PROJECT

Predication of Bike Rental count based On the Environmental and Seasonal settings.

Submitted By:

DIBYA JYOTI MANDAL

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION

1.1 PROBLEM STATEMENT

The project is about a bike rental company who has its historical data, and now our objective of this Project is to predict the bike rental count on daily basis, considering the environmental and seasonal settings. These predicted values will help the business to meet the demand on those particular days by maintain the amount of supply.

Nowadays there are number of bike renting companies like, Ola Bikes, Rapido etc. And these bike renting companies deliver services to lakhs of customers daily. Now it becomes really important to manage their data properly to come up with new business ideas to get best results. In this case we have to identify in which days there can be most demand, such that we have enough strategies met to deal with such demand.

1.2 DATA

The given dataset contains 16 variables and 731 observations. The "cnt" is the target variable and remaining all other variables are the independent variables.

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. A snapshot of the data is mentioned following.

instant	dteday	season	yr	mnth	holiday	weekday	workingda	weathersit	temp	atemp	hum	windspeed	casual	registered	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
8	1/8/2011	1	0	1	0	6	0	2	0.165	0.162254	0.535833	0.266804	68	891	959
9	1/9/2011	1	0	1	0	0	0	1	0.138333	0.116175	0.434167	0.36195	54	768	822
10	########	1	0	1	0	1	1	1	0.150833	0.150888	0.482917	0.223267	41	1280	1321

Table: Data

CHAPTER 2: METHODOLOGY

After going through the dataset in detail and pre-understanding the data the next step is, Methodology that will help achieve our goal.

In Methodology following processes are followed:

- Pre-processing:

  It includes missing value analysis, outlier analysis, feature selection and feature scaling.

- Model development:

  It includes identifying suitable Machine learning Algorithms and applying those algorithms in our given dataset.

2.1 Pre-processing

Here, we will use techniques like missing value analysis, outlier analysis, feature selection, feature scaling. This techniques are 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. As, normally the data we get can be messy i.e.: it can include many missing values, inconsistent values etc. And this things needs to be checked prior developing a model.

2.1.1 Missing Value Analysis

Missing value is availability of incomplete observations in the dataset. This is found because of reasons like, incomplete submission, wrong input, manual error etc. These Missing values affect the accuracy of model. So, it becomes important to check missing values in our given data.

Here, in this project, after checking the data it is found that the data doesn't consist any missing values.

```
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 values found

Plot: Missing Values

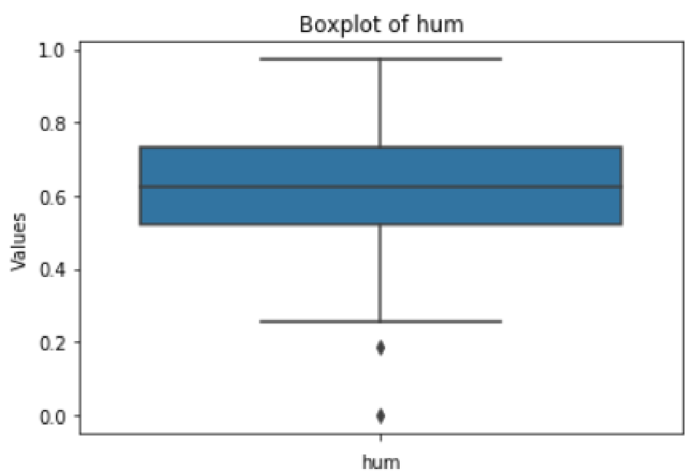
As there is no missing values found in our given data, thus we don't need to follow imputation processes here. So, we can directly move to our next step that is outlier analysis.

2.1.2 Outlier Analysis

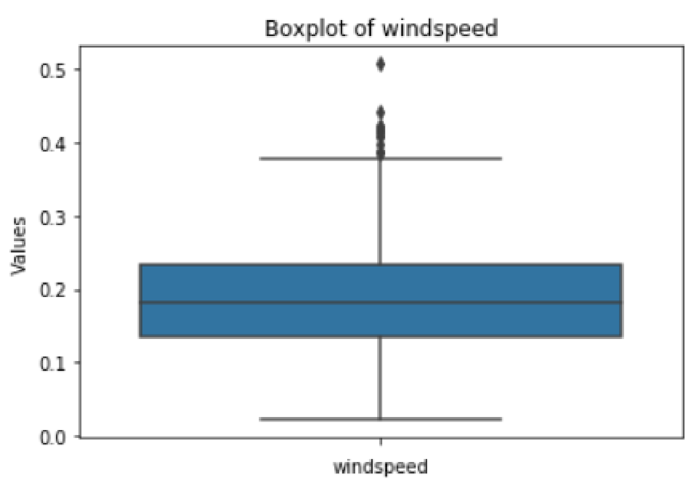
Outlier is an abnormal observation that stands or deviates away from other observations. These happens because of manual error, poor quality of data and it is correct but exceptional data. But, it can cause an error in predicting the target variables. So we have to check for outliers in our data set and also remove or replace the outliers wherever required.

In this project, outliers are found in only two variables this are Humidity and windspeed, following are the box plots for both the variables and dots outside the quartile ranges are outliers.

hum



windspeed



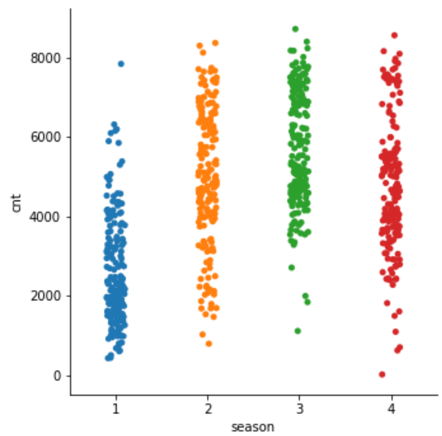
Plot: Outliers

All this outliers mentioned above happened because of manual error, or interchange of data, or may be correct data but exceptional. But 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 Data Understanding

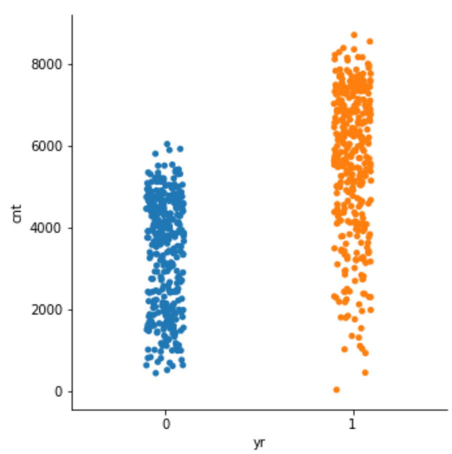
Data Understand is a process where we know our data in a better way by the help of visual representations and come up with initial ideas to develop our model. Here, the specific variables are plotted with respect to the target variable. In some cases two variables are compared, whereas in some cases three variables are plotted together for our better understanding and visualization.

a. Season



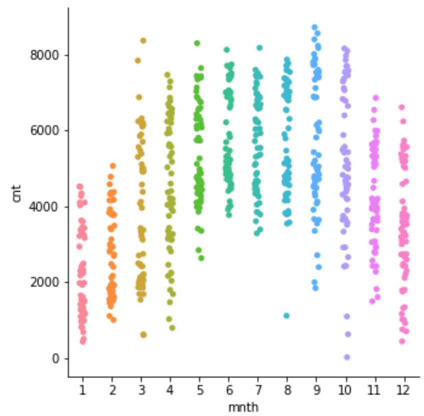
Here, it is found that in Season 2, 3 and 4 has the highest count

b. Year



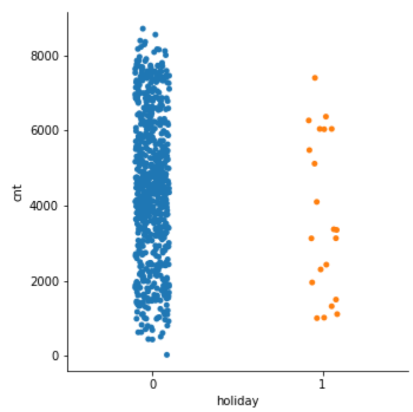
Here, it is found that in Year 1 has high count than 0

c. Month



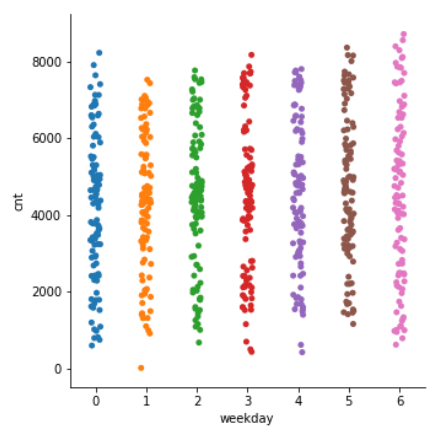
Here, it is observed that in Months 3 to 10 we got a good number of count

d. Holidays and Non-Holidays



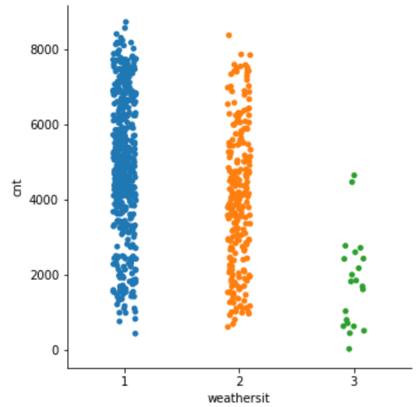
Here, it is found that, on holidays the count is higher when compared non-holidays

e. Weekdays



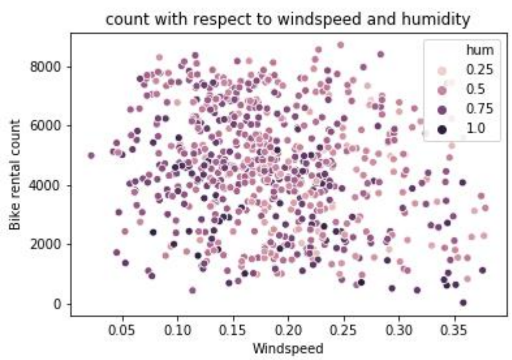
Here, it is observed that in weekdays, 0 and 6 i.e. Monday to Saturday the count is highest.

f. Weather



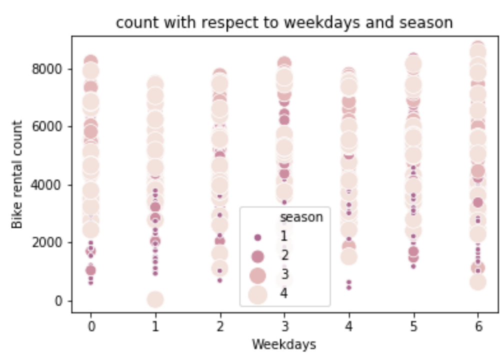
Here, in weather it is observed that, weather 1 has the highest count

g. Windspeed and Humidity vs count



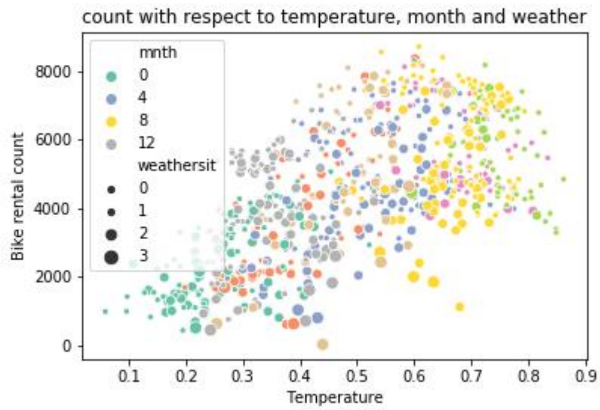
Here, it is found that in count vs windspeed and humidity, Count is High in ranges of windspeed 0.10 to 0.25 and humidity 0.5 to 0.75

h. Weekdays and Season vs count



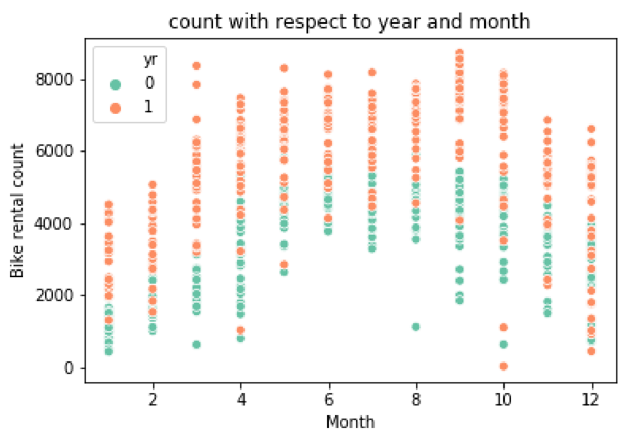
Here, it is observed that in count vs weekdays and season, Count is high in 4th season and 1st and 6th of weekdays

i. Temperature, month and weathers vs count



Here, 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.

j. Year and month vs count



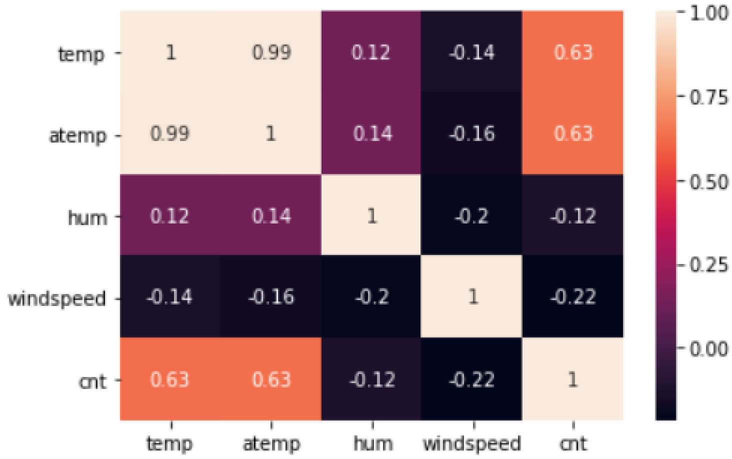
Here, it is found that count vs respect to year and month, count is high in year 1, particularly from season 3 to 12 excluding 9th.

2.1.4 Feature Selection

Sometimes it happens that, all the variables in our data may not be accurate enough to predict the target variable, in such cases we need to analyze our data, understand our data and select the dataset variables that can be most useful for our model. In such cases we follow feature selection. 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 variables can hamper the accuracy of the model.

a. Correlation Analysis for Numerical Variables.



Plot: Correlation Analysis

Observing here, it is found that temperature and atemp are highly correlated with each other. So, in further processes we can drop atemp as it is similar to temperature.

b. ANOVA Test for Categorical Variables

```
               sum_sq     df           F        PR(>F)
season    4.517974e+08    1.0  143.967653  2.133997e-30
Residual  2.287738e+09  729.0         NaN           NaN
               sum_sq     df           F        PR(>F)
yr        8.798289e+08    1.0  344.890586  2.483540e-63
Residual  1.859706e+09  729.0         NaN           NaN
               sum_sq     df          F        PR(>F)
mnth      2.147445e+08    1.0  62.004625  1.243112e-14
Residual  2.524791e+09  729.0        NaN           NaN
               sum_sq     df         F    PR(>F)
holiday   1.279749e+07    1.0  3.421441  0.064759
Residual  2.726738e+09  729.0       NaN       NaN
               sum_sq     df         F    PR(>F)
weekday   1.246109e+07    1.0  3.331091  0.068391
Residual  2.727074e+09  729.0       NaN       NaN
                 sum_sq     df         F    PR(>F)
workingday  1.024604e+07    1.0  2.736742  0.098495
Residual    2.729289e+09  729.0       NaN       NaN
                 sum_sq     df          F        PR(>F)
weathersit  2.422888e+08    1.0  70.729298  2.150976e-16
Residual    2.497247e+09  729.0        NaN           NaN
```

Plot: ANOVA Test

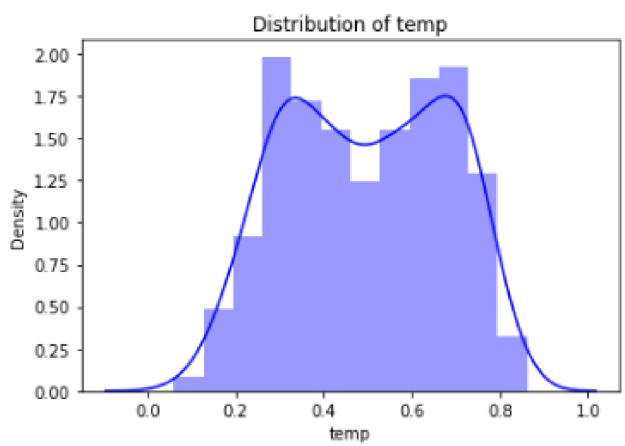
From the observations, it is found that the variables holiday, weekday, and working day has p value > 0.05. Here, null hypothesis is accepted. I.e. this variables has no dependency over target variable. So, in further processes this variables can be dropped before modeling. And this process of deducting the variables is also called as dimension reduction.

2.1.5 Feature Scaling

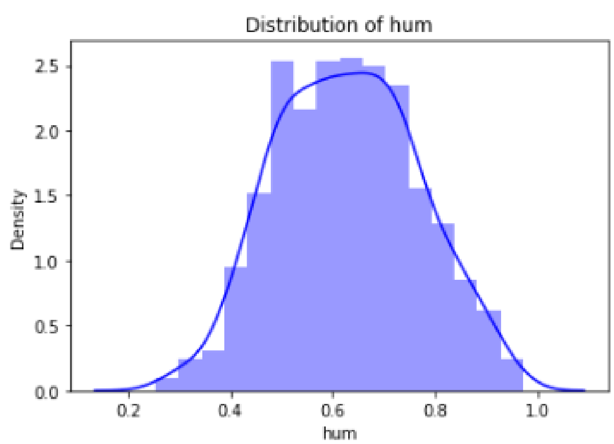
Here, In Feature Scaling ranges of variables are normalized or standardized, such that variables can be compared with same range. This is done for an unbiased and accurate model.

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.

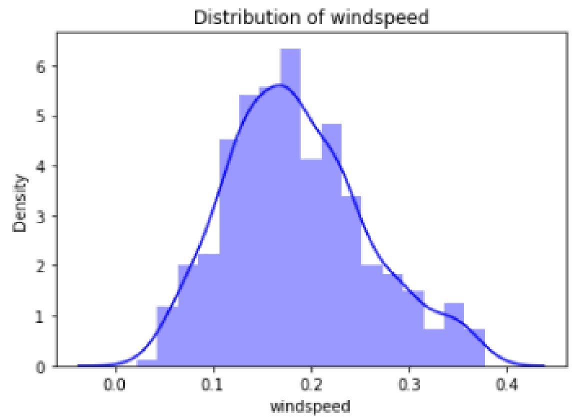
a. Categorical Variables Distribution plot



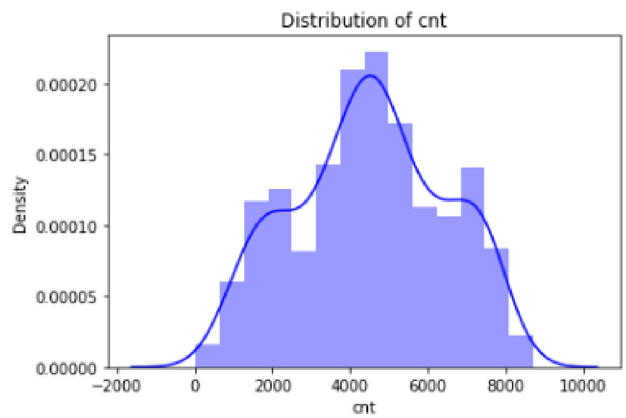
hum



windspeed



cnt



Plot: Distribution of Categorical Variables

b. For Numerical Variables Range check

	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

everything is normalized, no need of scaling

Table: Distribution of Numerical Variables

2.2 Model Development

The next step after Exploratory Data Analysis and Data Pre-Processing is Model Development. Now we have our data ready to be implemented to develop a model. There are number of models and Machine learning algorithms that are used to develop model, some are like decision tree, random forest, SVM, KNN, Naïve Bayes, Linear regression, Logistic Regression etc. So, before implementing any model we have to choose precisely our model. So, the first step in Model Development is selection of model.

2.2.1 Model Selection

As per industry standards, there are four categories of models that are derived by classifying problem statement and goal of the project. These categories are:

- 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.

Basis of understanding the criteria and given data's problem statement. In this project Decision Tree, Random Forest and Linear Regression are models selected for Model Development.

2.2.2 Decision Tree

Decision Tree is a supervised learning predictive model that uses a set of binary rules to calculate the target value/dependent variable.
Decision trees are divided into three main parts this are:

- **Root Node** **:** performs the first split
- **Terminal Nodes** **:** that predict the outcome, these are also called leaf nodes
- **Branches** **:** arrows connecting nodes, showing the flow from root to other leaves.

In this project Decision tree is applied in both R and Python, details are described following.

a. Decision Tree in R

The Decision tree Method is used R with the structured data found after Data Preprocessing

```
> 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 *
>
```

Plot: Decision Tree Fit 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, And the MAPE, RSQUARE and Accuracy is noted below.

MAPE = 26.4225
RSQUARE = 0.7612102
ACCURACY = 73.51 %

b. Decision Tree 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')
```

Plot: 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. After this the fit is used to predict in test data and the error rate, R-Square and accuracy is calculated.

MAPE = 36.948
RSQUARE = 0.6544
ACCURACY = 63.05 %

2.2.3 Random Forest

The next model to be followed in this project is Random forest. 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.

In this project Random Forest is applied in both R and Python, details are described following.

a) Random Forest in R

In a RandomForest model the importance contributed by individual variables can be seen using importance function, it is mentioned below.

```
> importance(RFModel)
                %IncMSE
season1      28.0952214
season2       9.9382321
season3       9.4057697
season4      17.4654533
yr0          21.8926669
yr1          29.4499631
mnth1        10.8787924
mnth2        10.7517743
mnth3        13.6432241
mnth4        13.0848454
mnth5         4.6377261
mnth6         7.6869807
mnth7        -0.0972309
mnth8         3.2663988
mnth9        10.2088852
mnth10        3.7535286
mnth11        7.0704458
mnth12        9.0703647
weathersit1 11.1685822
weathersit2 10.9657916
weathersit3 14.9649487
temp         55.7773526
hum          28.5997555
windspeed    17.1264750
```

Plot: importance of variables

The above RF Model describes about the variable contributing most for predicting the target Variable. Few instances are like Temperature, humidity, season and year contributes most developing the model.

After the trained fit is used to predict the test data and error rate, accuracy and R-Square is noted.

MAPE = 19.32104
RSQUARE = 0.8685008
ACCURACY = 80.67 %

b) Random Forest 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=100, n_jobs=None,
           oob_score=False, random_state=None, verbose=0, warm_start=False)
```

Plot: Random Forest in 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 attributes can be altered to get a model with a better score. After this the error rate, R Square and accuracy of the model is noted.

MAPE = 20.4007
RSQUARE = 0.885114
ACCURACY = 79.05%

2.2.3 Linear Regression

The next method in the process is linear regression. 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.

a) Linear regression in R

After running the model the details I got are as follows.

```
> summary(LRModel)

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

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 ' ' 1

Residual standard error: 781.7 on 563 degrees of freedom
Multiple R-squared:  0.8392,	Adjusted R-squared:  0.8335 
F-statistic: 146.9 on 20 and 563 DF,  p-value: < 2.2e-16
```

Plot: Summary 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% of the variance of count is explained by the input variables. This explains the model well enough. After this the error metrics and Accuracy is noted.

MAPE = 21.56792
RSQUARE = 0.8191175
ACCURACY = 78.44 %

b) Linear Regression in Python

After this the model is developed following details are found.

```
                            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:                Thu, 25 Jul 2019   Prob (F-statistic):          1.63e-203
Time:                        21:08:08   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.                     1.54e+16
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The smallest eigenvalue is 5.01e-30. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.
```

Plot: Linear regression 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. After this predictions are done and error metrics are calculated.

MAPE = 18.80069603
RSQUARE = 0.84360400
ACCURACY = 81.19 %

Model Summary:

From the above mentioned various models that can be developed for the given data. At first place, The Data is divided into train and test. Then the models are developed on the train data. After that the model is fit into it to test data to predict the target variable. After predicting the target variable in test data, the actual and predicted values of target variable are compare to get the error and accuracy. And looking over the error and accuracy rates, the best model for the data is identified and it is kept for future usage.

CHAPTER 3: EVALUATION OF THE MODEL

So, now we have developed few models for predicting the target variable, now the next step is evaluate 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 model of all.

3.1 Mean Absolute Error (MAE)

MAE or Mean Absolute Error, it is one of the error measures that is used to calculate the predictive performance of the model. It is the sum of calculated errors. In this project we will apply this measure to our models.

a) In R :

Method	Mape Error(in Percentage)
Decision Tree	26.4225
Random Forest	19.32104
Linear Regression	21.56792

Table: Mape in R

b) In Python :

Method	Mape Error(in Percentage)
Decision Tree	36.9480
Random Forest	20.9466
Linear Regression	18.8006

Table: Mape in Python

If we observe the above tables, we choose the model with lowest MAPE as a suitable Model. Here, from R we get Random Forest as a better model, whereas from Python we get Linear Regression as a better model. So following this we can conclude that Both Random Forest 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.

3.2 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

a. In R

Method	Accuracy (in Percentage)
Decision Tree	73.57
Random Forest	80.67
Linear Regression	78.43

Table: Accuracy in R Models

b. In Python

Method	Accuracy (in Percentage)
Decision Tree	63.051
Random Forest	79.053
Linear Regression	81.199

Table: Accuracy in Python Models

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 Random Forest and Linear Regression are good models for the given data set.

3.3 R Square

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

a. In R

Method	R – Square (in Percentage)
Decision Tree	76.12
Random Forest	86.85
Linear Regression	81.91

Table: R Square in R

b. In Python

Method	R – Square (in Percentage)
Decision Tree	65.44
Random Forest	88.43
Linear Regression	84.36

Table: R Square in R

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, from both R and Python it is found that Random Forest is a best fit model for the given data.

3.4 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:

By the help of caret package in R the cross-validation is done for various model and results are plotted.

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 split is good with R-Square of 86.9 %

```
> print(RF_KF)
Random Forest

584 samples
 24 predictor

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

  mtry  RMSE      Rsquared   MAE
  2     889.5567  0.8480267  692.1735
  3     753.6377  0.8642956  564.7550
  4     708.5888  0.8696301  518.4594

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 is 86.8 %

```
> print(DT_KF)
Stochastic Gradient Boosting

584 samples
 24 predictor

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

  interaction.depth  RMSE      Rsquared   MAE
  1                  728.8031  0.8578157  539.8345
  2                  702.5039  0.8675989  513.4690
  3                  702.3213  0.8680605  511.8224

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 82.6 %

```
> print(LR_KF)
Linear Regression

584 samples
 24 predictor

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

  RMSE      Rsquared   MAE
  803.2391  0.8268951  600.8336

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

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.

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 48.73 %. Thus, the model is not upto mark it can be tuned further, and if tuning also doesn’t improve the accuracy of the model, we will drop this model.

```
cross_val_score(RandomForestRegressor(), X_kf,y_kf, cv = 3)
#array([0.69521348, 0.27999794, 0.452253  ])
RF_Score = cross_val_score(RandomForestRegressor(n_estimators = 100), X_kf,y_kf, cv = 3)
np.average(RF_Score)
```

```
0.4873964218480966
```

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 upto mark it can be tuned further, and if tuning also doesn’t improve the accuracy of the model, we will drop this model.

```
cross_val_score(DecisionTreeRegressor(max_depth=2), X_kf,y_kf, cv = 3)
#array([ 0.23365401, -0.23313404,  0.15690143])

DT_Score = cross_val_score(DecisionTreeRegressor(max_depth=2), X_kf,y_kf, cv = 3)
np.average(DT_Score)
```

```
0.05247379896663843
```

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 62.80 %. Thus, the model is upto mark. it can also be tuned further to get better accuracy.

```
from sklearn.linear_model import LinearRegression
cross_val_score(LinearRegression(), X_kf,y_kf, cv = 3)
#array([0.73477372, 0.6035598 , 0.54577344])

LR_Score = cross_val_score(LinearRegression(), X_kf,y_kf, cv = 3)
np.average(LR_Score)
```

```
0.6280356539519311
```

From the above cross-validation it is found that, in some cases Random Forest 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 models 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 Random Forest are better for our given data set.

APPENDIX A

R Code

```
rm(list=ls())

#Set Working Directory
setwd("C:/Users/Lenovo/Documents/LM/EdWisor/Projects/Project 2")

getwd()


#Load Libraries
x = c("ggplot2", "corrgram", "DMwR", "caret", "randomForest", "unbalanced", "C50",
"dummies", "e1071", "Information",
      "MASS", "rpart", "gbm", "ROSE", 'sampling', 'DataCombine', 'inTrees')


install.packages(x)
lapply(x, require, character.only = TRUE)
rm(x)

############Load Data########################################

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


#Exploratory Data Analysis

class(Data_Day)
dim(Data_Day)
head(Data_Day)
names(Data_Day)
str(Data_Day)
summary(Data_Day)


#From the above observations

#Droping few columns
Data_Day = subset(Data_Day, select = -c(instant, dteday, casual, registered))

dim(Data_Day)
names(Data_Day)


#separate numeric and categorical variables

numeric_var = c('temp', 'atemp', 'hum', 'windspeed', 'cnt')

categorical_var = c('season', 'yr', 'mnth', 'holiday', 'weekday', 'workingday',
'weathersit')
```

```
######################## Missing Value analysis #############################

summary(is.na(Data_Day))
sum(is.na(Data_Day))

#there is no missing values


###############Outlier Analysis ##################################

df = Data_Day
Data_Day = df

# BoxPlots - Distribution and Outlier Check


library(ggplot2)

for (i in 1:length(numeric_var))
{
  assign(paste0("gn",i), ggplot(aes_string(y = (numeric_var[i]), x = "cnt"), data =
subset(Data_Day))+
           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=numeric_var[i],x="count")+
           ggtitle(paste("Box plot of count for",numeric_var[i])))
}

## Plotting plots together
gridExtra::grid.arrange(gn1,gn2,gn3,ncol=3)
gridExtra::grid.arrange(gn4,gn5, ncol=2)


# outliers found in windspeed and humidity variables.


#replacing outliers with NA

for(i in numeric_var){
  print(i)
  outlier = Data_Day[,i][Data_Day[,i] %in% boxplot.stats(Data_Day[,i])$out]
  print(length(outlier))
  Data_Day[,i][Data_Day[,i] %in% outlier] = NA
}

sum(is.na(Data_Day))
```

```
#Impute NA values with KNN

library(DMwR)
library(rpart)

Data_Day = knnImputation(Data_Day, k = 5)

sum(is.na(Data_Day))


#################### Data Understanding ###################


# Time to plot some graphs, so let's install few libraries

library(ggplot2)
library(scales)
library(psych)
library(gplots)


# Barplot with x axis as season and y axis as count

ggplot(Data_Day, aes(x = Data_Day$season, y = Data_Day$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"))

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


# Barplot with x axis as year and y axis as count

ggplot(Data_Day, aes(x = Data_Day$yr, y = Data_Day$cnt))+
  geom_bar(stat = "identity", fill = "red")+
  labs(title = "Number of bikes rented with respect to year", x = "yr", y = "cnt")+

  theme(panel.background = element_rect("white"))+
  theme(plot.title = element_text(face = "bold"))

# It is found that Year 1 has the highest count while year 0 has lowest count.


# Barplot with x axis as weekday and y axis as count

ggplot(Data_Day, aes(x = Data_Day$weekday, y = Data_Day$cnt))+
```

```
  geom_bar(stat = "identity", fill = "navyblue")+
  labs(title = "Number of bikes rented with respect to days", x = "Days of the
week", y = "count")+
  theme(panel.background = element_rect("white"))+
  theme(plot.title = element_text(face = "bold"))

#It is found that on day 5 there is highest count and on day 0 its lowest count of
bikes rented


#Count with respect to temperature and humidity together

ggplot(Data_Day,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 0.50 to 0.75, count is high.


# Count with respect to windspeed and weather together

ggplot(Data_Day, 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_Day, 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 3 to
season 4.
```

```
################Feature Selection ###########################################

df2 = Data_Day
Data_Day = df2

#Correlation Analysis and Anova test is done identify if variables can be reduced
or notis perfo

# Correlation Analysis for numeric variable

library(corrgram)

corrgram(Data_Day[,numeric_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.


# Anova Test for categorical variables

for(i in categorical_var){
  print(i)
  Anova_test_result = summary(aov(formula = cnt~Data_Day[,i],Data_Day))
  print(Anova_test_result)
}

#it is found that holiday, weekday and workingday has p value > 0.05. null
hypothesis accepted


# Dimension redusction , removing variables that ar not required

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


############Feature Scaling ######################################


numeric_var = c("temp","hum","windspeed","cnt")
catergorical_var = c("season", "yr", "mnth", "weathersit")


# Skewness test

library(propagate)
```

```
for(i in numeric_var){
  print(i)
  skew = skewness(Data_Day[,i])
  print(skew)
}

#dataset is approximately symmetric. values are found ranging between -0.5 to +0.5.


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

for(i in numeric_var){
  print(summary(Data_Day[,i]))
}

#dat is found as normalized, scaling not required


# visualizing normality check

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

hist(Data_Day$hum, col="Blue", xlab="Humidity", ylab="Frequency",
     main="Humidity Distribution")

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

# the distribution is approximately symmetric



####################MODELING ##########


library(DataCombine)
rmExcept("Data_Day")


df3 = Data_Day
Data_Day = df3

#Develop error metrics

#R Square

Rsquare = function(y,y1){
  cor(y,y1)^2
}
```

```
#MAPE

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



########Dummy creation #############

categorical_var = c("season","yr","mnth","weathersit")

library(dummies)

Data_Day = dummy.data.frame(Data_Day, categorical_var)

#Save Data for KFold CV
KFData = Data_Day


#divide data

set.seed(123)
train_index = sample(1:nrow(Data_Day),0.8*nrow(Data_Day))
train= Data_Day[train_index,]
test= Data_Day[-train_index,]


###############check multicollinearity ###########################

numeric_var = c("temp","hum","windspeed", "cnt")

numeric_var2 = Data_Day[,numeric_var]

library(usdm)

vifcor(numeric_var2, th = 0.7)

#No collinearity problem.


#############DECISION TREE ####################

library(rpart)

DTModel = rpart(cnt~., train, method = "anova" , minsplit=5)


# Predictions
```

```
DTTest = predict(DTModel, test[-25])

summary(DTModel)

#MAPE

DTMape_Test = MAPE(test[,25], DTTest)
DTMape_Test   #26.4225


#RSquare

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


#############RANDOM FOREST###################

library(randomForest)
set.seed(123)

RFModel = randomForest(cnt~., train, ntree = 500, importance = TRUE)

# Predictions

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


# MAPE

RFMape_Test = MAPE(test[,25], RFTest)
RFMape_Test  #  19.32104

#RSquare

RF_RSquare = Rsquare(test[,25], RFTest)
RF_RSquare   # 0.8685008

#################LINEAR REGRESSION###################

LRModel = lm(cnt~., train)

summary(LRModel)


# Predictions on test

LRTest = predict(LRModel, test[-25])
```

```
#MAPE

LRMape_Test = MAPE(test[,25], LRTest)
LRMape_Test #  21.56792


#RSquare

LR_RSquare = Rsquare(test[,25], LRTest)
LR_RSquare  #  0.8191175


###########################Model Selection & Evaluation ##########################

print("MAPE Statistics")
print(DTMape_Test)
print(RFMape_Test)
print(LRMape_Test)

print("Accuracy")
print(100 - DTMape_Test)
print(100 - RFMape_Test)
print(100 - LRMape_Test)


print("R Square Statistics")
print(DT_RSquare)
print(RF_RSquare)
print(LR_RSquare)


#######################Cross Validation ###########################

#Load Data
library(caret)

KFData

#divide data

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


#Random Forest Cross Validation

RF_KF = 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

RFPreds_RSquare = Rsquare(test[,25], RFpreds)
RFPreds_RSquare





#Decision Tree Cross Validation


DT_KF = 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

DTPreds_RSquare = Rsquare(test[,25], DTpreds)
DTPreds_RSquare





#Linear Regression CV


LR_KF = 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

LRPreds_RSquare = Rsquare(test[,25], LRpreds)
LRPreds_RSquare
```

APPENDIX B

Python Code

In [1]:

```
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from random import randrange,uniform
from sklearn.metrics import r2_score
from scipy import stats
```

In [2]:

```
# Set working directory
os.chdir("C:/Users/Lenovo/Documents/LM/EdWisor/Python")
```

In [3]:

```
os.getcwd()
```

Out[3]:

```
'C:\\Users\\Lenovo\\Documents\\LM\\EdWisor\\Python'
```

In [4]:

```
# Load Data
Data_Day = pd.read_csv("day.csv")
```

In [5]:

```
Data_Day.head()
```

Out[5]:

	instant	dteday	season	yr	mnth	holiday	weekday	workingday	weathersit	temp	a
0	1	2011-01-01	1	0	1	0	6	0	2	0.344167	0.3
1	2	2011-01-02	1	0	1	0	0	0	2	0.363478	0.3
2	3	2011-01-03	1	0	1	0	1	1	1	0.196364	0.1
3	4	2011-01-04	1	0	1	0	2	1	1	0.200000	0.2
4	5	2011-01-05	1	0	1	0	3	1	1	0.226957	0.2

EXPLORATORY DATA ANALYSIS

In [6]:

```
#Check Type of DataFrame
print(type(Data_Day))
```

```
<class 'pandas.core.frame.DataFrame'>
```

In [7]:

```
#Data Types of Varaibles

print(Data_Day.dtypes)
```

```
instant          int64
dteday          object
season           int64
yr               int64
mnth             int64
holiday          int64
weekday          int64
workingday       int64
weathersit       int64
temp           float64
atemp          float64
hum            float64
windspeed      float64
casual           int64
registered       int64
cnt              int64
dtype: object
```

In [8]:

```
#Dimension
print(Data_Day.shape)
```

```
(731, 16)
```

In [9]:

```
# Index range

print(Data_Day.index)
```

```
RangeIndex(start=0, stop=731, step=1)
```

In [10]:

```
#columns

print(Data_Day.columns)
```

```
Index(['instant', 'dteday', 'season', 'yr', 'mnth', 'holiday', 'weekday',
       'workingday', 'weathersit', 'temp', 'atemp', 'hum', 'windspeed',
       'casual', 'registered', 'cnt'],
      dtype='object')
```

In [11]:

```
#unique values present in each variable

print(Data_Day.nunique())
```

```
instant       731
dteday        731
season          4
yr              2
mnth           12
holiday         2
weekday         7
workingday      2
weathersit      3
temp          499
atemp         690
hum           595
windspeed     650
casual        606
registered    679
cnt           696
dtype: int64
```

In [12]:

```
# variable "instant" can be dropped as it simply represents the index
# Variable "dteday" can be ignored as output is not based on time series analysis
# casual and registered variables can be removed, as these two sums to dependent variable c

Data_Day = Data_Day.drop(Data_Day.columns[[0, 1, 13, 14]], axis = "columns")

print(Data_Day.shape)
```

```
(731, 12)
```

In [13]:

```
#Defining numeric and categorical variables and saving in specific array

numeric_var = ['temp', 'atemp', 'hum', 'windspeed', 'cnt']

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

DATA PRE PROCESSING

Missing Value Analysis

In [14]:

```
#sum of missing values

Data_Day.isnull().sum()
```

Out[14]:

```
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
```

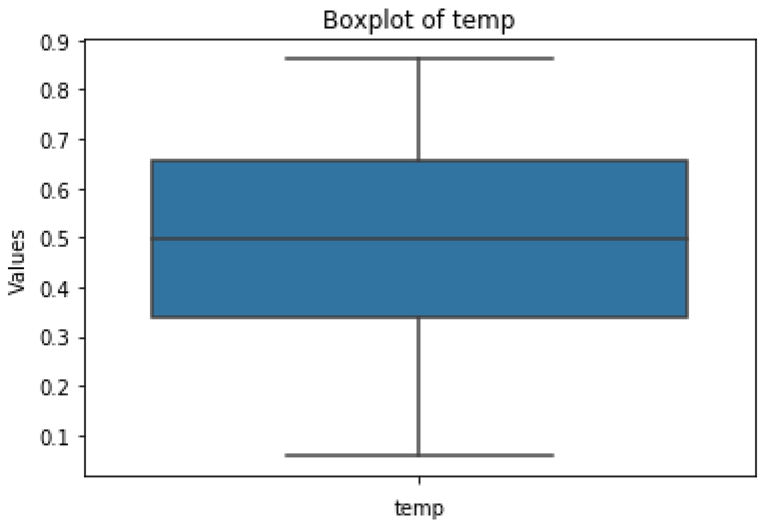
No missing values found

Outlier Analysis

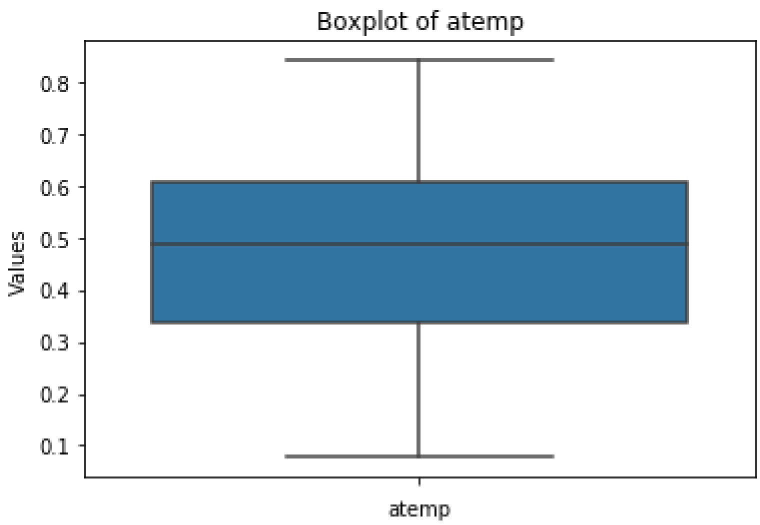
In [15]:

```
for i in numeric_var:
    print(i)
    sns.boxplot(y = Data_Day[i])
    plt.xlabel(i)
    plt.ylabel("Values")
    plt.title("Boxplot of " + i)
    plt.show()
```

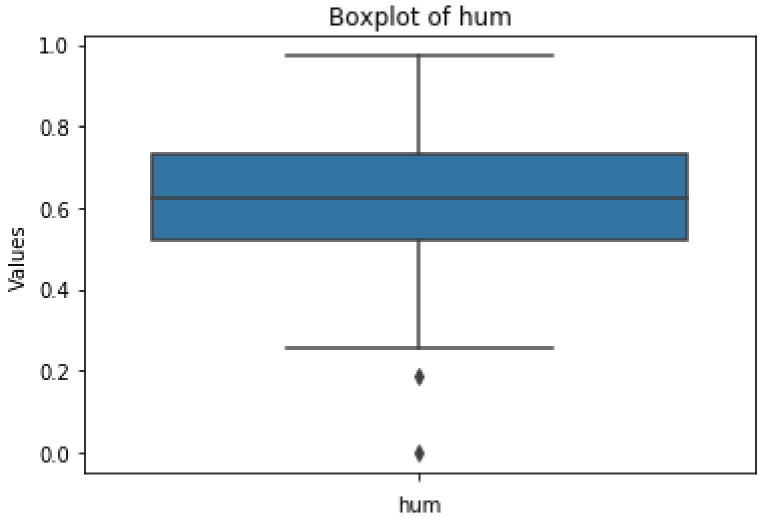
temp



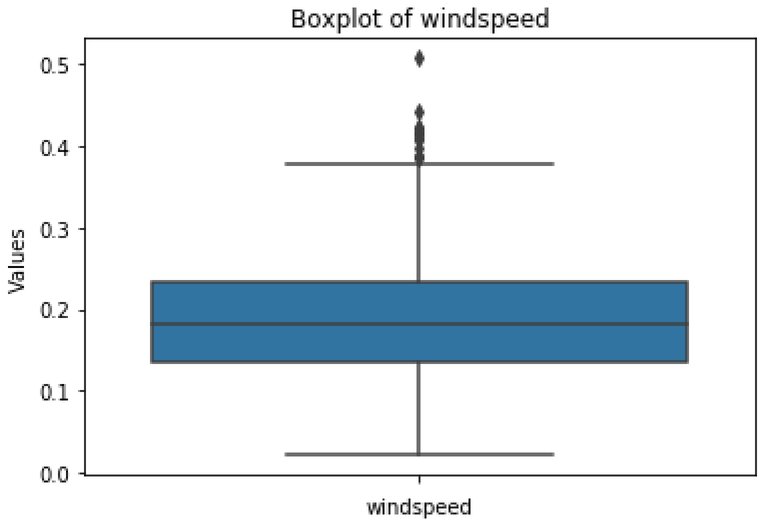
atemp



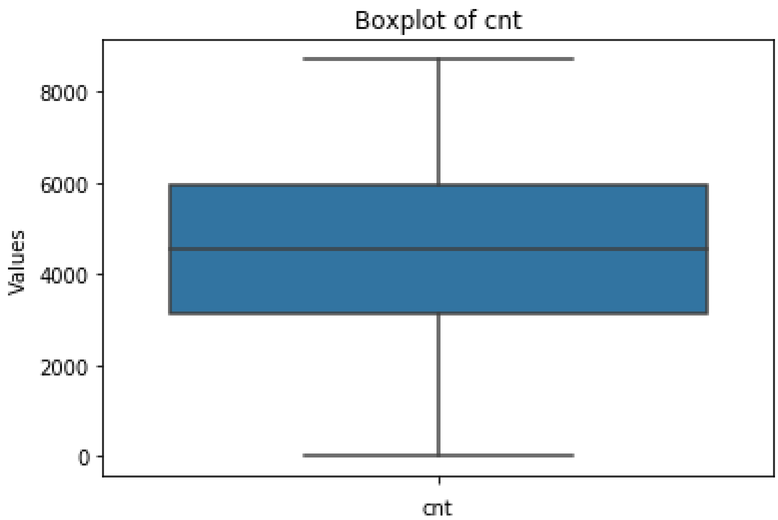
hum




```
windspeed
```




```
cnt
```



outliers are found in windspeed and humidity variables.

In [16]:

```
# Identify outliers
#calculate Inner Fence, Outer Fence, and IQR

for i in numeric_var:
    print(i)
    q75, q25 = np.percentile(Data_Day.loc[:,i], [75, 25])
    iqr = q75 - q25
    Innerfence = q25 - (iqr*1.5)
    Upperfence = q75 + (iqr*1.5)
    print("Innerfence= "+str(Innerfence))
    print("Upperfence= "+str(Upperfence))
    print("IQR ="+str(iqr))


# replace outliers with NA

    Data_Day.loc[Data_Day[i]<Innerfence, i] = np.nan
    Data_Day.loc[Data_Day[i]>Upperfence, i] = np.nan
```

```
temp
Innerfence= -0.14041600000000015
Upperfence= 1.1329160000000003
IQR =0.3183330000000001
atemp
Innerfence= -0.06829675000000018
Upperfence= 1.0147412500000002
IQR =0.2707595000000001
hum
Innerfence= 0.20468725
Upperfence= 1.0455212500000002
IQR =0.21020850000000002
windspeed
Innerfence= -0.012446750000000034
Upperfence= 0.38061125
IQR =0.0982645
cnt
Innerfence= -1054.0
Upperfence= 10162.0
IQR =2804.0
```

In [17]:

```
Data_Day.isnull().sum()
```

Out[17]:

```
season         0
yr             0
mnth           0
holiday        0
weekday        0
workingday     0
weathersit     0
temp           0
atemp          0
hum            2
windspeed     13
cnt            0
dtype: int64
```

In [18]:

```
# total 15 outliers found. Now, impute the values, by the help of median.

Data_Day['hum'] = Data_Day['hum'].fillna(Data_Day['hum'].median())
Data_Day['windspeed'] = Data_Day['windspeed'].fillna(Data_Day['windspeed'].median())
```

In [19]:

```
# Check NA Values

Data_Day.isnull().sum()
```

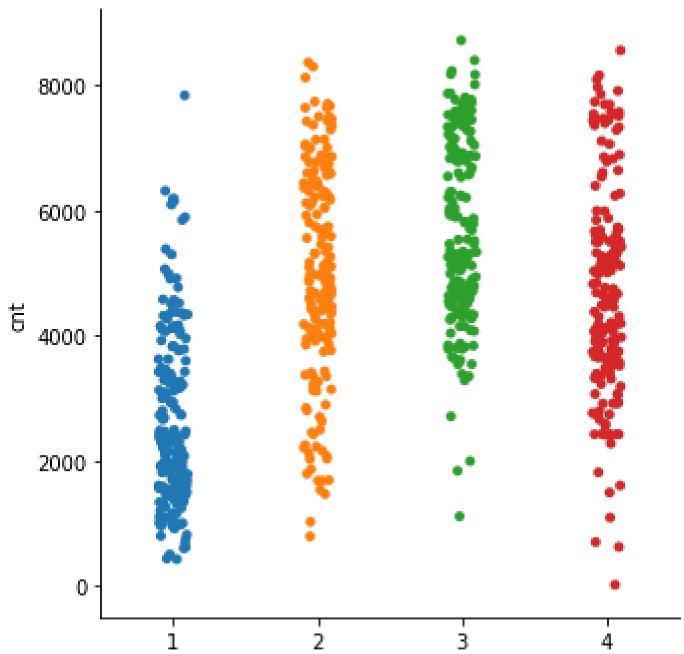
Out[19]:

```
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
```

DATA UNDERSTANDING

In [20]:

```
for i in categorical_var:
    sns.catplot(x = i, y = "cnt", data=Data_Day)
```



It is found that

In Season 2, 3 and 4 has the highest count

In Year 1 has high count than 0

In Months 3 to 10 has got pretty good count

On holidays the count is higher compared non-holidays

In weekdays, 0 and 6 has the highest count

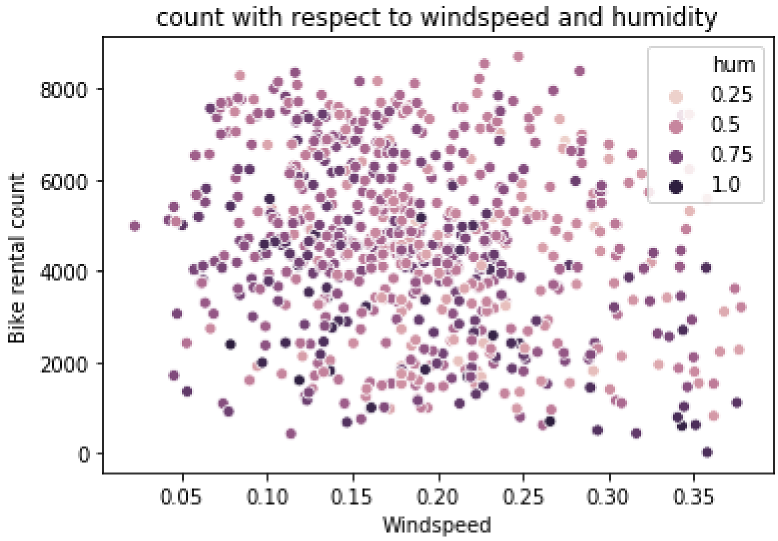
In weather, 1 has the highest count

In [21]:

```
scatter_plot1 = sns.scatterplot(x="windspeed", y="cnt", hue="hum", data= Data_Day)

plt.title("count with respect to windspeed and humidity")
plt.ylabel("Bike rental count")
plt.xlabel("Windspeed")
```

Out[21]:

```
Text(0.5, 0, 'Windspeed')
```



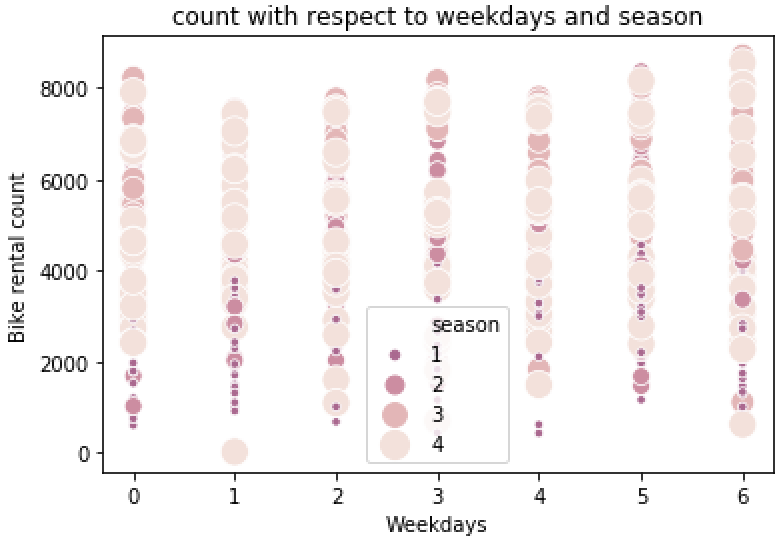
count vs windspeed and humidity, Count is High in ranges, windspeed 0.10 to 0.25 and humidity 0.5 to 0.75

In [22]:

```
cmap = sns.cubehelix_palette(dark=.9, light=.5, as_cmap=True)

scatter_plot2 = sns.scatterplot(x="weekday", y="cnt",
            hue="season", size="season", sizes = (20, 200), palette=cmap, data= Data_Day)

plt.title("count with respect to weekdays and season")
plt.ylabel("Bike rental count")
plt.xlabel("Weekdays")
```

Out[22]:

```
Text(0.5, 0, 'Weekdays')
```



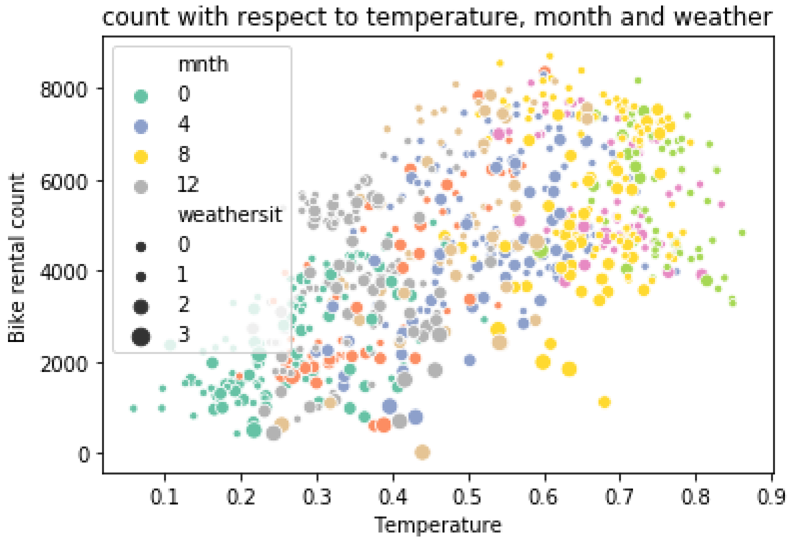
count vs weekdays and season, Count is high in 4th season and 1st and 6th weekdays

In [23]:

```
cmap2 = sns.cubehelix_palette(dark=.3, light=.8, as_cmap=True)

scatter_plot3 = sns.scatterplot(x="temp", y="cnt", hue="mnth", size="weathersit", palette="

plt.title("count with respect to temperature, month and weather")
plt.ylabel("Bike rental count")
plt.xlabel("Temperature")
```

Out[23]:

```
Text(0.5, 0, 'Temperature')
```



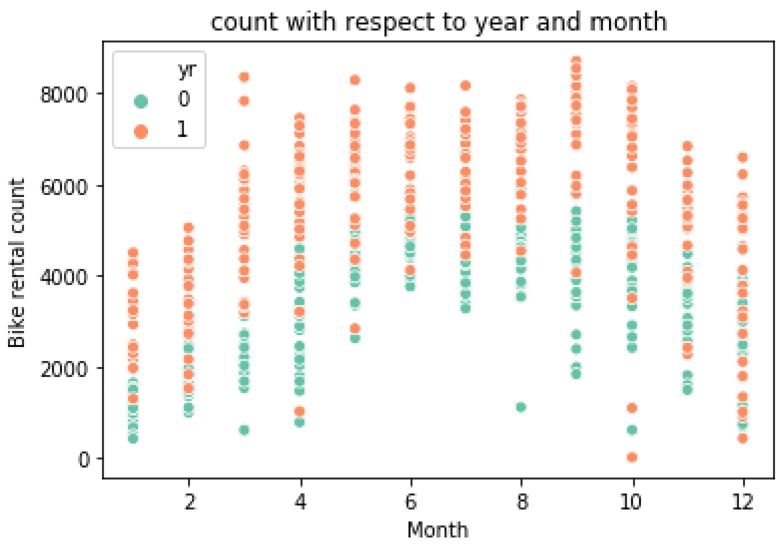
count vs temperature, month and weather, Count is high in range temperature 0.5 to 0.8, in 8th month and weather is 0.

In [24]:

```
cmap3 = sns.cubehelix_palette(dark=.3, light=.8, as_cmap=True)

scatter_plot4 = sns.scatterplot(x="mnth", y="cnt", hue="yr", palette="Set2", data= Data_Day

plt.title("count with respect to year and month")
plt.ylabel("Bike rental count")
plt.xlabel("Month")
```

Out[24]:

```
Text(0.5, 0, 'Month')
```



count vs respect to year and month, count is high in year 1, particularly from season 3 to 12 excluding 9

FEATURE SELECTION

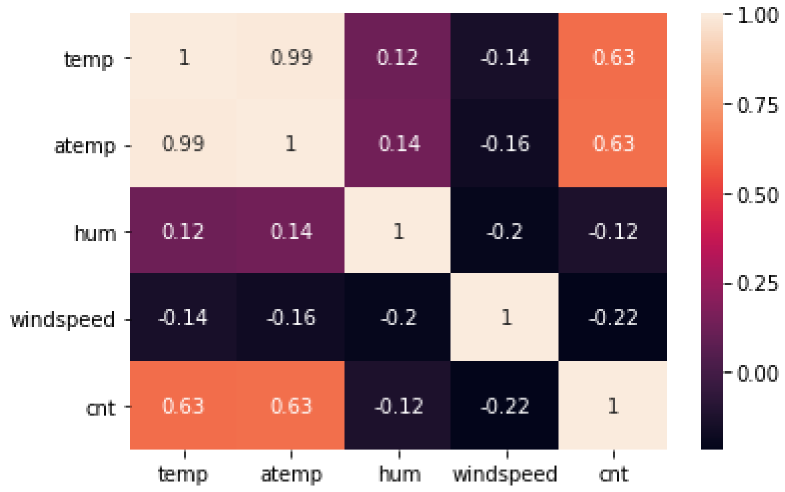
In [25]:

```
# Correlation Analysis and Anova test to find varaibles which can be excluded

Data_Day_cor = Data_Day.loc[:, numeric_var]
correlation_result = Data_Day_cor.corr()
print(correlation_result)
```

```
               temp     atemp       hum  windspeed       cnt
temp       1.000000  0.991702  0.123723  -0.138937  0.627494
atemp      0.991702  1.000000  0.137312  -0.164157  0.631066
hum        0.123723  0.137312  1.000000  -0.200237 -0.121454
windspeed -0.138937 -0.164157 -0.200237   1.000000 -0.215203
cnt        0.627494  0.631066 -0.121454  -0.215203  1.000000
```

In [26]:

```
heatmap = sns.heatmap(correlation_result, annot=True)
```



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

In [27]:

```
# Anova Test

import statsmodels.api as sm
from statsmodels.formula.api import ols

for i in categorical_var:
    mod = ols('cnt' + '~' + i, data = Data_Day).fit()
    anova_table = sm.stats.anova_lm(mod, typ = 2)
    print(anova_table)
```

```
                sum_sq     df           F        PR(>F)
season    4.517974e+08    1.0  143.967653  2.133997e-30
Residual  2.287738e+09  729.0         NaN           NaN
                sum_sq     df           F        PR(>F)
yr        8.798289e+08    1.0  344.890586  2.483540e-63
Residual  1.859706e+09  729.0         NaN           NaN
                sum_sq     df          F        PR(>F)
mnth      2.147445e+08    1.0  62.004625  1.243112e-14
Residual  2.524791e+09  729.0        NaN           NaN
                sum_sq     df         F    PR(>F)
holiday   1.279749e+07    1.0  3.421441  0.064759
Residual  2.726738e+09  729.0       NaN       NaN
                sum_sq     df         F    PR(>F)
weekday   1.246109e+07    1.0  3.331091  0.068391
Residual  2.727074e+09  729.0       NaN       NaN
                  sum_sq     df         F    PR(>F)
workingday  1.024604e+07    1.0  2.736742  0.098495
Residual    2.729289e+09  729.0       NaN       NaN
                  sum_sq     df          F        PR(>F)
weathersit  2.422888e+08    1.0  70.729298  2.150976e-16
Residual    2.497247e+09  729.0        NaN           NaN
```

It is found that holiday, weekday and workingday has p value > 0.05, by which,

we accept null hypothesis.

In [28]:

```
#Dimension Reduction

Data_Day = Data_Day.drop(['atemp', 'holiday', 'weekday', 'workingday'], axis = "columns")
print(Data_Day.shape)
```

```
(731, 8)
```

In [29]:

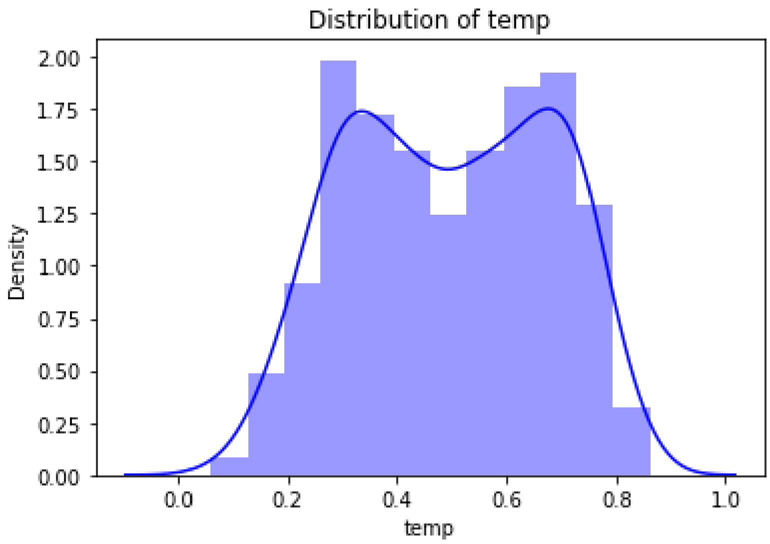
```
#Final Variables

numeric_var = ["temp","hum","windspeed","cnt"]   # numeric variables

categorical_var = ["season", "yr", "mnth", "weathersit"]   # categorical variables
```

FEATURE SCALING

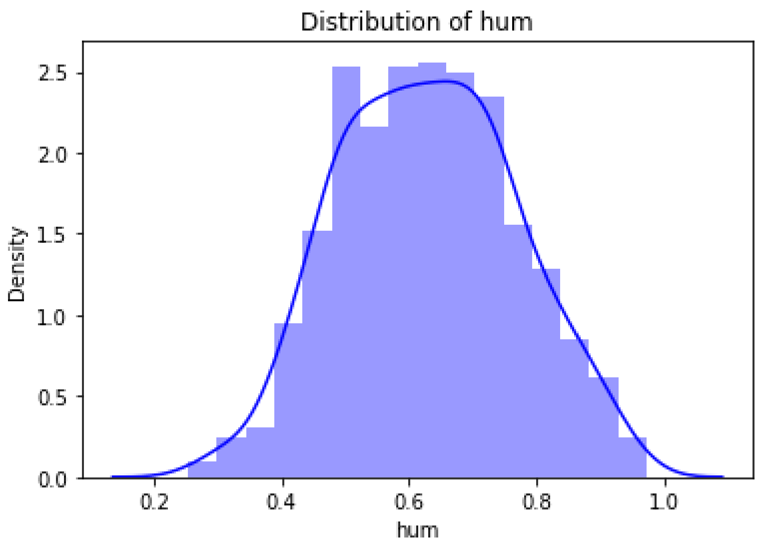
In [30]:

```
# Check normality
for i in numeric_var:
    print(i)
    sns.distplot(Data_Day[i], bins = 'auto', color = 'blue')
    plt.title("Distribution of "+i)
    plt.ylabel("Density")
    plt.show()
```

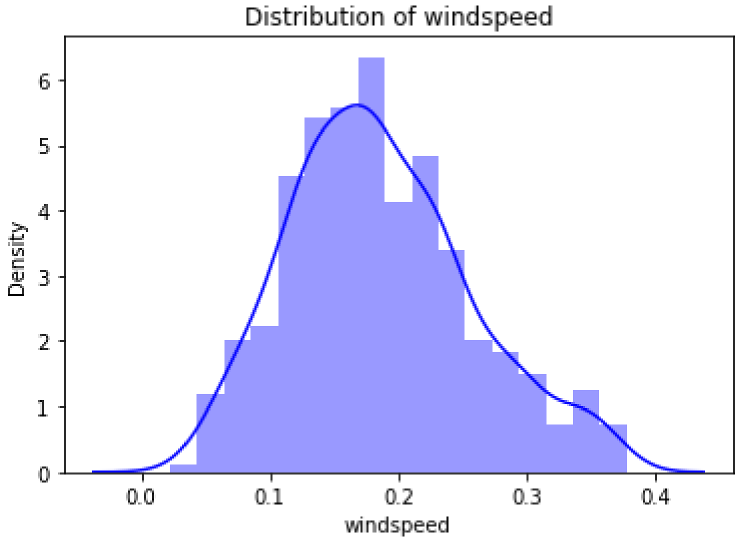
temp



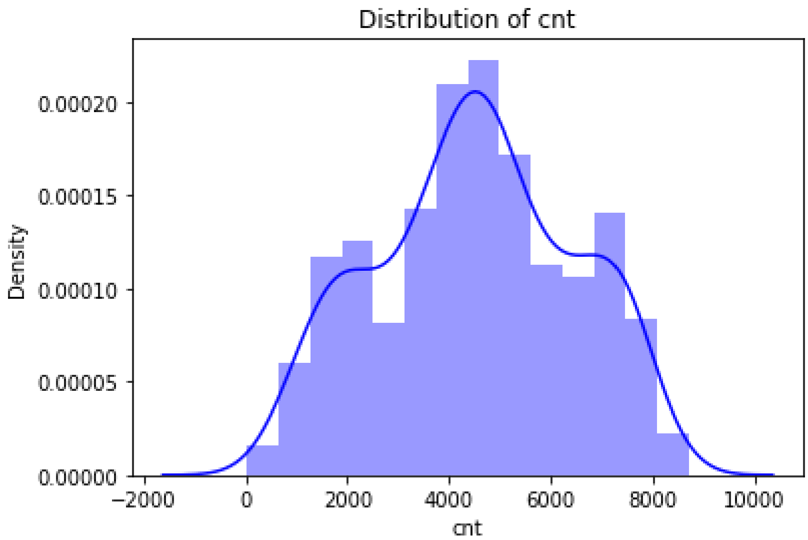
hum



windspeed



cnt



distributions are, approximately symmetric

In [31]:

```
# Check min and max values

Data_Day.describe()
```

Out[31]:

	season	yr	mnth	weathersit	temp	hum	windspeed	
count	731.000000	731.000000	731.000000	731.000000	731.000000	731.000000	731.000000	7
mean	2.496580	0.500684	6.519836	1.395349	0.495385	0.629354	0.186257	45
std	1.110807	0.500342	3.451913	0.544894	0.183051	0.139566	0.071156	19
min	1.000000	0.000000	1.000000	1.000000	0.059130	0.254167	0.022392	
25%	2.000000	0.000000	4.000000	1.000000	0.337083	0.522291	0.134950	31
50%	3.000000	1.000000	7.000000	1.000000	0.498333	0.627500	0.178802	45
75%	3.000000	1.000000	10.000000	2.000000	0.655417	0.730209	0.229786	59
max	4.000000	1.000000	12.000000	3.000000	0.861667	0.972500	0.378108	87

everything is normalized, no need of scaling

MODEL DEVELOPMENT

In [32]:

```
df = Data_Day.copy()
Data_Day = df.copy()
```

In [33]:

```
# Create dummy variables

Data_Day = pd.get_dummies(Data_Day, columns = categorical_var)

Data_Day.shape
```

Out[33]:

```
(731, 25)
```

In [34]:

```
Data_Day.head()
```

Out[34]:

	temp	hum	windspeed	cnt	season_1	season_2	season_3	season_4	yr_0	yr_
0	0.344167	0.805833	0.160446	985.0	1	0	0	0	1	
1	0.363478	0.696087	0.248539	801.0	1	0	0	0	1	
2	0.196364	0.437273	0.248309	1349.0	1	0	0	0	1	
3	0.200000	0.590435	0.160296	1562.0	1	0	0	0	1	
4	0.226957	0.436957	0.186900	1600.0	1	0	0	0	1	

5 rows × 25 columns

In [35]:

```
df_for_KFCV = Data_Day
```

In [36]:

```
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from scipy.stats.stats import pearsonr
```

In [37]:

```
#define Error Metrics.

def MAPE(y_actual, y_predicted):
    MAPE = np.mean(np.abs(y_actual-y_predicted)/y_actual)*100
    return MAPE

def Rsquare(y_actual, y_predicted):
    Rsquare = np.corrcoef(y_actual,y_predicted)**2
    return Rsquare
```

In [38]:

```
#predictors and target

X = Data_Day.drop(['cnt'], axis = "columns")
y = Data_Day['cnt']
```

In [39]:

```
#divide the data into train and test

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=.20, random_state=0)
```

DECISION TREE

In [40]:

```
from sklearn.tree import DecisionTreeRegressor
DTModel = DecisionTreeRegressor(max_depth=2).fit(X_train,y_train)

# Prediction

DTTest = DTModel.predict(X_test)

# MAPE
DTMape_Test = MAPE(y_test, DTTest)


# Rsquare - Test Data

DTR2_Test = Rsquare(y_test, DTTest)

DTR2_Test1 = DTR2_Test.ravel()

DTR2_Test2 = float(DTR2_Test1[1])


print("MAPE ="+str(DTMape_Test))
print("Accuracy =" + str(100 - DTMape_Test))
print("Rsquare ="+str(DTR2_Test2))
```

```
MAPE =36.94809301452646
Accuracy =63.05190698547354
Rsquare =0.6544606873373328
```

In [41]:

```
DTModel
```

Out[41]:

```
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')
```

RANDOM FOREST

In [42]:

```
from sklearn.ensemble import RandomForestRegressor

RFModel = RandomForestRegressor(n_estimators=100).fit(X_train,y_train)

# Predictions
RFTest = RFModel.predict(X_test)

# MAPE
RFMape_Test = MAPE(y_test, RFTest)

# Rsquare - Test Data

RFR2_Test = Rsquare(y_test, RFTest)

RFR2_Test1 = RFR2_Test.ravel()

RFR2_Test2 = float(RFR2_Test1[1])

print("MAPE ="+str(RFMape_Test))
print("Accuracy =" + str(100 - RFMape_Test))
print("Rsquare ="+str(RFR2_Test2))
```

```
MAPE =20.556838115351628
Accuracy =79.44316188464838
Rsquare =0.8871760288904629
```

In [43]:

```
RFModel
```

Out[43]:

```
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=100, n_jobs=None,
           oob_score=False, random_state=None, verbose=0, warm_start=False)
```

LINEAR REGRESSION MODEL

In [44]:

```
import statsmodels.api as sm
LRModel= sm.OLS(y_train, X_train).fit()
print(LRModel.summary())
```

```
                            OLS Regression Results
==============================================================================
==
Dep. Variable:                    cnt   R-squared:                       0.8
33
Model:                            OLS   Adj. R-squared:                  0.8
27
Method:                 Least Squares   F-statistic:                      14
0.2
Date:                Fri, 26 Jul 2019   Prob (F-statistic):          1.63e-2
03
Time:                        19:49:15   Log-Likelihood:                 -471
6.2
No. Observations:                 584   AIC:                             947
4.
Df Residuals:                     563   BIC:                             956
6.
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     574
5.398
hum          -1840.0359    351.762     -5.231      0.000   -2530.963    -114
9.109
windspeed    -2692.7145    509.781     -5.282      0.000   -3694.019    -169
1.410
season_1      -160.8963    149.431     -1.077      0.282    -454.407      13
2.615
season_2       735.4147    149.261      4.927      0.000     442.239     102
8.591
season_3       756.5640    170.170      4.446      0.000     422.319     109
0.809
season_4      1424.2811    170.259      8.365      0.000    1089.860     175
8.702
yr_0           409.9681    152.821      2.683      0.008     109.799      71
0.137
yr_1          2345.3954    151.325     15.499      0.000    2048.166     264
2.625
mnth_1          -1.9341    197.841     -0.010      0.992    -390.531      38
6.663
mnth_2          45.1383    186.947      0.241      0.809    -322.060      41
2.337
mnth_3         510.8770    141.897      3.600      0.000     232.166      78
9.588
mnth_4         233.3586    174.311      1.339      0.181    -109.021      57
5.738
mnth_5         659.7195    183.392      3.597      0.000     299.503     101
9.936
mnth_6         250.5066    180.098      1.391      0.165    -103.239      60
4.252
```

```
mnth_7        -222.2685    220.988     -1.006      0.315    -656.331     21
1.794
mnth_8         271.1265    207.045      1.310      0.191    -135.548     67
7.801
mnth_9         888.8861    173.978      5.109      0.000     547.161    123
0.611
mnth_10        382.5832    187.383      2.042      0.042      14.528     75
0.639
mnth_11       -183.6576    194.752     -0.943      0.346    -566.188     19
8.873
mnth_12        -78.9721    168.303     -0.469      0.639    -409.550     25
1.606
weathersit_1  1643.7280     90.978     18.067      0.000    1465.030    182
2.426
weathersit_2  1302.9232    110.447     11.797      0.000    1085.985    151
9.862
weathersit_3  -191.2876    221.771     -0.863      0.389    -626.886     24
4.311
============================================================================
==
Omnibus:                       97.249   Durbin-Watson:                   1.8
97
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              248.0
35
Skew:                          -0.849   Prob(JB):                     1.38e-
54
Kurtosis:                       5.704   Cond. No.                     1.54e+
16
============================================================================
==

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is corre
ctly specified.
[2] The smallest eigenvalue is 5.01e-30. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.
```

In [45]:

```
#Prediction

LRTest = LRModel.predict(X_test)

#MAPE

LRMape_Test = MAPE(y_test, LRTest)


#Rsquare -Test Data

LRR2_Test = Rsquare(y_test, LRTest)

LRR2_Test1 = LRR2_Test.ravel()

LRR2_Test2 = float(LRR2_Test1[1])


print("MAPE ="+str(LRMape_Test))
print("Accuracy =" + str(100 - LRMape_Test))
print("Rsquare ="+str(LRR2_Test2))
```

```
MAPE =18.800696038206944
Accuracy =81.19930396179305
Rsquare =0.8436040019904946
```

KFold Cross Validation

In [46]:

```
df_for_KFCV.head()
```

Out[46]:

	temp	hum	windspeed	cnt	season_1	season_2	season_3	season_4	yr_0	yr_
0	0.344167	0.805833	0.160446	985.0	1	0	0	0	1	
1	0.363478	0.696087	0.248539	801.0	1	0	0	0	1	
2	0.196364	0.437273	0.248309	1349.0	1	0	0	0	1	
3	0.200000	0.590435	0.160296	1562.0	1	0	0	0	1	
4	0.226957	0.436957	0.186900	1600.0	1	0	0	0	1	

5 rows × 25 columns

In [47]:

```
X_kf = df_for_KFCV.drop(['cnt'], axis = "columns")
y_kf = df_for_KFCV['cnt']
```

In [48]:

```
X_kf.head()
```

Out[48]:

	temp	hum	windspeed	season_1	season_2	season_3	season_4	yr_0	yr_1	mnth
0	0.344167	0.805833	0.160446	1	0	0	0	1	0	
1	0.363478	0.696087	0.248539	1	0	0	0	1	0	
2	0.196364	0.437273	0.248309	1	0	0	0	1	0	
3	0.200000	0.590435	0.160296	1	0	0	0	1	0	
4	0.226957	0.436957	0.186900	1	0	0	0	1	0	

5 rows × 24 columns

In [49]:

```
from sklearn.model_selection import cross_val_score
```

In [50]:

```
cross_val_score(DecisionTreeRegressor(max_depth=2), X_kf,y_kf, cv = 3)
#array([ 0.23365401, -0.23313404,  0.15690143])

DT_Score = cross_val_score(DecisionTreeRegressor(max_depth=2), X_kf,y_kf, cv = 3)
np.average(DT_Score)
```

Out[50]:

0.05247379896663843

In [51]:

```
cross_val_score(RandomForestRegressor(), X_kf,y_kf, cv = 3)
#array([0.69521348, 0.27999794, 0.452253  ])
RF_Score = cross_val_score(RandomForestRegressor(n_estimators = 100), X_kf,y_kf, cv = 3)
np.average(RF_Score)
```

```
C:\Users\Lenovo\Anaconda3\lib\site-packages\sklearn\ensemble\forest.py:246:
FutureWarning: The default value of n_estimators will change from 10 in vers
ion 0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
C:\Users\Lenovo\Anaconda3\lib\site-packages\sklearn\ensemble\forest.py:246:
FutureWarning: The default value of n_estimators will change from 10 in vers
ion 0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
C:\Users\Lenovo\Anaconda3\lib\site-packages\sklearn\ensemble\forest.py:246:
FutureWarning: The default value of n_estimators will change from 10 in vers
ion 0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
```

Out[51]:

```
0.49638830190520294
```

In [52]:

```
from sklearn.linear_model import LinearRegression
cross_val_score(LinearRegression(), X_kf,y_kf, cv = 3)
#array([0.73477372, 0.6035598 , 0.54577344])

LR_Score = cross_val_score(LinearRegression(), X_kf,y_kf, cv = 3)
np.average(LR_Score)
```

Out[52]:

```
0.6280356539519311
```

In [ ]:

```

```

REFERENCES

Websites:

- www.edwisor.com
- https://rpubs.com/
- https://www.r-bloggers.com/
- www.youtube.com
- www.wikipedia.com