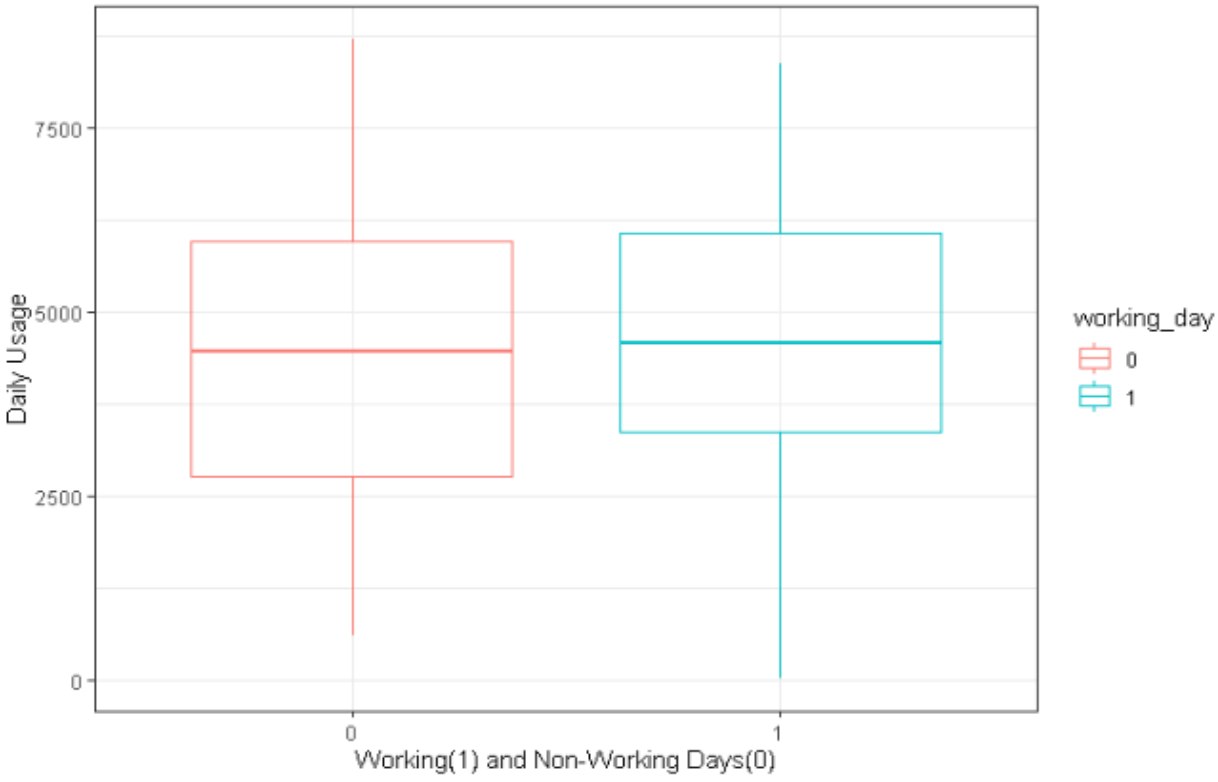

Figure 2.6: Box Plot- Working Day vs Usage

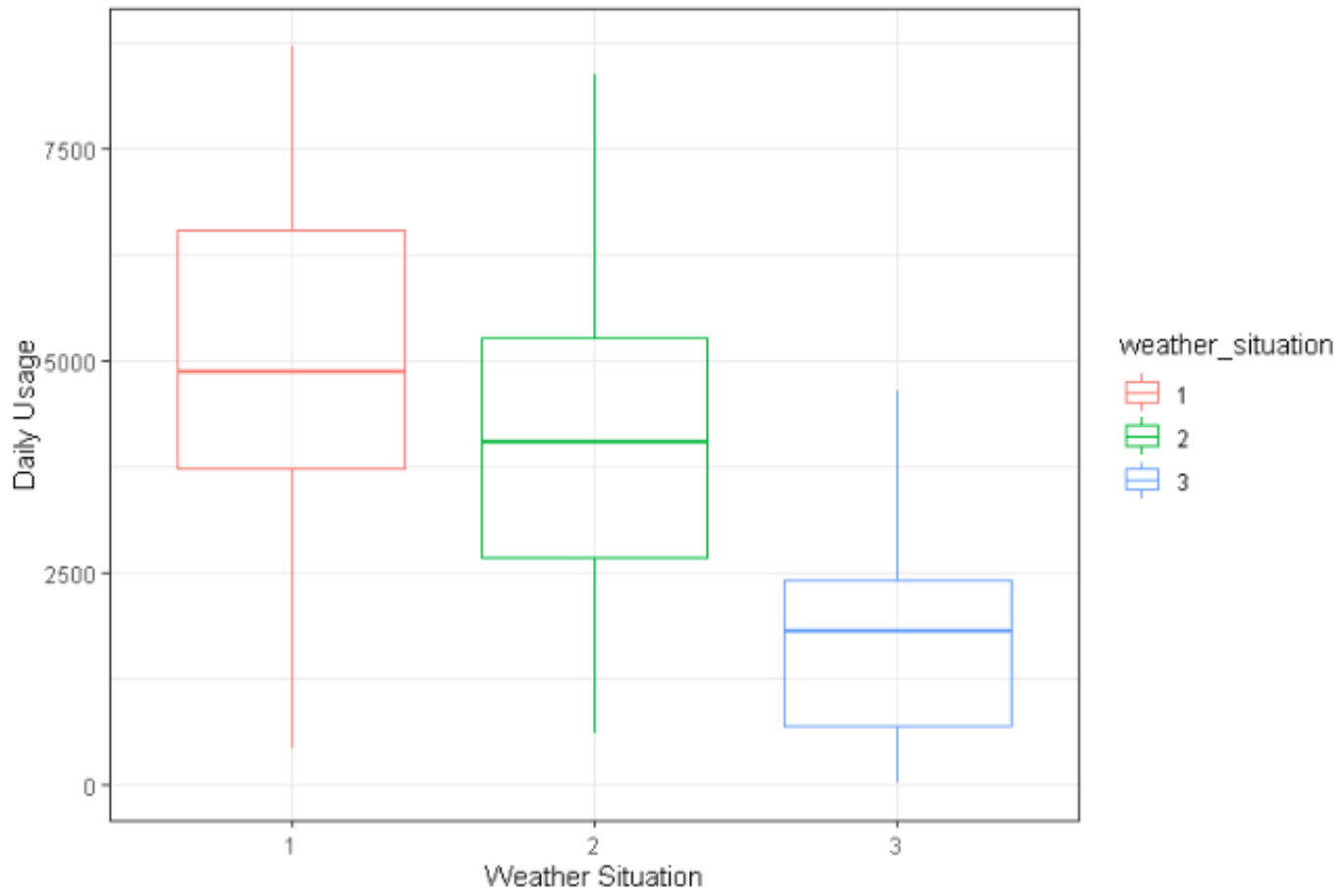

Figure 2.7: Box Plot- Weather Situation vs Usage

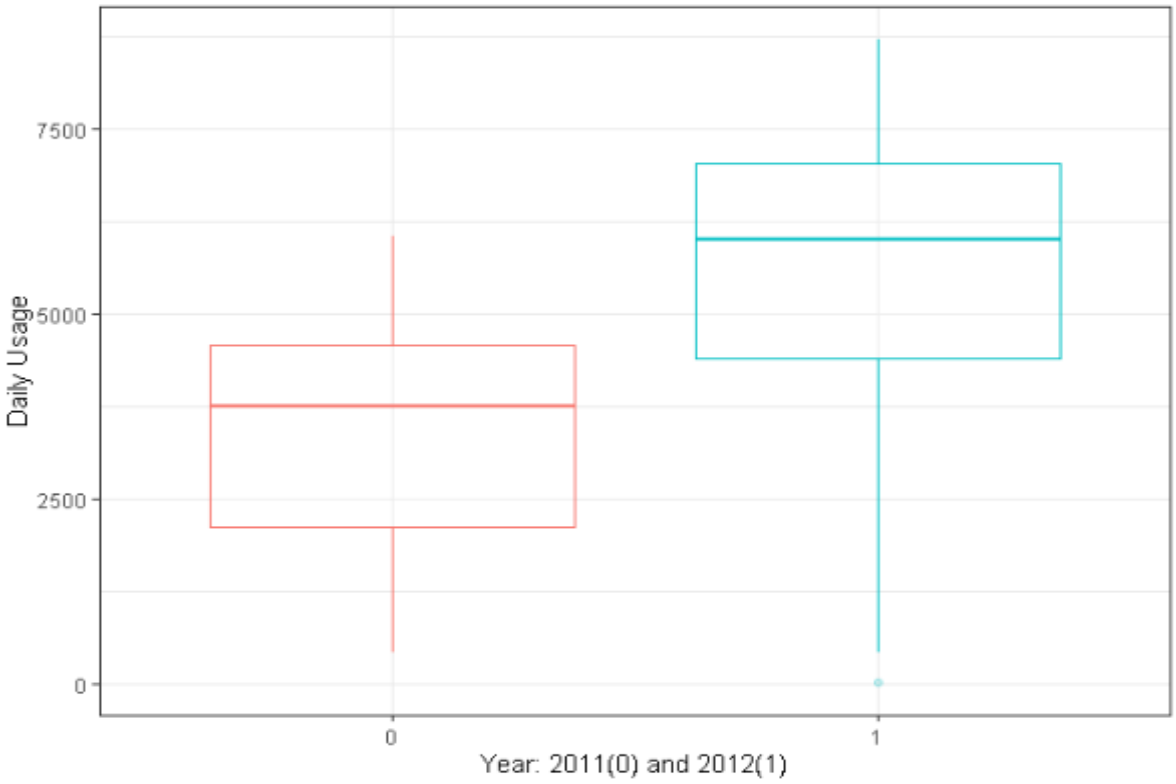

Figure 2.8: Box Plot- Year vs Usage

2.3 Data Pre-processing:

For any Machine Learning project, pre-processing the data is a crucial step. Data Pre-processing is a data mining technique which is used to transform the raw data in a useful and efficient format. Techniques involved in data pre-processing are:

2.3.1 Feature Engineering:

Feature Engineering is the process of using domain knowledge to extract features from raw data via data mining techniques. These features can be used to improve performance of the machine learning algorithms.

Extracting date from dteday column:

Though date columns usually provide valuable information about the model target, they are neglected as an input for the machine learning algorithm. Building an ordinal relationship between these date values is very challenging for a machine learning algorithm if we leave the date column without manipulation.so, this might be the reason for this that we extract day from the date given in the dteday column and then after having engineered new column named 'daynum', we delete the dteday column. We can use this 'daynum' variable as a predictor variable for the usage count.

2.3.2 Missing Value Analysis:

Missing Values are one fof the most common problems one can encounter when we try to prepare our data for machine learning. The reasons for the missing value might be human errors, interruptions in the data flow, privacy concerns and so on. Whatever the reason is, missing values affect the performance of the machine learning models.

I performed missing value analysis on the data set by calculating the percentage of missing values present in each column. And as a result there are no missing values in the dataset.

2.3.3 One hot encoding:

One hot encoding is one of the most common encoding methods in machine learning. This method spreads the values in a column to multiple flag columns and assign 0 or 1 to them.

These binary values express the relationship between grouped and encoded columns. This method changes the categorical data, which is challenging to understand for algorithms that might create misleading results, to a numerical format and enable us to group our categorical data without losing any information. So by using this method we split our non-binary categorical data, i.e. "season","month","weekday","weather_situation", "daynum", into multiple binary sub features where each sub-feature indicates whether a certain category for the original feature is True or Not(1 or 0).

2.3.4 Outlier Analysis:

`Outliers are extreme values that deviate from other observations on data, they may indicate the variability in a measurement, experimental errors or a novelty. In other words, an outlier is an observation that diverges from an overall pattern on a sample. Detecting outlier is of major importance for almost any quantitative discipline. In Machine Learning and in any other quantitative discipline the quality of data is as important as the quality of a prediction or classification model.

One of the simplest ways of detecting outliers is the use of Box plots. Box plot method uses the median and the lower and upper quartiles. The Tukey's method defines an outlier as those values of the data set that fall far from the central point, the median. The maximum distance to the centre of the data that is going to be allowed is called the cleaning parameter. If the cleaning parameter is very large, the test becomes less sensitive to outliers. On the contrary, if it is too small, a lot of values will be detected as outliers.

We can clearly observe from the probability distribution of the continuous predictor variables that some variables are skewed, for example wind speed and humidity to a certain extent. The skew in this distribution can be most likely explained by the presence of outliers and extreme values in the data.

In figure below we have plotted the boxplots of the 4 continuous predictor variable. A lot of useful inferences can be made from these plots. It can clearly be seen from the plots that wind_speed is having lot of values which are lying far away from the median, which are represented by the red dots. These are the outliers in wind_speed variable. We

Can also see outliers in humidity variable. But since they are a very few, there effect on the model will be negligible. Hence, we only consider the outliers present in the wind_speed variable.

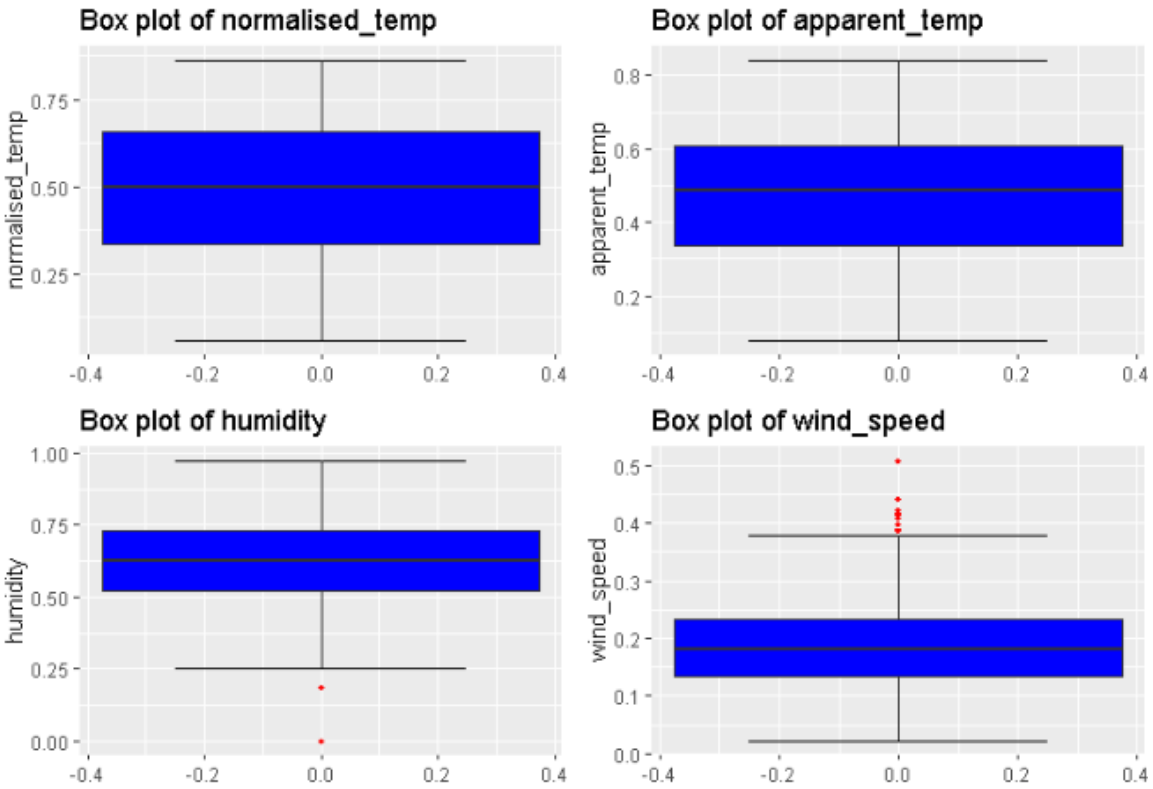

Figure 2.9: Outlier Analysis of Normalised Temperature, Apparent Temperature, Humidity and Windspeed

Outliers can be handles in ,any ways. One of the simplest ways to handle outliers is to drop that entire record from your dataset to keep that event from skewing your analysis. So I deleted entire record that has outliers in wind_speed.In the graphs below, we can clearly see the difference in the probability distribution of wind_speed with and without outliers.

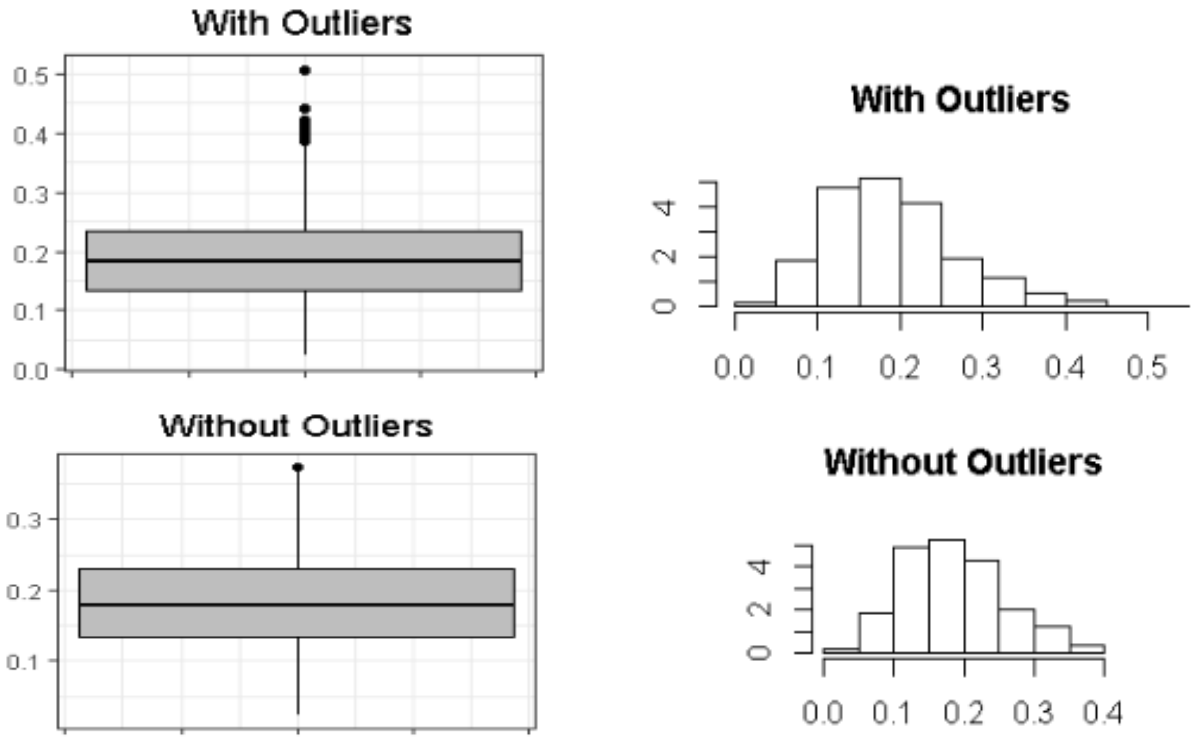

Figure 2.10: Distribution of Windspeed variable with and without outliers

2.4 Feature Selection

Feature Selection is the technique where we automatically or manually select those features which contribute most to our prediction variable or output which we are interested in. It is desirable to reduce the number of input variables to both reduce the computational cost of modelling and to improve the performance of the model.

We may face this problem of identifying the related features from the set of data and removing the irrelevant or less important features which do not contribute much to our target variable in order to achieve better accuracy for our model. Feature Selection is one of the core concepts in machine learning which hugely impacts the performance of our model. Irrelevant or partially relevant features can negatively impact the model performance. Here, I have used two techniques to select relevant features and remove the irrelevant ones.

2.4.1 Correlation Analysis

Correlation Analysis is a statistical method used to evaluate the strength of relationship between two quantitative variables. It's a good idea to look for data correlation to see how variables are interconnected. A high correlation means that two or more variables have a strong relationship with each other, while a weak correlation means the variables are hardly related. It is basically a statistical approach for modelling the association between a dependant variable, called response, and one or more independent variables. The correlation of the continuous variables can be analysed with the use of Correlation Matix. Correlation is expressed in the form of correlation coefficient. The correlation coefficient is measured on a scale of -1 to +1. Below is the correlation matrix plot between all the continuous variables in the dataset.

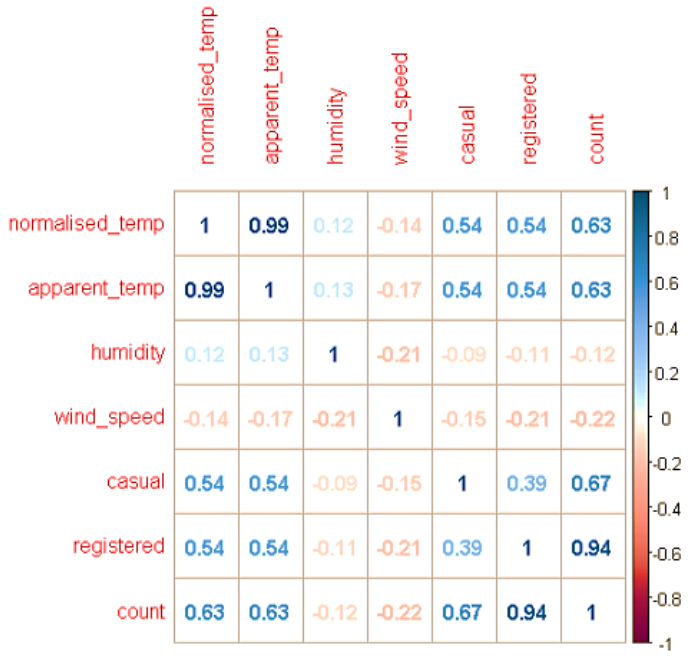

Figure 2.11: Correlation Matrix of continuous predictor variables.

We can derive following insights from this correlation matrix:

We find the correlation of 1 along the diagonal of the matrix. This is because each variable is highly and positively correlated with itself.

The is a positive correlation between apparent temperature and count and normalised temperature and count which means with increase in temperature, the count rate will increase and it has highest prediction power

Wind speed is negatively correlated with count which means that it has lowest prediction power.

We see that apparent temperature and normalised temperature are strongly and positively correlated.

Also, the total count, casual and registered are highly positively correlated with each other.

2.4.2 Multicollinearity

If we decide to use linear regression as our prediction model, we can't have multicollinearity and data distribution must be normal. In the presence of multicollinearity, the solution of the regression becomes unstable. Multicollinearity generally occurs when there are high correlations between two or more predictor variables. In other words, one predictor variable can be used to predict the other. This creates redundant information, skewing the results in regression model.

For a given predictor, multicollinearity can be assessed by computing a score called the Variance Inflation Factor(VIF), which measures how much the variance of the regression coefficient is inflated due to multicollinearity in the model. the smallest possible value of VIF is 1(absence of multicollinearity). As a rule of thumb, a VIF value that exceeds 5 or 10 indicates a problematic amount of collinearity. After performing multicollineaity test on the 6 continuous predictor variables namely apparent_temp, normalised_temp, wind_speed, humidity, casual and registered, we get the following result:

```
1 variables from the 6 input variables have collinearity problem:

apparent_temp

After excluding the collinear variables, the linear correlation coefficients ranges between:
min correlation ( casual ~ humidity ):  -0.09258584
max correlation ( casual ~ normalised_temp ):  0.5400295

---------- VIFs of the remained variables --------
        Variables      VIF
1 normalised_temp 1.845539
2        humidity 1.159188
3      wind_speed 1.119673
4          casual 1.496829
5      registered 1.545400
```

It tells us that apparent temperature and normalised temperature had very high VIF value, which means if both these variables are present in the model. This will cause the issue of multicollinearity. The similar result can also be seen from the correlation matrix where the correlation coefficient between these two variables is 0.99. Therefore, it is suggested to drop any one variable out of these to nullify the effect of multicollinearity in the model. The presence of multicollinearity implies that the information that apparent_temp provides about the count rate is redundant in the presence normalised_temp.

Therefore, before proceeding with the model development, we remove the features that add least information to our analysis. So, we drop:

sr.no: Using prior knowledge to remove features that don't add important information, which in this case is only the "sr.no" feature.

date: Using this feature, we have engineered a new column "daynum", which is more meaningful in predicting the bike rental demand, therefore "date" variable becomes of no use now.

season, month, week_day weather_situation, : As we have created binary columns for all these variables using One Hot Encoding method, we no longer require these variables.

apparent_temp : since apparent_temp results in multicollinearity, we drop this variable.

Casual, registered: we omit these variables as this is what we have to predict because count variable consist of aggregated values of these two columns.

Chapter 3

Modelling

3.1 Model Selection

In our early stages of analysis during pre-processing we have come to understand that our dependant variable is continuous, therefore, we need to do Regression Analysis.

Before diving into building statistical models, I partition my dataset into two sets, training and testing. Training set will be used to train statistical models and estimate coefficients while testing set will be used to validate the model we build with the training set. 80 % of the complete data is partitioned into training set, simply sampled randomly without replacement, and 20% of the data is partitioned into testing set. The resulting training set consists of 574 observations and testing set consists of 144 observations.

3.2. Model Development

For this project we are going to use three well-known Regression algorithm: Decision Tree Regression Model, Random Forest Regression Model and multiple Linear Regression Model.

3.2.1 Multiple Linear Regression Model

Multiple linear regression (MLR), also known simply as multiple regression, is a statistical technique that uses several explanatory variables to predict the outcome of a response variable. The goal of multiple linear regression (MLR) is to model the linear relationship between the explanatory (independent) variables and response (dependent) variable.

So here we train our training dataset by applying Linear regression model to it in order to predict the testing data set. We get the following results:

```
>
>     lm_model= lm(count~., data=training)
>     summary(lm_model)

Call:
lm(formula = count ~ ., data = training)

Residuals:
     Min       1Q   Median       3Q      Max
-3917.7  -361.1     70.2    478.0   2944.8

Coefficients: (4 not defined because of singularities)
                 Estimate Std. Error t value Pr(>|t|)
(Intercept)     -1252.028    561.339  -2.230 0.026124 *
year             1977.761     67.352  29.364  < 2e-16 ***
holiday          -278.669    231.534  -1.204 0.229275
working_day       388.424    126.314   3.075 0.002210 **
normalised_temp  4930.794    479.701  10.279  < 2e-16 ***
humidity        -1535.373    340.997  -4.503 8.21e-06 ***
wind_speed      -2658.337    489.407  -5.432 8.41e-08 ***
Month_1             1.021    208.411   0.005 0.996094
Month_2           110.899    215.875   0.514 0.607658
Month_3           334.169    210.945   1.584 0.113737
Month_4           335.855    281.473   1.193 0.233308
Month_5           527.224    297.089   1.775 0.076516 .
Month_6           304.805    300.119   1.016 0.310262
Month_7           -48.286    317.802  -0.152 0.879292
Month_8           289.752    305.204   0.949 0.342852
Month_9           869.102    250.704   3.467 0.000568 ***
Month_10          522.511    184.599   2.831 0.004818 **
Month_11          -87.555    168.108  -0.521 0.602698
Month_12               NA         NA      NA       NA
season_1        -1444.377    204.216  -7.073 4.67e-12 ***
season_2         -586.451    245.820  -2.386 0.017387 *
season_3         -782.091    225.443  -3.469 0.000563 ***
season_4               NA         NA      NA       NA
week_day_6        471.313    123.340   3.821 0.000148 ***
week_day_0             NA         NA      NA       NA
week_day_1       -108.261    130.276  -0.831 0.406328
week_day_2        -82.065    126.131  -0.651 0.515557
week_day_3        -14.030    126.636  -0.111 0.911824
week_day_4        -21.554    126.484  -0.170 0.864750
week_day_5             NA         NA      NA       NA
Weather_2        1636.332    200.048   8.180 2.01e-15 ***
Weather_1        2055.166    216.029   9.513  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 786.1 on 546 degrees of freedom
Multiple R-squared:  0.8428,    Adjusted R-squared:  0.8351
F-statistic: 108.4 on 27 and 546 DF,  p-value: < 2.2e-16
```

Here we get 4 variables that are not defined due to the singularity which means that these variables are not linearly independent. If we remove the variables that are giving NA in the above summary, we will obtain the same result for the rest of the variables.

Interpreting output:

Residuals: This section summarizes the residuals, the error between the prediction of the model and the actual result. Smaller residuals are better.

Coefficients: For each variable and the intercept, a weight is produced that has attributes like standard error, a t-test value and significance. For example: for every 1° C increase in normalised temperature, the model predicts an increase of 4931 count of bikes on daily basis.

Performance measure: Multiple or Adjusted R-Square: R-squared shows the amount of variance explained by the model. Therefore, R-squared indicates that 84.28 % of the variations in the count can be explained by the changes in the predictor variables, i.e., the count can be predicted with 84.28 % of error from the independent variables. Adjusted R-square tells about the goodness-of-fit of the model. Therefore, our MLR model is 83.5 % fit to predict the count rate.

MAPE of this multiple linear regression model is 15.26%. Hence the accuracy of this model is 84.74%. This model performs very well for this test data.

3.2.2 Decision Tree Regression Model

Decision tree is a type of supervised learning algorithm. It works for both categorical and continuous input and output variables. It beaks down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with decision nodes and leaf nodes.

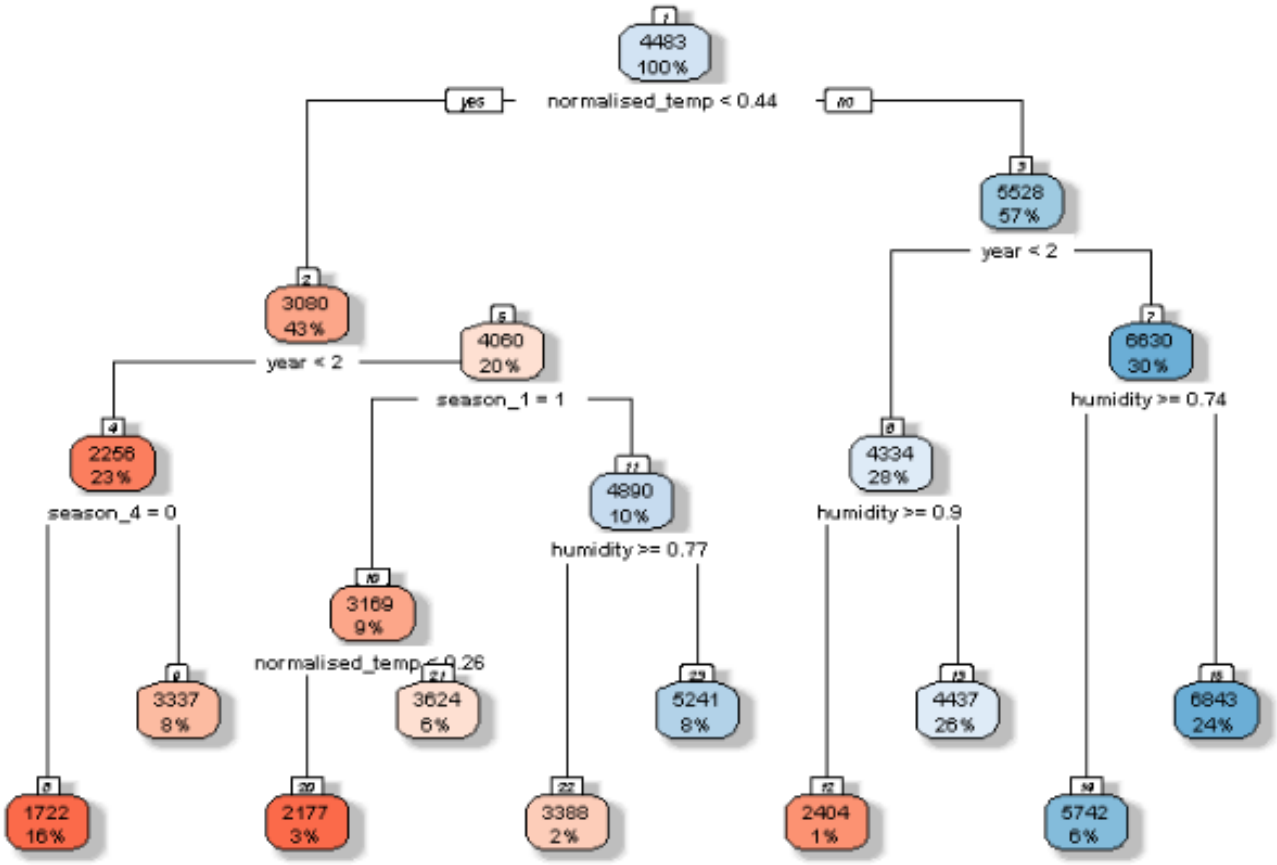

Figure 3.1: Decision Tree Model

Using decision tree, we can predict the value of bike count. The MAPE for this decision tree is 17.89%. Hence the accuracy for this model is 82.11%.

3.2.3 Random Forest

Decision trees are sensitive to the specific data on which they are trained. If the training data is changed the resulting decision tree can be quite different and in turn the predictions can be quite different. To address these weaknesses, we turn to Random Forest which illustrates the power of combining many decision trees into one model. Random forest is a Supervised Learning algorithm which uses ensemble learning method for classification and regression.

In this modelling, the number of decision trees used for prediction in the forest is 500. Using random forest, the MAPE was found to be 12.74%. Hence the accuracy is 87.26%.

Chapter 4

Conclusion

4.1 Model Evaluation

Ideally, the estimated performance of a model tells us how well it performs on unseen data. Making predictions on future data is often the main problem we want to solve. It's important to understand the context before choosing a metric because each machine learning model tries to solve a problem with a different objective using a different dataset. Model evaluation metrics are required to quantify model performance. The choice of evaluation metrics depends on a given machine learning task (such as classification, regression, ranking, clustering, topic modelling, among others).

4.1.1 MAE

Mean Absolute Error (MAE) measures the average magnitude of the errors in a set of predictions, without considering their direction. It's the average over the test sample of the absolute differences between prediction and actual observation where all individual differences have equal weight.

4.1.2 MAPE

Mean Absolute Percentage Error (MAPE) is a statistical measure of how accurate a forecast system is. It measures this accuracy as a percentage, and can be calculated as the average absolute percent error for each time period minus actual values divided by actual values. The mean absolute percentage error (MAPE) is the most common measure used to forecast error.

The MAE and MAPE values for all the three models are:

Decision Tree Model:	Random Forest Model:	Linear Regression Model:
MAE: 666.2107	MAE: 460.8144	MAE: 528.2087
MAPE: 17.89	MAPE: 12.77	MAPE: 15.26
Accuracy: 82.11 %	Accuracy: 87.23 %	Accuracy: 84.74 %

Based on the above error metrics, Random Forest is the better model for our analysis. Hence Random Forest is chosen as the model for prediction of bike rental count.

Below is the plot between the actual count values and the predicted values of count by Random Forest Model that we perform on the sample data.

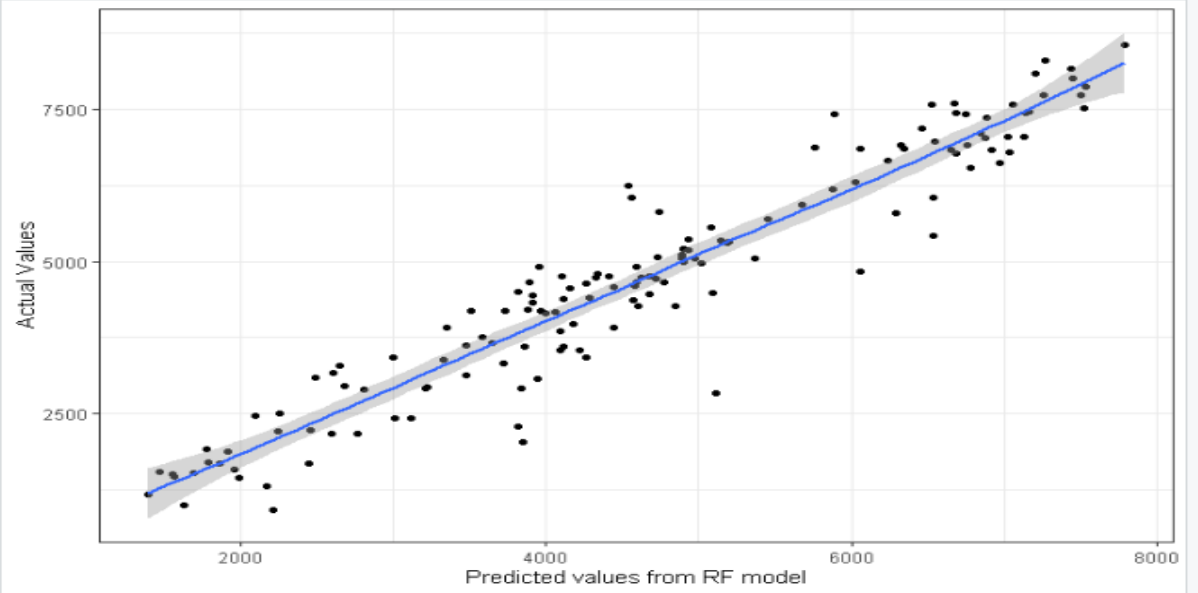

Figure 4.1: Plot between Actual Values and Predicted values of Random Forest Model

Appendix A - R Code

```
#Cleaning the Environment

rm(list=ls(all=T))

#Setting Working Directory

    setwd("D:/Data scientist/Project/Bike Rental")

    libraries=c("rpart.plot","tidyverse","ggplot2","ggExtra","corrplot","usdm","gridExtra","rpart","randomForest",'DMwR')

    lapply(libraries, require, character.only = TRUE)

    rm(libraries)

#Loading the csv file

    day=read.csv("day.csv", header=T, sep = ",")


#####################################Exploratory Analysis#################################

#First 10 rows of the dataset

    head(day,10)

#Contents of the data

    str(day)

    summary(day)

#Renaming Columns

    colnames(day)=c("sr_no","date","season","year","month","holiday","week_day","working_day","weather_situation","normalised_temp","apparent_temp","humidity","wind_speed","casual","registered","count")

#Datatype Conversion

    day$month=as.factor(as.numeric(day$month))

    day$working_day=as.factor(as.numeric(day$working_day))

    day$year=as.factor(as.numeric(day$year))

    day$season=as.factor(as.numeric(day$season))

    day$week_day=as.factor(as.numeric(day$week_day))

    day$month=as.factor(as.numeric(day$month))

    day$holiday=as.factor(as.numeric(day$holiday))

    day$weather_situation=as.factor(as.numeric(day$weather_situation))

#####################################Data Visualization#####################################

# Months affecting Count based on Season
```

```
ggplot(day,aes(month,count)) + labs(x='Months',y= 'Daily Usage')+geom_count(aes(color=season)) +
geom_point(alpha=0.5, color='green')+theme_bw()+ggtitle("Scatter Plot: Months vs Usage based on
Season")+theme(plot.title = element_text(hjust = 0.5))+geom_smooth()
# Year affecting Count
ggplot(day,aes(year,count), fill=year) + labs(x="Year: 2011(0) and 2012(1)",y= 'Daily Usage')+
geom_boxplot(aes(color=year),alpha=0.2) + theme_bw()+ggtitle("Box Plot: Year vs Usage")+
theme(plot.title = element_text(hjust = 0.5))
# Working and Non-Working Days affecting Count
ggplot(day,aes(working_day,count)) + labs(x='Working(1) and Non-Working Days(0)',y= 'Daily
Usage')+geom_boxplot(aes(color=working_day),alpha=0.2) + theme_bw()
# Normalised Feeling temperature affecting Count
ggMarginal(ggplot(day, aes(x=normalised_temp,y=count))+geom_point(aes(color=normalised_
temp),alpha=0.5)+scale_color_gradient(high='green',low='red')+labs(x='Normalised temperature',
y= 'Daily Usage')+ theme_bw()+geom_smooth(method = 'auto')+geom_smooth(method='lm',
color='red'),type = "histogram",fill="blue", margins = c("x"))
# Normalised normalised_temperature affecting Count
ggMarginal(ggplot(day, aes(x=apparent_temp, y=count))+geom_point(aes(color=apparent_temp),
alpha=0.5)+ scale_color_gradient(high='black',low='sky blue')+labs(x='Apparent temperature',y=
'Daily Usage')+geom_smooth(method = 'auto')+theme_bw()+geom_smooth(method='lm',
color='red'),type = "histogram",fill="blue", margins = c("x"))
# Weather situations affecting Count
ggplot(day,aes(weather_situation,count)) + labs(x='Weather Situation',y= 'Daily Usage')+
geom_boxplot(aes(color=weather_situation),alpha=0.2) + theme_bw()
# Humidity affecting Count
ggMarginal(ggplot(day, aes(x=humidity, y=count))+geom_point(aes(color=humidity),alpha=0.5)
+scale_color_gradient(high='blue',low='yellow')+labs(x='Humidity Level',y= 'Daily Usage')+
geom_smooth(method = 'auto')+geom_smooth(method='lm', color='red')+theme_bw(),type =
"histogram",fill="blue", margins = c("x"))
# Wind speed affecting Count
ggMarginal(ggplot(day, aes(x=wind_speed, y=count))+geom_point(aes(color=wind_speed),
alpha=0.9)+scale_color_gradient(high='orange',low='grey')+labs(x='Wind Speed',y= 'Daily Usage')
+ geom_smooth(method = 'auto')+geom_smooth(method='lm', color='red')+theme_bw(),type =
"histogram",fill="blue", margins = c("x"))
```

```
###################################Feature Engineering#####################################
#Extracting day number from the date
        day$daynum= format(as.Date(day$date,format="%d-%m-%Y"),"%d")
        day$daynum = as.factor(as.numeric(day$daynum))
###################################Missing Value Analysis#####################################
        apply(day,2,function(x){ sum(is.na(x))})
#########################One Hot Encoding for  Categorical Variables#############################
        for(i in unique(day$month)){
        day[[paste0("Month_",i)]]=ifelse(day$month==i,1,0)
        }
        for(i in unique(day$season)){
        day[[paste0("season_",i)]]=ifelse(day$season==i,1,0)
        }
        for(i in unique(day$week_day)){
        day[[paste0("week_day_",i)]]=ifelse(day$week_day==i,1,0)
        }
        for(i in unique(day$weather_situation)){
        day[[paste0("Weather_",i)]]=ifelse(day$weather_situation==i,1,0)
        }
###################################Outlier Analysis#########################################
# BoxPlots - Distribution and Outlier Check
        num_index = sapply(day,is.numeric)
        num_data = day[,num_index]
        var_name = colnames(num_data)
        for (i in 1:(length(var_name)))
        {
        assign(paste0("plot",i), ggplot(aes_string(y = (var_name[i])), data = subset(day))+
        stat_boxplot(geom = "errorbar", width = 0.5) +geom_boxplot(outlier.colour="red", fill = "blue"
        ,outlier.shape=18,outlier.size=1, notch=FALSE, orientation = "x") +theme(legend.position="bottom")
        +labs(y=var_name[i])+ ggtitle(paste("Box plot of",var_name[i])))
        }
        grid.arrange(plot2,plot3,plot4,plot5,ncol=2)
#Distribution of windspeed with outlier
```

```
ggplot(day,aes(y=wind_speed))+geom_boxplot(color="black",fill="grey",outlier.size=1.5)+theme_bw
()+ggtitle("With Outliers")+theme(plot.title = element_text(hjust = 0.5))+labs(x="",y="")
hist(day$wind_speed, main="With Outliers",xlab=NA, ylab=NA, prob=TRUE)
#Removing Outliers from wind_speed
val = day$wind_speed[day$wind_speed %in% boxplot.stats(day$wind_speed)$out]
day = day[which(!day$wind_speed %in% val),]
#Distribution of windspeed without outlier
ggplot(day,aes(y=wind_speed))+geom_boxplot(color="black",fill="grey", outlier.size=1.5)+
theme_bw()+ggtitle("Without Outliers")+theme(plot.title = element_text(hjust = 0.5))+labs
(x="",y="")
hist(day$wind_speed, main="Without Outliers",xlab=NA, ylab=NA, prob=TRUE)
#Reordering Columns by position
day=day[,c(1:15,17:42,16)]
####################################Feature Selection#########################################
#Methods used:
#1. Correlation Analysis
#2. Multicolinearity test
#Visualizing the Correlation Matrix
colnames(day)
numerical_var=day[c(10:15,42)]
corrplot(cor(numerical_var),method='number')
#Multicollinearity Test
mcoll_test=day[,c("normalised_temp","apparent_temp","humidity","wind_speed","casual",
"registered")]
vifcor(mcoll_test)
#Dimension Reduction
colnames(day)
day=day[-c(1,2,3,5,7,9,11,14,15,16)]
#Converting types of Categorical variables into numeric
day$year=as.numeric(as.factor(day$year))
day$holiday=as.numeric(as.factor(day$holiday))
day$working_day=as.numeric(as.factor(day$working_day))
####################################Model Development#########################################
```

```
  #1. Desicion Tree
  #2. Random Forest
  #3. Linear Regression
# Partitioning dataset into training/validation and test set
        set.seed(111)
        train_index = sample(1:nrow(day), 0.8 * nrow(day))    #Simple random sampling
        training = day[train_index,]
        testing = day[-train_index,]
#MULTIPLE LINEAR REGRESSION MODEL
testing = day[-train_index,]
#Train the data using Linear Regression model
        lm_model= lm(count~., data=training)
        summary(lm_model)
#Predict the test cases
        lm_predictions=predict(lm_model,testing[])
#Calculate MAE
        mae_LM=regr.eval(testing[,33],lm_predictions,stats= c('mae'))
#Writing a function to calculate MAPE
        mape=function(y, yhat)
                {
                mean(abs(y-yhat)/y)*100
                }
#Calculate MAPE
        mape_LM=mape(testing[,33],lm_predictions)
        cat("Linear Regression Model:\nMAE:",abs(mae_LM),"\nMAPE:",round(mape_LM,2),"\nAccuracy:",
round(100-mape_LM,2),"%")
#DECISION TREE REGRESSION
#Train the data using Decision Tree
        dt_model=rpart(count~.,data = training, method = "anova")
#Visualise the model
        rpart.plot(dt_model, box.palette = "RdBu",shadow.col="gray",nn=TRUE)
#Predict the Test cases
        dt_predictions= predict(dt_model,testing[,-33])
```

```
#Calculate MAE
        mae_DT=regr.eval(testing[,33],dt_predictions,stats= c('mae'))
#Calculate MAPE
        mape_DT=mape(testing[,33],dt_predictions)
        cat("Decision Tree Model:\nMAE:",abs(mae_DT),"\nMAPE:",round(mape_DT,2),"\nAccuracy:",
        round(100-mape_DT,2),"%")
#RANDOM FOREST REGRESSION
#Train the data using Random Forest
        rf_model=randomForest(count~.,training,imporatnce=TRUE, ntree=500)
#Plotting Error Graph
        plot(rf_model)
#Predict the Test Cases
        rf_predictions=predict(rf_model,testing[,-33])
#Calculate MAE
        mae_RF=regr.eval(testing[,33],rf_predictions,stats= c('mae'))
#Calculate MAPE
        mape_RF=mape(testing[,33],rf_predictions)
        cat("Random Forest Model:\nMAE:",abs(mae_RF),"\nMAPE:",round(mape_RF,2),"\nAccuracy:",
        round(100-mape_RF,2),"%")
# Checking Results on Sample Data
        sample_input_data=data.frame(testing)
        sample_output_data=cbind(testing$count,rf_predictions)
        sample_output_data=data.frame(sample_output_data)
        colnames(sample_output_data)=c("Actual_values","Predicted_Values")
#Plotting a graph between Actual Values and Predicted Values
        ggplot(aes(abs(Predicted_Values),Actual_values),data = sample_output_data)+geom_point()
        +geom_smooth(method='loess')+xlab("Predicted values from RF model")+ylab("Actual
         Values")+theme_bw()
        write.csv(sample_input_data,"Sample_input.csv")
        write.csv(sample_output_data,"Sample_output.csv")
```

Appendix B - Python Code

```
#Load libraries
import os
import datetime
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from statsmodels.stats.outliers_influence import variance_inflation_factor

#Set working directory
os.chdir("D:\Data scientist\Project\Bike Rental")


#Load data
day = pd.read_csv("day.csv")

#Read first 10 Data
day.head(10)

#Renaming the columns
day.rename(columns={'instant':'sr.no','dteday':'date','yr':'year','mnth':'month','weathersit':'weather_situatio
n','hum':'humidity','temp':'normalised_temp','atemp':'apparent_temp','cnt':'count'},inplace=True)

#Datatype Conversion
day['date']=pd.to_datetime(day.date, format= '%d-%m-%Y')
day['season']=day['season'].astype('category')
day['year']=day['year'].astype('category')
day['month']=day['month'].astype('category')
day['holiday']=day['holiday'].astype('category')
day['weekday']=day['weekday'].astype('category')
day['workingday']=day['workingday'].astype('category')
day['weather_situation']=day['weather_situation'].astype('category')
```

Exploratory Data Analysis

Data visualisation

```
# Months affecting Count based on Season
month_count=sns.catplot("month","count",hue="season",data=day)
month_count.set(xlabel="Non-Working and Working Day",ylabel="Daily count", title="Working Day affecting
Daily Count")

# Year affecting Count
year_cnt=sns.catplot("year","count",data=day)
year_cnt.set(xlabel="Year",ylabel="Daily count", title="Year affecting Daily Count")

# Season affecting Count based on working and non-working day
wday_count=sns.catplot("workingday","count",data=day)
wday_count.set(xlabel="Non-Working and Working Day",ylabel="Daily count", title="Working Day affecting
Daily Count")
```

```
# Normalised Feeling Temperature affecting Count
temp_cnt=sns.jointplot(x="normalised_temp",y="count",color='#000000',kind="reg",data=day)
temp_cnt.set_axis_labels("Normalised Feeling Temperature","Daily Count")
temp_cnt.fig.suptitle("Normalised Feeling Temperature affecting Count")
temp_cnt.fig.subplots_adjust(top=0.9)

# Apparent Temperature affecting Count
atemp_cnt=sns.jointplot(x="apparent_temp",y="count",color='#009999',kind="reg",data=day)
atemp_cnt.set_axis_labels("Apparent Temperature","Daily Count")
atemp_cnt.fig.suptitle("Apparent Temperature affecting Count")
atemp_cnt.fig.subplots_adjust(top=0.9)

# Weather situations affecting Count
weathersit_cnt=sns.catplot("weather_situation","count",data=day)
weathersit_cnt.set(xlabel="Weather Situation",ylabel="Daily count", title="Weather Situation affecting Daily Count")

#Humidity affecting Count
hum_cnt=sns.jointplot(x="humidity",y="count",color='#00CC00',kind="reg",data=day)
hum_cnt.set_axis_labels("Humidity Level","Daily Count")
hum_cnt.fig.suptitle("Humidity affecting Count")
hum_cnt.fig.subplots_adjust(top=0.9)

#Windspeed affecting Count
windspd_cnt=sns.jointplot(x="windspeed",y="count",color='#6666FF',kind="reg",data=day)
windspd_cnt.set_axis_labels("Wind Speed","Daily Count")
windspd_cnt.fig.suptitle("Wind Speed affecting Count")
windspd_cnt.fig.subplots_adjust(top=0.9)
```

Data Pre-processing

Feature Engineering

```
#Extracting day of the month from the date
day['daynum']=day['date'].dt.day

#Dropping dteday column
day.drop(["date"], axis = 1, inplace = True)
```

Missing Value Analysis

```
#Creating dataframe with missing percentagee
missing_val = pd.DataFrame(day.isnull().sum())

#Reset index
missing_val = missing_val.reset_index()

#Rename variable
missing_val = missing_val.rename(columns = {'index': 'Variables', 0: 'Missing_percentage'})

#Calculate percentage
missing_val['Missing_percentage'] = (missing_val['Missing_percentage']/len(day))*100
```

```
#descending order
missing_val = missing_val.sort_values('Missing_percentage', ascending = False).reset_index(drop = True)
missing_val
```

One Hot Encoding

```
day=pd.get_dummies(day, prefix=None, prefix_sep='_', dummy_na=False, columns=["season","month","weekday","weather_situation"], sparse=False, drop_first=False, dtype=int)
```

Outlier Analysis

```
# Plotting boxplot to visualize Outliers
#Checking for outliers in Apparent Temperature
%matplotlib inline
plt.boxplot(day['apparent_temp'])

#Checking for outliers in Normalised Temperature
%matplotlib inline
plt.boxplot(day['normalised_temp'])

#Checking for outliers in Normalised Humidity
%matplotlib inline
plt.boxplot(day['humidity'])

#Checking for outliers in Wind Speed
%matplotlib inline
plt.boxplot(day['windspeed'])

#Detect and delete outliers from data
x=["humidity","windspeed"]
for i in x:
    print(i)
    q75, q25 = np.percentile(day.loc[:,i], [75 ,25])

    #Calculating inner and outer fence
    iqr = q75 - q25
    min = q25 - (iqr*1.5)
    max = q75 + (iqr*1.5)
    print(min)
    print(max)

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

#Reordering Column by position
day.insert(37,"count",day.pop("count"))
day.head().transpose()
```

Feature Selection

Correlation Analysis

```
numerical_var= ["normalised_temp","apparent_temp","humidity","windspeed","casual","registered","count"]
```

```
day_corr = day.loc[:,numerical_var]

#Set the width and hieght of the plot
f,ax = plt.subplots(figsize=(7, 5))

#Generate correlation matrix
corr = day_corr.corr()

#Plot using seaborn library
sns.heatmap(cmap=sns.diverging_palette(220, 10, as_cmap=True),square=True, ax=ax, data=corr)
```

Multicollinearity

```
multi_corr_var= ["normalised_temp","apparent_temp","humidity","windspeed","casual","registered"]
multi_corr = day.loc[:,multi_corr_var]

#function calculating VIF for each value of x and save in dataframe
def calc_vif(x):
    vif=pd.DataFrame()
    vif["variables"]=x.columns
    vif["VIF"]=[variance_inflation_factor(x.values,i)
    for i in range(x.shape[1])]
    return (vif)
calc_vif(multi_corr)
```

Dimension Reduction

```
#Dropping Apparent_temp to reduce the multicollinearity
day=day.drop(['sr.no','daynum','apparent_temp','casual','registered'],axis=1)
```

Model Development

```
#Store the data in the form of dependent and independant variables seperately
X = day.values[:,0:32]
Y = day.values[:,32]

#Import Train_test_split
from sklearn.model_selection import train_test_split

#Split data into 80% train and 20 % test
X_train, X_test, Y_train, Y_test= train_test_split(X,Y,test_size=0.2,random_state=3)
```

Decision Tree Regression

```
#Import DecisionTreeRegressor
from sklearn.tree import DecisionTreeRegressor

#Initiate a Decision Tree Regressor
DT_model=DecisionTreeRegressor(max_depth=4,min_samples_leaf=0.1,random_state=3)

#Fit Decision Tree Regressor to training set
DT_model.fit(X_train,Y_train)

#Predict test set labels
prediction_DT=DT_model.predict(X_test)
```

```
# To Compute test set R-square value
from sklearn import metrics

#Creating a function to calculate MAPE
def MAPE(y_true, y_pred):
   mape=np.mean(np.abs((y_true-y_pred)/y_true))*100
   return mape

#print error metrics
r_square=metrics.r2_score(Y_test, prediction_DT)
print('R-square Value for Decision Tree Model is ',r_square)

mape_DT=MAPE(Y_test, prediction_DT)
print("The MAPE for Decision Tree Model is ",round(mape_DT,2),".")

accuracy_DT=100-mape_DT
print("The Accuracy for Decision Tree Model is ",round(accuracy_DT,2),"%.")
```

Random Forest Regression Model

```
from sklearn.ensemble import RandomForestRegressor

#Initiate a Random Forest Regressor
RF_model = RandomForestRegressor(n_estimators = 1000, random_state = 42)

#Fit Decision Tree Regressor to training set
RF_model.fit(X_train, Y_train);

# Use the forest's predict method on the test data
prediction_RF = RF_model.predict(X_test)

#print Error Metrics
r_square=metrics.r2_score(Y_test, prediction_RF)
print('R-square Value for Random Forest Regression Model is ',r_square)

mape_RF=MAPE(Y_test, prediction_RF)
print("The MAPE for Random Forest Regression Model is ",round(mape_RF,2),".")

accuracy_RF=100-mape_RF
print("The Accuracy for Random Forest Regression Model is ",round(accuracy_RF,2),"%.")
```

Multiple Linear Regression Model

```
#import Linear Regression machine learning library
from sklearn.linear_model import LinearRegression
from sklearn import linear_model

#invoke the linear Regression function
LR_model = linear_model.LinearRegression()

#Fit Linear Regression model to training set
LR_model.fit(X_train,Y_train)
```

```
#Use linear model prediction method on test data
prediction_LR=LR_model.predict(X_test)

#print Error Metrics
r_square=metrics.r2_score(Y_test, prediction_LR)
print('R-square Value for Linear Regression Model is ',r_square)

mape_LR=MAPE(Y_test, prediction_LR)
print("The MAPE for Linear Regression Model is ",round(mape_LR,2),".")

accuracy_LR=100-mape_LR
print("The Accuracy for Linear Regression Model is ",round(accuracy_LR,2),"%.")
```

References

The Element of Statistical Learning, by Trevor Hastie, Robert Tibshirani, Jerome Friedman.

R for Data Sciences, by Garret Grolemund and Hadley Wickham

A Byte of python, By C.H.Swaroop