Prediction of Bike Rental Count

Prakash B

26th May 2019

Contents

Chapter 1: Introduction

1.1 Problem Statement

The objective of this problem is to Predict number of bike rental count on daily based on the environmental and seasonal settings, by predicting the count it will be easy to manage the number bikes required on the daily basis and preparing the bike demand based on the Environmental Changes.

1.2 Data

Our task is to build Regression model which will predict the bike rental count on daily based on the environmental and seasonal settings. Given below is a sample of the data set that we are using to predict the bike Rental count:

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

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)

hr: Hour (0 to 23)

holiday: weather day is holiday or not (extracted fromHoliday Schedule)

weekday: Day of the week

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

weathersit: (extracted from Freemeteo)

1: Clear, Few clouds, Partly cloudy, Partly cloudy

2: Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist

3: Light Snow, 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_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

Table 1.1: Bike Count of Sample Data (Columns 1-9)

instant	dteday	season	yr	mnth	holiday	weekday	workingday	weathersit
1	2011-01-01	1	0	1	0	6	0	2
2	2011-01-02	1	0	1	0	0	0	2
3	2011-01-03	1	0	1	0	1	1	1
4	2011-01-04	1	0	1	0	2	1	1
5	2011-01-05	1	0	1	0	3	1	1
6	2011-01-06	1	0	1	0	4	1	1
7	2011-01-07	1	0	1	0	5	1	2

Table 1.2: Bike Count of Sample Data (Columns 10-16)

temp	atemp	hum	windspeed	casual	registered	cnt
0.3441670	0.3636250	0.805833	0.1604460	331	654	985
0.3634780	0.3537390	0.696087	0.2485390	131	670	801
0.1963640	0.1894050	0.437273	0.2483090	120	1229	1349
0.2000000	0.2121220	0.590435	0.1602960	108	1454	1562
0.2269570	0.2292700	0.436957	0.1869000	82	1518	1600
0.2043480	0.2332090	0.518261	0.0895652	88	1518	1606
0.1965220	0.2088390	0.498696	0.1687260	148	1362	1510

As you can see in the table below, we have the following 13 variables, using which we have to correctly predict the quality of the wines:

Table 1.3: Predictor variables

Sl.No	**Variables**
1	Instant
2	Dteday
3	Season
4	Yr
5	Month
6	Holiday
7	Weekday
8	Workingday
9	Weathersit
10	Temp
11	Atemp
12	Hum
13	windspeed

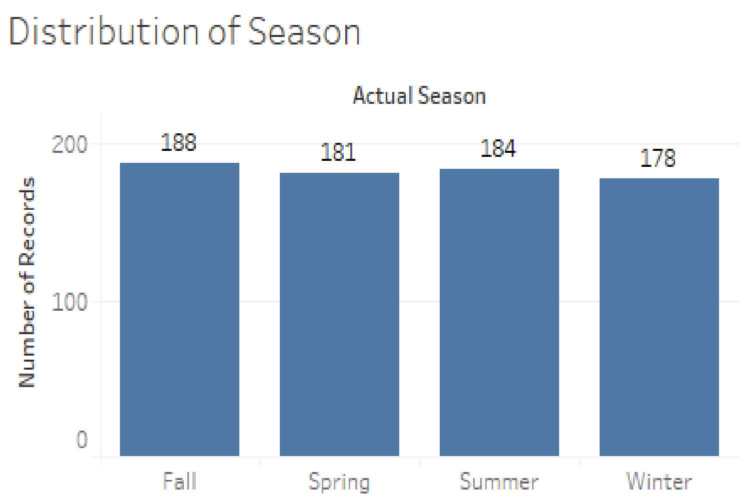
Chapter 2: Methodology

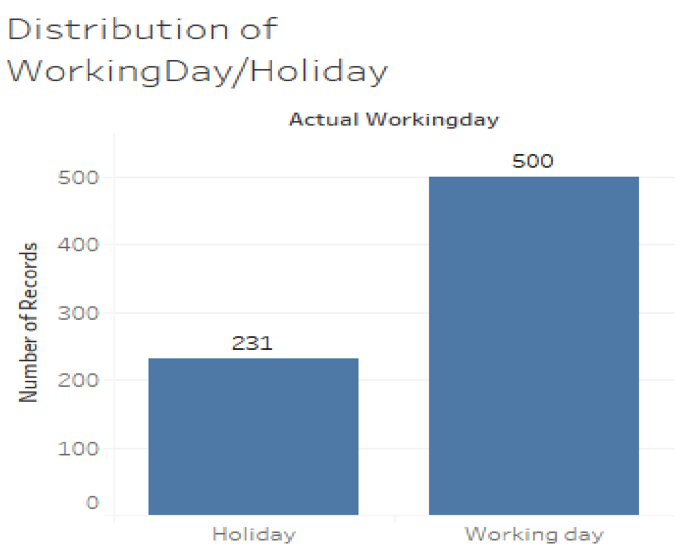
2.1 Data Pre-Processing

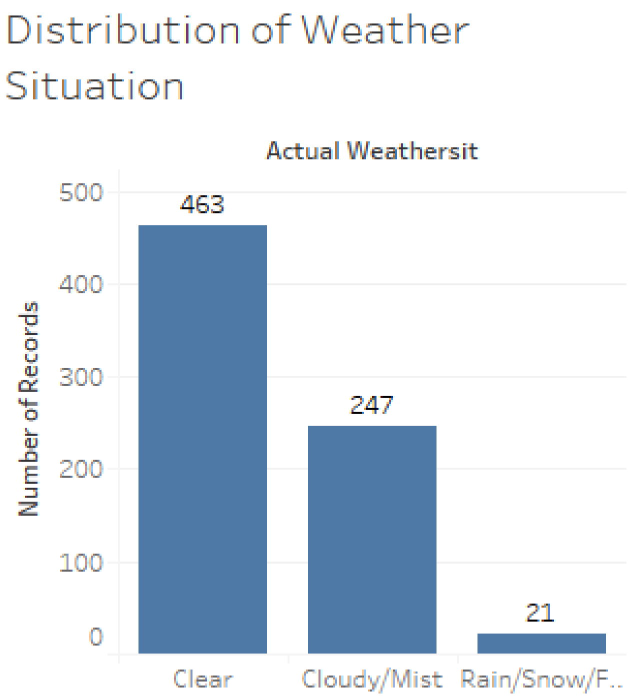
Any predictive modelling requires that we look at the data before we start modelling. However, in data mining terms looking at data refers to so much more than just looking, looking at data refers to exploring the data, cleaning the data as well as visualizing the data through graphs and plots. This 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.

2.1.1 Distribution of Categorical Variables

The below bar graph shows the Distribution of Categorical variables in the dataset.







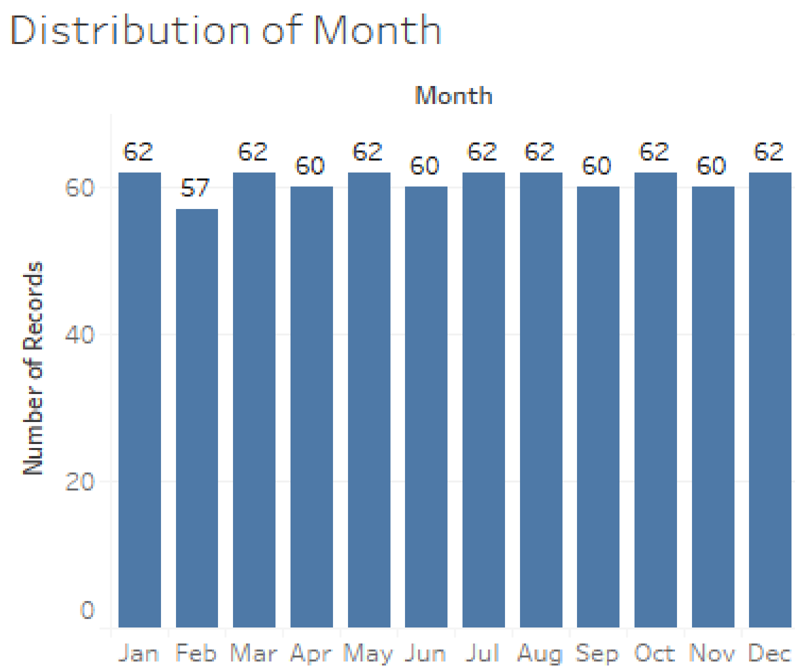
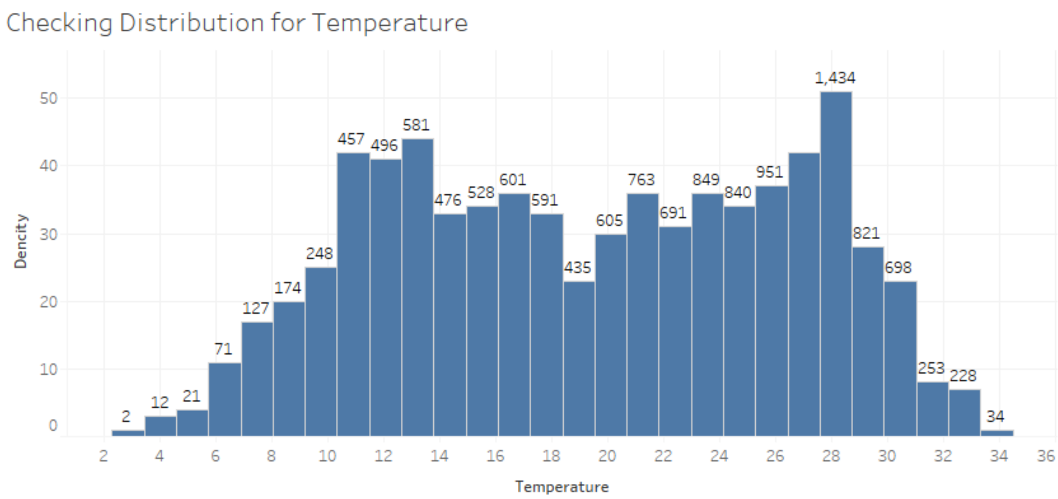


Fig 2.1: Distribution of categorical variables using bar graphs

2.1.2 Distribution of Continuous Variables

From the below distributions it can be observed that temperature and feel temperature are normally distributed, whereas the variables windspeed and humidity are slightly skewed. The skewness is likely because of the presence of outliers and extreme data in those variables.



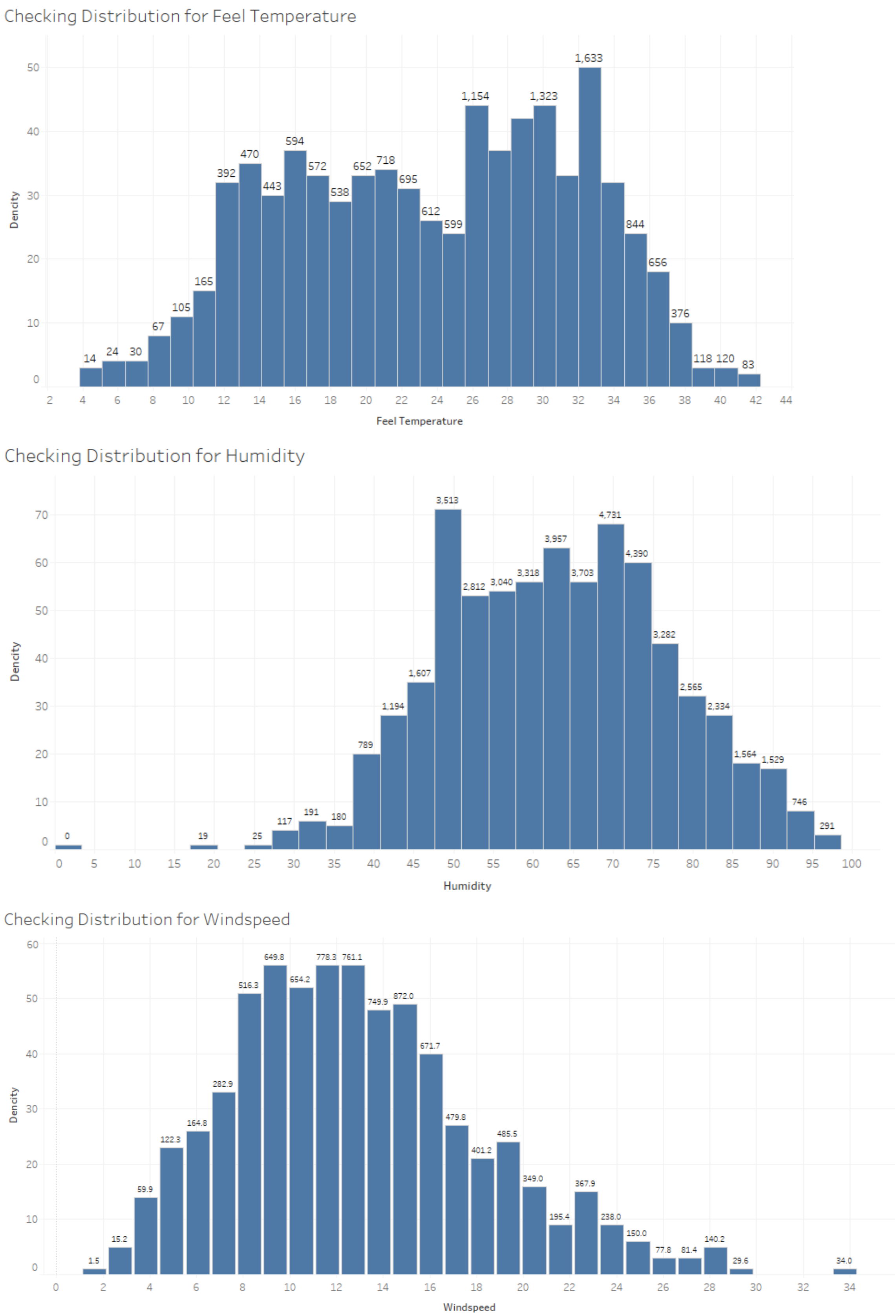
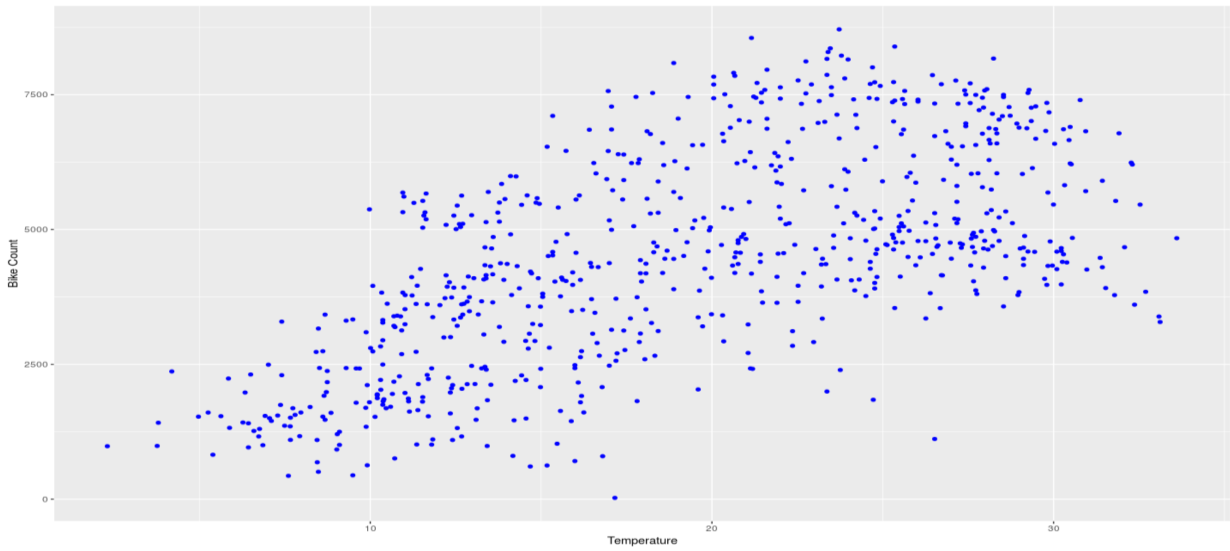


Fig 2.2: Distribution of continuous variables using histograms

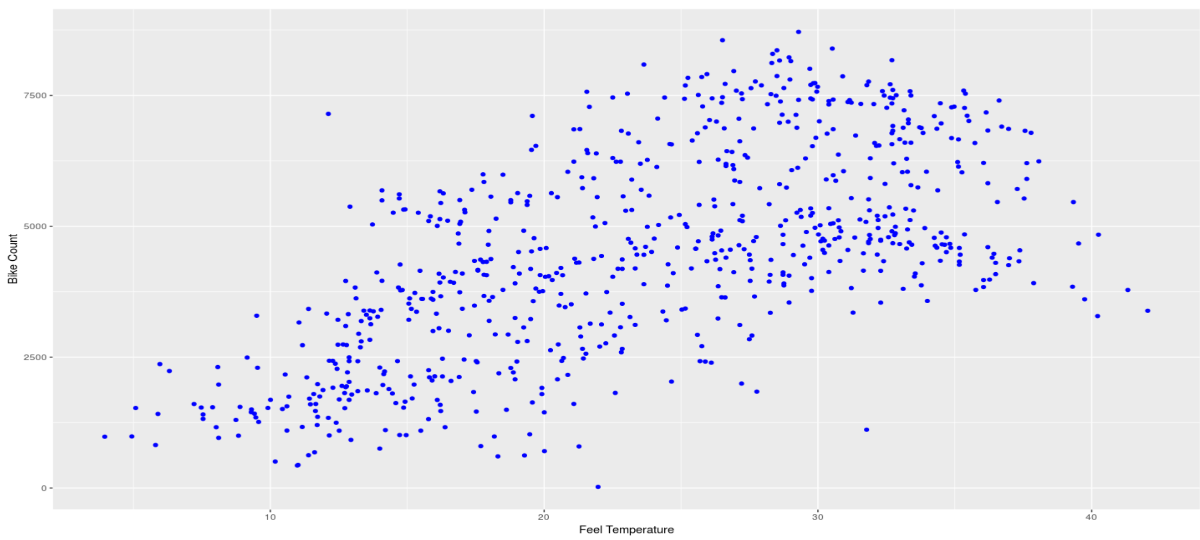
2.1.3 Relationship of Continuous Variable against target Variable

The Below figure shows the relationship between the continuous variables against target variable (bike count) using Scatter plot, from the figure we can see that how the data is scattered in temperature and feel temperature (atemp), and there exists a linear positive relationship between the variables temperature and feel temperature against the target variable, and also linear negative relationship between humidity and windspeed with the target variable.

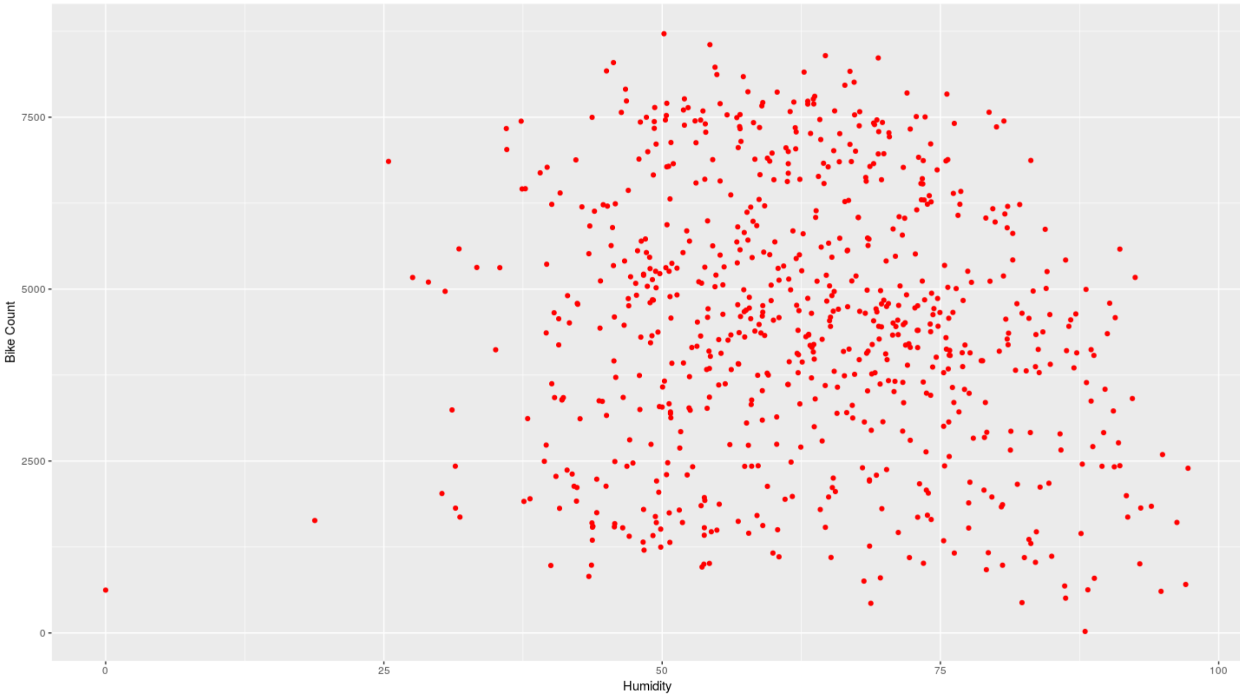
Scatterplot for Temperature



Scatterplot for Feel Temperature



Scatterplot for Humidity



Scatterplot for Wind speed

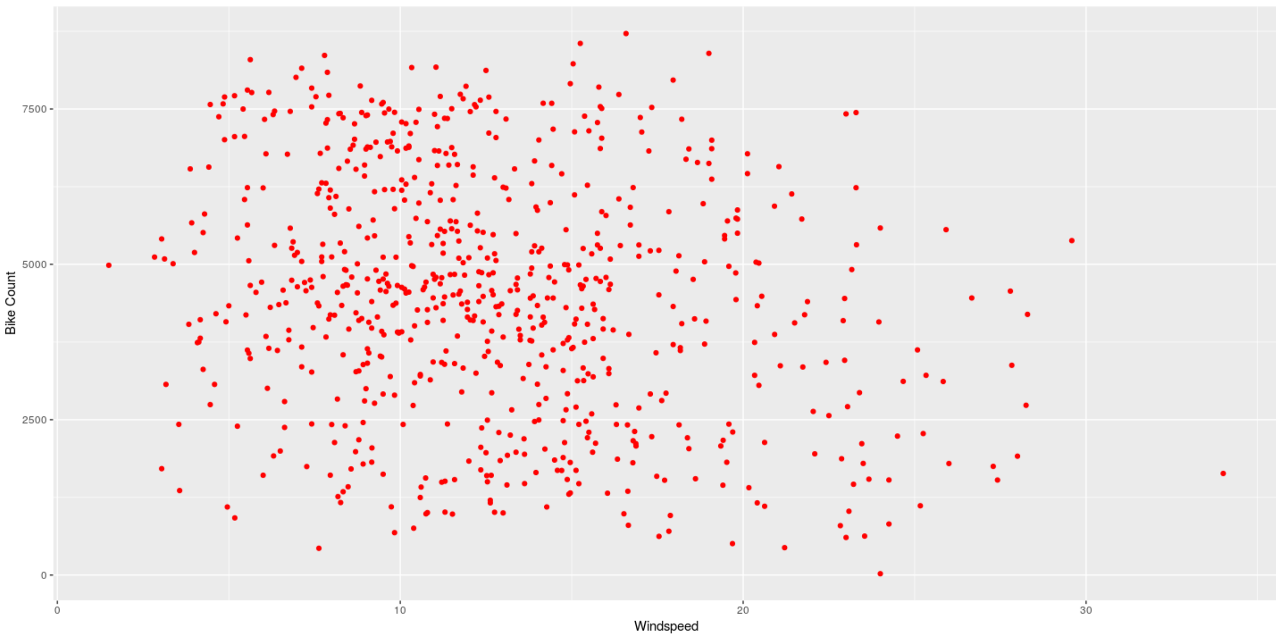
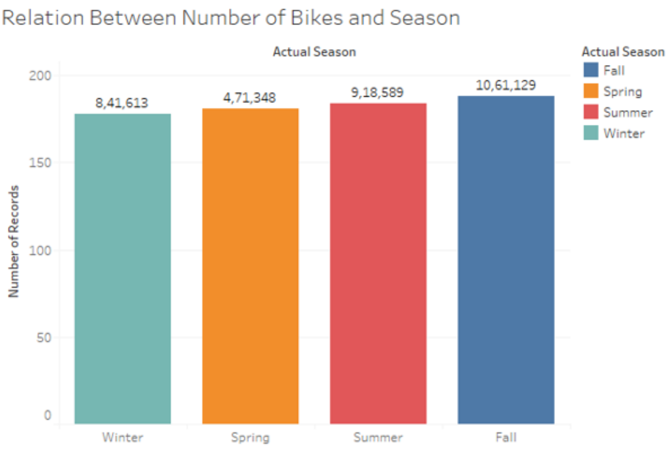
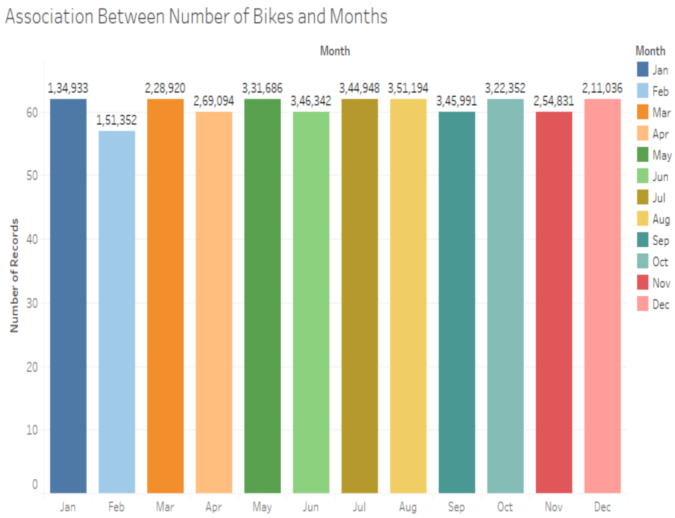
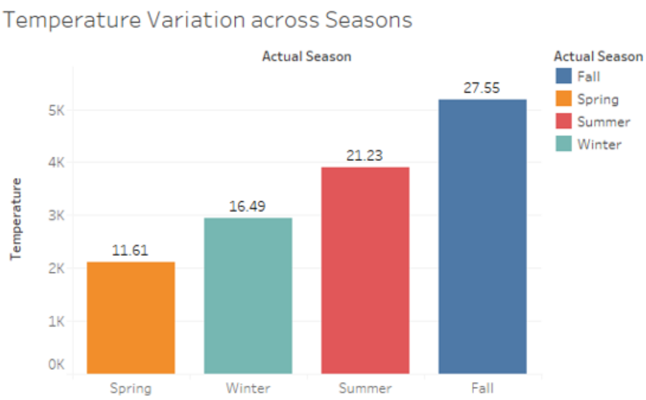


Fig 2.3: Scatter plot for continuous variables.







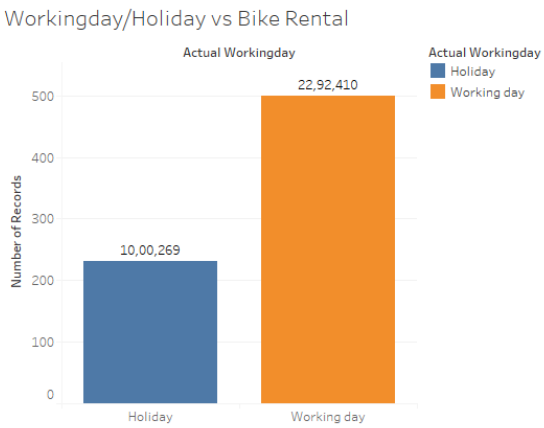


Fig 2.4: Relationship between the variables against bike count

From the above figures we can observed that how the bike rental count varies across the different seasons and in that fall season have more bike rental count, the temperature also varies across the season and its mean also changes in all seasons, when it comes to working day and holiday there are more number of bike rental count when the day is working day.

2.1.4 Missing Value Analysis

Missing data or missing values occur when no data value is stored for the variable in an observation. Missing values are a common occurrence in data analysis. These values can have a significant impact on the results or conclusions that would be drawn from these data. If a variable has more than 30% of its values missing, then those values can be ignored, or the column itself is ignored. In our case, there is no missing values occurred in the dataset.

	variable	Missing_values
0	dteday	0
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

Fig 2.5: Missing value analysis

2.1.5 Outlier Analysis

It can be observed from the distribution of variables that almost all of the variables are normally distributed. The skew in these distributions can be explained by the presence of outliers and extreme values in the data. One of the steps in pre-processing involves the detection and removal of such outliers. In this project, we use boxplot to visualize and remove outliers.

Outliers can be removed using the Boxplot stats method, wherein the Inter Quartile Range (IQR) is calculated and the minimum and maximum value are calculated for the variables. Any value ranging outside the minimum and maximum value are discarded

Variables windspeed and humidity contain outliers, the below figure shows the boxplot representation of the variables with outliers.

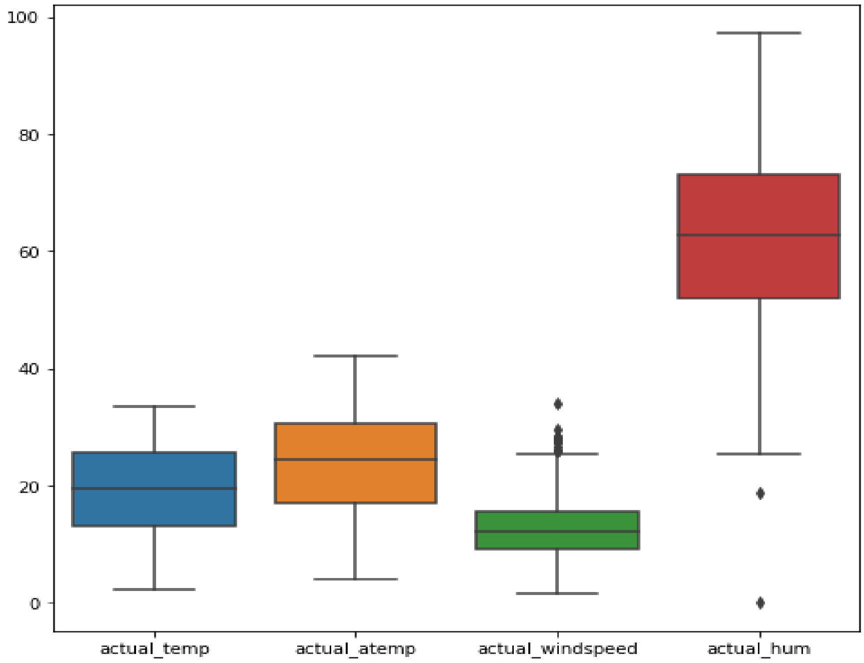


Fig 2.6: Boxplot of continuous variables with outliers

The below figure shows the boxplot representation of the variables without outliers.

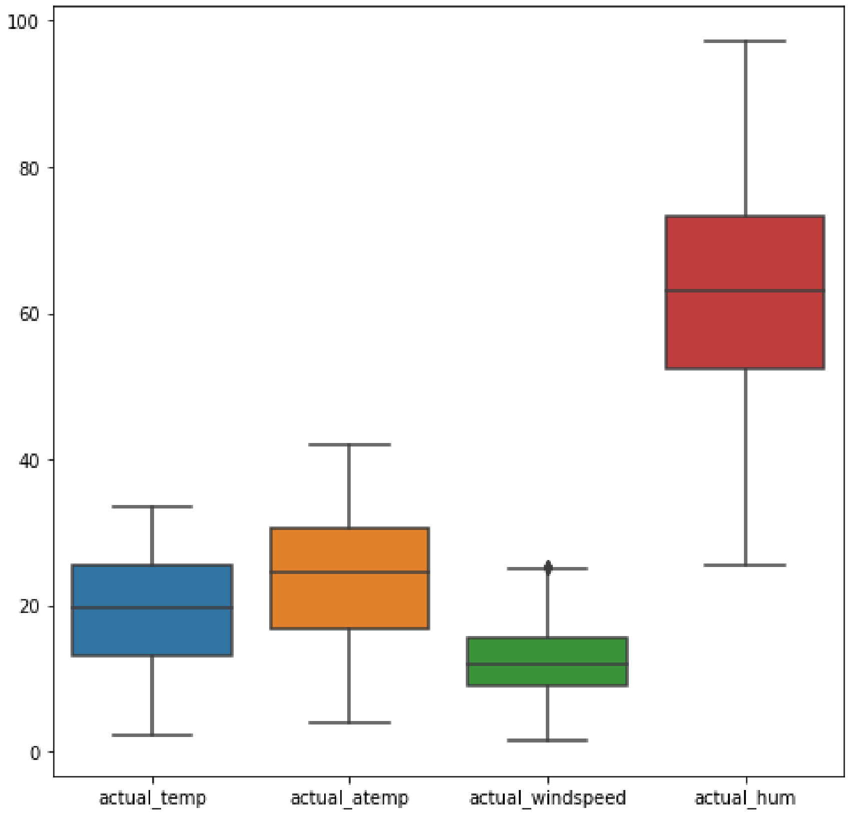
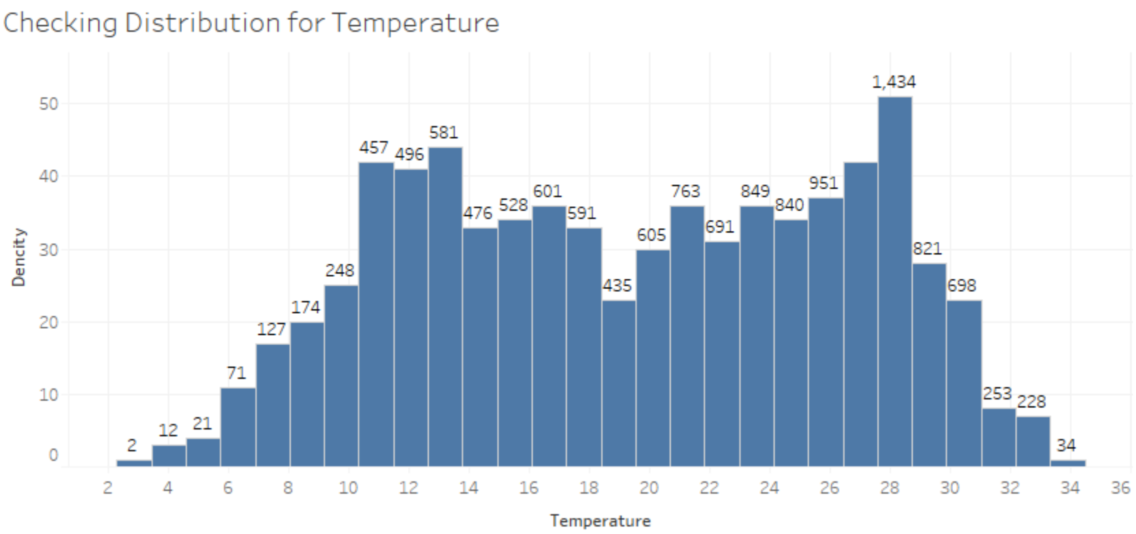
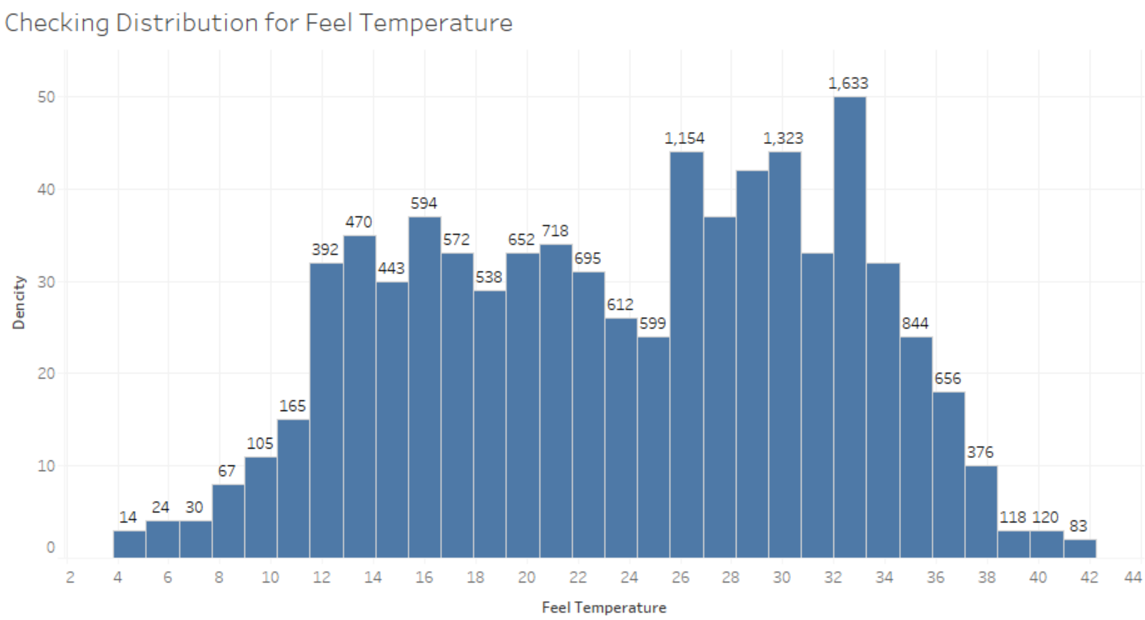
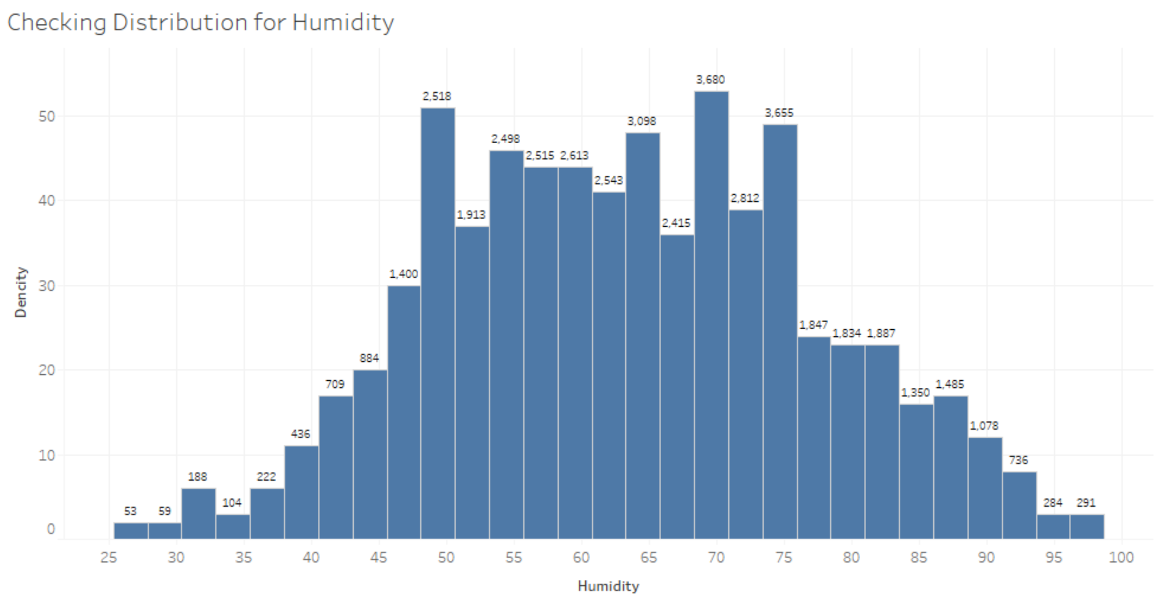


Fig 2.7: Boxplot of continuous variables without outliers

The Below histograms shows the distribution of continuous variables without outliers







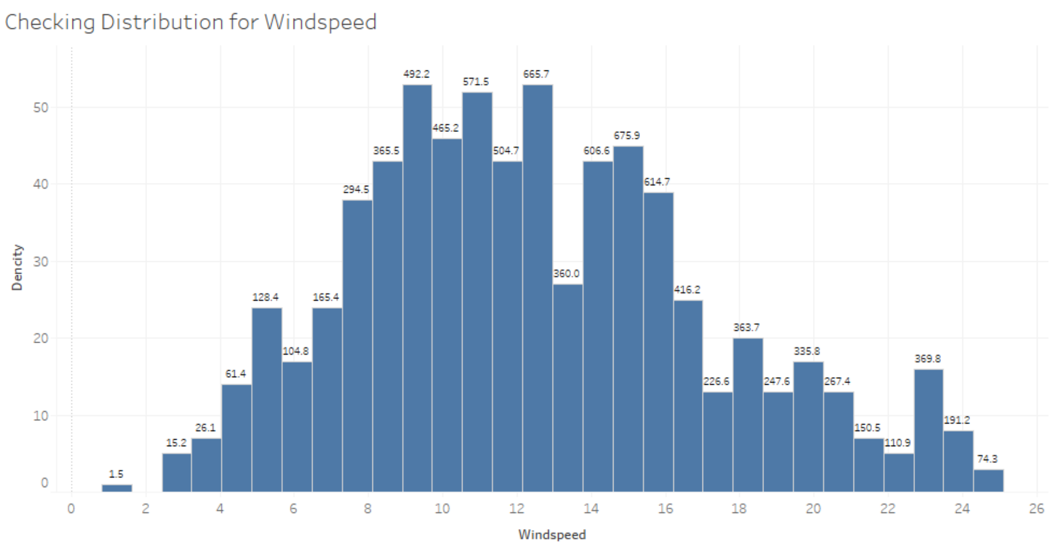


Fig 2.8: Distribution of continuous variables after removing outliers

2.1.6 Feature Selection

Feature Selection reduces the complexity of a model and makes it easier to interpret. It also reduces overfitting. Features are selected based on their scores in various statistical tests for their correlation with the outcome variable. Correlation plot is used to find out if there is any multicollinearity between variables. The highly collinear variables are dropped and then the model is executed.

From correlation analysis we have found that **Temperature** and **atemp** have highly correlated each other (>=0.9), so we have excluded the **atemp**, and **workingday** variable carrying holiday information so excluded **holiday** variable and also casual, registered are not required.

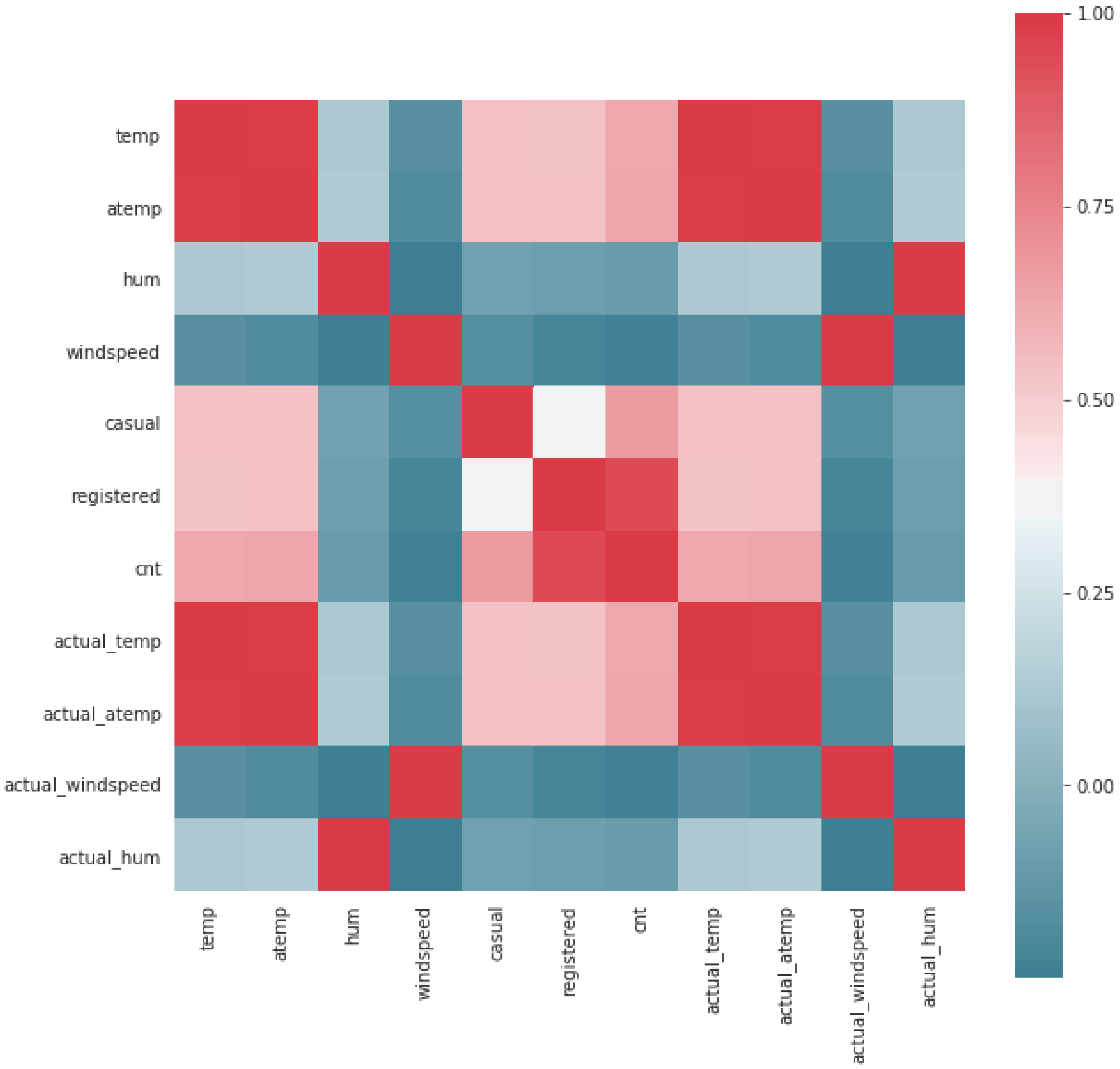


Fig 2.9: Correlation plot of Continuous variables

Let us analyse for categorical variables:

We would not use instant and dteday column, as instant is index only and information of dteday i.e. yr, month and day we have already columns for that. Dteday is important factor while doing time series analysis and we are not doing time series analysis.

For getting dependency between categorical variables we would use chi-square test of independence test. Basically we have mnth, yr, weekday, holiday, workingday, weathersit and season as categorical variables. We will use all the variables to check their dependency.

Null Hypothesis for Chi-square test of Independence:

Two variables are independent.

Alternate Hypothesis:

Two variables are not independent.

So, we want that our categorical variable should be independent. If we get p-value less than 0.05, it means we need to reject null hypothesis and accept alternate hypothesis which means variables are dependent. So, we would check for every combination and would print p-value.

	season	yr	mnth	holiday	weekday	workingday	weathersit
season	-	0.999	0.0	0.641	1.0	0.946	0.013
yr	0.999	-	1.0	0.995	1.0	0.956	0.183
mnth	0.0	1.0	-	0.571	1.0	0.993	0.01
holiday	0.641	0.995	0.571	-	0.0	0.0	0.599
weekday	1.0	1.0	1.0	0.0	-	0.0	0.249
workingday	0.946	0.956	0.993	0.0	0.0	-	0.294
weathersit	0.013	0.183	0.01	0.599	0.249	0.294	-

Analysis:

Here, from the table we can see some values are less than 0.05 indicating them they are dependent, **holiday** is showing collinearity with **weekday** and **workingday**, **month** is showing collinearity with **season** and **weathersit**, **weathersit** showing collinearity with season.

Now, we need to drop them to remove multicollinearity. But we have to be sure that we are not losing any information. So, we tried with dropping multicollinear column and building models and we got below result:

- On dropping weathersit or season we are losing accuracy with significant amount.
- On dropping holiday we are losing accuracy which is not significant amount.

- On dropping weekday or working day we are losing accuracy with little amount. And if we drop holiday, that information is also included in working day.

We just can't be sure by looking at test result and deciding, we need to get all factors. Here two columns most of the time showing collinearity but some data points would not be showing collinearity and at that data point there could be abrupt difference in bike renting count column which is very important information for our model.

We will drop only holiday column as we have working day column which has information of holiday also, that is why on dropping holiday we are not losing our accuracy.

Let us analyse for continuous variables using VIF (Variance inflation Factor):

Checking VIF for continuous variables:

```
const        46.4
temp         63.3
atemp        63.9
hum           1.1
windspeed     1.1
dtype: float64
```

From above results the atemp have multicollinear with temp.

After removing atemp lets check again once for VIF.

```
const        41.6
temp          1.0
hum           1.1
windspeed     1.1
dtype: float64
```

Now we have good value of VIF, don't have multicollinearity and const not a part of our dataset it was just added as it is required to calculate VIF

2.2 Modelling

2.2.1 Model Selection

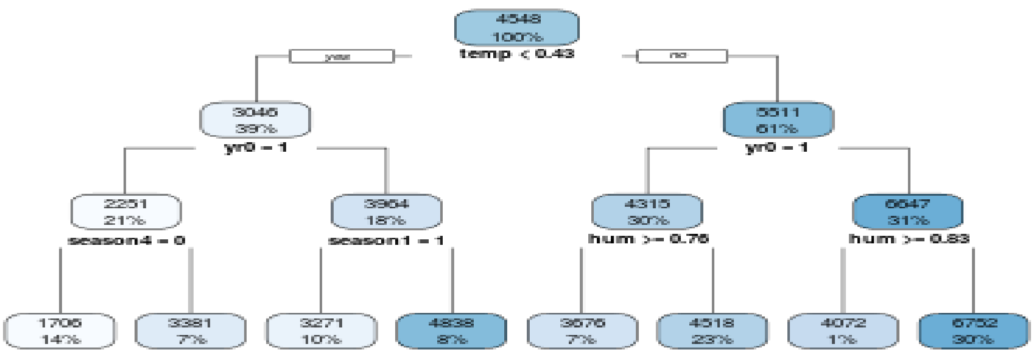
After a thorough pre-processing, we will be using some regression models on our processed data to predict the target variable. The target variable in our model is a continuous variable i.e. **cnt** (Bike count).Hence the models that we choose are Linear Regression, Decision Tree and Random Forest. The error metric chosen for the given problem statement is Root Mean Squared Error(RMSE) and R^2(R-Squared).

2.2.2 Decision Tree

Decision Tree algorithm belongs to the family of supervised learning algorithms. Decision trees are used for both classification and regression problems.
A decision tree is a tree where each node represents a feature (attribute), each link (branch) represents a decision (rule) and each leaf represents an outcome (categorical or continues value). The general motive of using Decision Tree is to create a training model which can use to predict class or value of target variables by learning decision rules inferred from prior data (training data).

The RMSE values and R^2 values for the given project in R and Python are:

Decision Tree	RMSE	R^2
R	961	0.76
PYTHON	962	0.73



Using decision tree in R, we can predict the value of bike count. RMSE and R2 for this model is 961 and 0.76, The MAPE for this decision tree is 25.93 %. Hence the accuracy for this model is 74.07%.

2.2.3 Multiple Linear Regression

Multiple linear regression is the most common form of linear regression analysis. Multiple linear regression is used to explain the relationship between one continuous dependent variable and two or more independent variables. The independent variables can be continuous or categorical.

Multiple Linear Regression	**RMSE**	**R^2**
R	**847**	**0.84**
PYTHON	**876**	**0.78**

The below figure shows the complete summary of the model:

```
Call:
lm(formula = cnt ~ ., data = train)

Residuals:
    Min      1Q  Median      3Q     Max 
-3996.7  -361.1    72.5   423.1  2908.4 

Coefficients: (6 not defined because of singularities)
             Estimate Std. Error t value Pr(>|t|)    
(Intercept)   4139.13     451.93   9.159  < 2e-16 ***
season1      -1142.91     209.39  -5.458 7.31e-08 ***
season2       -637.05     244.41  -2.606 0.009399 ** 
season3       -646.36     217.77  -2.968 0.003128 ** 
season4            NA         NA      NA       NA    
yr0          -2013.19      65.28 -30.839  < 2e-16 ***
yr1                NA         NA      NA       NA    
mnth1         -381.68     214.34  -1.781 0.075515 .  
mnth2         -208.13     213.48  -0.975 0.330030    
mnth3          318.59     217.37   1.466 0.143321    
mnth4          603.80     283.35   2.131 0.033543 *  
mnth5          937.89     299.64   3.130 0.001841 ** 
mnth6          691.88     301.19   2.297 0.021987 *  
mnth7          194.25     315.69   0.615 0.538587    
mnth8          609.74     304.55   2.002 0.045769 *  
mnth9         1194.56     250.14   4.775 2.31e-06 ***
mnth10         645.67     182.75   3.533 0.000446 ***
mnth11         -65.44     170.63  -0.384 0.701472    
mnth12             NA         NA      NA       NA    
weekday0      -398.95     121.82  -3.275 0.001124 ** 
weekday1      -822.75     195.11  -4.217 2.90e-05 ***
weekday2      -793.43     218.92  -3.624 0.000317 ***
weekday3      -712.84     218.35  -3.265 0.001165 ** 
weekday4      -696.34     217.36  -3.204 0.001436 ** 
weekday5      -691.77     217.82  -3.176 0.001579 ** 
weekday6           NA         NA      NA       NA    
workingday0   -654.59     184.10  -3.556 0.000410 ***
workingday1        NA         NA      NA       NA    
weathersit1   2077.30     231.93   8.957  < 2e-16 ***
weathersit2   1570.94     213.83   7.347 7.46e-13 ***
weathersit3        NA         NA      NA       NA    
temp          3969.38     466.69   8.505  < 2e-16 ***
hum          -1358.58     344.36  -3.945 9.02e-05 ***
windspeed    -2868.88     477.30  -6.011 3.38e-09 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 758.1 on 545 degrees of freedom
Multiple R-squared:  0.8512,	Adjusted R-squared:  0.8438 
F-statistic: 115.4 on 27 and 545 DF,  p-value: < 2.2e-16
```

As you can see the Adjusted R-squared value, we can explain 84.38% of the data using our multiple linear regression model. By looking at the F-statistic and combined p-value we can reject the null hypothesis that target variable does not depend on any of the predictor variables. This model explains the data very well and is considered to be good.

Even after removing the non-significant variables, the accuracy, Adjusted R-squared and F-statistic do not change by much, RMSE is calculated and found to be 847.82.

MAPE of this multiple linear regression model is 20.20%. Hence the accuracy of this model is 79.80%. This model performs very well for this test data.

2.2.4 Random Forest

Random Forest is a supervised learning algorithm. Random forest builds multiple decision trees and merges them together to get a more accurate and stable prediction. It can be used for both classification and regression problems. The method of combining trees is known as an ensemble method. Assembling is nothing but a combination of weak learners (individual trees) to produce a strong learner.
The number of decision trees used for prediction in the **forest** is 500.

Random Forest	**RMSE**	**R^2**
R	**693**	**0.88**
PYTHON	**669**	**0.87**

Using Random forest for prediction analysis in this case the number of decision trees used for prediction in the forest is 500. RMSE for this model is 693.

Using random forest, the MAPE was found to be 16.42%, Hence the accuracy is 83.58%

2.2.5 HyperParameter Tuning

Now, we will tune our model i.e. Random Forest, we will tune our model for whole dataset i.e. bike_data. With the help of hyperparameter tuning we would find optimum values for parameter used in function and would increase our accuracy.

Base model performance

```
<---Model Performance--->
R-Squared Value = 0.87
RMSE = 669.23
MAPE = 12.89
Accuracy = 87.11%.
```

Hyperparameter tuned model performance

```
<---Model Performance--->
R-Squared Value = 0.94
RMSE = 461.70
MAPE = 9.61
Accuracy = 90.39%.
```

Parameters of the Hyperparameter tuned model

```
{'bootstrap': True,
 'max_depth': 16,
 'max_features': 'auto',
 'min_samples_leaf': 2,
 'min_samples_split': 8,
 'n_estimators': 1000}
```

Analysis:

From above result (on tuned parameter), we have increased our model accuracy from 87.11% to 90.39%. Also we can observe that previously it was giving R^2 0.87 and now it is giving 0.94, also the RMSE also reduced from 669.23 to 461.70, On the test dataset we have slightly increased accuracy. In the parameters we can observed that n_estimators (tree numbers) changed from 500 to 1000.

So, with the help of hyperparameter tuning we have increased performance of model.

Chapter 3: Conclusion

3.1 Model Evaluation

In the previous chapter we have seen the Root Mean Square Error (RMSE) and R-Squared Value of different models. Root Mean Square Error (RMSE) is the standard deviation of the residuals (prediction errors). Residuals are a measure of how far from the regression line data points are, RMSE is a measure of how spread out these residuals are. In other words, it tells you how concentrated the data is around the line of best fit. Whereas R-squared is a relative measure of fit, RMSE is an absolute measure of fit. As the square root of a variance, RMSE can be interpreted as the standard deviation of the unexplained variance and has the useful property of being in the same units as the response variable. Lower values of RMSE and higher value of R-Squared Value indicate better fit.

3.1.1 RMSE and R-Squared

Root Mean Square Error (RMSE) is the standard deviation of the residuals (prediction errors). Residuals are a measure of how far from the regression line data points are, RMSE is a measure of how spread out these residuals are. In other words, it tells you how concentrated the data is around the line of best fit. Root mean square error is commonly used in climatology, forecasting, and regression analysis to verify experimental results.

$$\textbf{RMSE} = \sqrt{\overline{(\text{ predicted values} - \text{observed values })^2}}$$

R-squared (R2) is a statistical measure that represents the proportion of the variance for a dependent variable that's explained by an independent variable or variables in a regression model. Whereas correlation explains the strength of the relationship between an independent and dependent variable, R-squared explains to what extent the variance of one variable explains the variance of the second variable. So, if the R2 of a model is 0.50, then approximately half of the observed variation can be explained by the model's inputs.

3.1.2 Model selection

Multiple Linear Regression: RMSE = 847, R^2 = 0.84

Decision Tree: RMSE = 961, R^2 = 0.76

Random Forest: RMSE = 693, R^2 = 0.88

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.

Appendix A

Extra Figures

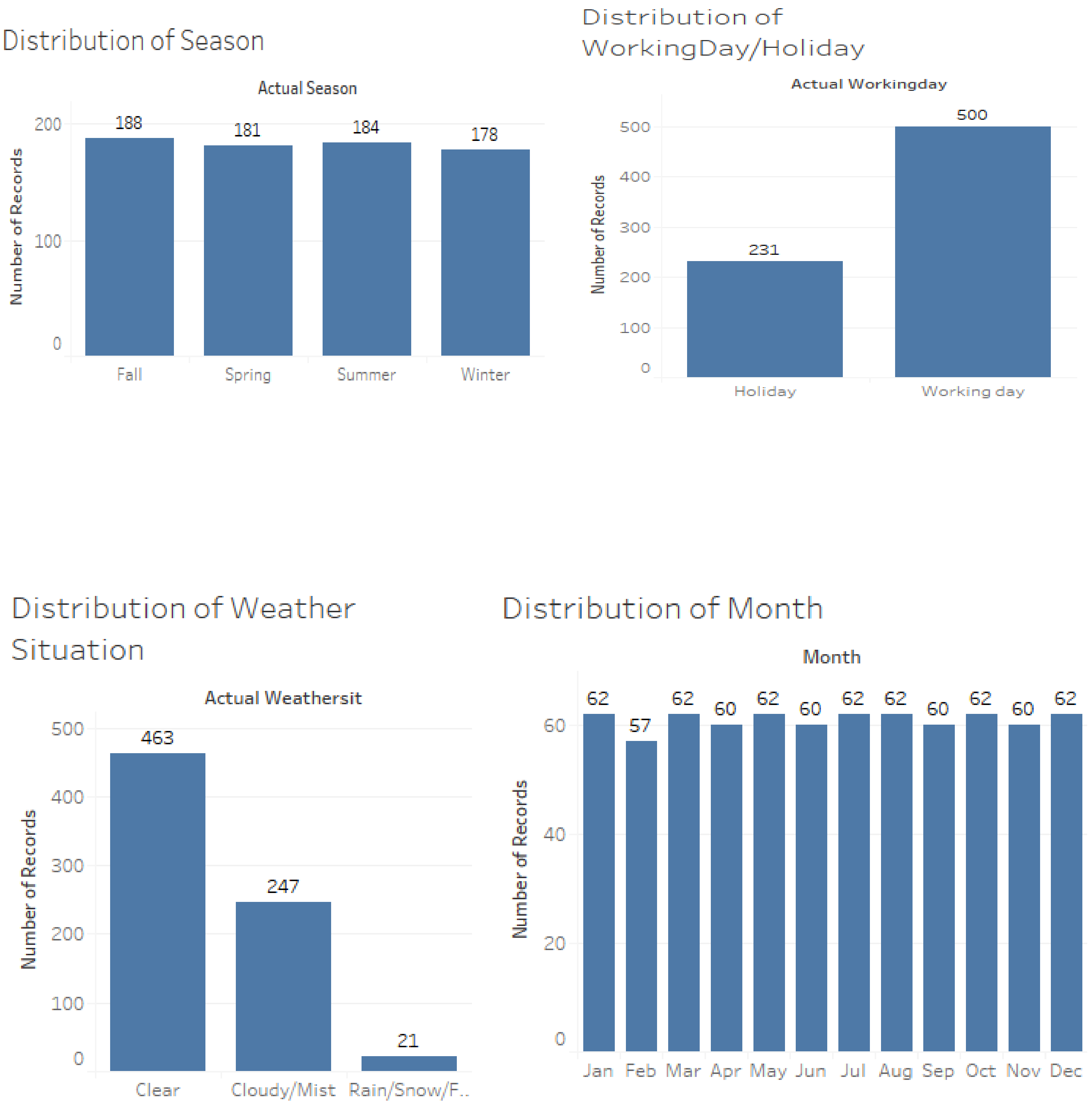
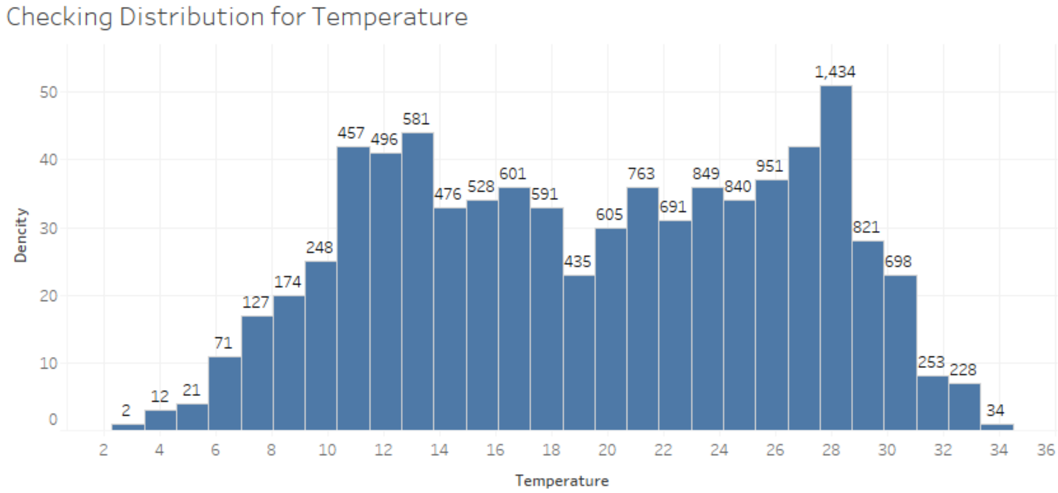
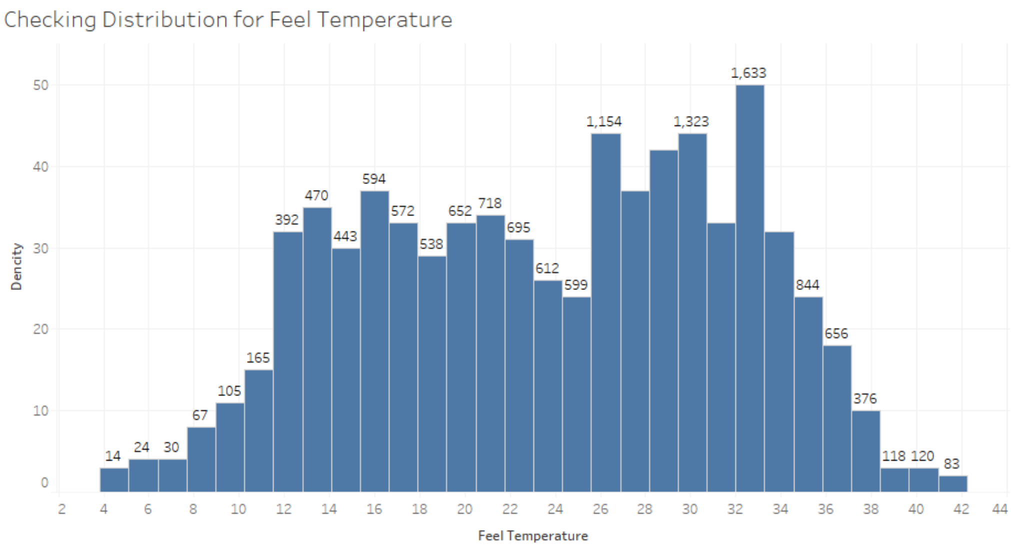


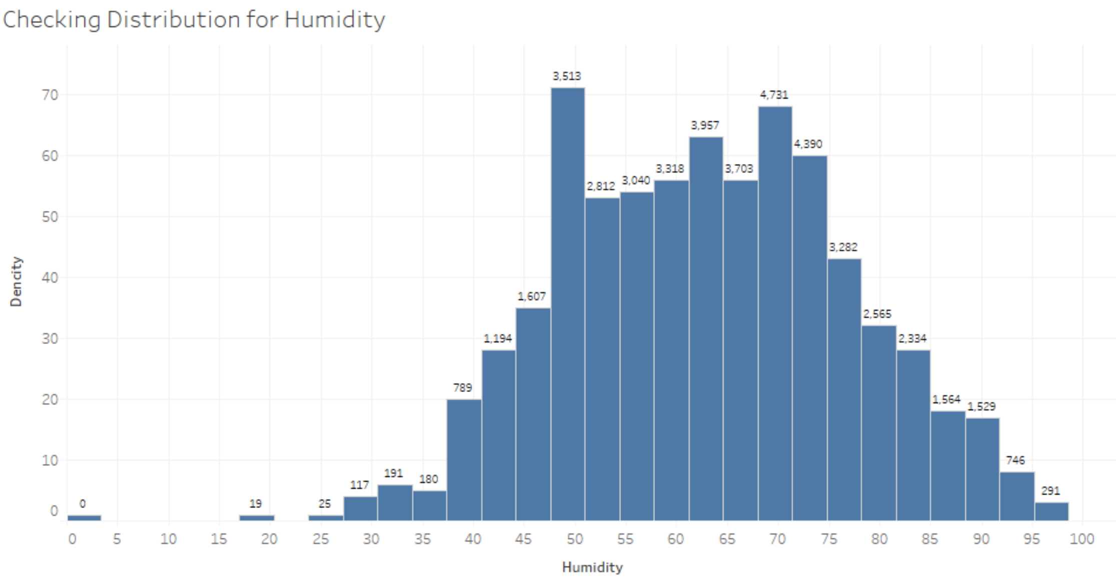
Fig 2.1: Distribution of categorical variables using bar graphs



Checking Distribution for Temperature
Dencity
Temperature



Checking Distribution for Feel Temperature
Dencity
Feel Temperature



Checking Distribution for Humidity
Dencity
Humidity

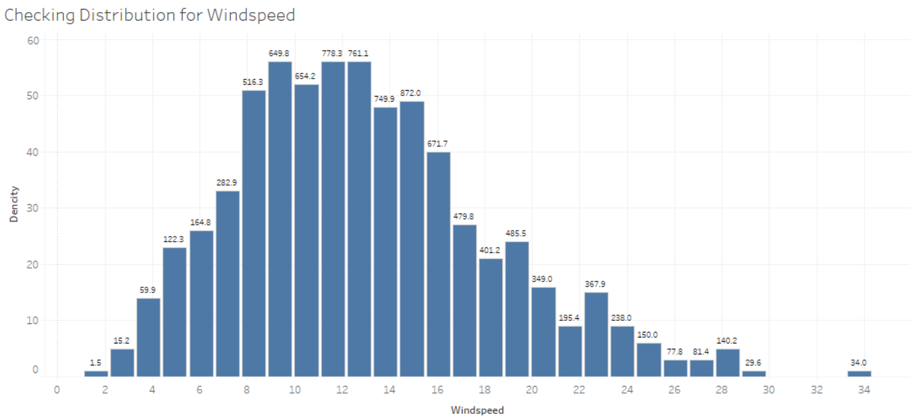
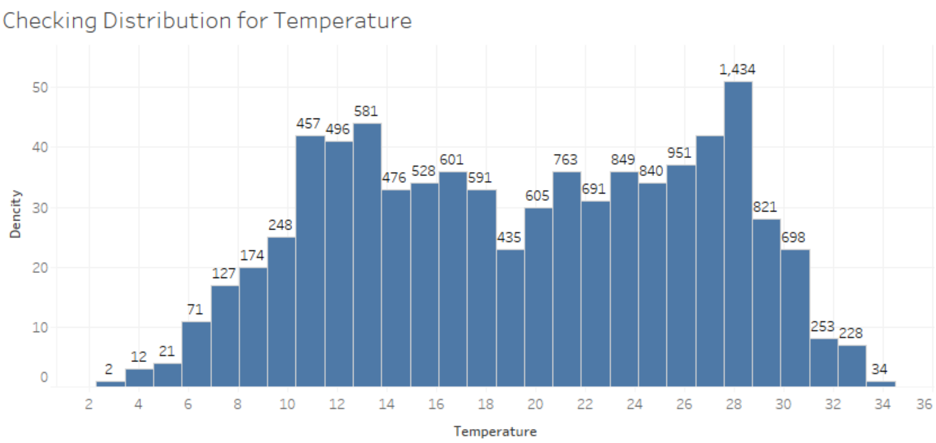
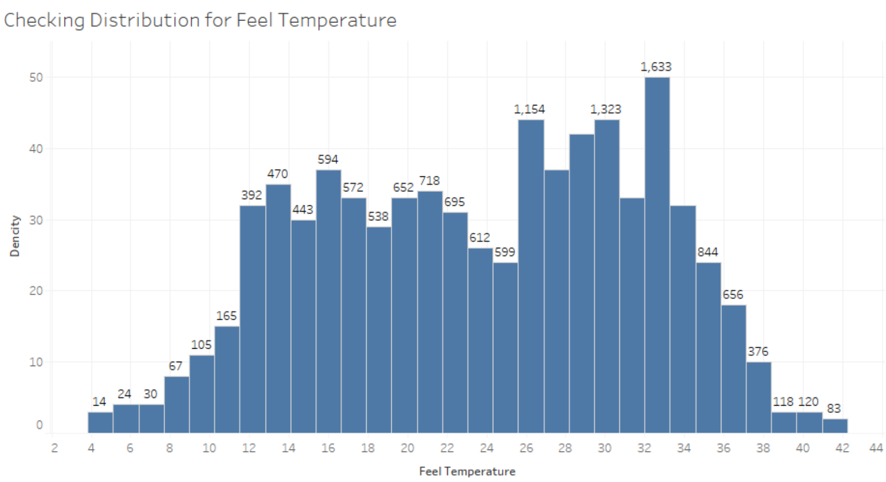
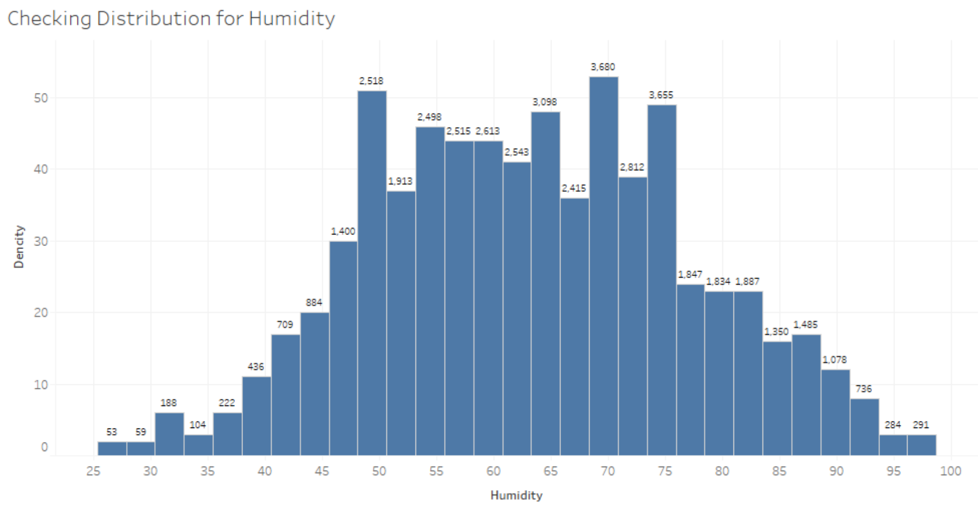


Fig 2.2: Distribution of continuous variables using histograms







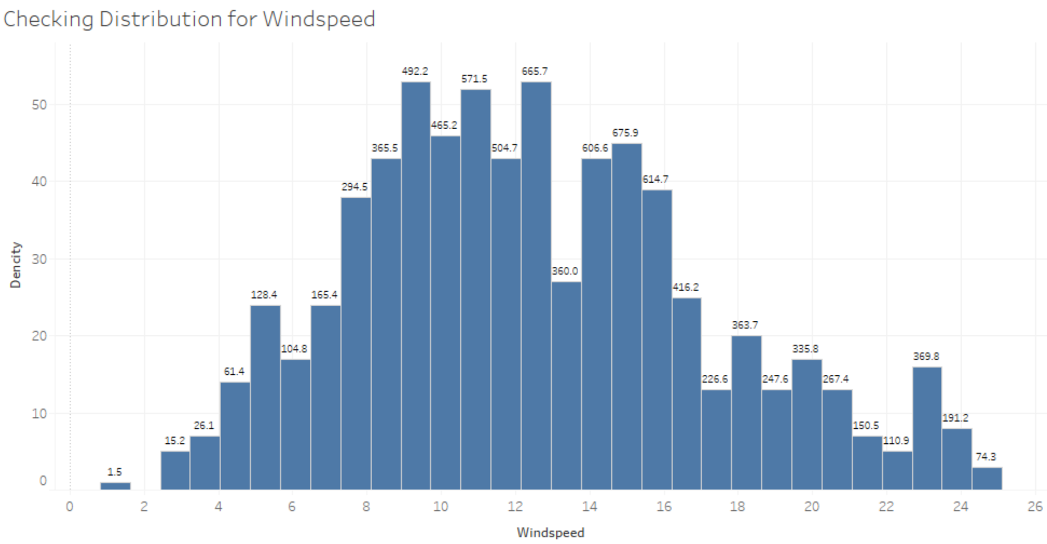
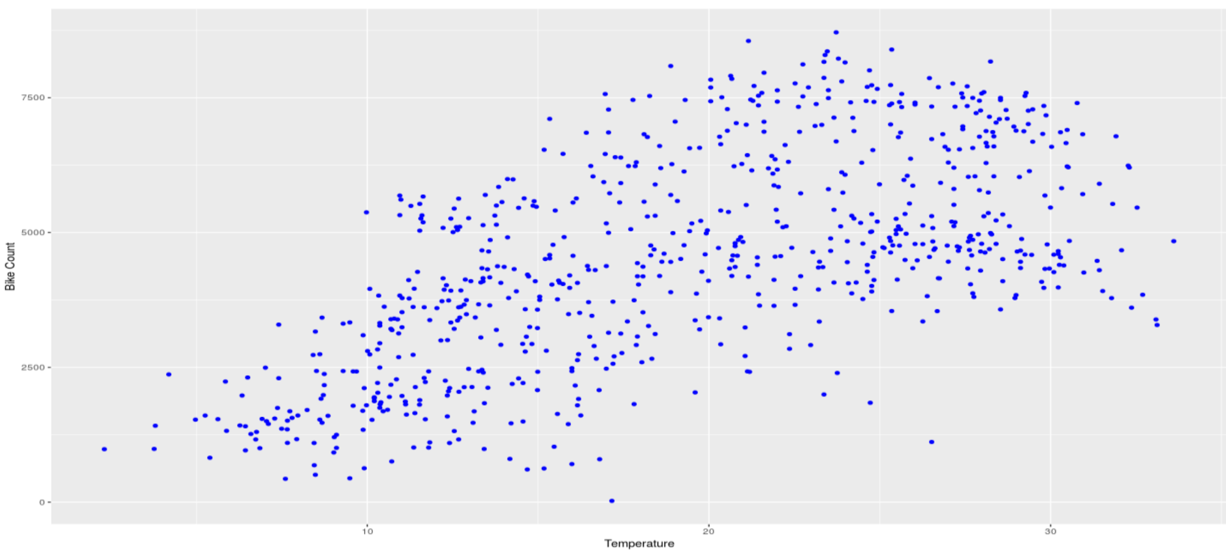
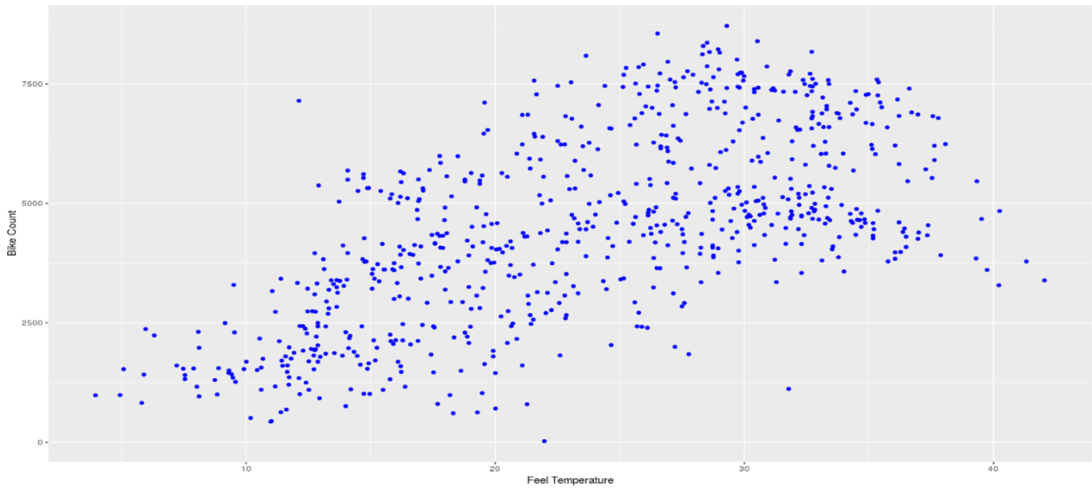


Fig 2.8: Distribution of continuous variables after removing outliers

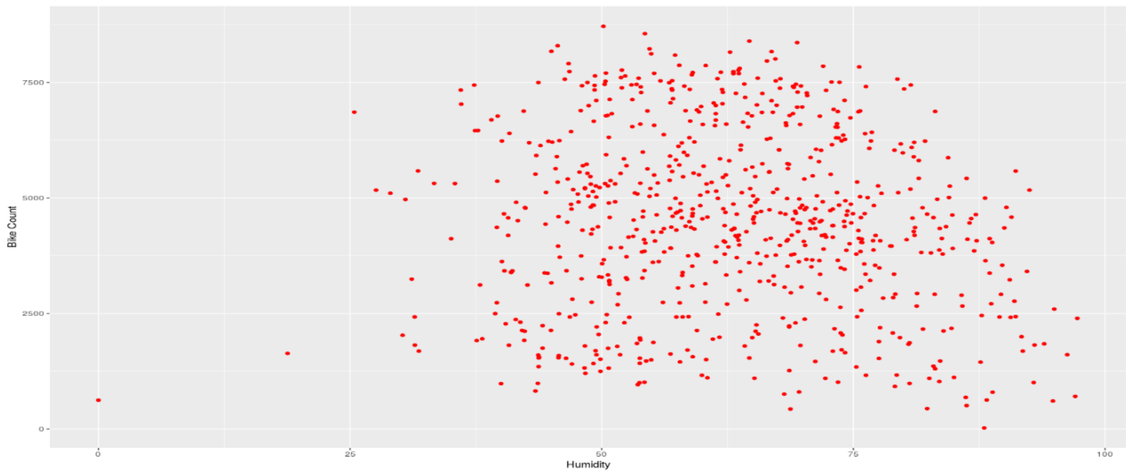
Scatterplot for Temperature



Scatterplot for Feel Temperature



Scatterplot for Humidity



Scatterplot for Wind speed

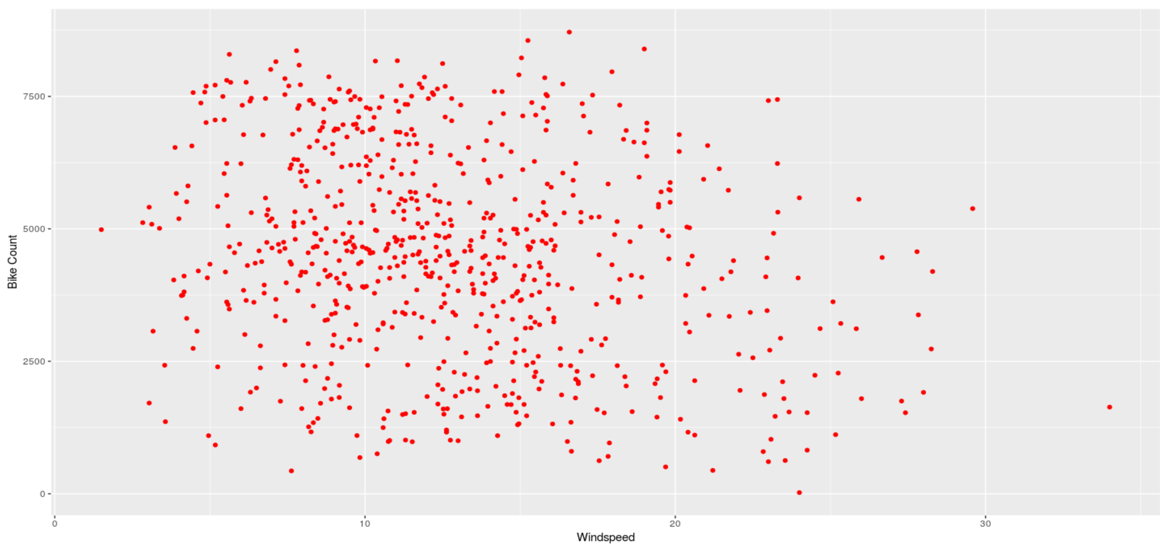


Fig 2.3: Scatter plot for continuous variables.

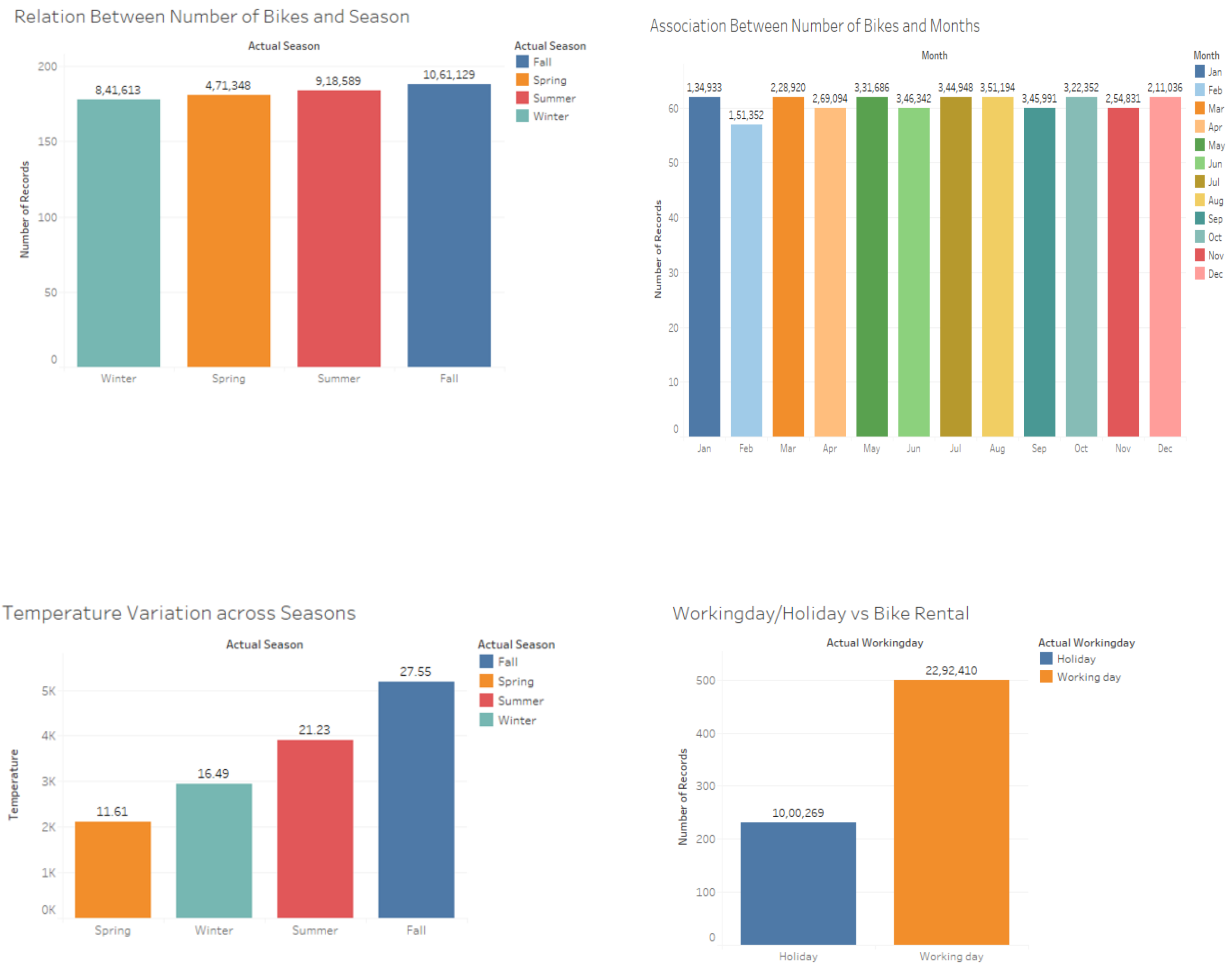


Fig 2.4: Relationship between the variables against bike count

	variable	Missing_values
0	dteday	0
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

Fig 2.5: Missing value analysis

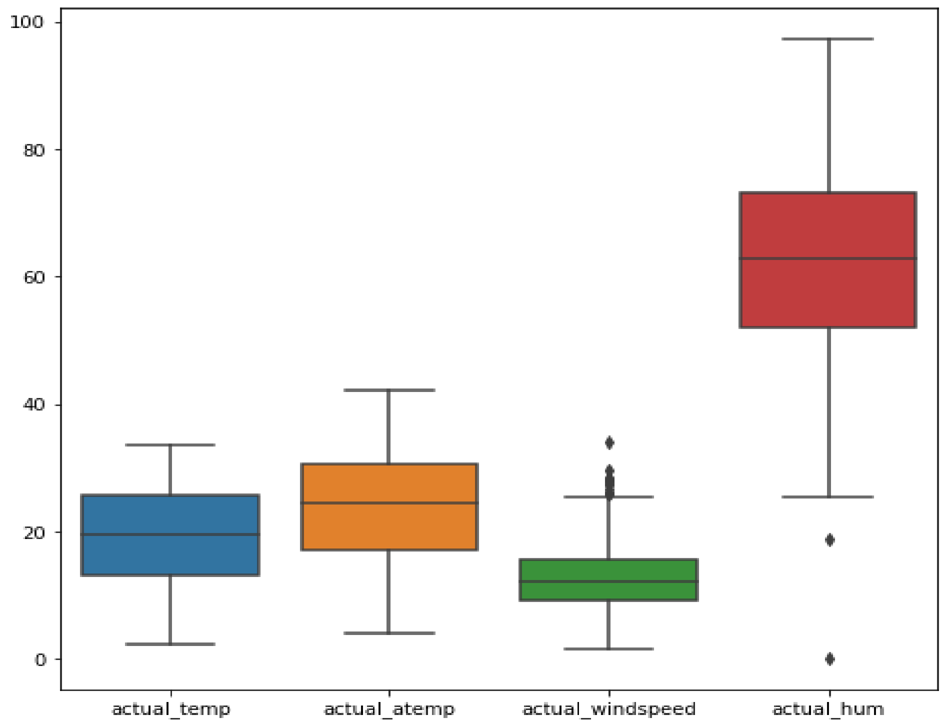


Fig 2.6: Boxplot of continuous variables with outliers

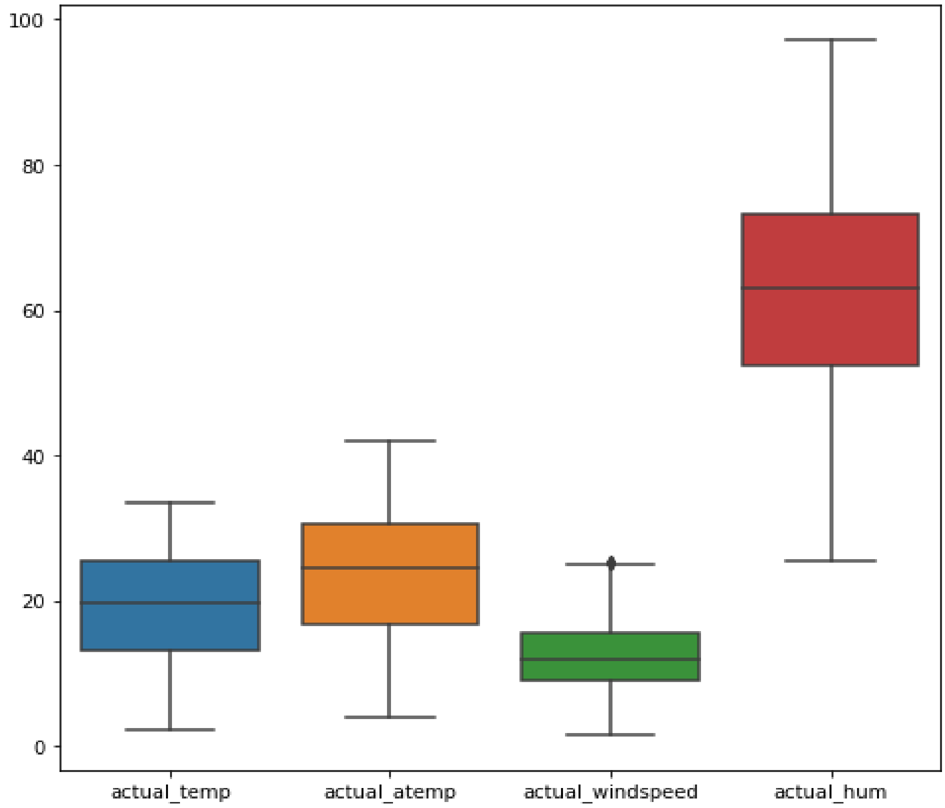


Fig 2.7: Boxplot of continuous variables without outliers

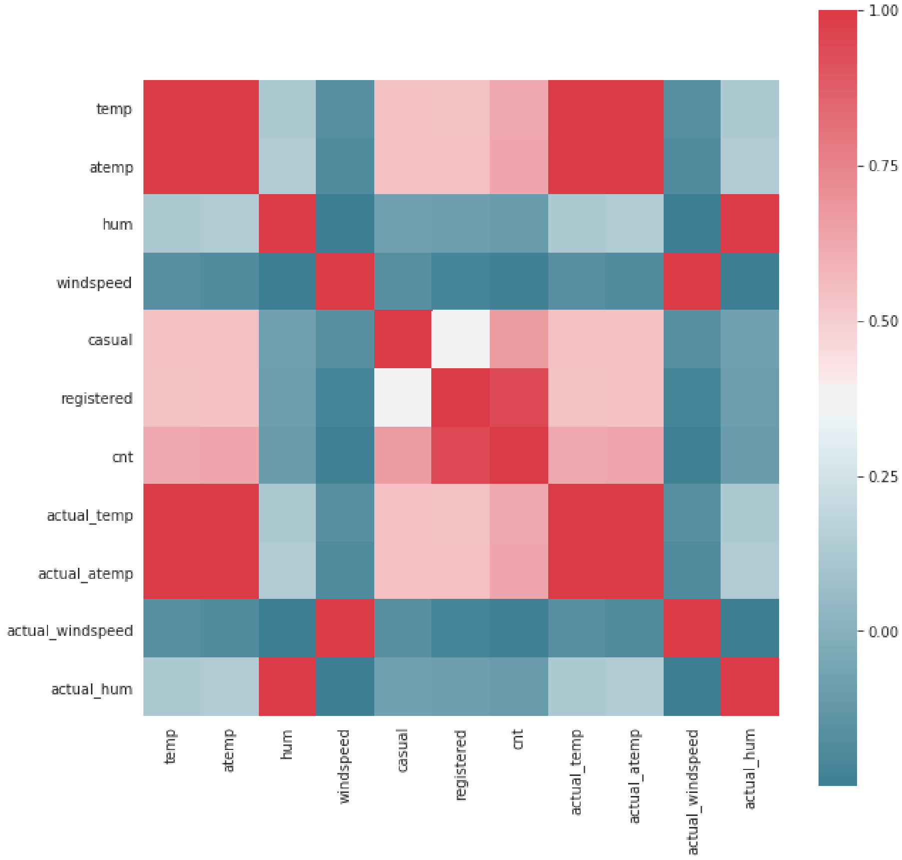


Fig 2.8: Correlation plot of Continuous variables

Appendix B

R – Code

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

#Loading Libraries
libraries = c("plyr","dplyr", "ggplot2","rpart","dplyr","DMwR","randomForest",
        "usdm","corrgram","DataCombine")
lapply(X = libraries,require, character.only = TRUE)
rm(libraries)

library(dummies)
library(caret)
library(rpart.plot)

#Loading the data
df = read.csv("day.csv",header=T)

#first few rows of data
head(df)

#str names of data
str(df)
################# Exploratory Data Analysis ##############
df$dteday = as.Date(df$dteday)
df$season = as.factor(df$season)
df$yr = as.factor(df$yr)
df$mnth = as.factor(df$mnth)
df$holiday = as.factor(df$holiday)
```

df$weekday = as.factor(df$weekday)

df$workingday = as.factor(df$workingday)

df$weathersit = as.factor(df$weathersit)

################## Future Engineering #####################

#Creating a new variables

df$actual_temp = df$temp * 39

df$actual_atemp = df$atemp * 50

df$actual_windspeed = df$windspeed * 67

df$actual_hum = df$hum * 100

df$actual_season = factor(x = df$season, levels = c(1,2,3,4), labels = c("Spring","Summer","Fall","Winter"))

df$actual_year = factor(x = df$yr, levels = c(0,1), labels = c("2011","2012"))

df$actual_workingday = factor(x = df$workingday, levels = c(0,1), labels = c("Holiday","Working day"))

df$actual_weathersit = factor(x = df$weathersit, levels = c(1,2,3,4),

labels = c("Clear","Cloudy/Mist","Rain/Snow/Fog","Heavy Rain/Snow/Fog"))

################# Missing value analysis ##############

missing_val = sapply(df,function(x){sum(is.null(df))})

missing_val

#There is no missing values so will move forward to data distribution

```
############# Exploring data distribution by graphs ###############

#checking data distribution of categorical variables using bar graphs
bar_season = ggplot(data = df,aes(x = actual_season)) +
  geom_bar(fill = 'blue')+ggtitle("Count of Season")

bar_season

bar_weather = ggplot(data = df,aes(x = actual_weathersit)) +
  geom_bar(fill = 'blue')+ggtitle("Count of Weatherlist")

bar_weather

bar_workingday = ggplot(data = df,aes(x = actual_workingday)) +
  geom_bar(fill = 'blue')+ggtitle("Count of Workingday")

bar_workingday

gridExtra::grid.arrange(bar_season,bar_weather,bar_workingday,ncol=2)

#Checking data distribution numerical variables using histograms
hist_temp = ggplot(data = df, aes(x =actual_temp)) + ggtitle("Distribution of Temperature") + geom_histogram(bins = 25,fill='blue')
hist_atemp = ggplot(data = df, aes(x =actual_atemp)) + ggtitle("Distribution of feeled Temprature") + geom_histogram(bins = 25,fill='blue')
hist_windspeed = ggplot(data = df, aes(x =actual_windspeed)) + ggtitle("Distribution of Windspeed") + geom_histogram(bins = 25,fill='blue')
hist_hum = ggplot(data = df, aes(x =actual_hum)) + ggtitle("Distribution of Humidity") + geom_histogram(bins = 25,fill='blue')
gridExtra::grid.arrange(hist_temp,hist_atemp,hist_windspeed,hist_hum,ncol=2)
```

```
########### Outlier Analysis ################
continous_var = c("actual_temp","actual_atemp","actual_windspeed","actual_hum")

for (i in 1:length(continous_var))
{
  assign(paste0("gn",i), ggplot(aes_string(y = continous_var[i]), data = df)+
        stat_boxplot(geom = "errorbar", width = 0.5) +
        geom_boxplot(outlier.colour="red", fill = "grey" ,outlier.shape=18,
                outlier.size=1, notch=FALSE) +
        theme(legend.position="bottom")+
        labs(y=continous_var[i])+
        ggtitle(paste("Box plot for",continous_var[i])))
}
gridExtra::grid.arrange(gn1,gn3,gn2,gn4,ncol=2)

#Found that outliers in actual_windspeed and actual_hum
#Removing outliers in Windspeed

val = df[,19][df[,19] %in% boxplot.stats(df[,19])$out]
df = df[which(!df[,19] %in% val),]

#Removing outliers in humidity
val = df[,20][df[,20] %in% boxplot.stats(df[,20])$out]
df = df[which(!df[,20] %in% val),]
colnames(df)
```

######################## Future Selection ########################

```
#Checking multicollinearity using VIF

df_vif = df[,c('temp','atemp','windspeed','hum')]
vifcor(df_vif)
#vifstep(df_vif)

#Checking Collinearity by using Correlation graph
corrgram(df, order = FALSE,lower.panel = panel.shade,upper.panel = panel.pie,
         text.panel = panel.txt, main="Correlation Graph")

#From above 2 Correlation analysis observed 'atemp' variable has multicollinearity problem
#Removing unwanted variables
df = subset(df,select = -c( holiday,instant,dteday,atemp,casual,registered,
                            actual_temp,actual_atemp,actual_windspeed,
                            actual_hum,actual_season,actual_year,actual_workingday,
                            actual_weathersit))

#Taking copy of data
df2 = df
#df = df2
colnames(df)

#Creating dummyvariables for categorical variables to trick the Regression models
catagorical_var = c('season','yr','mnth','weekday','workingday','weathersit')
df = dummy.data.frame(df, catagorical_var)
colnames(df)
rmExcept(keepers = 'df')
```

###################### Model Development ##########################

#Splitting data into train and test data

train_index = sample(1:nrow(df), 0.8*nrow(df))

train = df[train_index,]

test = df[-train_index,]

#------------------Decision Tree---------------------#

#training the data with rpart

dt_model = rpart(cnt ~ ., data = train,method = "anova")

rpart.plot(dt_model)

#Predicting test data

dt_predictions = predict(dt_model,test[,-34])

#Create dataframe for actual and predicted values

df_pred = data.frame("actual"=test[,34], "pred"=dt_predictions)

head(df_pred)

summary(dt_model)

#Calculate RMSE and other error metrics

regr.eval(trues = test[,34], preds = dt_predictions, stats = c("mae","mse","rmse","mape"))

#Calculate R-Squared

print(postResample(pred = dt_predictions, obs = test[,34]))

RMSE Rsquared MAE

#961.0064188 0.7676469 736.2084616

#-------------------Linear Regression--------------------#

#Train the data using linear regression

lr_model = lm(formula = cnt~., data = train)

#Check the summary of the model

summary(lr_model)

#Predict the test cases

lr_predictions = predict(lr_model, test[,-34])

#Create dataframe for actual and predicted values

df_lin = cbind(df_pred,lr_predictions)

head(df_lin)

#Calculate RMSE and other error metrics

regr.eval(trues = test[,34], preds = lr_predictions, stats = c("mae","mse","rmse","mape"))

#Calculate R-Squared

print(postResample(pred = lr_predictions, obs = test[,34]))

RMSE Rsquared MAE

#847.8232089 0.8209717 629.3466018

```
#--------------------------Random Forest-------------------#

#Train the data using random forest
rf_model = randomForest(cnt~., data = train, ntree = 500)

#Predict the test cases
rf_predictions = predict(rf_model, test[,-34])

#Create dataframe for actual and predicted values
df_lin = cbind(df_lin,rf_predictions)
head(df_lin)

#Calculate RMSE and other error metrics
regr.eval(trues = test[,34], preds = rf_predictions, stats = c("mae","mse","rmse","mape"))
MAPE(test[,34], rf_predictions)

#Calculate R-Squared
print(postResample(pred = rf_predictions, obs = test[,34]))
#        RMSE    Rsquared        MAE
#693.8210337   0.8804323   512.0623566

#-------Hyperparameter Tuning ---------------#
# Tuning Random Forest
control <- trainControl(method="repeatedcv", number=10, repeats=3)
reg_fit <- caret::train(cnt~., data = train, method = "rf",trControl = control)
reg_fit$bestTune
y_pred <- predict(reg_fit, test[,-34])
print(caret::R2(y_pred, test[,34]))
```

Python – Code

Importing packages for the model development and data processing

```
import datetime
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

#Loading dataset

```
df = pd.read_csv("day.csv",index_col = 0)
```

#Checking the data

```
df.head(10)
```

#Exploratory Data Analysis

#Checking datatypes of the variables

```
df.dtypes
```

```
#Converting variables datatype to required datatypes
#Categorical variables
df['dteday'] = pd.to_datetime(df['dteday'],yearfirst = True)
df['season'] = df['season'].astype('category')
df['yr']     = df['yr'].astype('category')
df['mnth']   = df['mnth'].astype('category')
df['holiday']= df['holiday'].astype('category')
df['weekday']= df['weekday'].astype('category')
df['workingday']= df['workingday'].astype('category')
df['weathersit']= df['weathersit'].astype('category')

#Continuous variables
df['temp'] = df['temp'].astype('float')
df['atemp']= df['atemp'].astype('float')
df['hum']  = df['hum'].astype('float')
df['windspeed'] = df['windspeed'].astype('float')
df['casual'] = df['casual'].astype('float')
df['registered'] = df['registered'].astype('float')
df['cnt'] = df['cnt'].astype('float')
```

#Checking datatypes of the variables after updating columns datatypes

```
df.dtypes
```

#Missing Value Analysis

```
missing_val = pd.DataFrame(df.isnull().sum())
missing_val = missing_val.reset_index()
missing_val = missing_val.rename(columns={'index':'variable',0:'Missing_values
'})
missing_val
```

```
#Craeting new variables from existing variables for visualizations (Future Eng
ineering)
df['actual_temp'] = df['temp'] * 39
df['actual_atemp'] = df['atemp'] * 50
df['actual_windspeed'] = df['windspeed'] * 67
df['actual_hum'] = df['hum'] * 100
```

#Checking the columns after Future Engineering

```
df.columns

#Cheking the Distribution of data by using Histograms
continuous_variables = ['actual_temp','actual_atemp','actual_windspeed','actua
l_hum','cnt']
for i in continuous_variables:
    plt.hist(df[i],bins='auto')
    plt.title("Checking Distribution for Variable "+str(i))
    plt.ylabel("Density")
    plt.xlabel(i)
    plt.show()
```

```
#Bike Rentals Per Monthly
monthly_sales = df.groupby('mnth').size()
print(monthly_sales)
#Plotting the Graph
plot = monthly_sales.plot(title='Monthly Sales',xticks=(1,2,3,4,5,6,7,8,9,10,1
1,12))
plot.set_xlabel('Months')
plot.set_ylabel('Total Number of Bikes')
```

#Checking the distribution categorical Data using factorplot

```
categorical_var = ['dteday','yr', 'mnth', 'season','weathersit', 'workingday']
sns.set_style("whitegrid")
for i in categorical_var[i]:
        sns.factorplot(data=df, x=df[i], kind= 'count',size=4,aspect=2)
```

```
#Scatter plot for temprature against bike rentals
sns.scatterplot(data=df,x='actual_temp',y='cnt')
```

```
#Scatter plot for humidity against bike rentals
sns.scatterplot(data=df,x='actual_hum',y='cnt')
```

```
#Scatter plot for atemp(feeled_temparature) against bike rentals
sns.scatterplot(data=df,x='actual_atemp',y='cnt')
```

```
#Scatter plot for windspeed against bike rentals
sns.scatterplot(data=df,x='actual_windspeed',y='cnt')
```

#Outlier Analysis

```
#Checking Outliers in  data using boxplot
sns.boxplot(data=df[['actual_temp','actual_atemp','actual_windspeed','actual_h
um']])
fig=plt.gcf()
fig.set_size_inches(8,8)
```

```
sns.boxplot(data=df[['season','mnth','holiday','weekday']])
fig=plt.gcf()
fig.set_size_inches(8,8)
```

```
sns.boxplot(data=df[['workingday','weathersit','casual','registered']])
fig=plt.gcf()
fig.set_size_inches(8,8)
```

```
#Variables that are used to remove outliers
#Not considering casual because this is not predictor variable
#Not considering holiday because workingday variable includes holiday,
#so there is no useful of considering holiday variables.

out_names = ['actual_windspeed','actual_hum']
```

```
#Detecting and Removing Outliers
for i in out_names :
    print (i)
    q75,q25 = np.percentile(df.loc[:,i],[75,25])
    iqr = q75-q25

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

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

#Future Selection

```
#Checking Outliers in data after outliers removel using boxplot
sns.boxplot(data=df[['actual_temp','actual_atemp','actual_windspeed','actual_h
um']])
fig=plt.gcf()
fig.set_size_inches(8,8)
```

```
continuous_variables = [ 'temp','atemp', 'hum', 'windspeed', 'casual',
                        'registered', 'cnt', 'actual_temp', 'actual_atemp',
                        'actual_windspeed', 'actual_hum'
                        ]
```

```
#Future selection on the basis of Correlation, multcollinearity
#cnames = ["actual_temp","actual_atemp","actual_hum","acttual_windspeed"]
#cnames = ["temp","atemp","hum","windspeed"]

df_cor = df.loc[:,continuous_variables]
f, ax = plt.subplots(figsize=(10,10))

#Generate correlation matrix
cor_mat = df_cor.corr()

#Plot using seaborn library
sns.heatmap(cor_mat, mask=np.zeros_like(cor_mat, dtype=np.bool), cmap=sns.dive
rging_palette(220, 10, as_cmap=True),
            square=True, ax=ax)
plt.plot()
```

Hypothesis Testing

Null Hypothesis

Two variables are independent

Alternate Hypothesis

Two variables are not independent

> If p-value is less than 0.05 then reject null hypothesis, that means two are variables are dependant (not independent)

> But in our case most of the p-value are greater than 0.05, hence we need to accept that we failed to reject null hypothesis

```
cat_columns = ['season', 'yr', 'mnth', 'holiday', 'weekday','workingday', 'wea
thersit']
# making every combinationfrom cat_columns
factors_paired = [(i,j) for i in cat_columns for j in cat_columns]
factors_paired
p_values = []
from scipy.stats import chi2_contingency
for factor in factors_paired:
    if factor[0] != factor[1]:
        chi2, p, dof, ex = chi2_contingency(pd.crosstab(df[factor[0]], df[fact
or[1]]))
        p_values.append(p.round(3))
    else:
        p_values.append('-')
p_values = np.array(p_values).reshape((7,7))
p_values = pd.DataFrame(p_values, index=cat_columns, columns=cat_columns)
print(p_values)
```

```
# checking vif of numerical column without dropping multicollinear column
from statsmodels.stats.outliers_influence import variance_inflation_factor as
vf
from statsmodels.tools.tools import add_constant
continuous = add_constant(df[['temp', 'atemp', 'hum', 'windspeed']])
vif = pd.Series([vf(continuous.values, i) for i in range(continuous.shape[1])]
, index = continuous.columns)
print(vif.round(1))

# Checking VIF values of numeric columns after dropping column atemp
from statsmodels.stats.outliers_influence import variance_inflation_factor as
vf
from statsmodels.tools.tools import add_constant
continuous = add_constant(df[['temp', 'hum', 'windspeed']])
vif = pd.Series([vf(continuous.values, i)  for i in range(continuous.shape[1])
], index = continuous.columns)
vif.round(1)
```

```
#Removing variables atemp beacuse it is highly correlated with temp,
#Removing weekday,holiday because they don't contribute much to the independen
t cariable
#Removing Causal and registered becuase that's what we need to predict.
df = df.drop(columns=['holiday','dteday','atemp','casual','registered',
                       'actual_temp','actual_atemp',
                     'actual_windspeed','actual_hum'
                       ])
```

#taking copy of the data

```
df2 = df.copy()

categorical_var = ['season','yr','mnth', weekday', 'workingday','weathersit']
```

```
#Dummy Variable creation for categorical variables
df2 = pd.get_dummies(data = df2,columns=categorical_var)
```

#Checking Columns

```
df2.columns
```

```
df_plt_tree = df2.drop('count',axis=1)
df2.shape
```

#Model Development

```
#Import Libraries for decision tree
from sklearn.tree import DecisionTreeRegressor,export_graphviz
from sklearn.metrics import accuracy_score,r2_score,mean_squared_error
from sklearn.model_selection import train_test_split
from sklearn import tree


#Splitting data into train and test data
train,test = train_test_split(df2,test_size = 0.2, random_state = 123)
```

```
#Function for Performing all the tasks such as Error metrix rmse,mape,r-square
d,accuracy,predictions
def evaluate(model, test_features, test_actual):
    predictions = model.predict(test_features)
    #Creating new data frame with comparing actual and predicted values
    df_Dt = pd.DataFrame({'actual':test_actual,'predicted':predictions})
    errors = abs(predictions - test_actual)
    mape = 100 * np.mean(errors / test_actual)
    accuracy = 100 - mape
    rmse = np.sqrt(mean_squared_error(test_actual,predictions))
    rsquared = r2_score(test_actual, predictions)
    print('<---Model Performance--->')
    print('R-Squared Value = {:0.2f}'.format(rsquared))
    print('RMSE = {:0.2f}'.format(rmse))
    print('MAPE = {:0.2f}'.format(mape))
    print('Accuracy = {:0.2f}%.'.format(accuracy))
    return
```

#Decision Tree

```
#Decision Tree model development
#Training the model with train data
model = DecisionTreeRegressor(random_state = 123).fit(train.iloc[:,0:33],train
.iloc[:,33])

#Function for predictions, Error metrix rmse,mape,r-squared,accuracy
evaluate(model, test.iloc[:,0:33], test.iloc[:,33])

dotfile = open("pt.dot",'w')
df = tree.export_graphviz(model,out_file=dotfile,feature_names = df_plt_tree.c
olumns)
```

#Linear Regression

```
#import libraries for Linear regression
from sklearn.linear_model import LinearRegression

#Create model Linear Regression using LinearRegression
model = LinearRegression().fit(train.iloc[:,0:33],train.iloc[:,33])

#Function for predictions, Error metrix rmse,mape,r-squared,accuracy
evaluate(model, test.iloc[:,0:33], test.iloc[:,33])
```

#Random forest

```
#Import the libraries for Random Forest
from sklearn.ensemble import RandomForestRegressor
#Train the model
Rf_model = RandomForestRegressor(n_estimators=500,random_state=123).fit(train.
iloc[:,0:33], train.iloc[:,33])

#Function for predictions, Error metrix rmse,mape,r-squared,accuracy
evaluate(Rf_model, test.iloc[:,0:33], test.iloc[:,33])
```

#Hyperparameter Tuning

```
#Hyperparameter tuning using GridSearchCV
from sklearn.model_selection import GridSearchCV
# Create the parameter grid based on the results of random search
param_grid = {
    'bootstrap': [True],
    'max_depth': [12,14,16],
    'max_features': ['auto', 'sqrt'],
    'min_samples_leaf': [2,3],
    'min_samples_split': [8, 10, 12],
    'n_estimators': [900,1000,1200]
}
# Create a based model
rf = RandomForestRegressor()
# Instantiate the grid search model
grid_search = GridSearchCV(estimator = rf, param_grid = param_grid,
                          cv = 5, n_jobs = -1, verbose = 2)
```

```
grid_search.fit(test.iloc[:,0:33], test.iloc[:,33])
grid_search.best_params_
best_grid = grid_search.best_estimator_

#Applying gridsearchcsv to test data
grid_accuracy = evaluate(best_grid,test.iloc[:,0:33],test.iloc[:,33])
```