BIKE RENTING ANALYSIS

Himachala Venkata

15-10-19

Content

CHAPTERS **Page No:**

1. Introduction

1.1. Problem Statement :

The objective of this case is to prediction of bike rental count based on daily environmental and seasonal settings:
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 from Holiday 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_max- t_min), t_min=-16, t_max=+50 (only in hourly scale) hum: Normalized humidity. The values are divided to 100 (max)
windspeed: Normalized wind speed. The values are divided to 67 (max)
casual: count of casual users
registered: count of registered users
cnt: count of total rental bikes including both casual and registered

1.2. Data:

This data consists of 731 rows and 16 columns. Here I have shown just a sample data of 6 rows and 16 columns.

	INSTANT	DTEDAY	SEASON	YR	MNTH	HOLIDAY	WEEKDAY	WORKINGDAY	WEATHERSIT
0	1	01-01-2011	1	0	1	0	6	0	2
1	2	02-01-2011	1	0	1	0	0	0	2
2	3	03-01-2011	1	0	1	0	1	1	1
3	4	04-01-2011	1	0	1	0	2	1	1
4	5	05-01-2011	1	0	1	0	3	1	1

	TEMP	ATEMP	HUM	WINDSPEED	CASUAL	REGISTERED	CNT
5	0.344167	0.363625	0.805833	0.160446	331	654	985
6	0.363478	0.353739	0.696087	0.248539	131	670	801
7	0.196364	0.189405	0.437273	0.248309	120	1229	1349
8	0.200000	0.212122	0.590435	0.160296	108	1454	1562
9	0.226957	0.229270	0.436957	0.186900	82	1518	1600

2. Analysis

2.1. Pre-processing:

Any predictive Modeling requires that we look at the data before we start modeling. 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 Descriptive Analysis:

Descriptive analysis is the first step for conducting statistical analysis. Descriptive statistics are used to describe the basic features of the data in a study. They provide summaries about the sample and the measures. It gives you an idea about the distribution of the data, to detect the outliers and types. It summarize data in a meaningful way.

In the program I have divided data into Category and numeric features along with the target variable “cnt”.

Later, calculated the data statistics for each column of numeric features.

	Temp	Atemp	Hum	Windspeed
Count	731.000000	731.0000000	731.000000	731.000000
Mean	0.495385	0.474354	0.627894	0.190486
Std	0.183051	0.162961	0.142429	0.077498
Min	0.059130	0.079070	0.000000	0.022392
25%	0.337083	0.337842	0.520000	0.134950
50%	0.498333	0.486733	0.626667	0.180975
75%	0.655417	0.608602	0.730209	0.233214
Max	0.861667	0.840896	0.972500	0.507463

2.1.2 Missing Value Analysis*:*

In statistics missing value (or) missing data occur when no data is stored for the variable in an observation. Missing data is a common occurrence and can have a significant effect on the conclusion that can be drawn on the data.

Missing data reduces the representativeness of the sample and can therefore distort inferences about the population. Generally speaking, there are three main approaches to handle missing data:

1. Imputation—where values are filled in the place of missing data
2. Omission—where samples with invalid data are discarded from further analysis and
3. Analysis—by directly applying methods unaffected by the missing values

In this data set, there are no missing values. If we have missing values we can impute values by KNN imputation (or) by imputing the mean, median (or) mode of the data and saved into a file for further analysis.

Checking Null values using this syntax:
print(fullData.isnull().any())

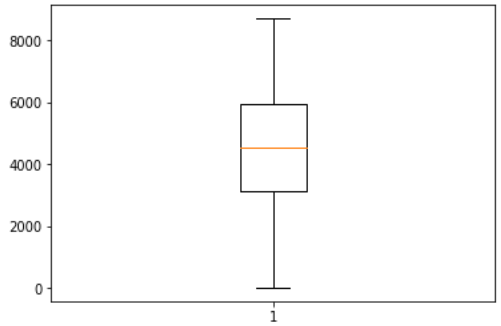
Instant	**False**
dteday	False
season	False
Yr	False
Mnth	False
Holiday	False
Weekday	False
Workingday	False
Weathersit	False
Temp	False
Atemp	False
Hum	False
Windspeed	False
Casual	False
Registered	False
count	False

dtype: bool

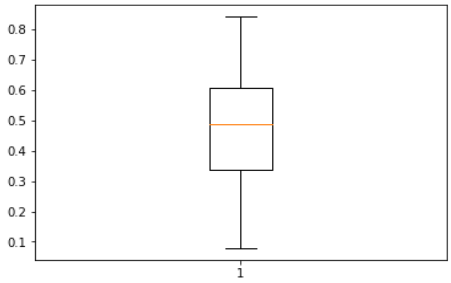
False indicates that there are no null values (or) missing values in any variable of the data

2.1.3 **Outlier Analysis**:

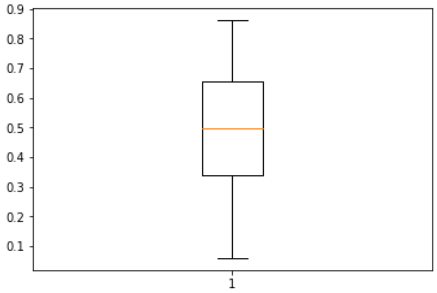
An outlier is an element of a data set that distinctly stands out from the rest of the data. In other words, outliers are those data points that lie outside the overall pattern of distribution as shown in figure below. The easiest way to detect outliers is to create a graph.



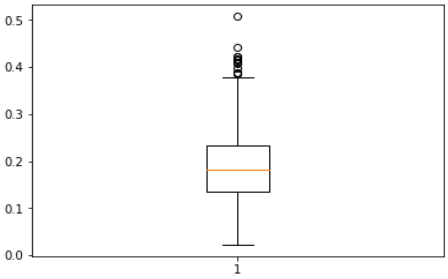
Cnt



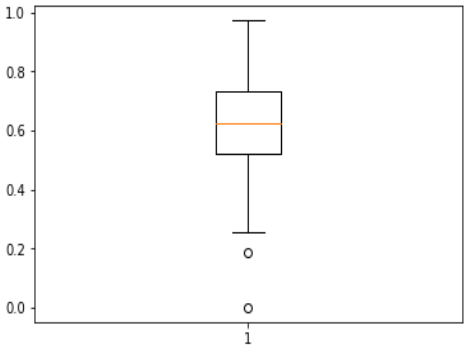
atemp



temp



windspeed



hum

Above, we have find outliers in the windspeed. We have to detect and delete the outliers from the data by using this code:

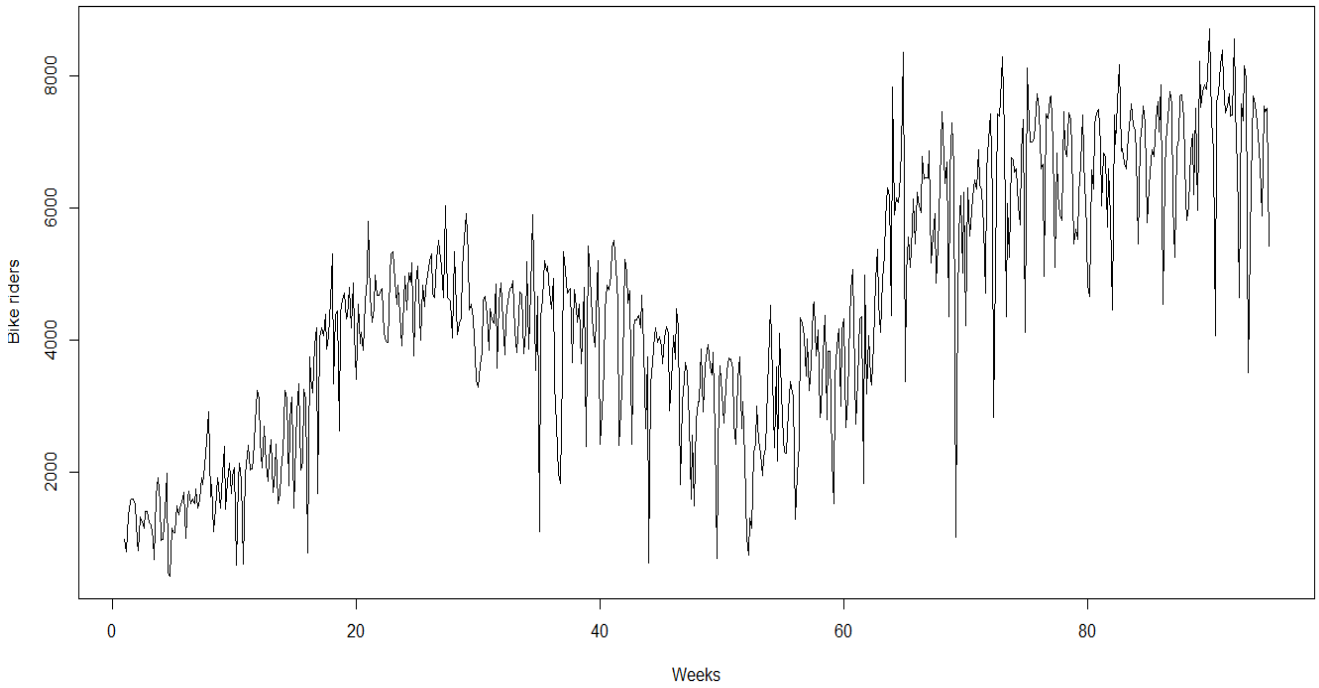
```
for i in cnames:
    print(i)
    q75, q25 = np.percentile(day_df.loc[:,i], [75 ,25])
    iqr = q75 - q25
    min = q25 - (iqr*1.5)
    max = q75 + (iqr*1.5)
    print(min)
    print(max)
    day_df = day_df.drop(day_df[day_df.loc[:,i] < min].index)
    day_df = day_df.drop(day_df[day_df.loc[:,i] > max].index)


#Detect and replace with NA
# #Extract quartiles
q75, q25 = np.percentile(day_df['windspeed'], [75 ,25])

# #Calculate IQR
#iqr = q75 - q25

# #Calculate inner and outer fence
minimum = q25 - (iqr*1.5)
maximum = q75 + (iqr*1.5)

# #Replace with NA
day_df.loc[day_df['windspeed'] < minimum,:'windspeed'] = np.nan
day_df.loc[day_df['windspeed'] > maximum,:'windspeed'] = np.nan
```

2.1.4 **Time Series Analysis*:***

Time series analysis is a statistical technique that deals with time series data, or trend analysis. Time series data means that data is in a series of particular time periods or intervals.

In this data set I have calculated the time series analysis of count

```
data_ts <-msts(day[,'cnt'], seasonal.periods=c(7))
train_ts <- head(data_ts, round(length(data_ts) * 0.9))
test_ts <- tail(data_ts, round(length(data_ts) * 0.1))
plot(train_ts, xlab="Weeks", ylab="Bike riders")
```



2.1.5 **Feature Selection*:***

In machine learning and statistics, feature selection, also known as variable selection, attribute selection or variable subset selection, is the process of selecting a subset of relevant features (variables, predictors) for use in model construction.

2.1.5.1 Correlation Analysis:

Correlation analysis is a method of statistical evaluation used to study the strength of a relationship between two, numerically measured, continuous variables (e.g. height and weight). This particular type of analysis is useful when a researcher wants to establish if there are possible connections between variables.

It is the scaled version of the covariance. It tells us about the dependency of two independent variables. It applies only on the numerical data. It tells us how two continuous variables are connected to each other.

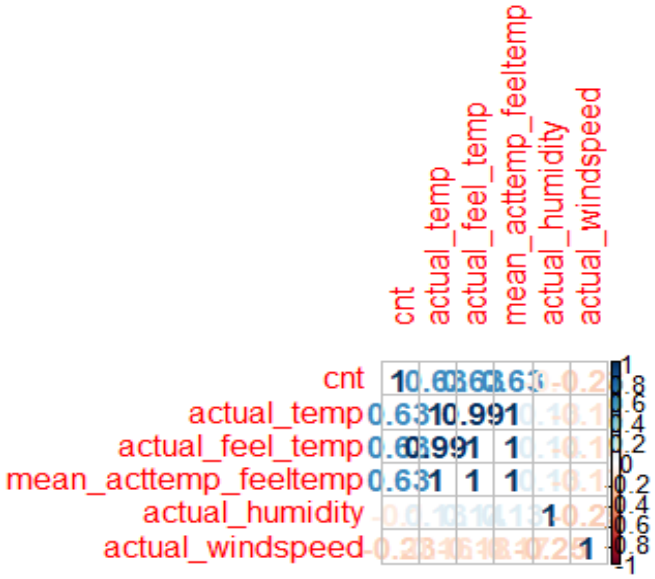
There will be high correlation between the independent and dependent variables and less correlation between the two dependent variables.

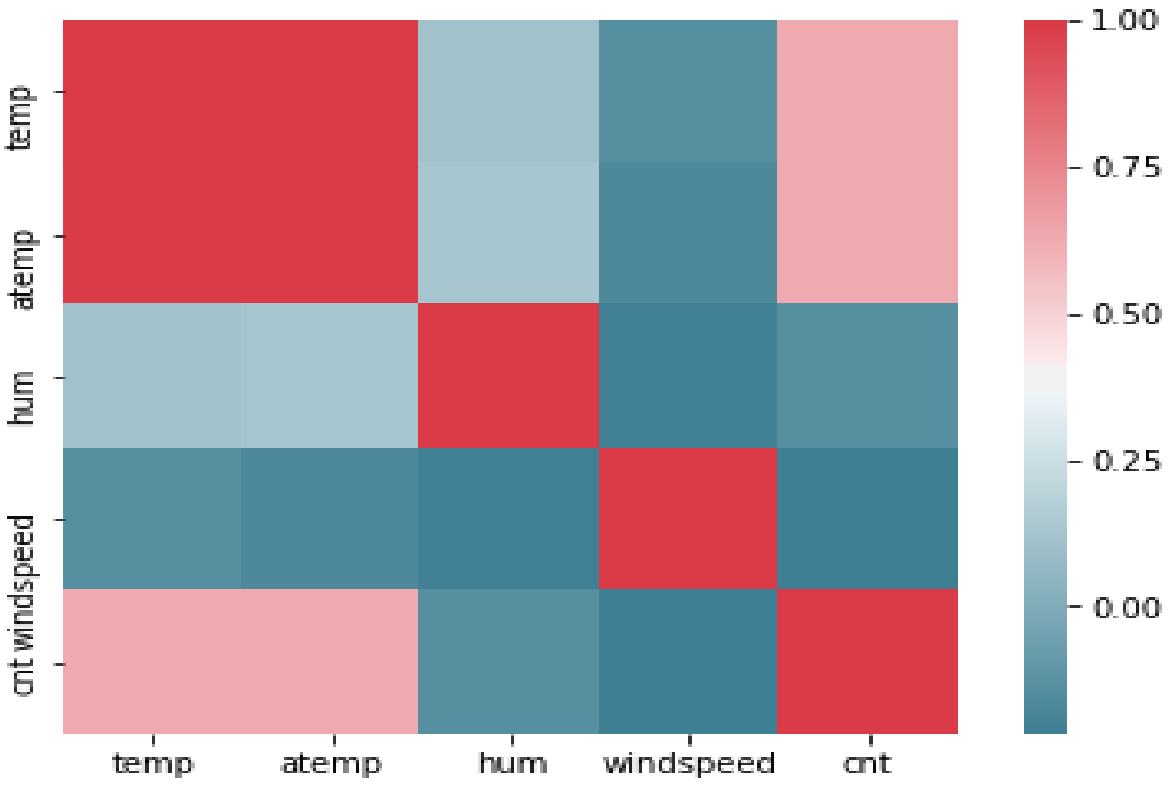
It varies from -1 to +1.

	Cnt	Act_ temp	Actual_feel _temp	Mean_ acttemp_ feeltemp	Actual_ humidity	Actual_win dspeed
Cnt	1.0000000	0.6274940	0.6310657	0.6306607	-0.1006586	-0.2345450
Act_temp	0.6274940	1.0000000	0.9917016	0.9977489	0.1269629	-0.1579441
Actual_feel_temp	0.6310657	0.9917016	1.0000000	0.9980905	0.1399881	-0.1836430
Mean_acttemp_feeltemp	0.6306607	0.9977489	0.9980905	1.0000000	0.1340209	-0.1716773
Actual_humidity	-0.1006586	0.1269629	0.1399881	0.1340209	1.0000000	-0.2484891
Actual_windspeed	-0.2345450	-0.1579441	-0.1836430	-0.1716773	-0.2484891	1.0000000

Correlation Analysis

2.1.5.2 **Correlation Plot:**





2.2. Modeling:

Data modeling is the process of producing a descriptive diagram of relationships between various types of information that are to be stored in a database. It is a representation of the data structures in a table for a company's database and is a very powerful expression of the company's business requirements. It is the process of creating a data model for an information system by applying certain formal techniques.

2.2.1 Model Development:

To develop a model we have to divide our data into train and test sets.

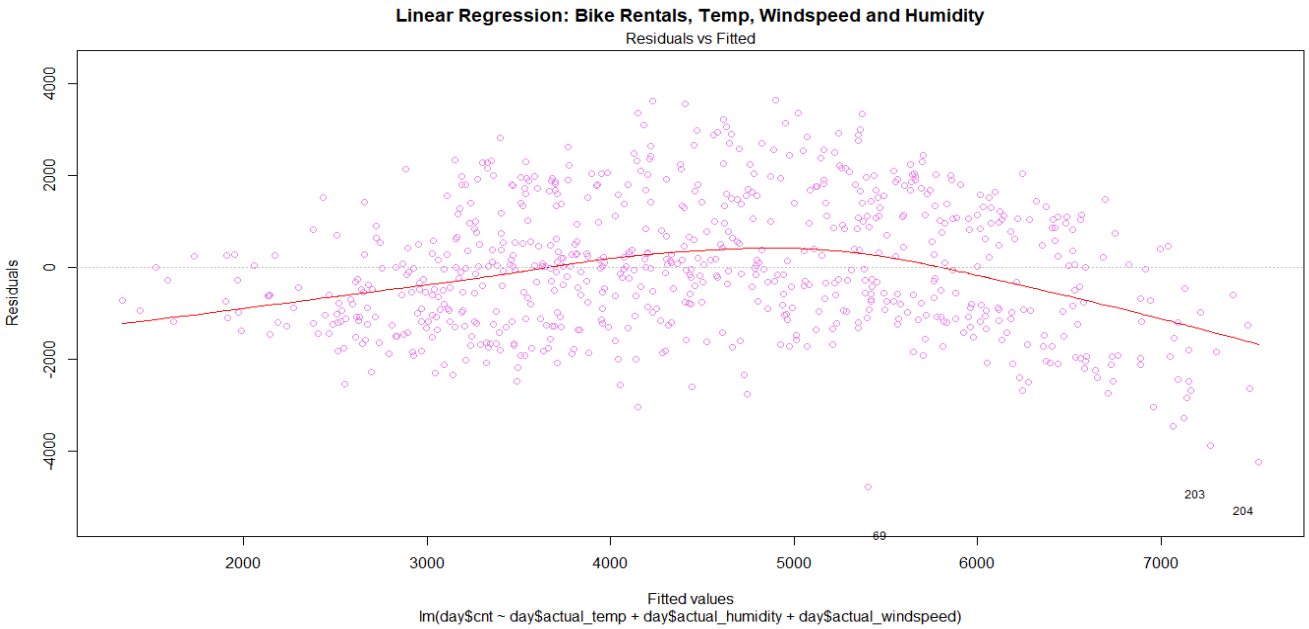
```
from sklearn.model_selection import train_test_split
#Divide data into train and test
X = day_df.values[:, 3:15]
Y = day_df.values[:, 15]
Y=Y.astype(int)

X_train, X_test, y_train, y_test = train_test_split( X, Y, test_size = 0.2)
```

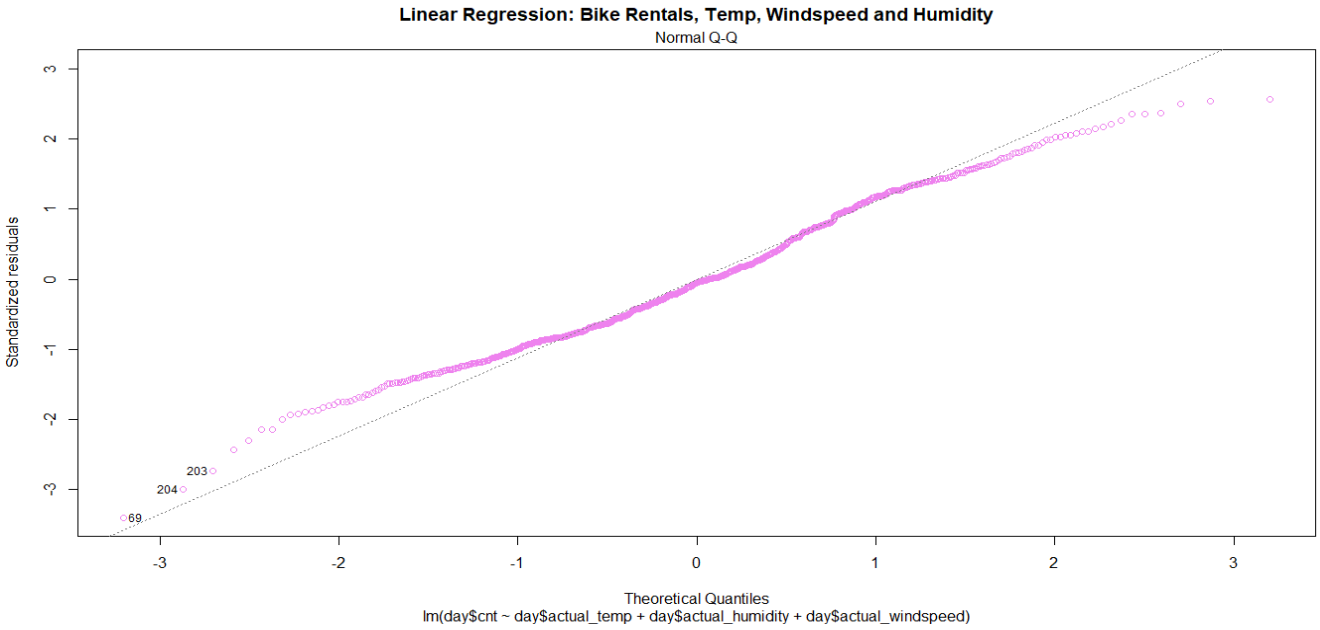
With this code we can develop a model.

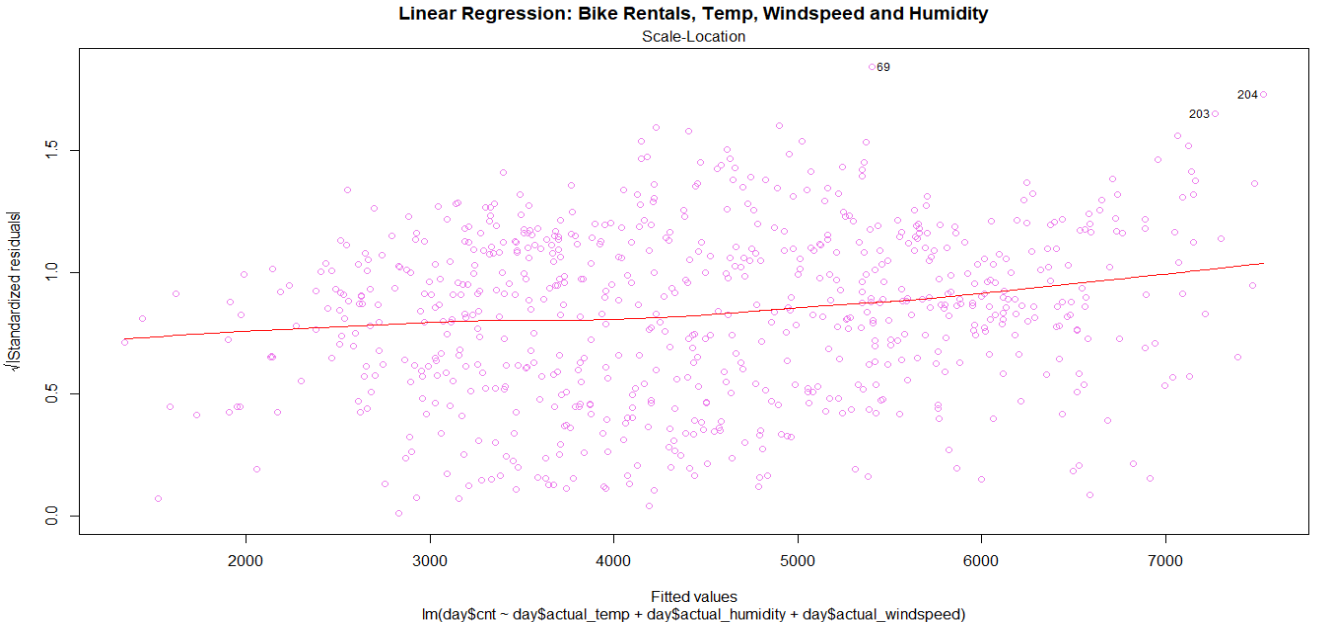
2.2.2 Linear Regression Model:

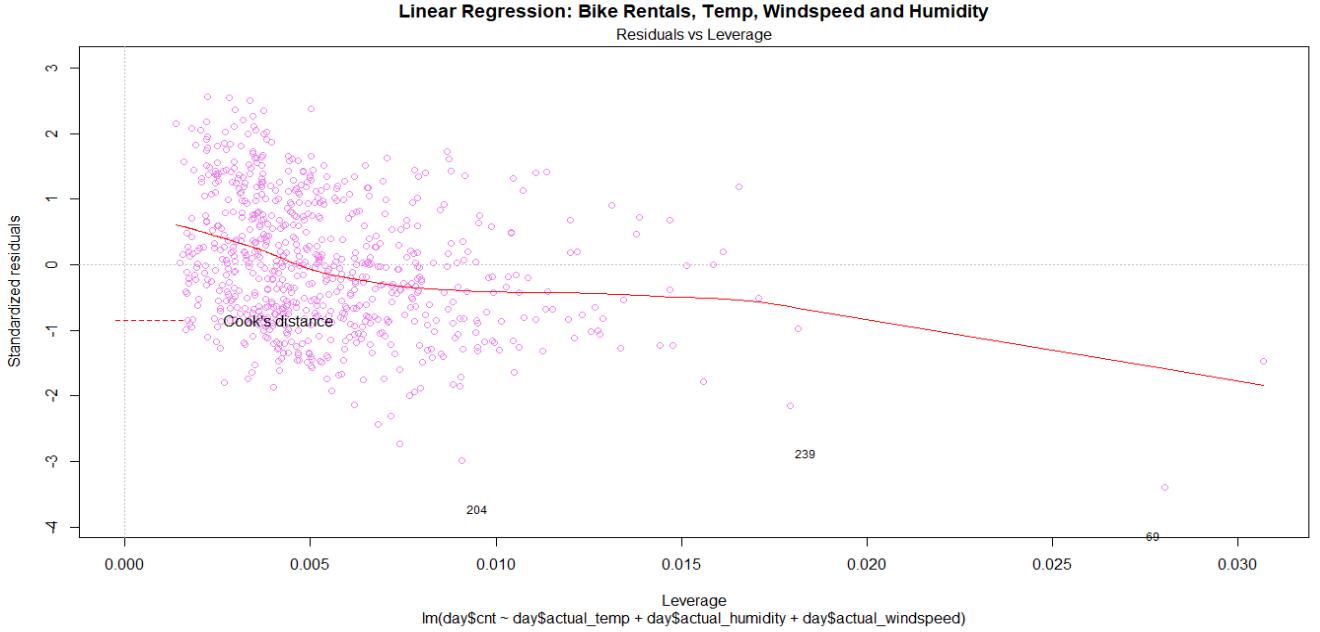
In statistics, linear regression is a linear approach to modeling the relationship between a scalar response (or dependent variable) and one or more explanatory variables (or independent variables). The case of one explanatory variable is called simple linear regression.

Linear Regression is the process of finding a line that best fits the data points available on the plot, so that we can use it to predict output values for inputs that are not present in the data set we have, with the belief that those outputs would fall on the line.




Linear Regression: Bike Rentals, Temp, Windspeed and Humidity
Normal Q-Q
Standardized residuals
Theoretical Quantiles
lm(day$cnt ~ day$actual_temp + day$actual_humidity + day$actual_windspeed)
203
204
69


Linear Regression: Bike Rentals, Temp, Windspeed and Humidity
Scale-Location
√|Standardized residuals|
Fitted values
lm(day$cnt ~ day$actual_temp + day$actual_humidity + day$actual_windspeed)
69
204
203


Linear Regression: Bike Rentals, Temp, Windspeed and Humidity
Residuals vs Leverage
Standardized residuals
Cook's distance
Leverage
lm(day$cnt ~ day$actual_temp + day$actual_humidity + day$actual_windspeed)
239
204
69

2.2.3. Decision Tree:

A decision tree is a decision support tool that uses a tree-like graph or model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility. It is one way to display an algorithm that only contains conditional control statements.

It belongs to the family of supervised Machine learning. It is like a flow chart. Each leaf represents a tribute

In this data set, I created the decision tree with the number of leafs 2 and 5.

For no. of leafs = 5

```
from sklearn.tree import DecisionTreeRegressor
dt = DecisionTreeRegressor(min_samples_leaf=5)
dt.fit(train[predictors], train["cnt"])
```

Output: DecisionTreeRegressor(criterion='mse', max_depth=None, max_features=None, max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, min_samples_leaf=5, min_samples_split=2, min_weight_fraction_leaf=0.0, presort=False, random_state=None, splitter='best')

```
predictions = dt.predict(test[predictors])
np.mean((predictions - test["cnt"]) ** 2)
```

Output:

614927.365592092

For no. of leafs = 2

```
from sklearn.tree import DecisionTreeRegressor
dt = DecisionTreeRegressor(min_samples_leaf=2)
dt.fit(train[predictors], train["cnt"])
```

Output: DecisionTreeRegressor(criterion='mse', max_depth=None, max_features=None, max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, min_samples_leaf=2, min_samples_split=2, min_weight_fraction_leaf=0.0, presort=False, random_state=None, splitter='best')

```
predictions = dt.predict(test[predictors])
np.mean((predictions - test["cnt"]) ** 2)
```

Output: 814263.0835275834

2.2.4 Random Forest Model:

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of overfitting to their training set.

In RF model we have to split the data into train and test data sets.

R- Syntax

```
Call:
lm(formula = cnt ~ temp + workingday + weathersit + atemp + hum +
   windspeed, data = train_df)
```

```
By calling the above syntax we will get the below :
Coefficients:
             (Intercept)                     temp
                 2491.59                 -5480.32
              workingday  weathersitModerate:Cloudy/Mist
                   39.69                  -252.08
 weathersitBad: Rain/Snow/Fog                    atemp
                -1463.84                 13035.28
                     hum                windspeed
                -2193.27                 -2119.61
```

```
# Predicting on test set

predTrain <- predict(lm_model, test_df)

library(randomForest)

        train_df <- day[1:547, ]
        test_df <- day[547:nrow(day), ]

# Create a Random Forest model

library(randomForest)

rf_model <- randomForest(cnt ~ temp+ workingday + weathersit + atemp + hum + windspeed, data =
train_df, ntree = 10)

print(rf_model)

Call:
 randomForest(formula = cnt ~ temp + workingday + weathersit +      atemp + hum + windspeed, data
= train_df, ntree = 10)
```

By calling the above syntax we will get below:

Type of random forest: regression
Number of trees: 10
No. of variables tried at each split: 2
Mean of squared residuals: 1429108
% Var explained: 51.13

Python Syntax:

```
from sklearn.ensemble import RandomForestRegressor
rf = RandomForestRegressor(min_samples_leaf=2)
rf.fit(train[predictors], train["cnt"])
```

Output:

```
RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=None, max_features='auto', max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, min_samples_leaf=2, min_samples_split=2, min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=None, oob_score=False, random_state=None, verbose=0, warm_start=False)
```

```
predictions = rf.predict(test[predictors])
np.mean((predictions - test["cnt"]) ** 2)
```

Output: 437690.7193734511

3 Correlation Matrix:

A correlation matrix is a table showing correlation coefficients between variables. Each cell in t he table shows the correlation between two variables. A correlation matrix is used as a way to s ummarize data, as an input into a more advanced analysis, and as a diagnostic for advanced an alysis.

	Temp	atemp	hum	windspeed	cnt
Temp	1.000000	0.991702	0.126963	-0.157944	0.627494
Atemp	0.991702	1.000000	0.139988	-0.183643	0.631066
Hum	0.126963	0.139988	1.000000	-0.248489	-0.100659
Windspeed	-0.157944	-0.183643	-0.248489	1.000000	-0.234545
cnt	0.627494	0.631066	-0.100659	-0.234545	1.000000

4 Chi-Square Test:

A chi-squared test, also written as χ^2 test, is any statistical hypothesis test where th e sampling distribution of the test statistic is a chi-squared distribution when the null hypothesis is true. W ithout other qualification, 'chi-squared test' often is used as short for *Pearson's* chi-squared test. The chi-s quared test is used to determine whether there is a significant difference between the expected frequenc ies and the observed frequencies in one or more categories.

```
#Chisquare test of independence
#Save categorical variables
cat_names = ['season', 'holiday', 'mnth', 'weekday', 'workingday', 'weathersit']
```

In this dataset, I have divided the data into the categorical names and I applied the chi-square test on this variables.

```
from scipy.stats import chi2_contingency
#loop for chi square values
for i in cat_names:
    print(i)
    chi2, p, dof, ex = chi2_contingency(pd.crosstab(day_df['temp'], day_df[i]))
    print(p)
```

Output:

```
season 4.315570887979698e-05
holiday 0.21343418010240936
mnth 0.00012805631695426737
weekday 0.5333494444007716
workingday 0.7152546016917956
weathersit 0.05640620552827349
```

5 Feature Scaling:

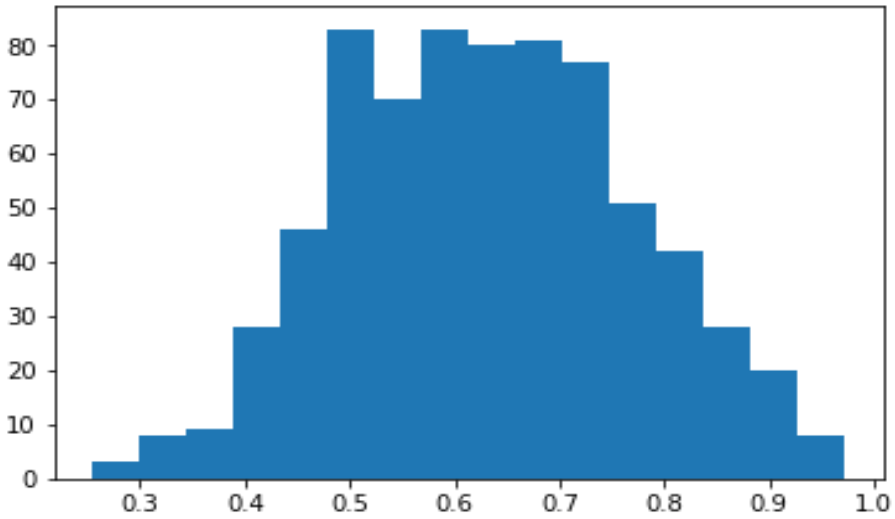
Feature scaling is a method used to normalize the range of independent variables or features of data. In data processing, it is also known as data normalization and is generally performed during the data pre-processing step.

In this I have calculated the normality check of the “hum” variable and created the histogram of hum and cnt variable.

5.1 Normality check:

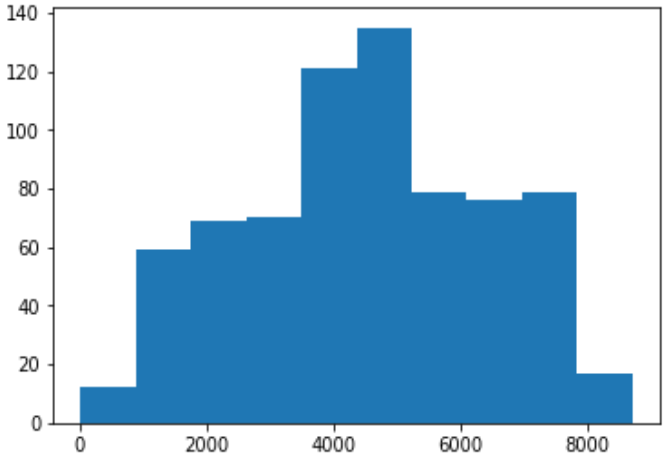
```
%matplotlib inline
plt.hist(day_df['hum'], bins='auto')
```

```
Output:
(array([ 3.,  8.,  9., 28., 46., 83., 70., 83., 80., 81., 77., 51., 42.,
        28., 20.,  8.]),
 array([0.254167  , 0.29906281, 0.34395863, 0.38885444, 0.43375025,
        0.47864606, 0.52354188, 0.56843769, 0.6133335 , 0.65822931,
        0.70312513, 0.74802094, 0.79291675, 0.83781256, 0.88270837,
        0.92760419, 0.9725    ]),
 <a list of 16 Patch objects>)
```




```
plt.hist(day_df['cnt'])
```

```
Output:
(array([ 12., 59., 69., 70., 121., 135., 79., 76., 79., 17.]), array([ 22. , 891.2, 1760.4, 2629.6, 3498.8,
4368. , 5237.2, 6106.4, 6975.6, 7844.8, 8714. ]), <a list of 10 Patch objects>)
```



6 Contingency Table:

In statistics, a contingency table (also known as a cross tabulation or crosstab) is a type of table in a matrix format that displays the (multivariate) frequency distribution of the variables. ... They provide a basic picture of the interrelation between two variables and can help find interactions between them

#showing correlation between working day and holiday

day_crosstab = pd.crosstab(day_df['weekday'], day_df['holiday'],margins = False)
day_crosstab

holiday	0.0	1.0
weekday		
0.0	103	0
1.0	87	15
2.0	102	1
3.0	102	1
4.0	99	2
5.0	101	2
6.0	102	0

7 Cluster Analysis:

Cluster analysis or clustering is the task of grouping a set of objects in such a way that objects in the same group (called a cluster) are more similar (in some sense) to each other than to those in other groups (clusters).

In this we have to load the data and we have to load the required libraries.

```
#load data
df=pd.read_csv(r'C:\Users\Himachala_Venkata\Desktop\edwisor\Projects\Bike_rentals\day (1).csv')

#Load required libraries
from sklearn.cluster import Kmeans
```

We have to assume the optimum number of clusters:

```
cluster_range = range( 1, 20 )
cluster_errors = []

Cluster_errors=[11835.603283173736, 5277.856779255, 4055.652058678603, 3084.526021420196, 2337.5840721017694, 1851.8626415410185, 1648.6103941306728, 1482.26673976758, 1325.562970276756, 1224.603230570794, 1134.212597729793, 1043.35746267226, 973.4908328563333, 921.8834686127166, 858.7984493829784, 801.9701833657983, 771.0524798431378, 711.7773178175139, 697.5209064542446]

for num_clusters in cluster_range:
  clusters = KMeans(num_clusters).fit(df.iloc[:,3:7])
  cluster_errors.append(clusters.inertia_)

Next, we have to create dataframe with cluster errors
clusters_df = pd.DataFrame( { "num_clusters":cluster_range, "cluster_errors": cluster_errors } )
```

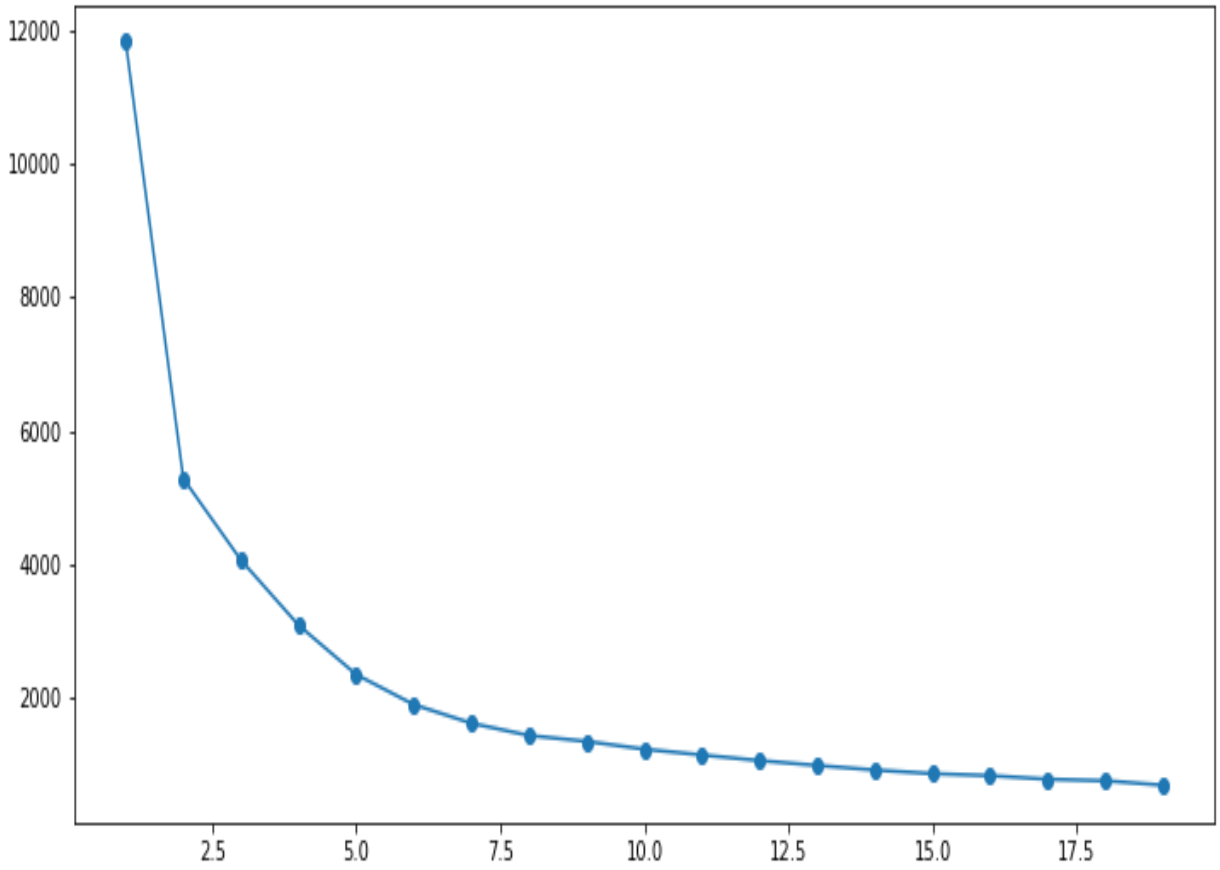
	Num_clusters	Cluster_errors
0	1	11835.603283
1	2	5277.856779
2	3	4055.652059
3	4	3084.526021
4	5	2337.584072
5	6	1851.862642
6	7	1648.610394
7	8	1482.266740
8	9	1325.562970
9	10	1224.603231
10	11	1134.212598
11	12	1043.357463
12	13	973.490833

13	14	921.883469
14	15	858.798449
15	16	801.970183
16	17	771.052480
17	18	711.777318
18	19	697.520906

Plot the line chart to visualize the number of clusters.

```
import matplotlib.pyplot as plt
%matplotlib inline
plt.figure(figsize=(12,6))
plt.plot( clusters_df.num_clusters, clusters_df.cluster_errors, marker = "o" )
```

```
[<matplotlib.lines.Line2D at 0x2bdcbd736a0>]
```



Implement kmeans

7.1 K-Mean Clustering:

K-means clustering is one of the simplest and popular unsupervised machine learning algorithms. ... In other words, the K-means algorithm identifies k number of centroids, and then allocates every data point to the nearest cluster, while keeping the centroids as small as possible.

```
kmeans_model = KMeans(n_clusters = 3).fit(df.iloc[:,3:7])
kmeans_model
```

Output: KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300, n_clusters=3, n_init=10, n_jobs=None, precompute_distances='auto', random_state=None, tol=0.0001, verbose=0)

Summarize the output

```
pd.crosstab(df['cnt'], kmeans_model.labels_)
```

After summarizing, again go through the cluster analysis

```
cluster_range = range( 1, 20 )
cluster_errors = []

for num_clusters in cluster_range:
    clusters = KMeans( num_clusters )
    clusters.fit( df.iloc[:,3:7] )
    cluster_errors.append( clusters.inertia_ )
```

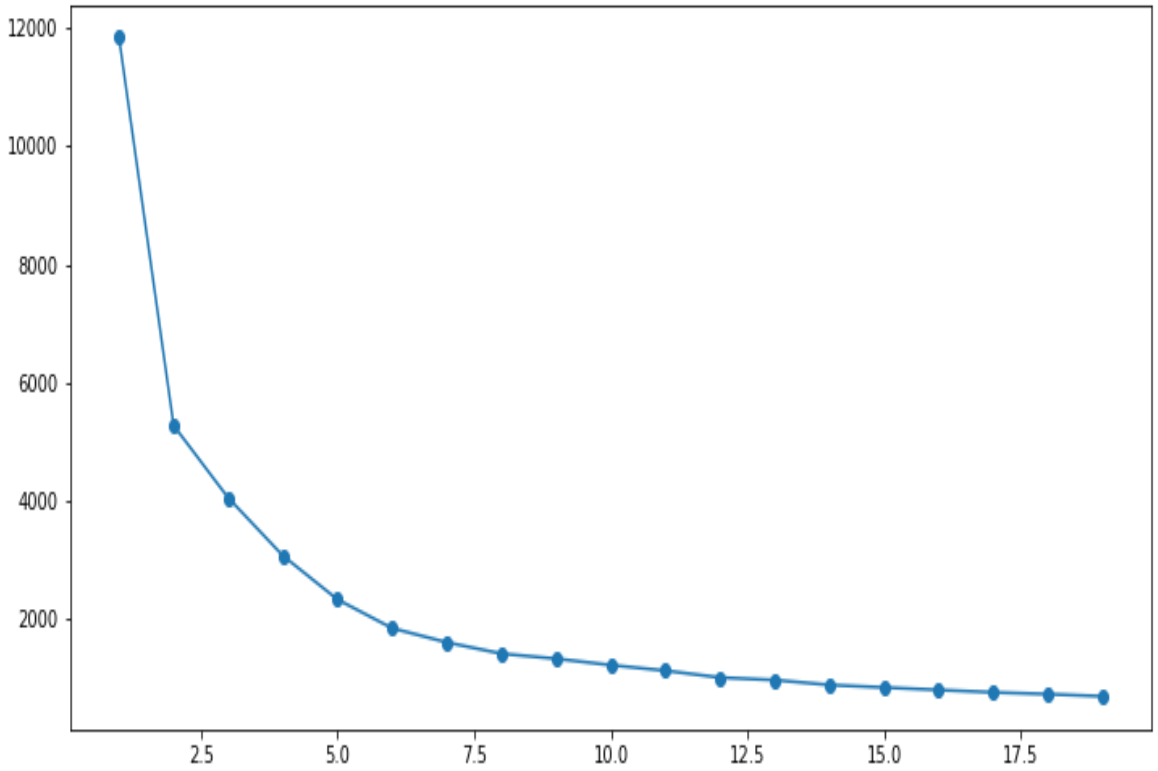
Create the dataframe for cluster errors

```
clusters_df = pd.DataFrame( { "num_clusters":cluster_range, "cluster_errors": cluster_errors } )
clusters_df
```

	Num_clusters	Cluster_errors
0	1	11835.603283
1	2	5277.856779
2	3	4055.652059
3	4	3078.441988
4	5	2337.584072
5	6	1852.784402
6	7	1613.160733
7	8	1422.748481
8	9	1337.088522
9	10	1229.872823
10	11	1135.331500
11	12	1018.935589
12	13	976.985265
13	14	894.524537
14	15	851.247801
15	16	809.702691
16	17	768.424979
17	18	738.935663
18	19	702.052788

```
%matplotlib inline

plt.figure(figsize=(12,6))
plt.plot( clusters_df.num_clusters, clusters_df.cluster_errors, marker = "o" )
```

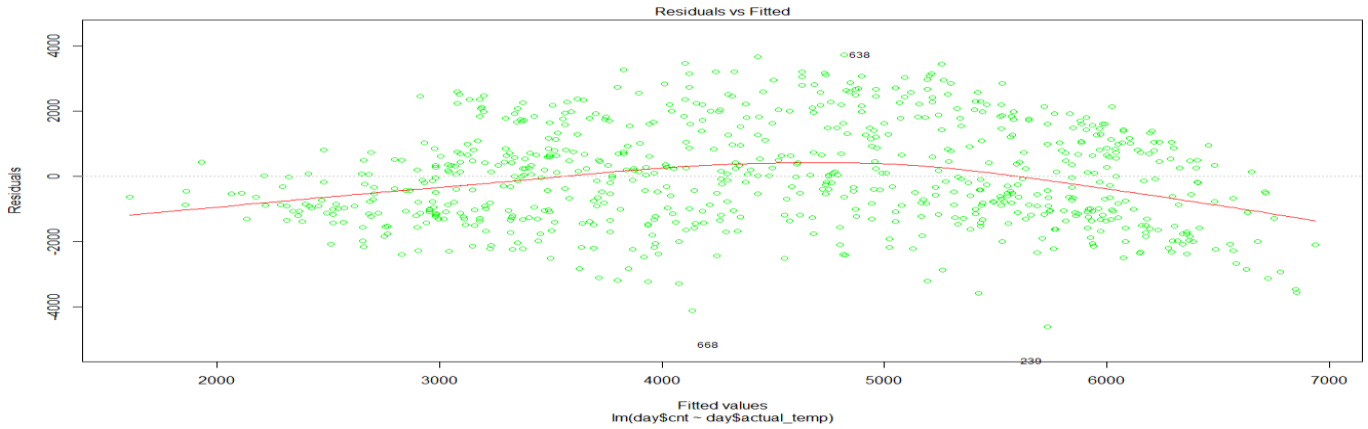
```
[<matplotlib.lines.Line2D at 0x2bdcbe8e5f8>]
```

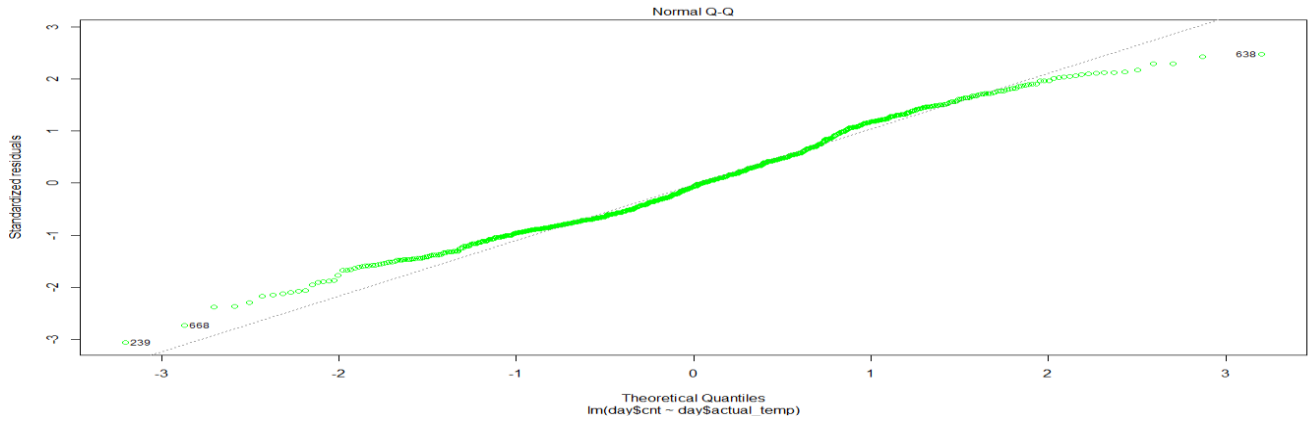


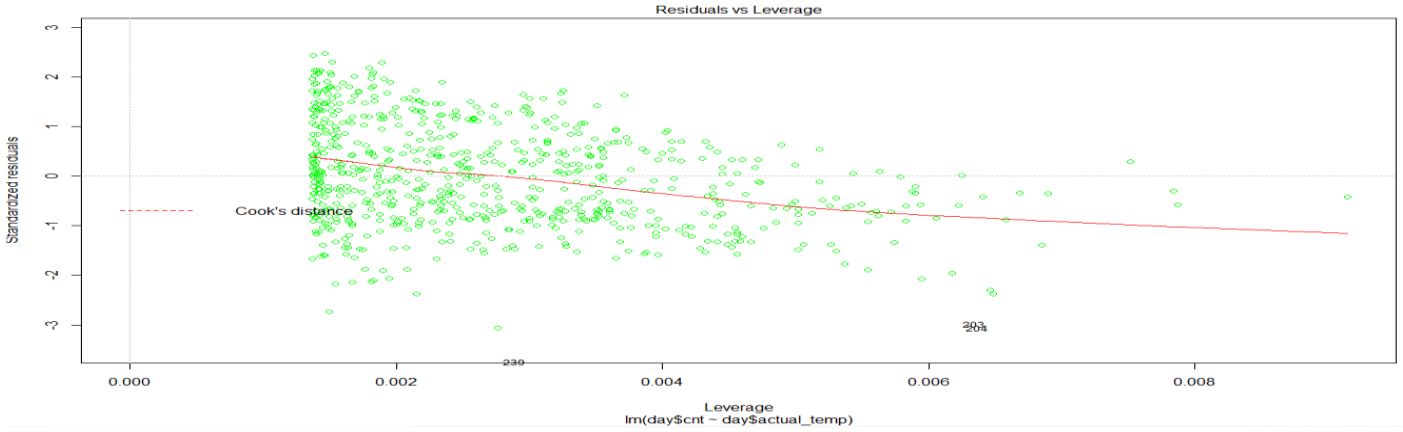
8 Lm Test:

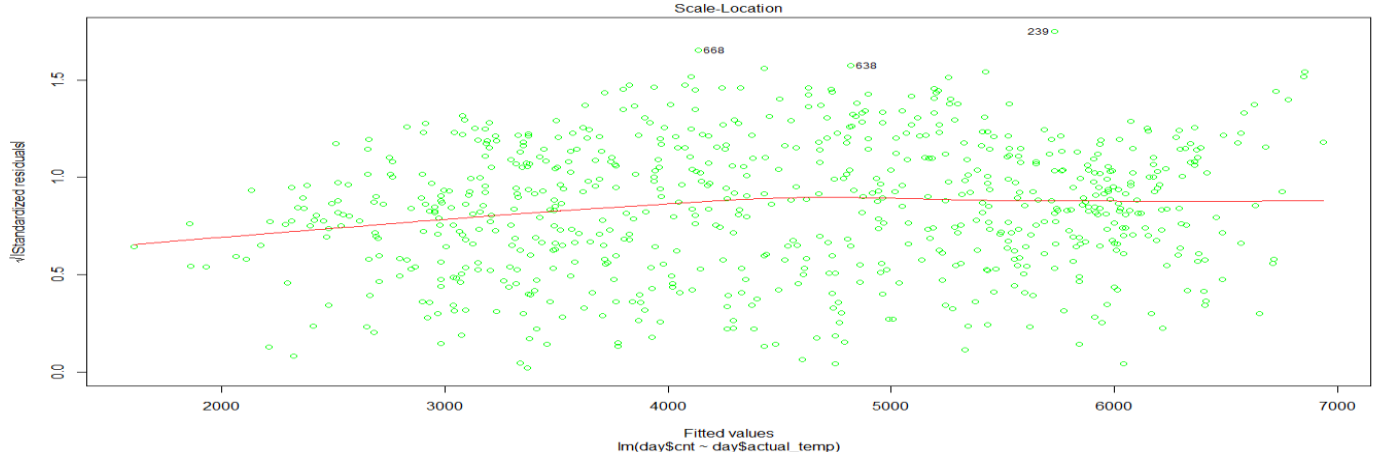
The Lagrange Multiplier (LM) test is a general principle for testing hypotheses about parameters in a likelihood framework. The hypothesis under test is expressed as one or more constraints on the values of parameters. To perform an LM test only estimation of the parameters subject to the restrictions is required.

```
dev.off()
null device
          1
lm_test<- lm(day$cnt~day$actual_temp)
summary(lm_test)

Call:
lm(formula = day$cnt ~ day$actual_temp)

Residuals:
    Min      1Q  Median      3Q     Max
-4615.3 -1134.9  -104.4  1044.3  3737.8

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)     1214.642    161.164   7.537 1.43e-13 ***
day$actual_temp  161.969      7.444  21.759  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1509 on 729 degrees of freedom
Multiple R-squared:  0.3937,    Adjusted R-squared:  0.3929
F-statistic: 473.5 on 1 and 729 DF,  p-value: < 2.2e-16
```




Normal Q-Q
Standardized residuals
Theoretical Quantiles
lm(day$cnt ~ day$actual_temp)
638
668
239


Residuals vs Leverage
Standardized residuals
Cook's distance
Leverage
lm(day$cnt ~ day$actual_temp)
239


Scale-Location
√|Standardized residuals|
Fitted values
lm(day$cnt ~ day$actual_temp)
239
668
638

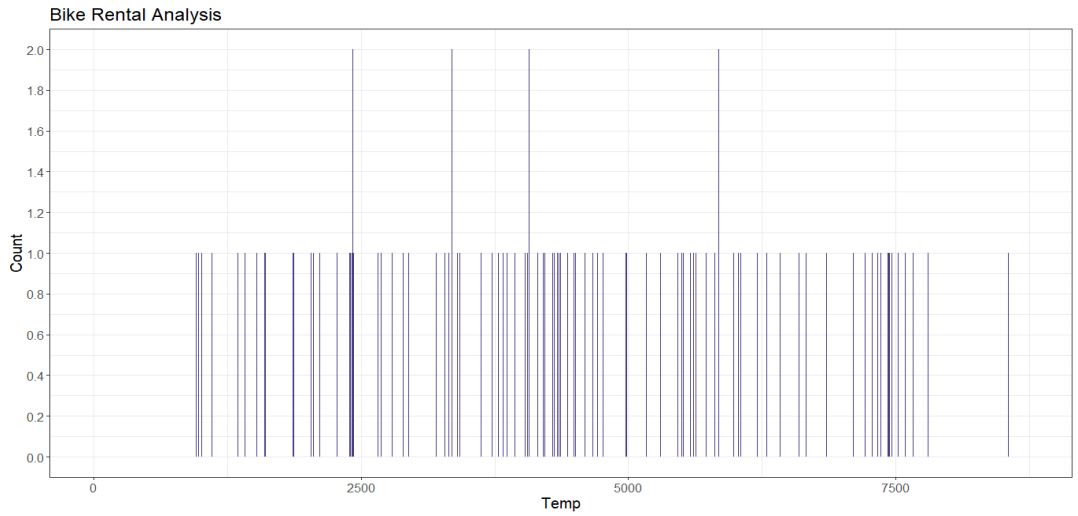
9 Visualizations:

It is a technique for creating images, diagrams, or animations to communicate a message. Load the data and load the required libraries for visualizations.

9.1 Bar Plot of count and temp :

In R:

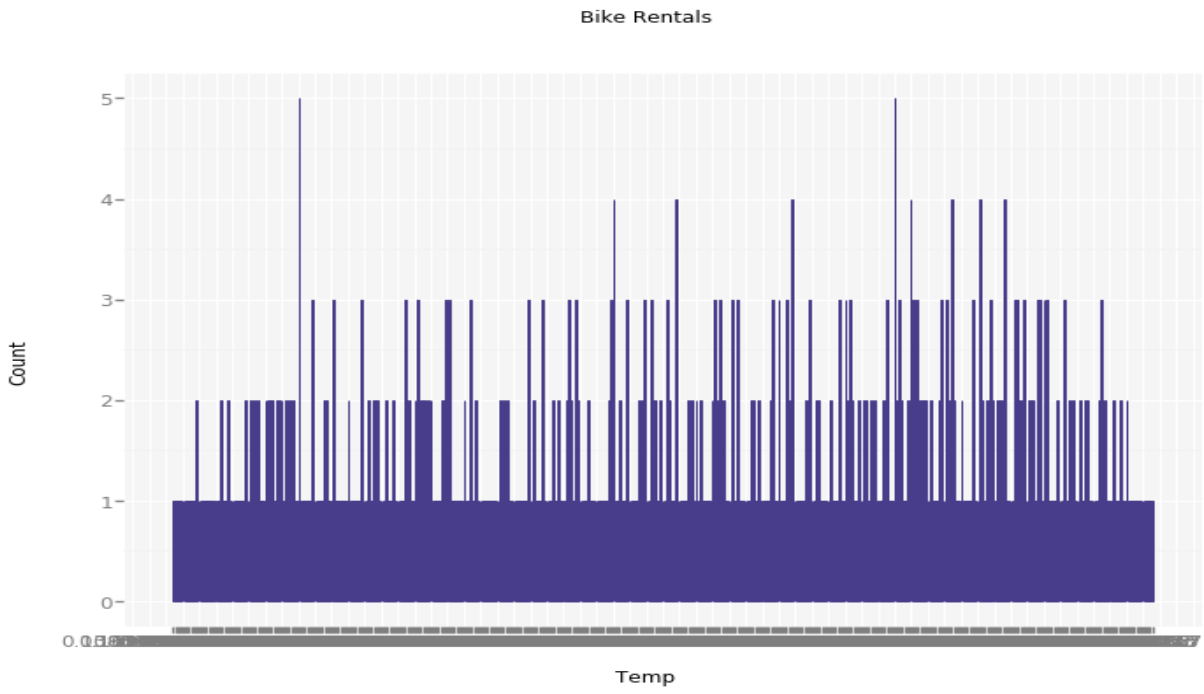
```
#Bar plot(categorical data)
 #If you want count then stat="bin"
      ggplot(day, aes_string(x = day$cnt)) +
       geom_bar(stat="count",fill =  "DarkSlateBlue") + theme_bw() +
       xlab("Temp") + ylab('Count') + scale_y_continuous(breaks=pretty_breaks(n=10)) +
       ggtitle("Bike Rental Analysis") +  theme(text=element_text(size=15))
```



In Python:

```
ggplot(day_df, aes(x='temp', y='cnt')) +\
  geom_bar(fill= "DarkSlateBlue") +\
  scale_color_brewer(type='diverging', palette=4) +\
  xlab("Temp") + ylab("Count") + ggtitle("Bike Rentals") + theme_bw()
```

```
<ggplot: (-9223371848465841934)>
```

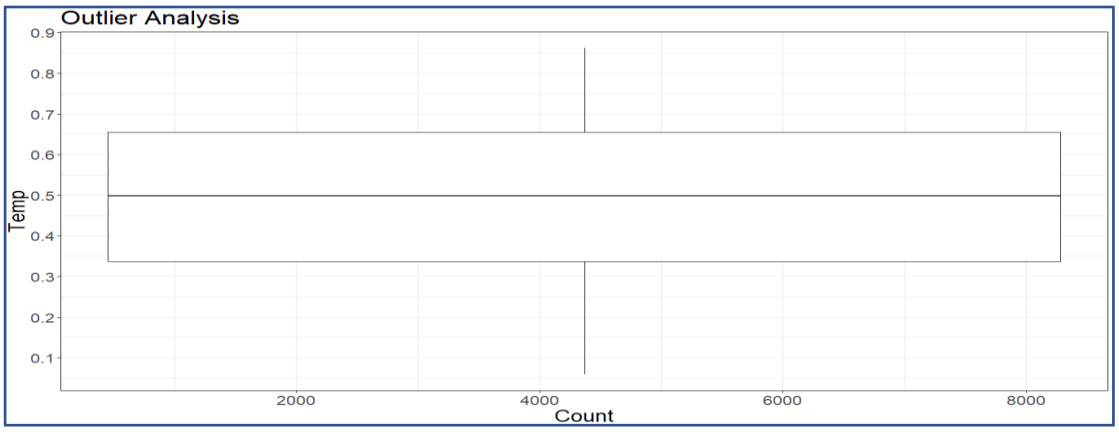


9.2 Box Plot of count and temp :

In R:

```
#Box plot
ggplot(day, aes_string(x = day$cnt, y = day$temp, fill = day$cnt)) +
 geom_boxplot(outlier.colour = "red", outlier.size = 3) +
 scale_y_continuous(breaks=pretty_breaks(n=10)) +
 guides(fill=FALSE) + theme_bw() + xlab("Count") + ylab("Temp") +
```

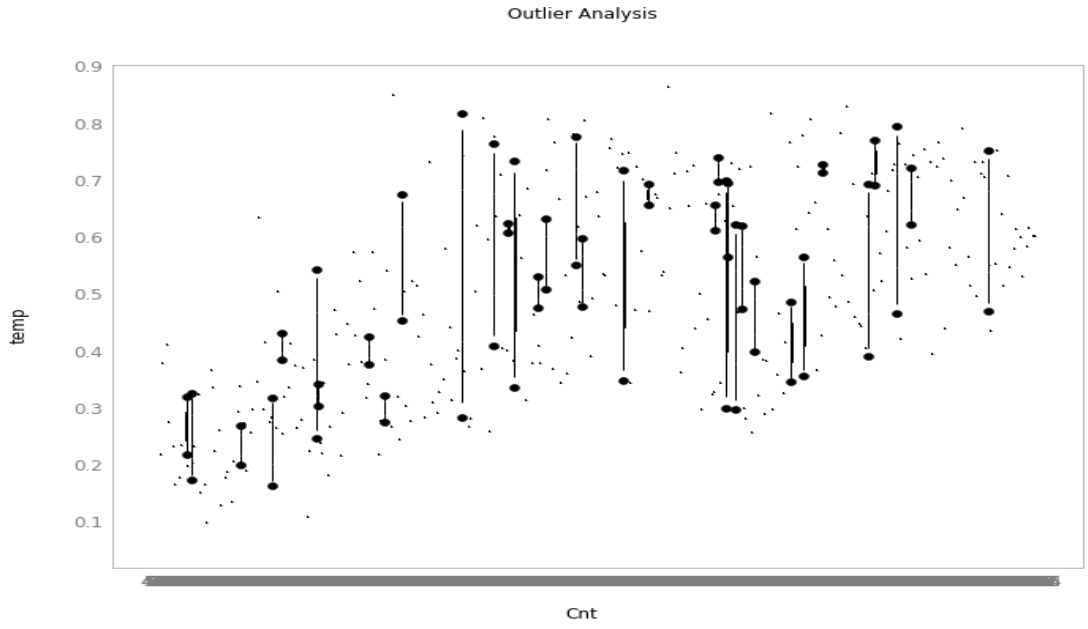
```
 ggtitle("Outlier Analysis") +
 theme(text=element_text(size=20))
```



In python:

```
ggplot(day_df, aes(x = 'cnt', y = 'temp', fill = 'cnt')) +\
  geom_boxplot(fill = 'cnt')+ theme_bw() + xlab("Cnt") + ylab("temp") +\
  ggtitle("Outlier Analysis") +\
  theme(text=element_text(size=30))
```

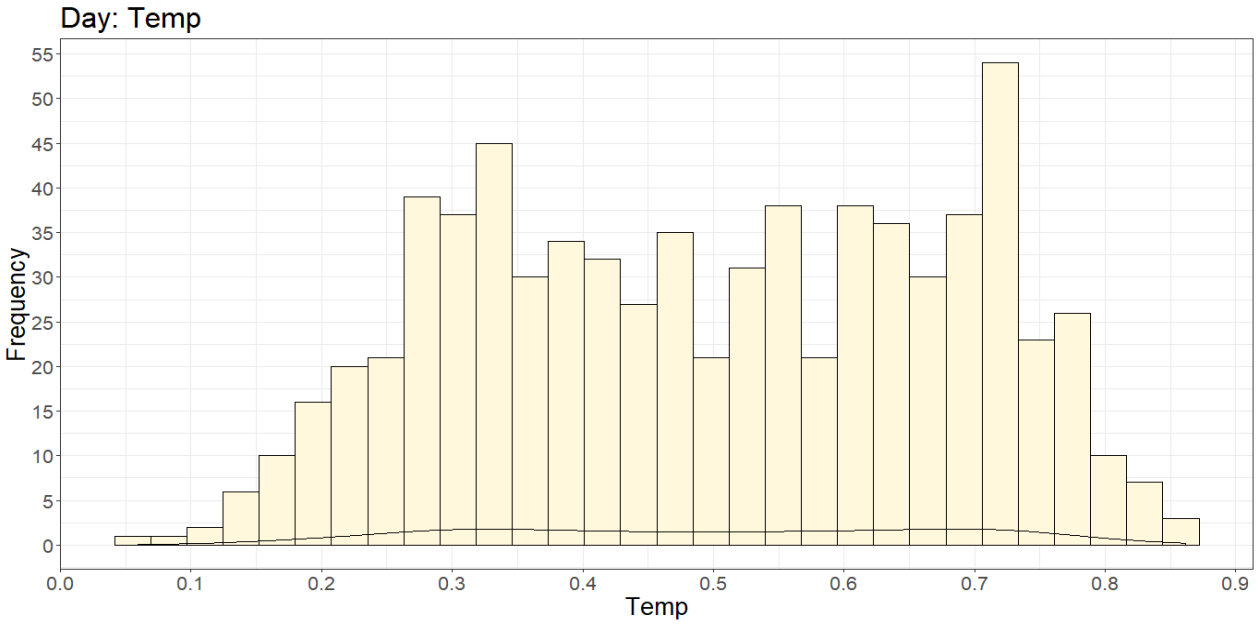
```
<ggplot: (-9223371848464144966)>
```



9.3 Histogram:

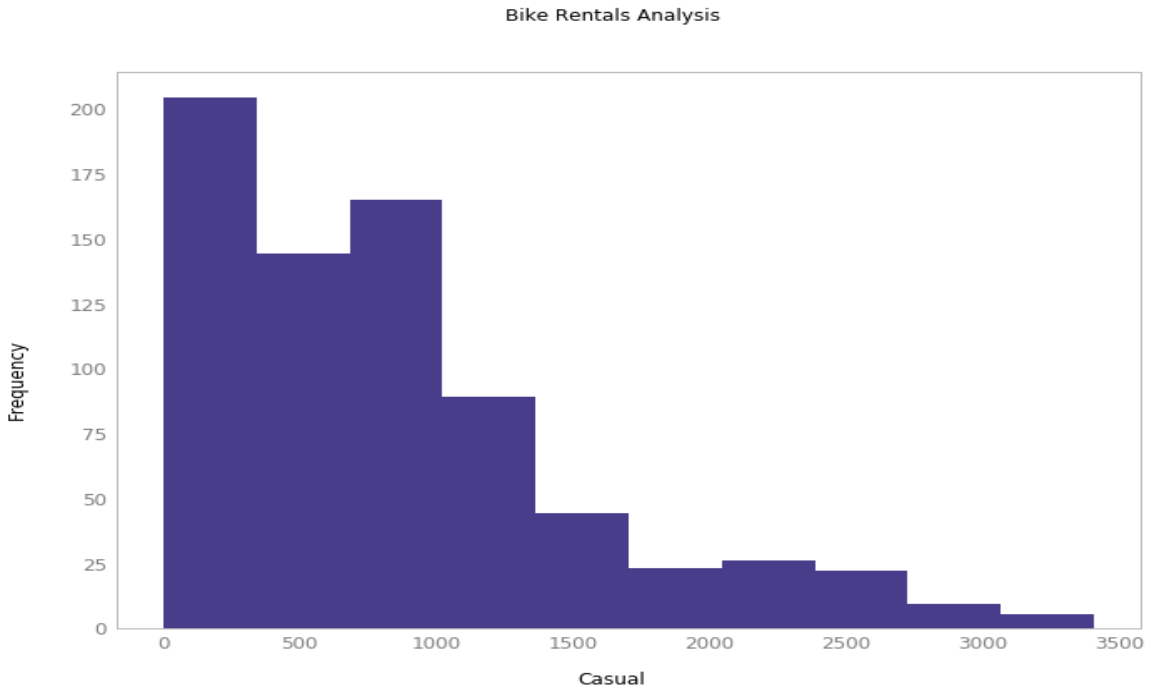
9.3.1 Histogram of Temp frequency in R:

```
ggplot(day, aes_string(x = day$temp)) +
 geom_histogram(fill="cornsilk", colour = "black") + geom_density() +
 scale_y_continuous(breaks=pretty_breaks(n=10)) +
 scale_x_continuous(breaks=pretty_breaks(n=10))+
 theme_bw() + xlab("Temp") + ylab("Frequency") + ggtitle("Day: Temp") +
 theme(text=element_text(size=20))
```



9.3.2 Histogram of casual frequency in python:

```
ggplot(day_df, aes(x = 'casual')) + geom_histogram(fill="DarkSlateBlue", colour = "black") +\
  geom_density() +\
  theme_bw() + xlab("Casual") + ylab("Frequency") + ggtitle("Bike Rentals Analysis") +\
  theme(text=element_text(size=20))
```



References

James, Gareth, Daniela Witten, Trevor Hastie, and Robert Tibshirani. 2013. An Introduction to Statistical Learning. Vol. 6. Springer.

Wickham, Hadley. 2009. Ggplot2: Elegant Graphics for Data Analysis. Springer Science & Business Media.