PROJECT

Bike Renting

SUBMITTED BY VISHAKHA GAIKWAD

Table of content

1 Chapter: - Introduction

1.1 Problem statement

The purpose of this project is to estimate daily bike rental calculations. Considering all factors such as changing session and environmental impact, the bike rental company that has historical data. we have to estimate bike rental count. These predicted values will help the business meet demand for those specific days to maintain supply levels.

There are currently several bike rental companies. Who provide their services to billions of customers every Day? It is important to manage their data properly for new business idea. In this case, we need to identify which days can be the most demanding, so we have better strategy to cope with that day's demand.

1.2 Data

In the given data have 16 variable and 731 observations. The 'cnt' is dependent variable or target variable and the rest are independent.

A snapshot of the data is mentioned.

instant	dteday	season	yr	mnth	holiday	weekday	workingda	weathersi	temp	atemp	hum	windspee	casual	registerec	cnt
1	1/1/2011	1	0	1	0	6	0	2	0.344167	0.363625	0.805833	0.160446	331	654	985
2	1/2/2011	1	0	1	0	0	0	2	0.363478	0.353739	0.696087	0.248539	131	670	801
3	1/3/2011	1	0	1	0	1	1	1	0.196364	0.189405	0.437273	0.248309	120	1229	1349
4	1/4/2011	1	0	1	0	2	1	1	0.2	0.212122	0.590435	0.160296	108	1454	1562
5	1/5/2011	1	0	1	0	3	1	1	0.226957	0.22927	0.436957	0.1869	82	1518	1600
6	1/6/2011	1	0	1	0	4	1	1	0.204348	0.233209	0.518261	0.089565	88	1518	1606
7	1/7/2011	1	0	1	0	5	1	2	0.196522	0.208839	0.498696	0.168726	148	1362	1510

Table 1: Bike rental data

Our objective is to develop a model that can determine the count for future test cases. And this model can be developed by the help of given data.

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 fromFreemeteo)

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

This mentioned above attribute are helping us to predicting the value of count variable ‘cnt’ perfectly.

2 Chapter: - Methodology

2.1 Data 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. In that I have explore the data by checking its data type and summery function, we get information here that some variable is not having much information to calculate the target variable, such as instant, dteday, casual, registered. By knowing the type of variable, I have place them in two categories.

Here, we will use techniques like missing value analysis, outlier analysis, feature selection, feature scaling. This technique is used to structure our data. Basically, pre-processing is done because and the model asks for structured data and preprocessing is used to structure the data we have got.

2.1.1 Missing value Analysis

Missing value is availability of incomplete observations in the dataset. These Missing values affect the accuracy of model. So, it becomes important to check missing values in our given data.

Here, in given project, after checking for missing data it is found that data don't have any missing value. the table of missing

```
season        0
yr            0
mnth          0
holiday       0
weekday       0
workingday    0
weathersit    0
temp          0
atemp         0
hum           0
windspeed     0
cnt           0
dtype: int64
```

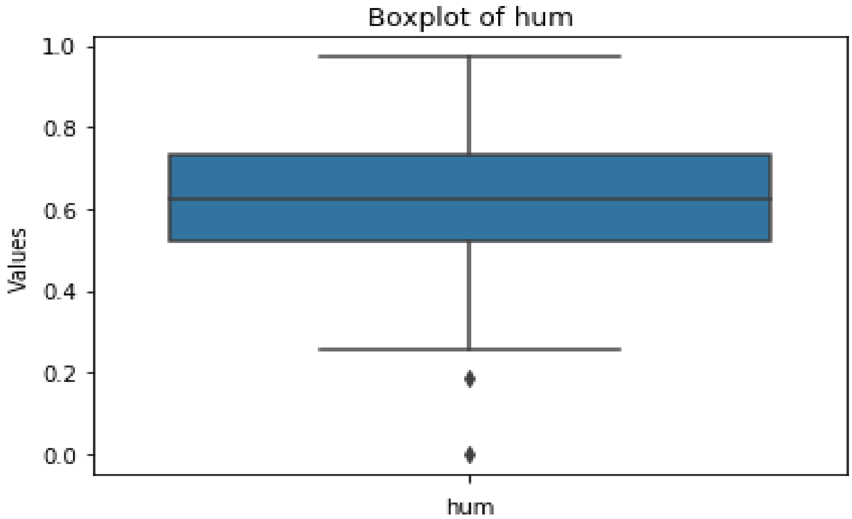
No missing value found

Table 2: -Missing value

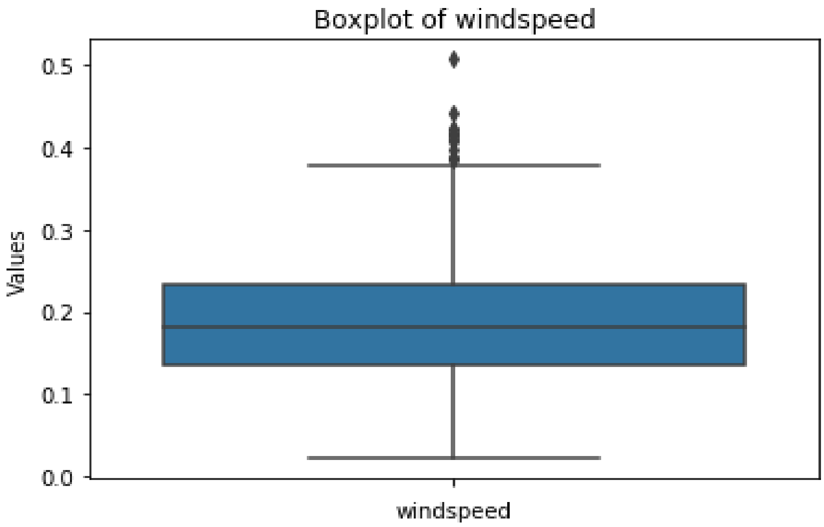
2.1.2 Outlier Analysis

Outlier is an abnormal observation that stands or deviates away from other observations. These happens because of manual error; poor quality of data and it is correct but exceptional data. But, it can cause an error in predicting the target variables.

After having outlier analysis with help of boxplot method, here it is found that in variable humidity and windspeed have outlier problem. The plot is mention shows the actual result of outlier.



Plot1: - humidity outlier



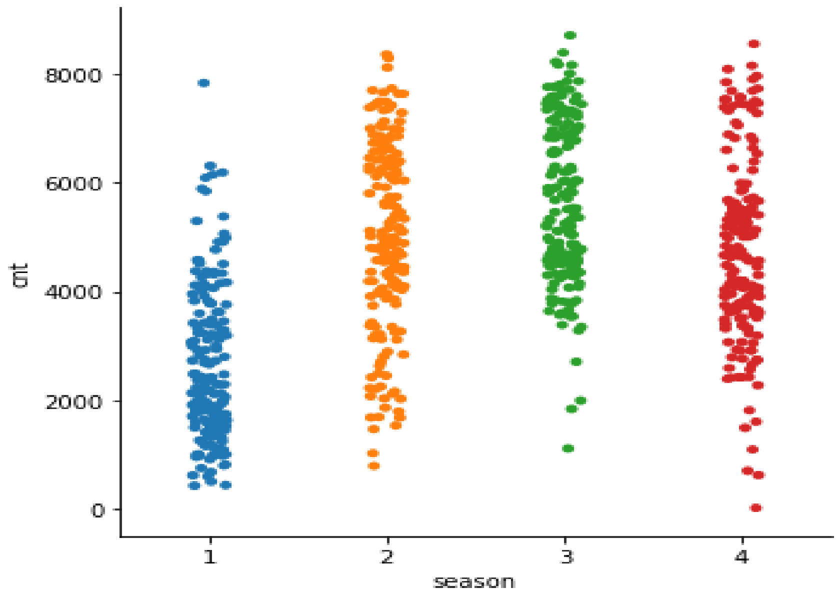
Plot2: - windspeed outlier

all these outliers can hamper our data model. So, there is a requirement to eliminate or replace such outliers and impute with proper methods to get better accuracy of the model. In this project, I used median method to impute the outliers in windspeed and humidity variables.

2.1.3 Exploring Data with visualization

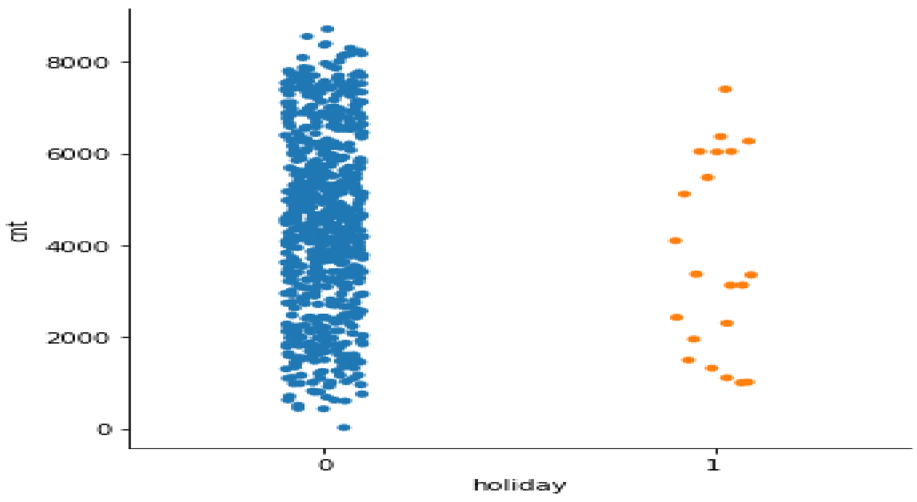
Understanding data is a process where you get better data with the help of visuals. Where it helps with the basic idea of developing models on data. Here with help seaborn family I have created some plots.

Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics. The new **catplot** function provides a new framework giving access to several types of plots that show relationship between numerical variable and one or more categorical variables.



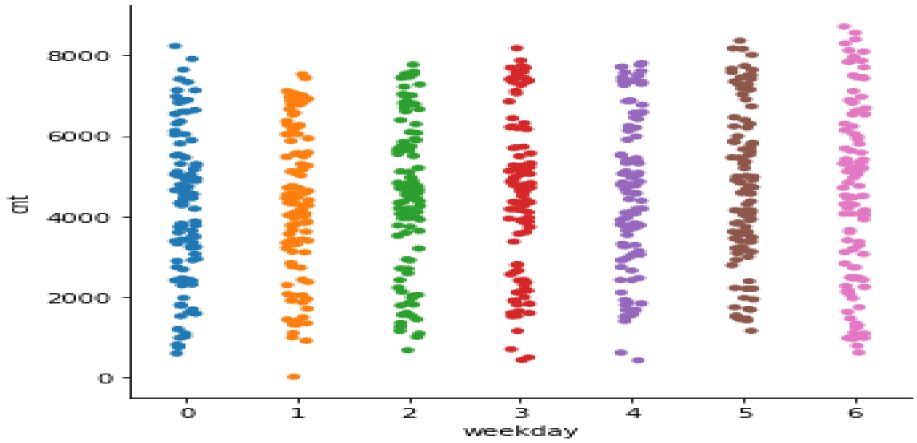
Plot 3: - count vs season

In this graph we have plot relation between season variable and cnt variable, here I have found that **season summer=2, fall=3, spring=4 have maximum count**



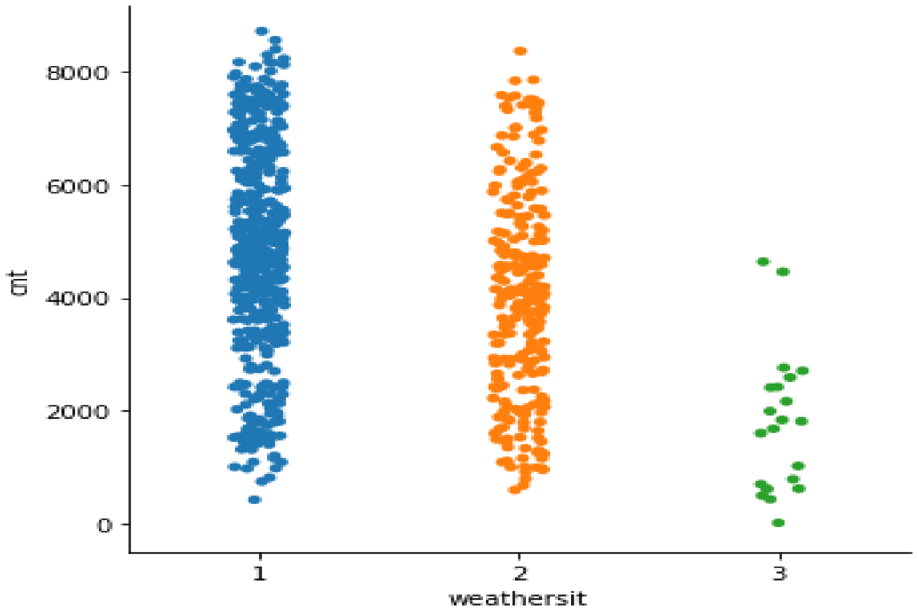
Plot4: - count vs holiday

In this graph we have plot relation between holiday variable and cnt variable, here I have found that **holiday = all holiday has maximum count**



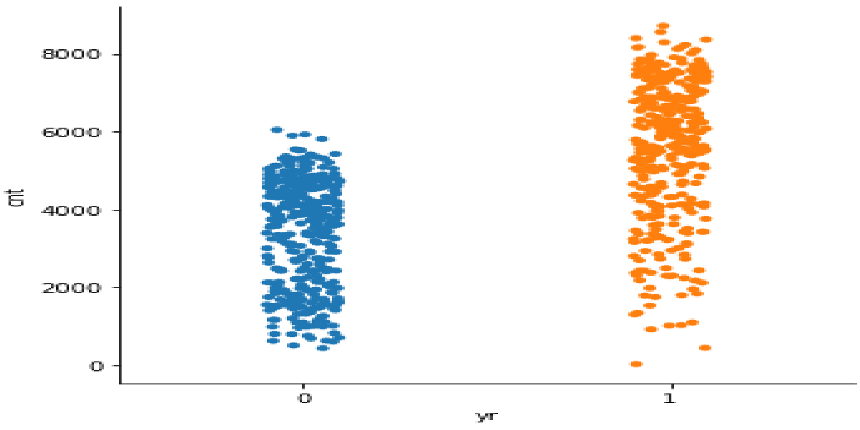
Plot 5: - count vs weekday

In this graph we have plot relation between weekday variable and cnt variable, here I have found that **0 and 6 mean Monday to Saturday the count is highest**



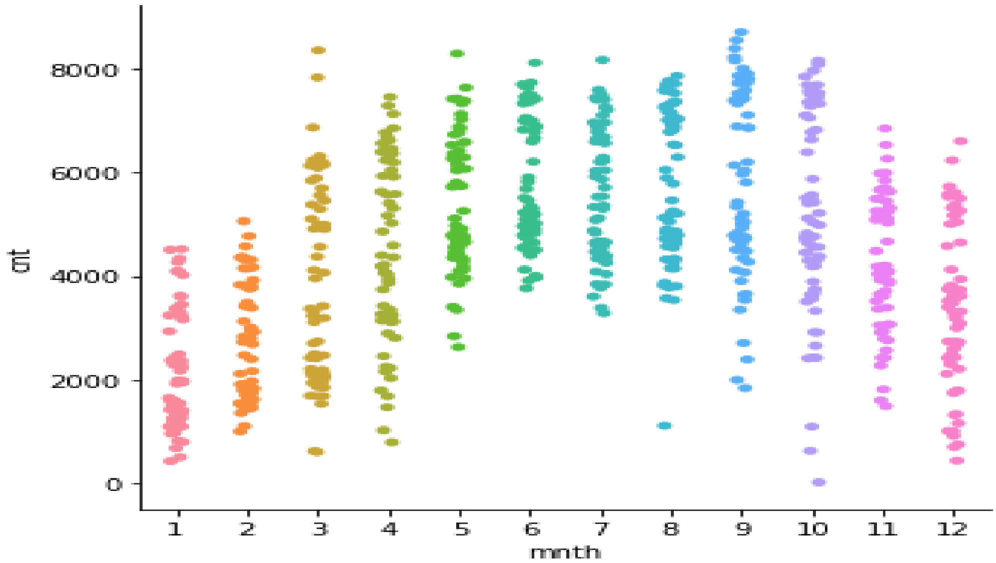
Plot 6: - count vs weathersit

In this graph we have plot relation between weathersit variable and cnt variable, here I have found that **weather 1 has the highest count**



Plot7: - count vs year

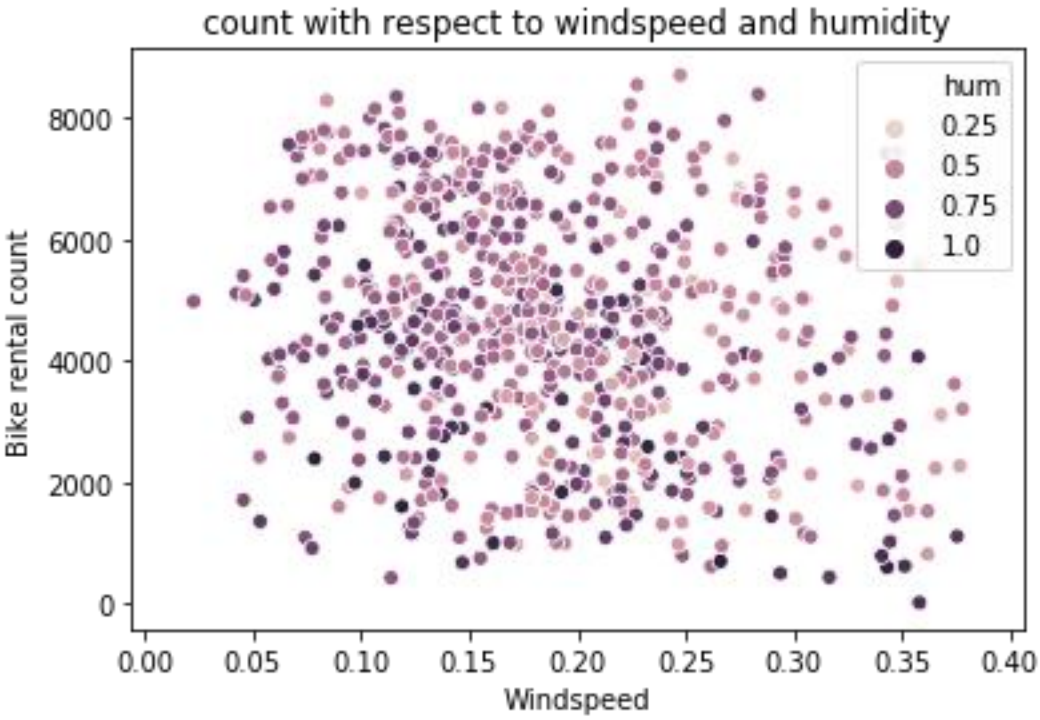
In this graph we have plot relation between yr variable and cnt variable, here I have found that **in yr1= 2012 has the highest count**



Plot8: - count vs month

In this graph we have plot relation between yr variable and cnt variable, here I have found that **Months 3 to 10 we got a good number of count**

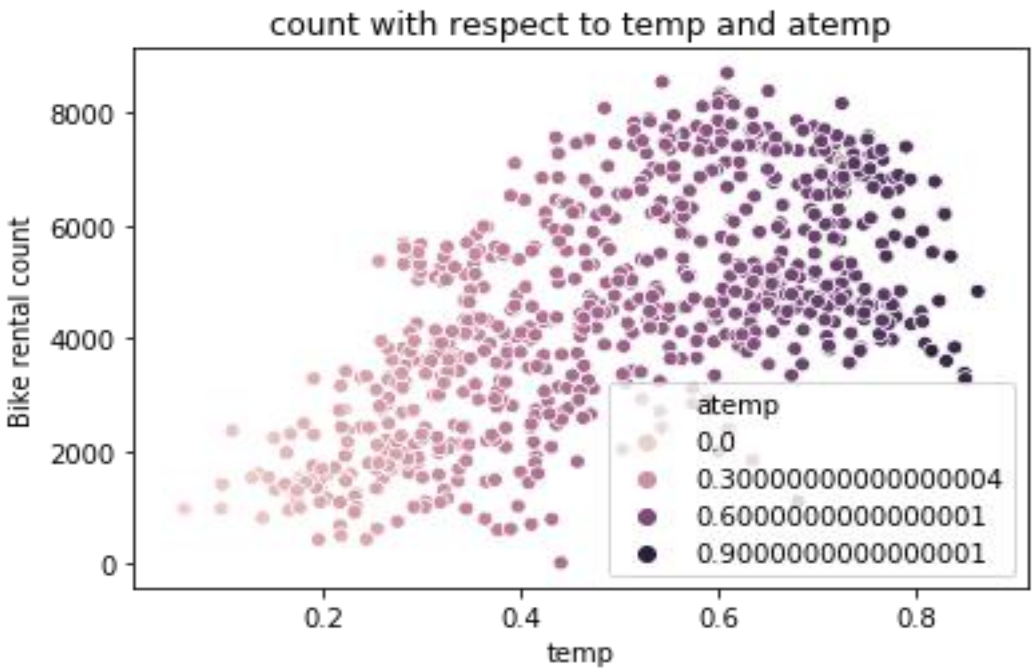
Some scatter plot which giving the information about environment effect on cnt variable.



Plot9: -count vs windspeed and hum

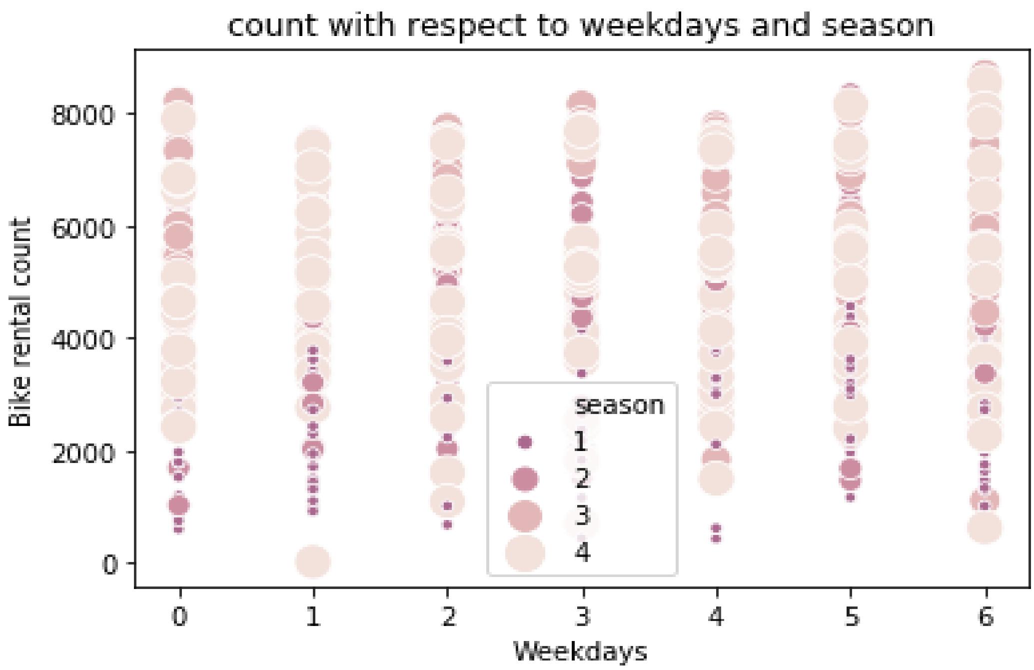
In this scatter plot the highest count of bike occurred when the range of windspeed is between 0.10 to 0.15 whereas humidity ranges from 0.5 to 0.75.

count vs weekdays and season, Count is high in 4th season and 1st and 6th weekday



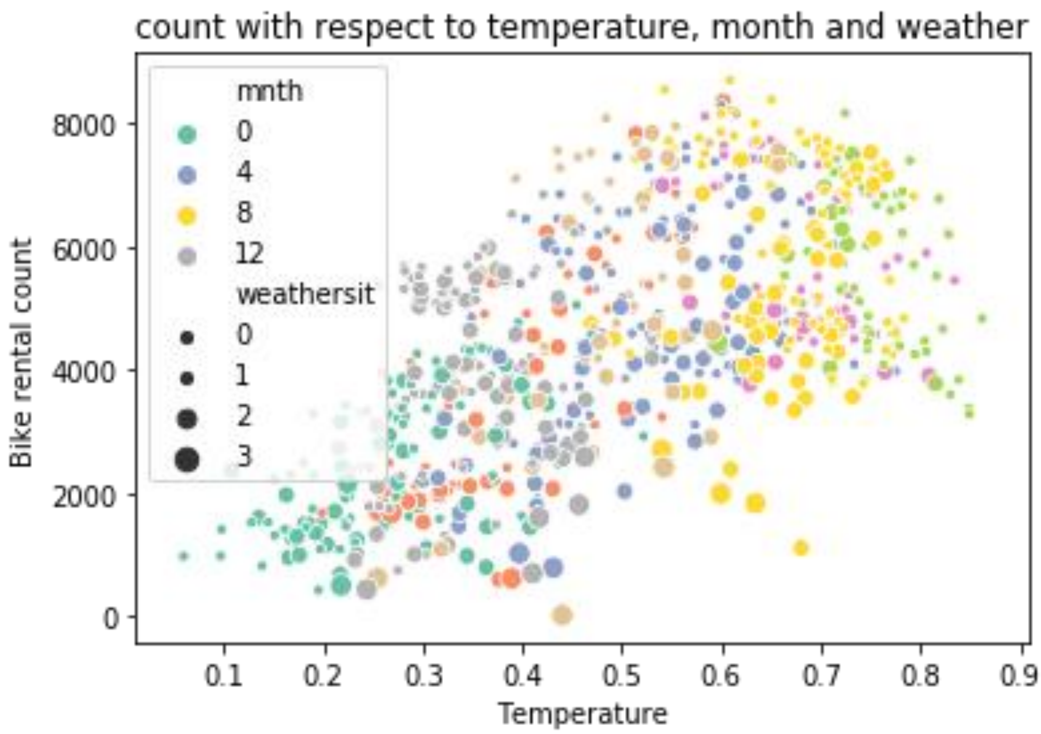
Plot 10: - count vs temp and atemp

In this scatter plot highest count of bike when the temp ranges 0.4 to 0.8 and atemp ranges from 0.3 to 0.9



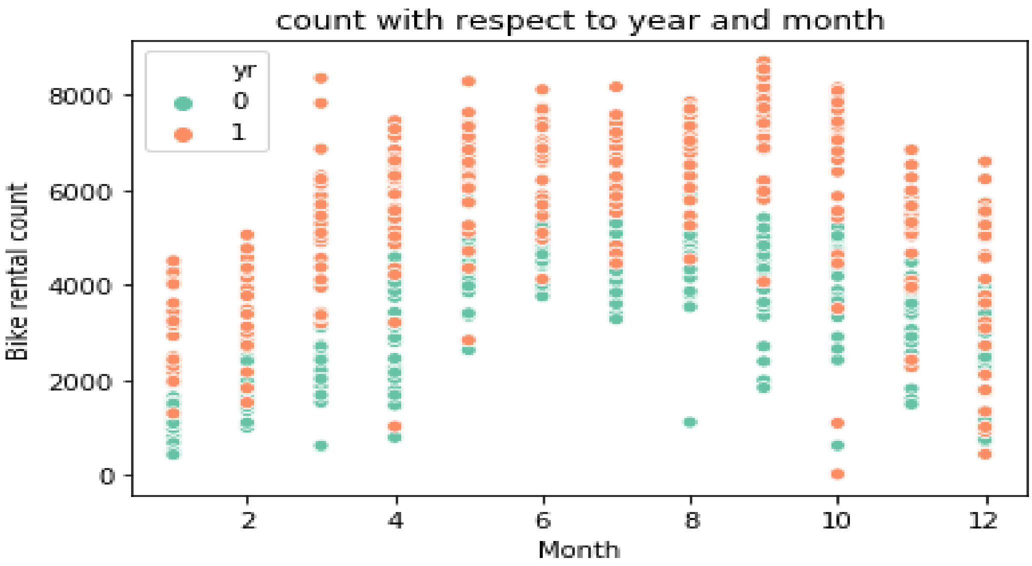
Plot11: -count vs weekdays and season

In the scatter plot of count vs weekdays and season, Count is high in 4th season and 1st and 6th weekday.



Plot12: -count vs temperature month and weather

In this scatter plot, it is found that in count vs temperature, month and weather, Count is high in range temperature 0.5 to 0.8, in 8th month and weather is 0.



Plot13: - count vs year and month

In this scatter plot, it is found that count respect to year and month, count is high in year 1, particularly from season 3 to 12 excluding 9.

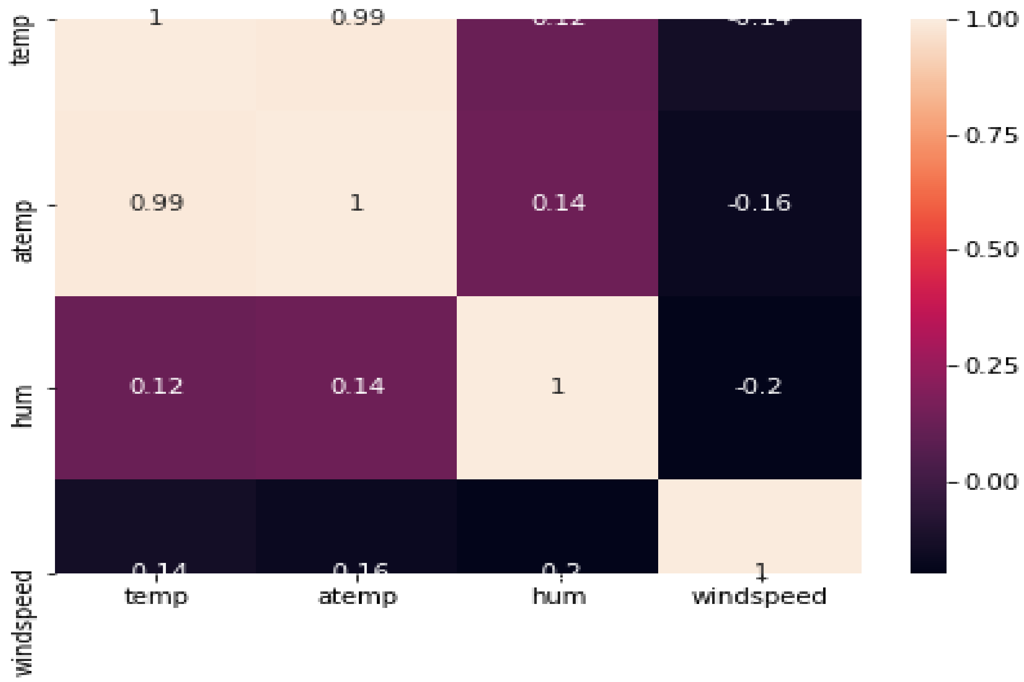
2.1.4 Feature Selection

Feature selection is the process where we go for selection of important variable and drop unwanted one. Feature selection helps by reducing time for computation of model and also reduces the complexity of the model.

Here, in this project correlation analysis is done with numerical variables and ANOVA test is done with categorical variables to check if there is collinearity among the variables. And if there is any collinearity it's better to drop such variables, else this redundant variable can hamper the accuracy of the model.

2.1.4.1 Correlation Plot

The correlation plot for numerical variable is mentioned below:



Plot14: - correlation heat map

Observing here, it is found that temperature and atemp are highly correlated with each other. So, for further processes I have drop atemp as it is similar to temperature. I have drop attempt for reducing complexity problem.

ANOVA Test: - Analysis of variance, it helps to select the important feature among variables. When your independent variable is categorical in nature and your dependent variable is continuous in nature.

```
 For target var = cnt
Anova table between cnt and season is
              df        sum_sq       mean_sq           F        PR(>F)
season       1.0  4.517974e+08  4.517974e+08  143.967653  2.133997e-30
Residual   729.0  2.287738e+09  3.138187e+06         NaN           NaN
Anova table between cnt and holiday is
              df        sum_sq       mean_sq         F    PR(>F)
holiday      1.0  1.279749e+07  1.279749e+07  3.421441  0.064759
Residual   729.0  2.726738e+09  3.740381e+06       NaN       NaN
Anova table between cnt and weekday is
              df        sum_sq       mean_sq         F    PR(>F)
weekday      1.0  1.246109e+07  1.246109e+07  3.331091  0.068391
Residual   729.0  2.727074e+09  3.740843e+06       NaN       NaN
Anova table between cnt and workingday is
                df        sum_sq       mean_sq         F    PR(>F)
workingday     1.0  1.024604e+07  1.024604e+07  2.736742  0.098495
Residual     729.0  2.729289e+09  3.743881e+06       NaN       NaN
Anova table between cnt and weathersit is
                df        sum_sq       mean_sq          F        PR(>F)
weathersit     1.0  2.422888e+08  2.422888e+08  70.729298  2.150976e-16
Residual     729.0  2.497247e+09  3.425578e+06        NaN           NaN
Anova table between cnt and yr is
              df        sum_sq       mean_sq           F        PR(>F)
yr           1.0  8.798289e+08  8.798289e+08  344.890586  2.483540e-63
Residual   729.0  1.859706e+09  2.551038e+06         NaN           NaN
Anova table between cnt and mnth is
              df        sum_sq       mean_sq          F        PR(>F)
mnth         1.0  2.147445e+08  2.147445e+08  62.004625  1.243112e-14
Residual   729.0  2.524791e+09  3.463362e+06        NaN           NaN
```

Table 3: -Result of ANOVA Test

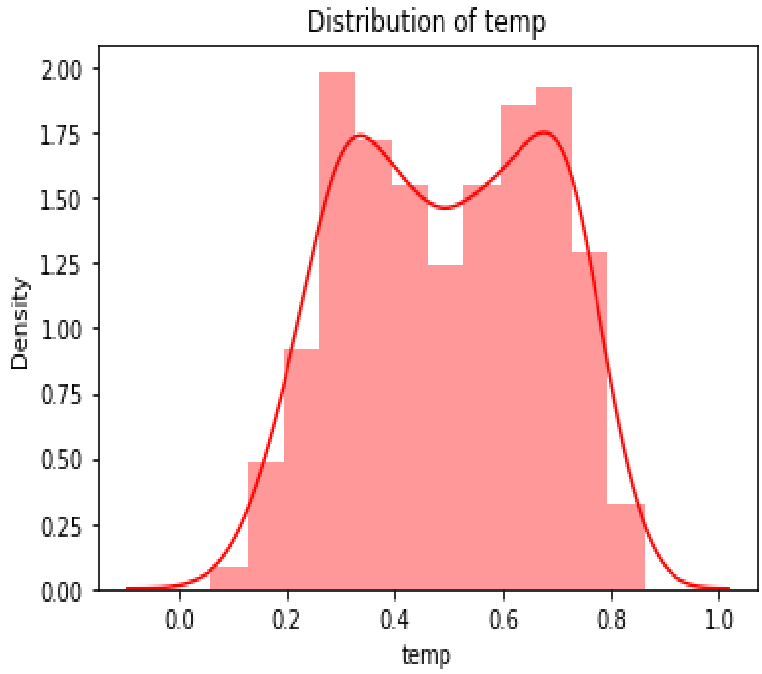
From the observations, it is found that the variables holiday, weekday, and working day has p value > 0.05.

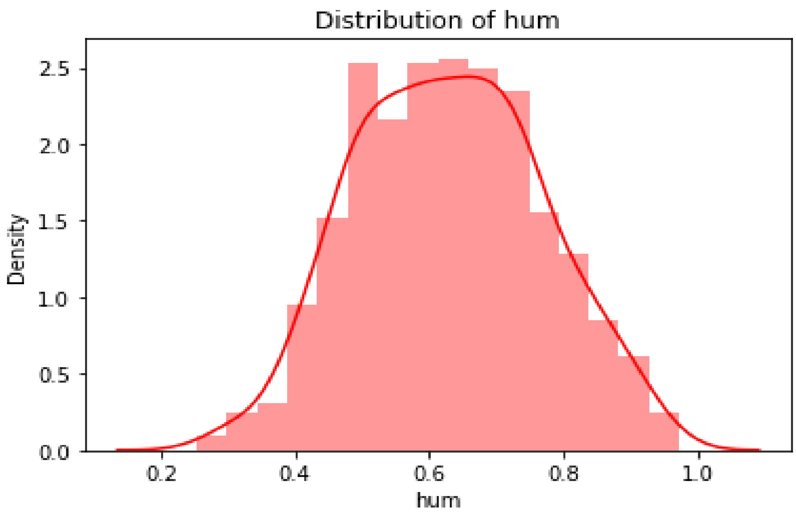
Here, I go for null hypothesis. these variables have no dependency over target variable. So, in further processes these variables can be dropped before modeling. And this process of deducting the variables is also called as dimension reduction.

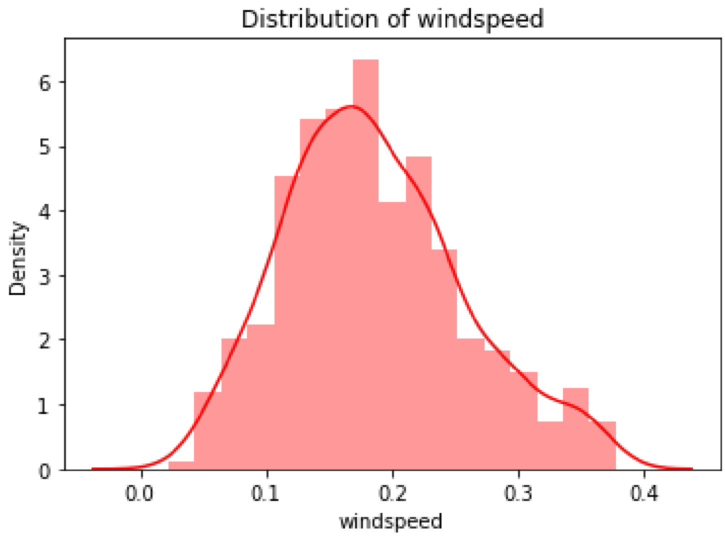
2.1.5 Feature scaling

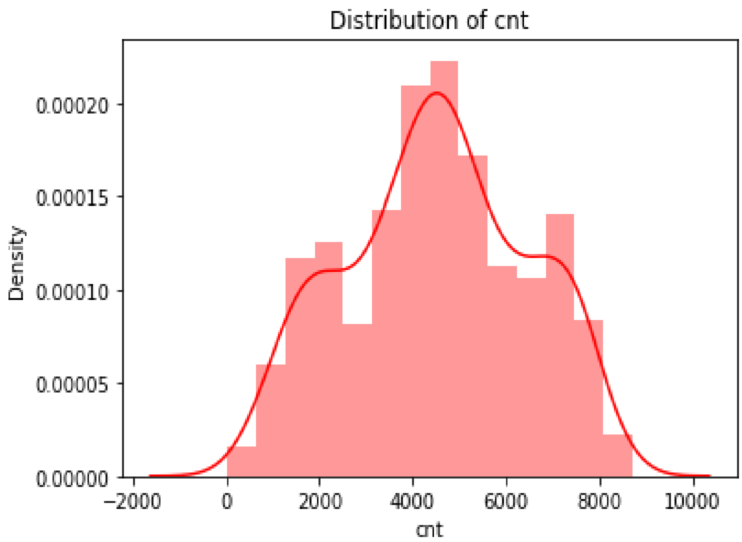
Feature scaling is done when there is relation between multiple columns but the scale of those columns is different at that time. In Feature Scaling ranges of variables are normalized or standardized, such that variables can be compared with same range. This is done for an unbiased and accurate model.

In this project, as the data are found as approximately symmetric. The feature scaling is not required. Following are the plots of approximately symmetric data visuals.









Plot15: - Distribution of variable

	season	yr	mnth	weathersit	temp	hum	windspeed	cnt
count	731.000000	731.000000	731.000000	731.000000	731.000000	731.000000	731.000000	731.000000
mean	2.496580	0.500684	6.519836	1.395349	0.495385	0.629354	0.186257	4504.348837
std	1.110807	0.500342	3.451913	0.544894	0.183051	0.139566	0.071156	1937.211452
min	1.000000	0.000000	1.000000	1.000000	0.059130	0.254167	0.022392	22.000000
25%	2.000000	0.000000	4.000000	1.000000	0.337083	0.522291	0.134950	3152.000000
50%	3.000000	1.000000	7.000000	1.000000	0.498333	0.627500	0.178802	4548.000000
75%	3.000000	1.000000	10.000000	2.000000	0.655417	0.730209	0.229786	5956.000000
max	4.000000	1.000000	12.000000	3.000000	0.861667	0.972500	0.378108	8714.000000

Table: - distribution of variables

By checking the distribution, we have found all the variable are normally distributed so there no need of scaling.

2.2 Model Development

After completing EDA and Data Pre- Processing step we have our next step to develop model on our given data.as per the industry there are several problem categories of the data problem such as Forecasting, Classification, Optimization, Unsupervised Learning.

The process of selecting precise model depends on our goal and the problem statement. In this project the problem statement is to predict the bike rental count on daily basis, considering the environmental and seasonal settings. Thus, the problem statement is an identified as regression problem and falls under the category of forecasting, where we have to forecast a numeric data or continuous variable for the target.

This forecasting problem is of regression type so here we have using some regression algorithm model to predict the accurate value of count.

2.2.1 Error Matrix to decide Accuracy of model

MAPE: - The 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. Where A_t is the actual value and F_t is the forecast value, this is given by:

$$\mathrm{M} = \frac{1}{n}\sum_{t=1}^{n}\left|\frac{A_t - F_t}{A_t}\right|$$

Formula: -MAPE

The mean absolute percentage error (MAPE) is the most common measure used to forecast error and works best if there are no extremes to the data (and no zeros).

Accuracy: -The second matric to identify or compare for better model is Accuracy. It is the ratio of number of correct predictions to the total number of predictions made.

Accuracy= number of correct predictions / Total predictions made

It can also be calculated from MAE as

Accuracy = 1- MAPE

R Squared: - It represent proportion of variance (of y) that has been explained by the independent variable in the model. It provides an indication of fit model and therefore a measure of how well unseen sample are likely to be predicted by the model through the proportion of the explained variable.

2.2.2 Linear Regression Model

It is used to predict the value of variable *Y* based on one or more input predictor variables *X*. The goal of this method is to establish a linear relationship between the predictor variables and the response variable. Such that, we can use this formula to estimate the value of the response *Y*, when only the predictors (*X-Values*) are known. In this project Linear Regression is applied in both R and Python, details are described following.

Code in python: -

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    cnt   R-squared:                       0.833
Model:                            OLS   Adj. R-squared:                  0.827
Method:                 Least Squares   F-statistic:                     140.2
Date:                Fri, 31 Jan 2020   Prob (F-statistic):         1.63e-203
Time:                        12:58:37   Log-Likelihood:                -4716.2
No. Observations:                 584   AIC:                             9474.
Df Residuals:                     563   BIC:                             9566.
Df Model:                          20
Covariance Type:            nonrobust
================================================================================
                   coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------
temp          4807.6605    477.418     10.070      0.000    3869.923    5745.398
hum          -1840.0359    351.762     -5.231      0.000   -2530.963   -1149.109
windspeed    -2692.7145    509.781     -5.282      0.000   -3694.019   -1691.410
season_1      -160.8963    149.431     -1.077      0.282    -454.407     132.615
season_2       735.4147    149.261      4.927      0.000     442.239    1028.591
season_3       756.5640    170.170      4.446      0.000     422.319    1090.809
season_4      1424.2811    170.259      8.365      0.000    1089.860    1758.702
yr_0           409.9681    152.821      2.683      0.008     109.799     710.137
yr_1          2345.3954    151.325     15.499      0.000    2048.166    2642.625
mnth_1          -1.9341    197.841     -0.010      0.992    -390.531     386.663
mnth_2          45.1383    186.947      0.241      0.809    -322.060     412.337
mnth_3         510.8770    141.897      3.600      0.000     232.166     789.588
mnth_4         233.3586    174.311      1.339      0.181    -109.021     575.738
mnth_5         659.7195    183.392      3.597      0.000     299.503    1019.936
mnth_6         250.5066    180.098      1.391      0.165    -103.239     604.252
mnth_7        -222.2685    220.988     -1.006      0.315    -656.331     211.794
mnth_8         271.1265    207.045      1.310      0.191    -135.548     677.801
mnth_9         888.8861    173.978      5.109      0.000     547.161    1230.611
mnth_10        382.5832    187.383      2.042      0.042      14.528     750.639
mnth_11       -183.6576    194.752     -0.943      0.346    -566.188     198.873
mnth_12        -78.9721    168.303     -0.469      0.639    -409.550     251.606
weathersit_1  1643.7280     90.978     18.067      0.000    1465.030    1822.426
weathersit_2  1302.9232    110.447     11.797      0.000    1085.985    1519.862
weathersit_3  -191.2876    221.771     -0.863      0.389    -626.886     244.311
==============================================================================
Omnibus:                       97.249   Durbin-Watson:                   1.897
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              248.035
Skew:                          -0.849   Prob(JB):                     1.38e-54
Kurtosis:                       5.704   Cond. No.                     1.37e+16
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The smallest eigenvalue is 6.3e-30. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.
```

Plot 16: - Result of linear Regression Model in Python

Here, F-Statistic explains about the quality of the model. AIC is Akkaine information criterion if we have multiple models with same accuracy then we need to refer this to choose the best model. The table three values containing Omnibus and JB test are mostly required for time variance analysis. Here, as we are not using any time values in our project we can ignore this table 3. T-statistic explain how much statistically significant the coefficient is. It is also used to calculate the P –Value. And if P-Value is less than 0.05 we reject null hypothesis and say that the variable is significant. Here, all the variables are less than 0.05 and are significant.

The R squared and adjusted R squared values show how much variance of the output variable is explained by the independent or input variables. Here the adjusted r square value is 82.7%, which explains that only 83% of the variance of count is explained by the input variables This shows that the model is performing well.

Result of linear regression in R: -

```
lm(formula = cnt ~ ., data = train1)

Residuals:
    Min      1Q  Median      3Q     Max 
-3690.6  -377.7    89.6   483.9  3063.1 

Coefficients: (4 not defined because of singularities)
             Estimate Std. Error t value Pr(>|t|)    
(Intercept)   3191.54     418.87   7.619 1.09e-13 ***
season1      -1672.57     199.20  -8.396 3.75e-16 ***
season2       -823.16     239.88  -3.432 0.000644 ***
season3       -920.26     223.59  -4.116 4.43e-05 ***
season4            NA         NA      NA       NA    
yr0          -2017.20      66.71 -30.236  < 2e-16 ***
yr1                NA         NA      NA       NA    
mnth1          263.32     203.95   1.291 0.197200    
mnth2          278.46     202.34   1.376 0.169310    
mnth3          821.60     205.51   3.998 7.24e-05 ***
mnth4          790.01     275.17   2.871 0.004246 ** 
mnth5         1061.10     294.90   3.598 0.000349 ***
mnth6         1009.55     300.93   3.355 0.000848 ***
mnth7          501.88     323.14   1.553 0.120951    
mnth8          969.69     306.42   3.165 0.001637 ** 
mnth9         1495.47     254.95   5.866 7.63e-09 ***
mnth10         773.08     186.40   4.147 3.88e-05 ***
mnth11         -48.84     174.88  -0.279 0.780117    
mnth12             NA         NA      NA       NA    
weathersit1   2042.10     231.94   8.805  < 2e-16 ***
weathersit2   1606.45     213.88   7.511 2.32e-13 ***
weathersit3        NA         NA      NA       NA    
temp          3906.00     474.15   8.238 1.23e-15 ***
hum          -1185.02     344.17  -3.443 0.000618 ***
windspeed    -2590.37     497.88  -5.203 2.75e-07 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’

Residual standard error: 781.7 on 563 degrees of freedom
Multiple R-squared:  0.8392,	Adjusted R-squared:  0.8335
```

Plot 17: -Result of linear Regression Model

The above plot shows how the target variable count varies with change in each individual variable. The P Value shows which values are significant in predicting the target variable. Here, we reject null hypothesis which is less than 0.05 and declare that the variable is significant for the model. F-Statistic explains about the quality of the model and describes the relationship among predictor and target variables. The R squared and adjusted R squared values shows how much variance of the output variable is explained by the independent or input variables. Here the adjusted r square value is 83.35%, which indicated that 83.92%

of the variance of count is explained by the input variables. This explains the model well enough. After this prediction are done and error metrics are calculated.

MAPE=18.800696038206937

Accuracy =81.19930396179306

R square =0.8436040019904946

2.2.3 Decision Tree

Decision Tree is a supervised learning predictive model that uses a set of binary rules to calculate the target value/dependent variable.

Code in python

```
DecisionTreeRegressor(criterion='mse', max_depth=2, max_features=None,
                      max_leaf_nodes=None, min_impurity_decrease=0.0,
                      min_impurity_split=None, min_samples_leaf=1,
                      min_samples_split=2, min_weight_fraction_leaf=0.0,
                      presort=False, random_state=None, splitter='best')
```

Plot18: - Decision tree fit in Python

The above fit plot shows the criteria that is used in developing the decision tree in Python. To develop the model in python, during modeling I have kept all the attributes at default, except the depth as 2. Although these attributes can be played around to derive better score of the model, which is called Hyper tuning of the model.

Decision Tree in R code

```
> DTModel
n= 584

node), split, n, deviance, yval
      * denotes terminal node

 1) root 584 2140008000.0 4535.288
   2) temp< 0.432373 240  527376400.0 3102.171
     4) yr1< 0.5 124  129321300.0 2248.524
       8) season4< 0.5 85   28532480.0 1737.753 *
       9) season4>=0.5 39   30282360.0 3361.744 *
     5) yr1>=0.5 116  211102600.0 4014.690
      10) temp< 0.2804165 32   21386190.0 2550.188 *
      11) temp>=0.2804165 84   94938170.0 4572.595
        22) season1>=0.5 35   20882460.0 3798.600 *
        23) season1< 0.5 49   38111590.0 5125.449 *
   3) temp>=0.432373 344  775817500.0 5535.137
     6) yr1< 0.5 165  111388900.0 4342.473
      12) weathersit3>=0.5 5     496603.2 2277.600 *
      13) weathersit3< 0.5 160  88907630.0 4407.000 *
     7) yr1>=0.5 179  213377700.0 6634.520
      14) hum>=0.771458 22   52841300.0 5267.318 *
      15) hum< 0.771458 157  113650600.0 6826.102 *
```

Plot19: -Decision Tree fit in R

The above plot shows the rules of splitting of trees. The main root splits into 2 nodes having temp <0.432373 240 and temp >=0.432373 344 as its conditions. Nodes further split, the line with * shows that it is the terminal node. These rules are then applied on the test data to predict values. After this the error rate, R Square and accuracy of the model is noted.

MAPE=36.94809301452646

Accuracy =63.05190698547354

Square =0.8436040019904946

2.2.4 Random Forrest

It is a process where the machine follows an ensemble learning method for classification and regression that operates by developing a number of decision trees at training time and giving output as the class that is the mode of the classes of all the individual decision trees.

Random Forrest Regressor code in Python

```
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=1, 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)
```

Plot19: -Random forest fitting Python

Like the Decision tree above are all the criteria values that are used to develop the Random Forest model in python. Everything is kept default only except n_estimators, which is tree numbers. Although this attribute can be altered to get a model with a better score. After this the error rate.

Random Forrest Regressor code in R

```
> RFModel

Call:
 randomForest(formula = cnt ~ ., data = train1, ntree = 500, importance = TRUE)
               Type of random forest: regression
                     Number of trees: 500
No. of variables tried at each split: 8

          Mean of squared residuals: 460286.8
```

Plot20: - Random Forrest Model in R

Here in this plot, random forest created 500 trees with 8 variables randomly check at each node which giving total variance of 87.44% explained.

```
> importance(RFModel, type = 1)
              %IncMSE
season1     27.3434751
season2     10.9457140
season3      9.9969168
season4     18.4304211
yr0         21.5938157
yr1         30.7886652
mnth1       10.2728694
mnth2       10.5726833
mnth3       12.7265449
mnth4       12.9659555
mnth5        5.9450748
mnth6        8.9854222
mnth7       -0.2186141
mnth8        2.6613930
mnth9       13.2157670
mnth10       3.0995267
mnth11       6.9810733
mnth12       9.9717654
weathersit1 11.4091623
weathersit2  9.5038849
weathersit3 15.4185748
temp        54.5577011
hum         27.2861873
windspeed   16.2951523
```

Plot21: -Importance factor contributing in to find best fit of Random forest

The above RF Model describes about the variable contributing most for predicting the target Variable. Few instances are like Temperature, humidity, season1 and year0 contributes most developing the model. After this the error rate, R Square and

accuracy of the model is noted.

MAPE=20.971461848606978

Accuracy =79.02853815139302

Square =0.654460687337333

2.2.5 XGBoost Regressor

XGBoost is an implementation of gradient boosted decision trees designed for speed and performance.

Code in Python: -

```
XGBRegressor(alpha=10, base_score=0.5, booster='gbtree', colsample_bylevel=1,
             colsample_bynode=1, colsample_bytree=0.3, gamma=0,
             importance_type='gain', learning_rate=0.1, max_delta_step=0,
             max_depth=5, min_child_weight=1, missing=None, n_estimators=10,
             n_jobs=1, nthread=None, objective='reg:squarederror',
             random_state=0, reg_alpha=0, reg_lambda=1, scale_pos_weight=1,
             seed=None, silent=None, subsample=1, verbosity=1)
```

Plot22: -XgbRegressor Fit

Here in this pot, the xgbregressor use gbtree for bossting method, colsample_bylevel ,colsample_bynode and colsample_bytree parameters have a range of (0, 1], the default value of 1, and specify the fraction of columns to be subsampled. Min_child_wight in this linear regression task, this simply corresponds to minimum number of instances needed to be in each node. This will check by using mean square error.by using all this parameter it will boost the train value to predict test value correctly.

Code in R: -

```
> xgboost_model
##### xgb.Booster
raw: 47.6 Kb
call:
  xgb.train(params = params, data = dtrain, nrounds = nrounds,
    watchlist = watchlist, verbose = verbose, print_every_n = print_every_n,
    early_stopping_rounds = early_stopping_rounds, maximize = maximize,
    save_period = save_period, save_name = save_name, xgb_model = xgb_model,
    callbacks = callbacks)
params (as set within xgb.train):
  silent = "1"
xgb.attributes:
  niter
callbacks:
  cb.evaluation.log()
# of features: 24
niter: 15
nfeatures : 24
evaluation_log:
    iter train_rmse
       1  3502.4768
       2  2515.1365
---
      14   264.9751
      15   253.1613
```

Plot23: -Xgboost Model in R

In this plot, by using model of xgboost model have boosted 24 features with root mean square matrix to get accurate value of prediction.

MAPE=18.494600043907855

Accuracy =81.50539995609215

Rsquare =0.8436040019904946

3 Chapter: Evaluation of The Model

So, now we have developed few models for predicting the target variable, now the next step is evaluating the models and identify which one to choose for deployment. To decide these, error metrics are used. In this project MAPE, R Square and Accuracy are used. And addition to these error metrics K Fold Cross validation is also applied to identify the best Fit model of all.

3.1.1 Evolution of Model with Error Matrix Value

MAPE of model: -

Value in R

Sr.no	Model Name	MAPE value in %
1	Linear Regression	21.57545
2	Decision Tree	26.4225
3	Random forest	19.38623
4	XGboost	18.27699

Table: - MAPE value with respective model in R

In Python

Sr.no	Model Name	MAPE value in %
1	Linear Regression	18.8006960
2	Decision Tree	36.9480930
3	Random forest	20.9714618
4	XGboost	18.494600

Table: - MAPE value with respective model in Python

If we observe the above tables, we choose the model with lowest MAPE as a suitable Model. Here, from R we get XGboost as a better model, whereas from Python we get Linear Regression and XGboost as a better model. So, following this we can conclude that Both XGboost and Linear Regression can be used as model for this data, if you evaluate on the basis of MAPE. But we need more error metrics to cross check this. So, we go for R Square which is a better error metric. Before that we r going through accuracy of each mode

3.1.1.1 Accuracy of model

Value in R

Sr.no	Model Name	Accuracy in %
1	Linear Regression	78.4320
2	Decision Tree	73.5775
3	Random forest	80.5348
4	XGboost	81.5655

Table: - Accuracy value with respective model in R

In Python

Sr.no	Model Name	Accuracy in %
1	Linear Regression	81.199303
2	Decision Tree	63.051906
3	Random forest	79.028538
4	XGboost	81.505399

Table: - Accuracy value with respective model in Python

As, Accuracy derives from MAE/MAPE its observations also suggest same models as better models as suggested by MAPE. Here, the models with highest accuracy are chosen, and from the observations it is found that both XGboost and Linear Regression are good models for the given data set.

3.1.1.2 R Square of Model

R Square is another metric that helps us to know about the Correlation between original and predicted values.

Value in R

Sr.no	Model Name	R square in %
1	Linear Regression	81.911
2	Decision Tree	76.121
3	Random forest	86.783
4	XGboost	86.025

Table: - R square value with respective model in R

In Python

Sr.no	Model Name	R square in %
1	Linear Regression	84.360
2	Decision Tree	65.446
3	Random forest	65.446
4	XGboost	84.360

Table: - R square value with respective model in Python

R Square is identified as a better error metric to evaluate models. If we observe the above tables, we choose the model with highest R Square as a suitable Model. Here, in R XGboost and Random forest has highest score, whereas in Python it is found that XGboost and Linear Regression has highest equal score.

So, by concluding all the error matrix analysis we conclude here that this model has XGboost and Linear as the best fit model.

3.1.2 Cross Validation

Cross-validation is a resampling procedure used to evaluate machine learning models on a limited data sample. Although we have followed above error metrics to identify a better model, there is always a chance that model is under fitting or over fitting the data. So, the problem with this evaluation technique is that it does not give an indication of how well the learner will generalize to an independent/ unseen data set. Getting this idea about our model is known as Cross Validation. So, it becomes important to cross validate our model in most cases. Cross – Validation are of different types. In this project K-Fold cross validation is used.

K-Fold Cross – Validation:

The procedure has a single parameter called k, that refers to the number of groups that a given data sample is to be split into. As such, the procedure is often called k-fold cross-validation. When a specific value for k is chosen, it may be used in place of k in the reference to the model, such as k=10 becoming 10-fold cross-validation. Basically, it distributes the data in various folds and averages the accuracy score of various folds to identify the best model. The model with highest cross validated average score of accuracy is termed as best model for the data.

In R:

Random Forest

5 folds are created and little hypertuning is done with mtry = 2,3,4 and the following observations are found, it says RF Model with 4 splits is good with R-Square of after predicting value for test 87.23 %

```
> print(RF_KF)
Random Forest

584 samples
 24 predictor

No pre-processing
Resampling: Cross-Validated (5 fold)
Summary of sample sizes: 468, 468, 466, 468, 466
Resampling results across tuning parameters:

  mtry  RMSE      Rsquared   MAE
  2     889.4056  0.8501722  694.4940
  3     765.1775  0.8633487  573.2052
  4     716.4231  0.8704569  524.2358

RMSE was used to select the optimal model using the smallest value.
The final value used for the model was mtry = 4.
```

Decision Tree

5 folds are created and little hyper tuning of interaction depth = 1,2,3, and n. trees = 200, and the following observations are found, it says DT Model with interaction depth with 3 and 200 n. trees the model performs better as R-Square after predicting value for test is 86.93%.

```
> print(DT_KF)
Stochastic Gradient Boosting

584 samples
 24 predictor

No pre-processing
Resampling: Cross-Validated (5 fold)
Summary of sample sizes: 468, 468, 467, 466, 467
Resampling results across tuning parameters:

  interaction.depth  RMSE      Rsquared   MAE
  1                  717.3242  0.8597332  535.0284
  2                  700.9235  0.8657042  519.9110
  3                  677.5933  0.8744386  502.8123

Tuning parameter 'n.trees' was held constant at a value of 200
Tuning parameter 'shrinkage'
 was held constant at a value of 0.1
Tuning parameter 'n.minobsinnode' was held constant at a
 value of 10
RMSE was used to select the optimal model using the smallest value.
The final values used for the model were n.trees = 200, interaction.depth = 3, shrinkage = 0.1
 and n.minobsinnode = 10.
```

Linear Regression

5 folds are created and the following observations are found for Linear Regression Cross Validation, it says LR Model performs well with as R-Square is after predicting value for test 81.19 %.

```
> print(LR_KF)
Linear Regression

584 samples
 24 predictor

No pre-processing
Resampling: Cross-Validated (5 fold)
Summary of sample sizes: 466, 468, 467, 468, 467
Resampling results:

  RMSE      Rsquared   MAE
  806.9495  0.8223286  600.506

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

XGBoost: - 5 folds are created and the following observations are found for XGBoost Regression Cross Validation, it says XGModel performs well with as R-Square is after predicting value for test 87.38%.

```
> print(XG_KF)
eXtreme Gradient Boosting

584 samples
 24 predictor

No pre-processing
Resampling: Cross-Validated (5 fold)
Summary of sample sizes: 468, 467, 468, 466, 467
Resampling results across tuning parameters:

  max_depth  colsample_bytree  RMSE      Rsquared   MAE
  3          0.5               676.0603  0.8772100  493.9459
  3          0.7               679.6241  0.8749255  484.9032
  6          0.5               682.7930  0.8741306  501.8300
  6          0.7               672.2735  0.8784990  490.4461

Tuning parameter 'nrounds' was held constant at a value of 100
Tuning parameter 'eta' was

Tuning parameter 'min_child_weight' was held constant at a value of 2
Tuning parameter
 'subsample' was held constant at a value of 1
RMSE was used to select the optimal model using the smallest value.
The final values used for the model were nrounds = 100, max_depth = 6, eta = 0.1, gamma =
 1, colsample_bytree = 0.7, min_child_weight = 2 and subsample = 1.
```

In Python:

Here in python the cross_val_score function is imported from scikit learn library, which performs K Fold Cross Validation in various models. The details are noted below.

Linear Regression:

3 Folds are created with no tuning, and 3 folds scores are found and the average accuracy score of the model is found as 62.63 %. Thus, the model is up to mark. it can also be tuned further to get better accuracy.

```
In [83]: #for linear regression model
         from sklearn.linear_model import LinearRegression
         lr = LinearRegression()

         lr_score=cross_val_score(lr,kv_x,kv_y, cv= 3)
```

```
In [84]: lr_score.mean()
Out[84]: 0.626364965698319
```

Decision Tree:

3 Folds are created with mac_depth = 2, and 3 folds scores are found and the average accuracy score of the model is found as 5.24 %. Thus, the model is not up to mark it can be tuned further, and if tuning also doesn't improve the accuracy of the model, we will drop this model.

```
In [80]: #for decision tree
         dt = DecisionTreeRegressor(max_depth=2)

         dt_score=cross_val_score(dt,kv_x,kv_y, cv= 3)
         dt_score.mean()
Out[80]: 0.05247379896663843
```

Random Forest:

3 Folds are created with n_estimators = 100, and 3 folds scores are found and the average accuracy score of the model is found as 51.23 %. Thus, the model is not up to mark it can be tuned further, and if tuning also doesn't improve the accuracy of the model, we will drop this model.

```
In [79]: #for Random forrest
         rf = RandomForestRegressor()

         rf_score=cross_val_score(rf,kv_x,kv_y, cv= 3)
         print(rf_score.mean())
```

```
0.512350842138848
```

XGboost:

3 Folds are created with the average accuracy score of the model is found as 55.95 %. Thus, the model is not up to mark it can be tuned further, and if tuning also doesn't improve the accuracy of the model, we will drop this model.

```
[78]: #for Random forrest
      xg =xgb.XGBRegressor()

      xg_score=cross_val_score(xg,kv_x,kv_y, cv=3)
      print(xg_score.mean())
```

```
0.5595401339182667
```

From the above cross-validation it is found that, in some cases XGboost is a better model and in some other cases Linear Regression is a better model for the given data set. We can go with any one of them or both. Thus, this model can be used for further processes and this model can also be further tuned to get optimum results.

And also, from all the criteria mentioned above, like MAPE, R Square, Accuracy and Cross- Validation, it is concluded that both the models Linear Regression and XGboost is are better for our given data set.

Appendix A:-R code

```
#................................project- Bike rental count
.............................................rm(list=ls(all=T))

# setting up working directory for project cab prediction

setwd("R:/vishakha r progaram/projects")

getwd()

#loading required libraries

x = c("ggplot2", "corrgram", "DMwR", "caret", "randomForest", "unbalanced", "C50",
"dummies", "e1071", "Information",

     "MASS", "rpart", "gbm", "ROSE", 'sampling', 'DataCombine', 'inTrees')

#installing packages(x)

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

rm(x)

#...............................Loading data....................................................#

#importing data from directory

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

View(Data_d)


#...........................Explorin Data Analysis............................................#

str(Data_d)

class(Data_d)

names(Data_d)

summary(Data_d)


#remove some unwanted variable

# isntant = giving information of index so not important variable

# dteday = having date tym there is no reqire of this variable

# casual, registered = by combining this two variable we getiing cnt
```

```
Data_d = subset(Data_d, select= -c(instant,dteday,registered,casual))
str(Data_d)
names(Data_d)
head(Data_d)

#checking for nunique value of each variable
apply(Data_d, 2,function(x) length(table(x)))


#grouping the categorical nad numerical variable
cat_var = c('season','yr','mnth','holiday','weekday','workingday','weathersit')
num_var = c('temp','atemp','hum','windspeed','cnt')



Data1 = Data_d
#................................Data Pre Processing .......................................#

# 1.Missing value analysis
# checking missing value of data

apply(Data_d,2,function(x){sum(is.na(x))})

# there is no  missing value

#2. Outliyer Analysis
#visualization of outlier with plot
```

```
for (i in 1:length(num_var))
{
  assign(paste0("gn",i), ggplot(aes_string(y = (num_var[i]), x = "cnt"), data =
subset(Data_d))+
        stat_boxplot(geom = "errorbar", width = 0.5) +
        geom_boxplot(outlier.colour="red", fill = "blue" ,outlier.shape=18,
                     outlier.size=1, notch=FALSE) +
        theme(legend.position="bottom")+
        labs(y=num_var[i],x="count")+
        ggtitle(paste("Box plot of count for",num_var[i])))
}

# Plotting plots together
gridExtra::grid.arrange(gn1,gn2,ncol=2)
gridExtra::grid.arrange(gn3,gn4,ncol=2)

# Here we have found that hum and windspeed has some outlier
# outlier treatment
# convert outlier into NA
for(i in num_var){
  val_outlier = Data_d[,i][Data_d[,i] %in% boxplot.stats(Data_d[,i])$out]
  print(length(val_outlier))
  Data_d[,i][Data_d[,i] %in% val_outlier] = NA
}

#imputing outlier with help of knn method
Data_d = knnImputation(Data_d, k = 5)
sum(is.na(Data_d))
```

```
#.................................Data Understanding with vizualisation............................#

# method which will plot barplot of a columns with respect to other column

#ploting graph season vs cnt

ggplot(Data_d, aes(x = Data_d$season, y = Data_d$cnt))+
  geom_bar(stat = "identity", fill = "blue")+
  labs(title = "Number of bikes rented with respect to season", x = "Seasons", y = "cnt")+
  theme(panel.background = element_rect("white"))+
  theme(plot.title = element_text(face = "bold"))

#here found that season 3, has the highest count of bikes and season 1 has lowest count of bikes


#ploting graphs yr vs cnt

ggplot(Data_d, aes(x = Data_d$yr, y = Data_d$cnt))+
  geom_bar(stat = "identity", fill = "grey")+
  labs(title = "Number of bikes rented with respect to yr", x = "yr", y = "cnt")+
  theme(panel.background = element_rect("white"))+
  theme(plot.title = element_text(face = "bold"))

#plot conlclude that 1=2011 has highestcount of renting bike than 0 = 2010


#ploting graphs month vs cnt

ggplot(Data_d, aes(x = Data_d$mnth, y = Data_d$cnt))+
  geom_bar(stat = "identity", fill = "pink")+
  labs(title = "Number of bikes rented with respect to mnth", x = "mnth", y = "cnt")+
  theme(panel.background = element_rect("white"))+
  theme(plot.title = element_text(face = "bold"))

#plot conclude that month 8 and 9 has highest count of renting bike and month 1 has lowest count of bike renting
```

#ploting of graph weekday vs cnt

ggplot(Data_d, aes(x = weekday, y = cnt))+

geom_bar(stat = "identity", fill = "green")+

labs(title = "Number of bikes rented with respect to weekday", x = "weekday", y = "cnt")+

theme(panel.background = element_rect("white"))+

theme(plot.title = element_text(face = "bold"))

#plot conclude that bike count is highest in the week day of 4th and 5th of week day

#Count with respect to temperature and humidity together

ggplot(Data_d,aes(temp,cnt)) +

geom_point(aes(color=hum),alpha=0.5) +

labs(title = "Bikes count vs temperature and humidity", x = "Normalized temperature", y = "Count")+

scale_color_gradientn(colors=c('blue','light blue','dark blue','light green','yellow','dark orange','black')) +

theme_bw()

#it is found that when normalized temperature is between 0.5 to 0.75 and humidity is between 0.4 to 0.8, count is high

Count with respect to windspeed and weather together

ggplot(Data_d, aes(x = windspeed, y = cnt))+

geom_point(aes(color= weathersit), alpha=0.5) +

labs(title = "Bikes count vs windspeed and weather", x = "Windspeed", y = "Count")+

scale_color_gradientn(colors=c('blue','light blue','dark blue','light green','yellow','dark orange','black')) +

```
  theme_bw()
```

It is found that count is at peak, when windspeed is from 0.1 to 0.3 and weather is from 1.0 to 1.5.

Count with respect to temperature and season together

```
ggplot(Data_d, aes(x = temp, y = cnt))+
  geom_point(aes(color=season),alpha=0.5) +
  labs(title = "Bikes count vs temperature and season", x = "Normalized temperature", y = "Count")+
  scale_color_gradientn(colors=c('blue','light blue','dark blue','light green','yellow','dark orange','black')) +
  theme_bw()
```

it is found that count is maximum when temperature is 0.50 to 0.75 & season 2 to season 4

#.....................................Feature Selection.......................................#

feature selection by the checking

#correlation among the variable in the case of numerical variable

#Correlation Plot

```
corrgram(Data_d[,num_var],order=FALSE,upper.panel = panel.pie,
        text.panel = panel.txt,
        main= "Correlation Analysis between numeric variables")
```

#it is found that temperature and atemp are highly correlated with each other.

```
# in given condition our target variable is continous
#for checking correlation, using anova test
for(i in cat_var){
  print(i)
  Anova_test_result = summary(aov(formula = cnt~Data_d[,i],Data_d))
  print(Anova_test_result)
}


#it is found that holiday, weekday and workingday has p value > 0.05.we go for null hypothesis.
#by the checking the hypothesisi we have found and selecting significant varialbe
#so we droping unwanted variable

Data_d = subset(Data_d,select = -c(atemp,workingday,weekday,holiday))

#so new variable now
cat_var = c('season','yr','mnth','weathersit')
num_var = c('temp','hum','windspeed','cnt')


#checking  multicollinearity

variable_n = Data_d[,num_var]

#importing reqired library
library(usdm)
```

```
vifcor(variable_n, th = 0.7)

#No variable from the 4 input variables has collinearity problem

#........................................feature Scaling...................................#

#checking for normality visualisation

hist(Data_d$temp, col="Navyblue", xlab="Temperature", ylab="Frequency",
     main="Temperature Distribution")

hist(Data_d$hum, col="Yellow", xlab="Humidity", ylab="Frequency",
     main="Humidity Distribution")

hist(Data_d$windspeed,col="Dark green",xlab="Windspeed",ylab="Frequency",
     main="Windspeed Distribution")

#all histogram showing that data is symmetric in nature

# Identify range and check min max of the variables to check noramility

for(i in num_var){
  print(summary(Data_d[,i]))
}
#data is found as normalized, no need to scaling
Data2 = Data_d

#......................................Model development...................................#
```

```
#collectint required data to perform model development

rmExcept("Data_d")

#creating some reqired dummies variable  for categorical variable
cat_var = c("season","yr","mnth","weathersit")

library(dummies)
Data_d = dummy.data.frame(Data_d, cat_var)



#mape error
MAPE = function(y,y1){
  mean(abs((y-y1)/y))*100
}

#R Square

Rsquare = function(x,x1){
  cor(x,x1)^2
}

#saving the data for cross validation

cv_data = Data_d

#so spliting the data into trian and test subset for modeling
set.seed(123)
```

```
train_index = sample(1:nrow(Data_d),0.8*nrow(Data_d))
train1= Data_d[train_index,]
test1= Data_d[-train_index,]



#...........................Decision tree................................................#

#deploying decison tree model
DTModel = rpart(cnt~., train1, method = "anova" , minsplit=5)

summary(DTModel)

# Predictions decesion tree

DTTest = predict(DTModel, test1[-25])

summary(DTModel)

#mape
DTMape_Test = MAPE(test1[,25], DTTest)

DTMape_Test #26.4225

#RSquare

DT_RSquare = Rsquare(test1[,25], DTTest)
DT_RSquare  #0.7612102
```

```
#Accuracy

Accuracy_DTModel=(100-DTMape_Test)

Accuracy_DTModel#73.5775

#............................linear Regression............................................#


# deploying linear regression model

lm_model = lm(cnt ~. , data = train1)



summary(lm_model)


#prediction of lm_mode

LMTest= predict(lm_model, test1[,-25])



#mape

LRMape_Test = MAPE(test1[,25], LMTest)

LRMape_Test

#21.57545


#RSquare

LR_RSquare = Rsquare(test1[,25],LMTest)

LR_RSquare

#0.8191175


#Accuracy

Accuracy_lmModel=(100-LRMape_Test)
```

```
Accuracy_lmModel#78.43208


#...............................Random Forrest.............................................#
#Deploying random forrest
RFModel = randomForest(cnt~., train1, ntree = 500, importance = TRUE)

summary(RFModel)

importance(RFModel, type = 1)

# Predictions of random forrest

RFTest = predict(RFModel, test1[-25])



# MAPE

RFMape_Test = MAPE(test1[,25], RFTest)
RFMape_Test
# 19.38623

#RSquare

RF_RSquare = Rsquare(test1[,25], RFTest)
RF_RSquare
#0.8678384

#Accuracy
```

```
Accuracy_Rfmodel =(100-RFMape_Test)

Accuracy_Rfmodel#80.53485




#............................XGBOOST algorithm..............................................#

library(xgboost)

train_data_matrix = as.matrix(sapply(train1[-25],as.numeric))

test_data_data_matrix = as.matrix(sapply(test1[-25],as.numeric))

xgboost_model = xgboost(data = train_data_matrix,label = train1$cnt,nrounds = 15,verbose
= FALSE)

summary(xgboost_model)

xgb_predictions = predict(xgboost_model,test_data_data_matrix)

xgMape_Test = MAPE(test1[,25], xgb_predictions)

xgMape_Test#18.27699

xg_RSquare = Rsquare(test1[,25], xgb_predictions)

xg_RSquare#0.859154



#Accuracy

Accuracy_xgbmodel =(100-xgMape_Test)

Accuracy_xgbmodel#81.72301
```

#...................................Cross Validation process....................................#

```
#Load Data
library(caret)


cv_data

#divide data

set.seed(123)
train_index2 = sample(1:nrow(cv_data),0.8*nrow(cv_data))
train_KF = cv_data[train_index,]
test_KF = cv_data[-train_index,]

#Random Forest Cross Validation

RF_KF = caret::train(cnt~.,
            data = train_KF,
            method = "rf",
            tuneGrid = expand.grid(mtry = c(2,3,4)),
            trControl = trainControl(method = "cv",
                        number = 5,
                        verboseIter = FALSE,))


print(RF_KF)
```

```
knitr::kable(head(RF_KF$results), digits = 3)

print(RF_KF$bestTune)



RFpreds = predict(RF_KF, test_KF[-25])


RFpreds_MAPE = MAPE(test_KF[,25], RFpreds)
RFpreds_MAPE#22.18992


RFPreds_RSquare = Rsquare(test_KF[,25], RFpreds)
RFPreds_RSquare# 0.8694244



#Decision Tree Cross Validation



DT_KF = caret::train(cnt~.,
          data = train_KF,
          method = "gbm",
          tuneGrid = expand.grid(n.trees = 200,
                    interaction.depth = c(1,2,3),
                    shrinkage = 0.1,
                    n.minobsinnode = 10 ),
          trControl = trainControl(method = "cv",
```

```
                        number = 5,
                        verboseIter = FALSE))

print(DT_KF)



knitr::kable(head(DT_KF$results), digits = 3)

print(DT_KF$bestTune)



DTpreds = predict(DT_KF, test_KF[-25])

DTpreds_MAPE = MAPE(test_KF[,25], DTpreds)
DTpreds_MAPE#18.19822

DTPreds_RSquare = Rsquare(test_KF[,25], DTpreds)
DTPreds_RSquare#0.8746362



#Linear Regression CV

LR_KF = caret::train(cnt~.,
       data = train_KF,
       method = "lm",
       tuneGrid = expand.grid(intercept = TRUE),
       trControl = trainControl(method = "cv",
                        number = 5,
```

```
                              verboseIter = FALSE))


print(LR_KF)
knitr::kable(head(LR_KF$results), digits = 3)
print(LR_KF$bestTune)
LRpreds = predict(LR_KF, test_KF[-25])
LRpreds_MAPE = MAPE(test_KF[,25], LRpreds)
LRpreds_MAPE#21.56792
LRPreds_RSquare = Rsquare(test_KF[,25], LRpreds)
LRPreds_RSquare#0.8191175
```

References :-

https://towardsdatascience.com/exploratory-data-analysis-8fc1cb20fd15

https://blog.exploratory.io/a-practical-guide-of-exploratory-data-analysis-with-linear-regression-part-1-9f3a182d7a9

https://www.youtube.com/watch?v=Gj-prU8aEgg

https://xgboost.readthedocs.io/en/latest/parameter.html

https://rdrr.io/cran/lmvar/man/cv.lm.html