Project Report

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

Gaurav Bagade

19th Nov 2021

TABLE OF CONTENTS

1. CHAPTER 1 : INTRODUCTION 3

a. 1.1 : Problem statement 2

b. 1.2 : Data 2

2. CHAPTER 2 : METHODOLOGY 4

a. 2.1 : Pre-processing 4

- 2.1.1 : Missing Value Analysis 4
- 2.1.2 : Outlier Analysis 5
- 2.1.3 : Data Understanding 6
- 2.1.4 : Feature Selection 9
- 2.1.5 : Feature Scaling 11

b. 2.2 : Model Development 13

- 2.2.1 : Model Selection 13
- 2.2.2 : Decision Tree 13
- 2.2.3 : Random Forest 14
- 2.2.4 : Linear Regression 14

3. CHAPTER 3 : EVALUATION OF THE MODEL 16

a. 3.2 : Accuracy 19

b. 3.1 : Mean Absolute Percentage Error (MAPE) 19

c. 3.3 : R Square 20

Appendix A – Python Code 18

References

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 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 day

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.

```
In [3]: Data.head()
```

Out[3]:

	instant	dteday	season	yr	mnth	holiday	weekday	workingday	weathersit	temp	atemp	hum	windspeed	casual	registered	cnt
0	1	2011-01-01	1	0	1	0	6	0	2	0.344167	0.363625	0.805833	0.160446	331	654	985
1	2	2011-01-02	1	0	1	0	0	0	2	0.363478	0.353739	0.696087	0.248539	131	670	801
2	3	2011-01-03	1	0	1	0	1	1	1	0.196364	0.189405	0.437273	0.248309	120	1229	1349
3	4	2011-01-04	1	0	1	0	2	1	1	0.200000	0.212122	0.590435	0.160296	108	1454	1562
4	5	2011-01-05	1	0	1	0	3	1	1	0.226957	0.229270	0.436957	0.186900	82	1518	1600

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 pre-processing 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 the data it is found that the data doesn't consist any missing values.

```
In [7]: Data.isnull().sum()
Out[7]: instant       0
        dteday        0
        season        0
        yr            0
        mnth          0
        holiday       0
        weekday       0
        workingday    0
        weathersit    0
        temp          0
        atemp         0
        hum           0
        windspeed     0
        casual        0
        registered    0
        cnt           0
        dtype: int64
```

No Missing Values Found

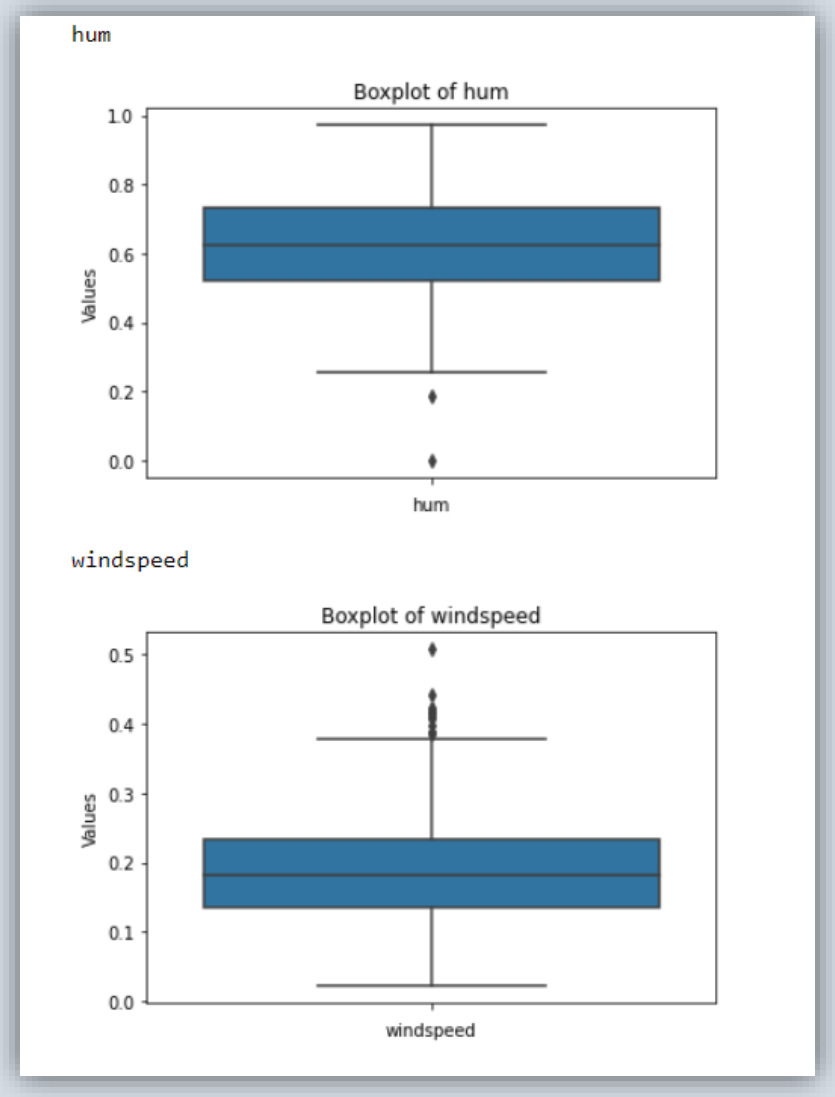
Plot: Missing Values

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 wind speed, following are the box plots for both the variables and dots outside the quartile ranges are outliers.



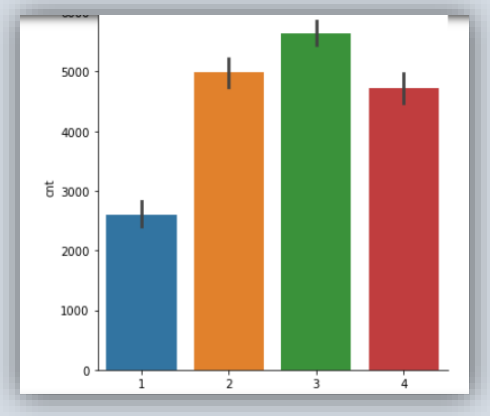
Plot: Outliers

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 wind speed 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.

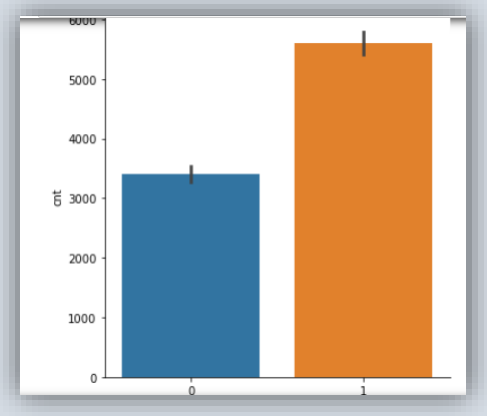
a. Season



Season

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

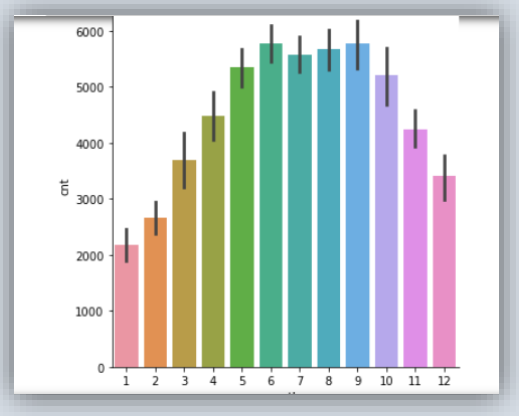
b. Year



Year

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

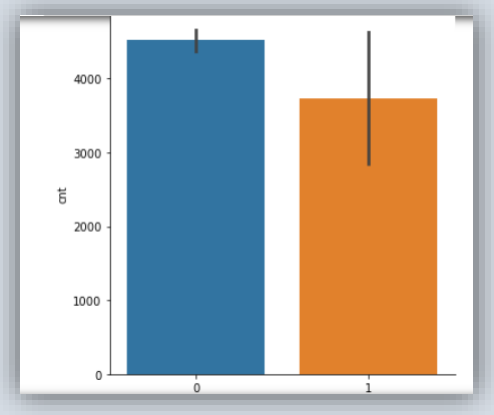
c. Month



Month

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

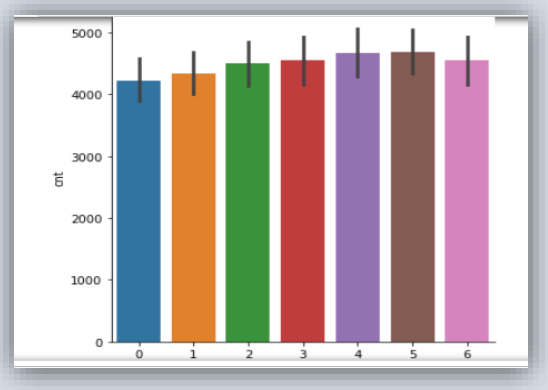
d. Holidays and Non-Holidays



Holiday

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

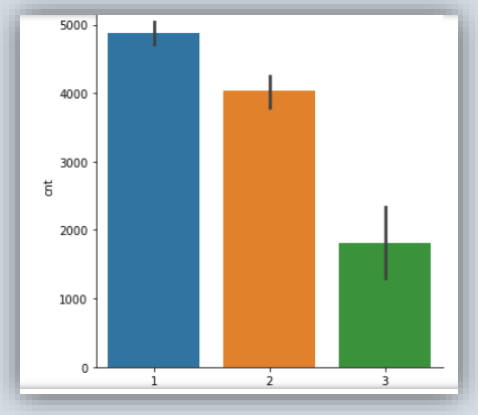
e. Weekdays



Weekdays

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

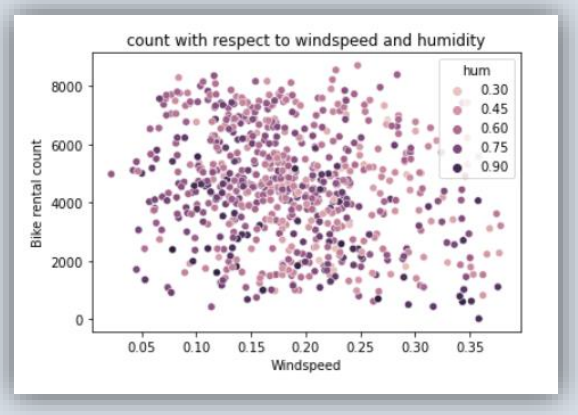
f. Weather



Weathersit

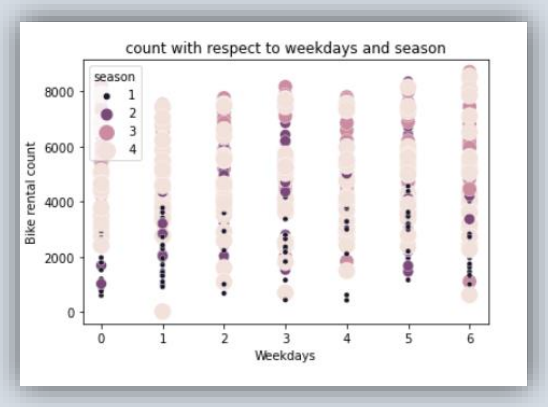
Here, in weather it is observed that, weather 1 has the highest count

g. Windspeed and Humidity vs count



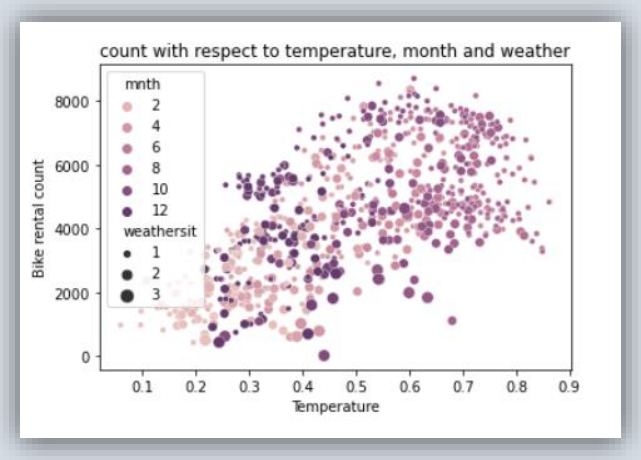
Here, it is found that in count vs windspeed and humidity, Count is high in ranges of windspeed 0.10 to 0.25 and humidity 0.5 to 0.75

h. . Weekdays and Season vs count



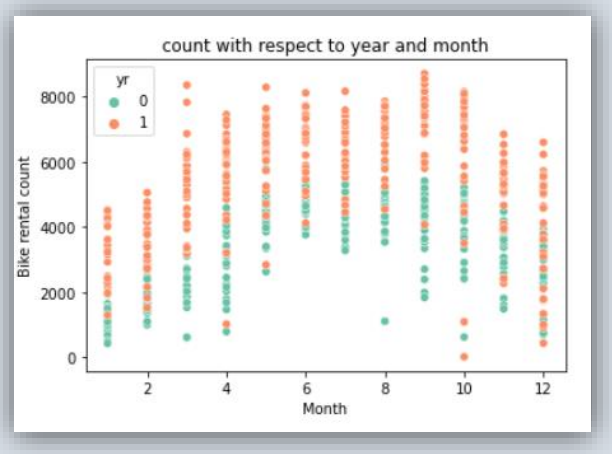
Here, it is observed that in count vs weekdays and season, Count is high in 4th season and 1st and 6th of weekdays.

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.

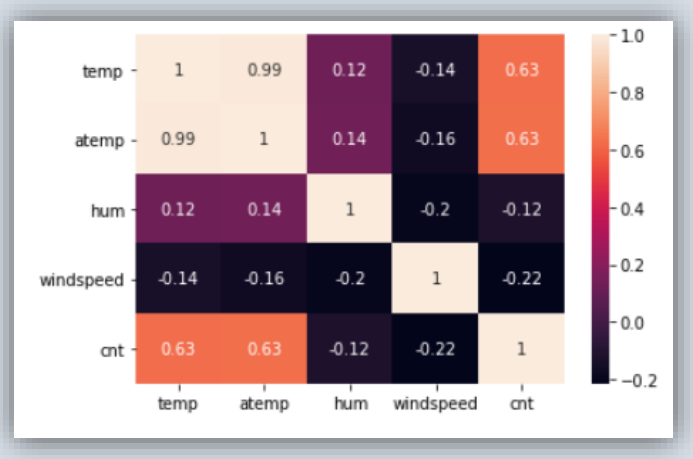
2.1.4 Feature Selection

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 analyse 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

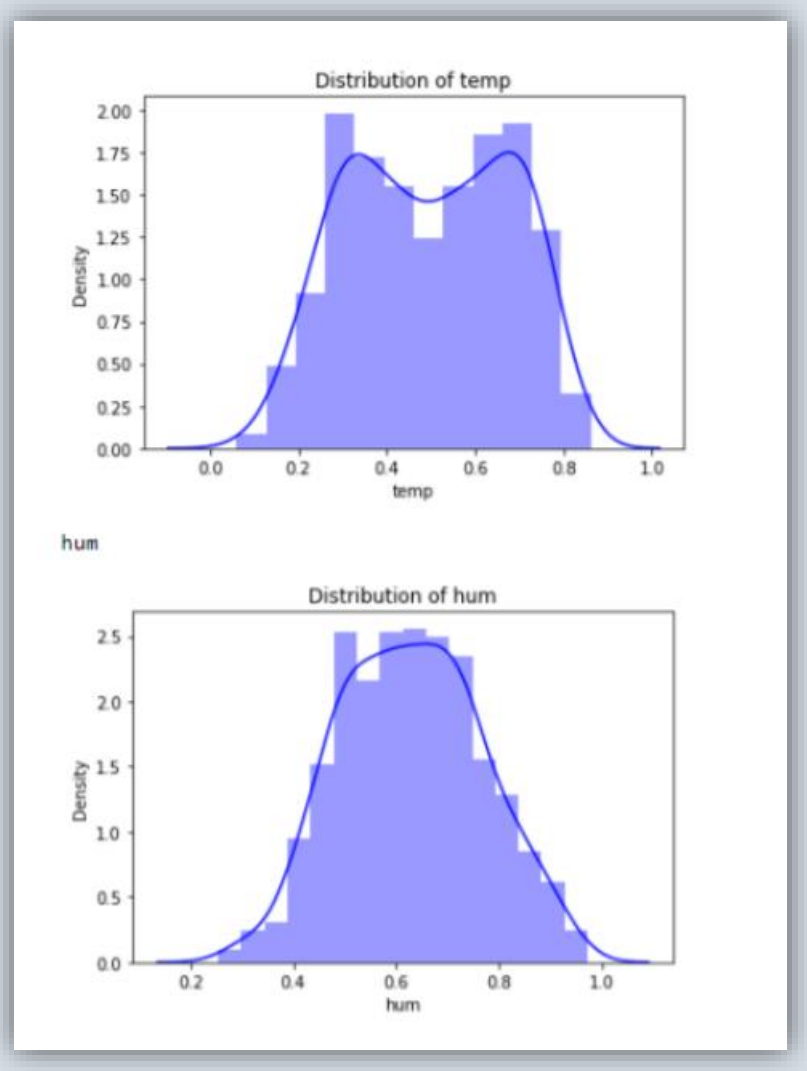
From the observations, it is found that the variables holiday, weekday, and working day has p value > 0.05. Here, null hypothesis is accepted. I.e. this variables has no dependency over target variable. So, in further processes this variables can be dropped before modelling. And this process of deducting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

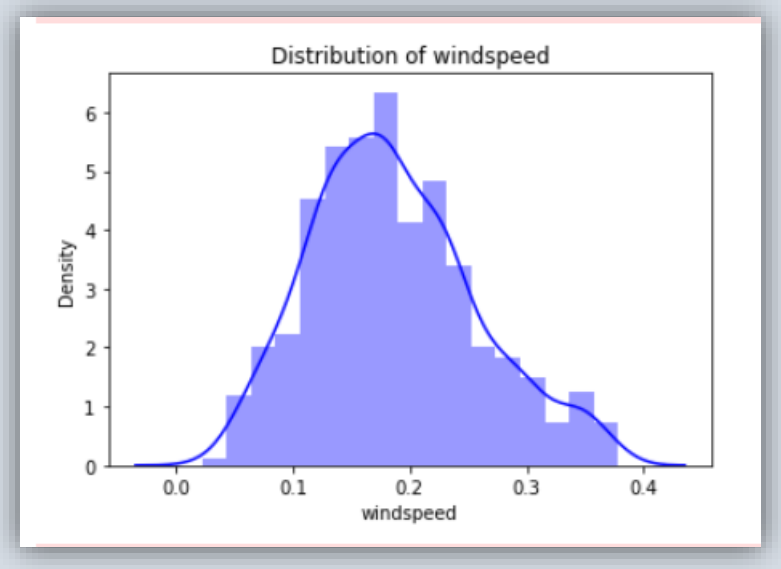
2.1.5 Feature Scaling

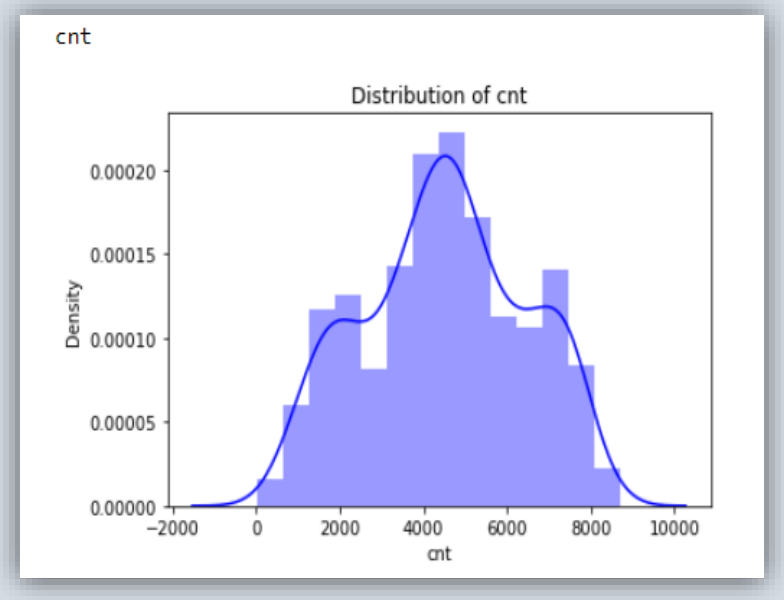
Here, In Feature Scaling ranges of variables are normalized or standardized, such that variables can be compared with same range. This is done for an unbiased and accurate model.

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







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	0.229785	5956.000000
max	4.000000	1.000000	12.000000	3.000000	0.861667	0.972500	0.378108	8714.000000

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 Python, details are described following.

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

To develop the model in python, during modelling 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:-36.94809301452646
ACCURACY:-63.05190698547354
Rsquare:-0.6464697716428665
```

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 Python, details are described following.

```
from sklearn.ensemble import RandomForestRegressor
RFModel = RandomForestRegressor(n_estimators=100).fit(X_train,y_train)
```

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:-20.274920141571627
ACCURACY:-79.72507985842837
Rsquare:-0.8881238894433683
```

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 Python, details are described following.

```
from sklearn.linear_model import LinearRegression
LRModel = LinearRegression().fit(X_train,y_train)
```

The above plot shows how the target variable count varies with change in each individual variable. The PValue 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:-18.782083030717057
ACCURACY:-81.21791696928294
Rsquare:-0.8410468277222658
```

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.

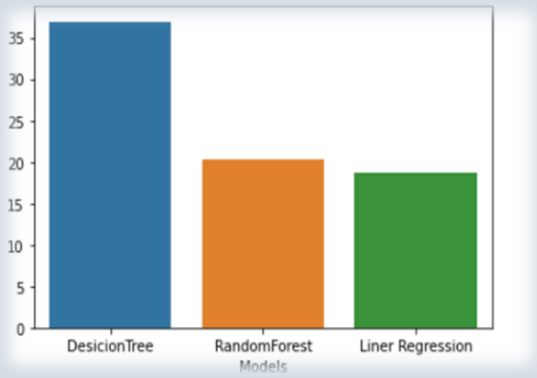
CHAPTER 3: EVALUATION OF THE MODEL

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.

	Models	MAPE
0	DesicionTree	36.948093
1	RandomForest	20.274920
2	Liner Regression	18.782083



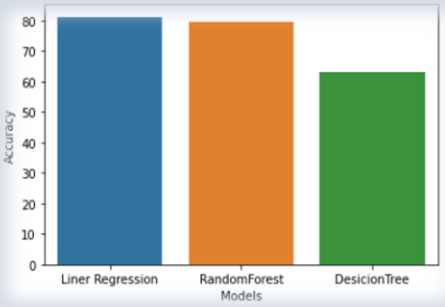
If we observe the above tables, we choose the model with lowest MAPE as a suitable Model. We get Linear Regression as a better model. So following this we can conclude that Both Random Forest and Linear Regression can be used as model for this data, if you evaluate on the basis of MAPE. But we need more 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

	Models	Accuracy
0	DesicionTree	63.051907
1	RandomForest	79.725080
2	Liner Regression	81.217917

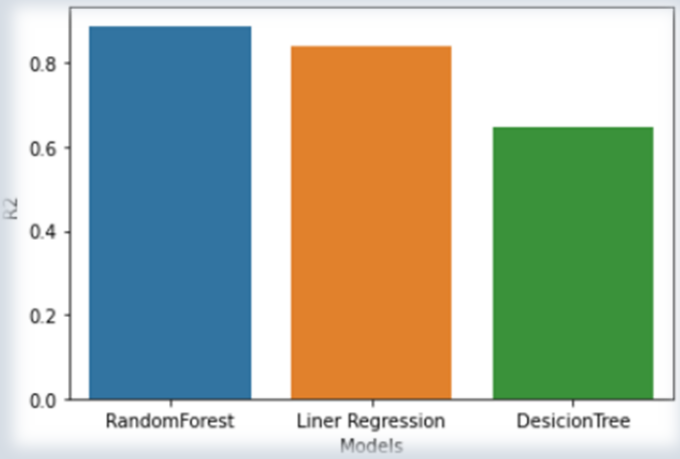


As, Accuracy derives from MAE/MAPE its observations also suggest same models as better models as suggested by MAPE. Here, the models with highest accuracy are chosen, and from the observations it is found that both Random Forest and Linear Regression are good models for the given data set.

3.3 R Square

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

	Models	R2
0	DesicionTree	0.646470
1	RandomForest	0.888124
2	Liner Regression	0.841047



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 Python it is found that Random Forest is a best fit model for the given data.

APPENDIX A

Python Code

Bike Rental Project

The objective of this Case is to Predication of bike rental count on daily based on the environmental and seasonal settings.

Import libraries

In [1]:

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

In [2]:

```
Data = pd.read_csv("day.csv")
```

In [3]:

```
Data.head()
```

Out[3]:

	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 [4]:

```
Data.shape
```

Out[4]:

```
(731, 16)
```

In [5]:

```
Data.dtypes
```

Out[5]:

```
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 [6]:

```
Data.nunique()
```

Out[6]:

```
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 [7]:

```
Data.isnull().sum()
```

Out[7]:

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

Removed instant it just represent index

Removed dteday date parameter mostly use for time series model

Removed casual and registered because we have total count as CNT

In [8]:

```
Data = Data.drop(Data.columns[[0, 1, 13, 14]], axis = "columns")
print(Data.shape)
```

```
(731, 12)
```

Defining Numeric and Categorical Varibales

In [9]:

```
numeric_var = ['temp', 'atemp', 'hum', 'windspeed', 'cnt']
categorical_var = ['season', 'yr', 'mnth', 'holiday', 'weekday', 'workingday', 'weathersit'
```

Missing Value Analysis

In [10]:

```
Data.isnull().sum()
```

Out[10]:

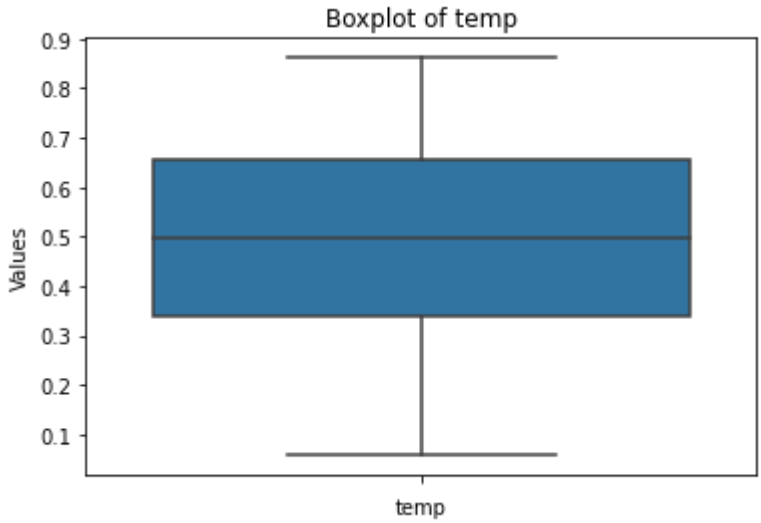
```
season        0
yr            0
mnth          0
holiday       0
weekday       0
workingday    0
weathersit    0
temp          0
atemp         0
hum           0
windspeed     0
cnt           0
dtype: int64
```

Outlier Analysis

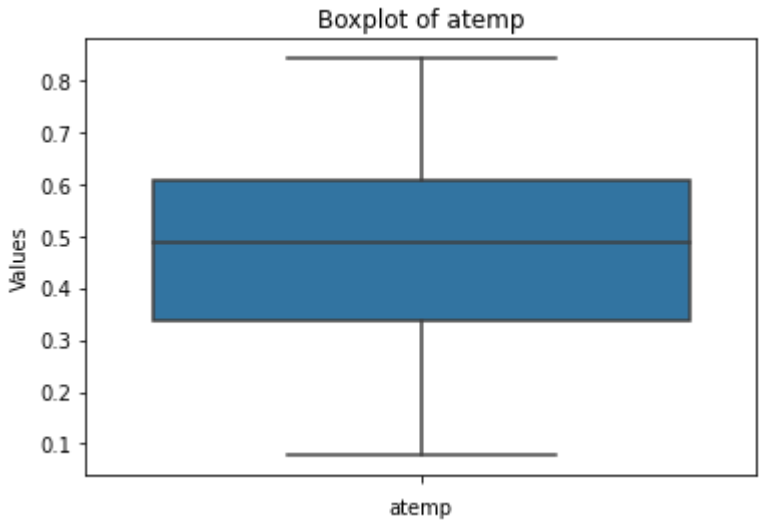
In [11]:

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

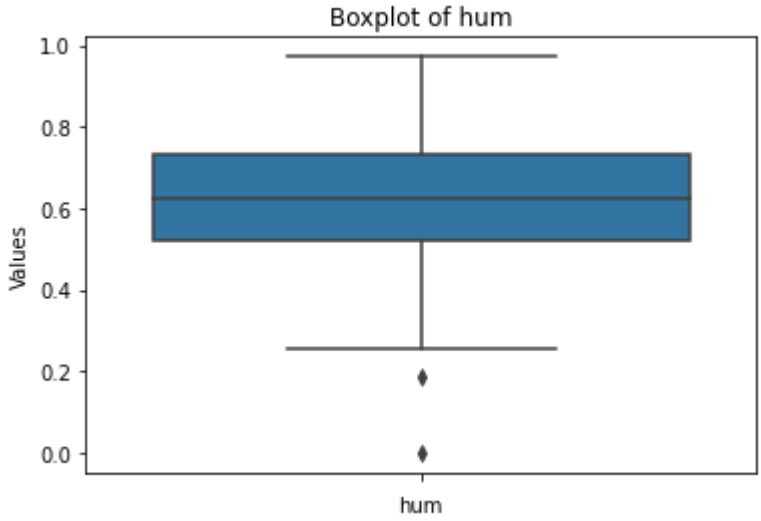
temp



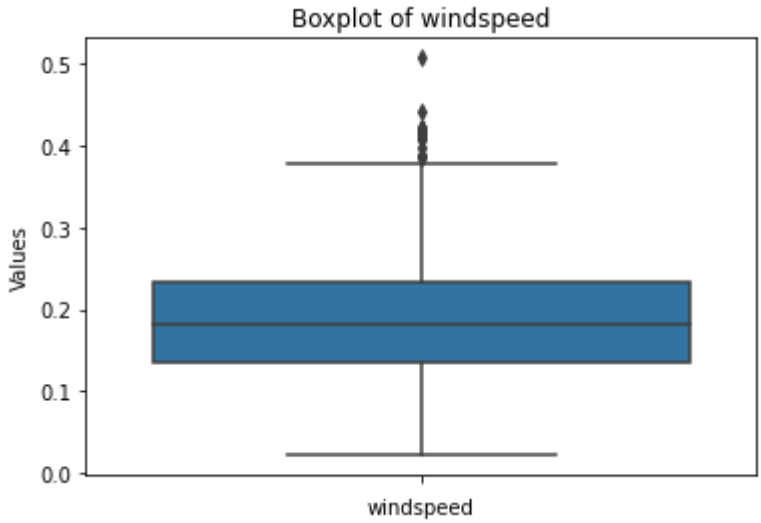
atemp



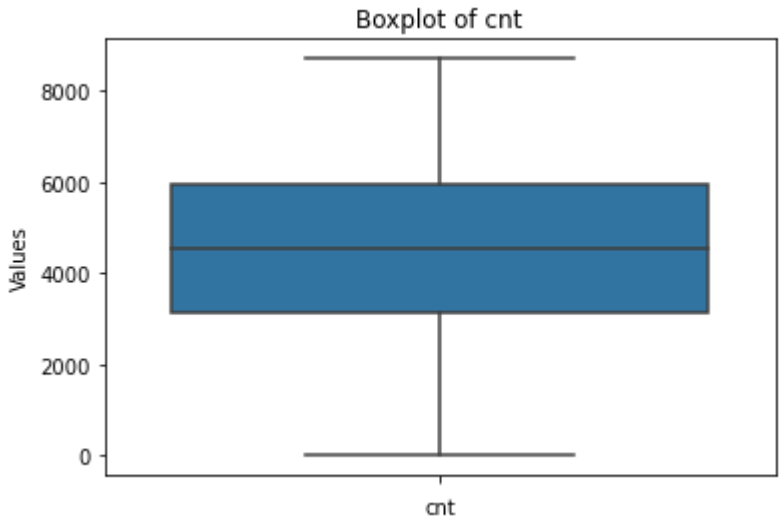
hum



windspeed



cnt



Humidity and Weendspeed have some outliers

Calculate inner fence, outer fence and IQR

In [12]:

```
# Identify outliers
#calculate Inner Fence, Outer Fence, and IQR
for i in numeric_var:
 print(i)
 q75, q25 = np.percentile(Data.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.loc[Data[i]<Innerfence, i] = np.nan
 Data.loc[Data[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 [13]:

```
Data.isnull().sum()
```

Out[13]:

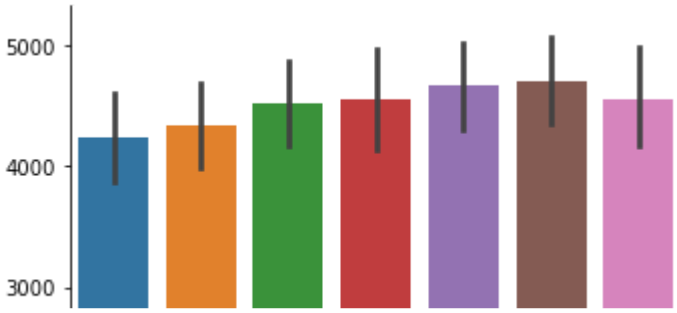
```
season         0
yr             0
mnth           0
holiday        0
weekday        0
workingday     0
weathersit     0
temp           0
atemp          0
hum            2
windspeed     13
cnt            0
dtype: int64
```

In [14]:

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

In [15]:

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



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

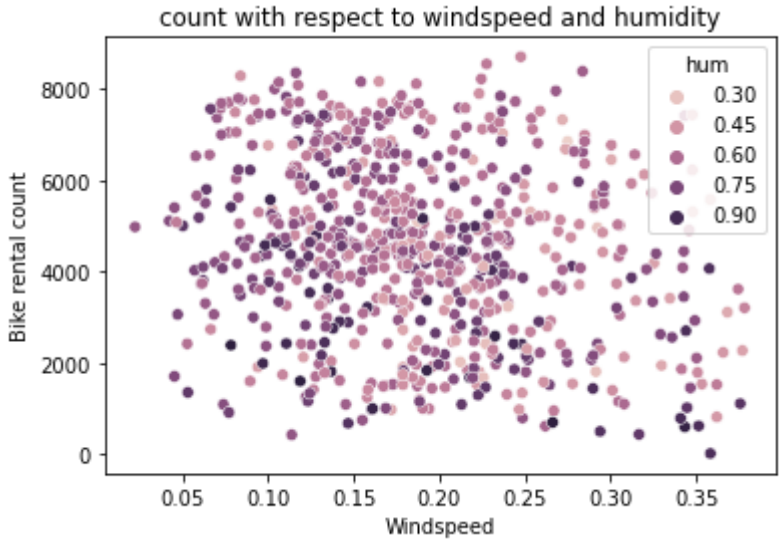
In weather, 1 has the highest count

In [16]:

```
scatter_plot1 = sns.scatterplot(x="windspeed", y="cnt", hue="hum", data= Data)
plt.title("count with respect to windspeed and humidity")
plt.ylabel("Bike rental count")
plt.xlabel("Windspeed")
```

Out[16]:

```
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 [17]:

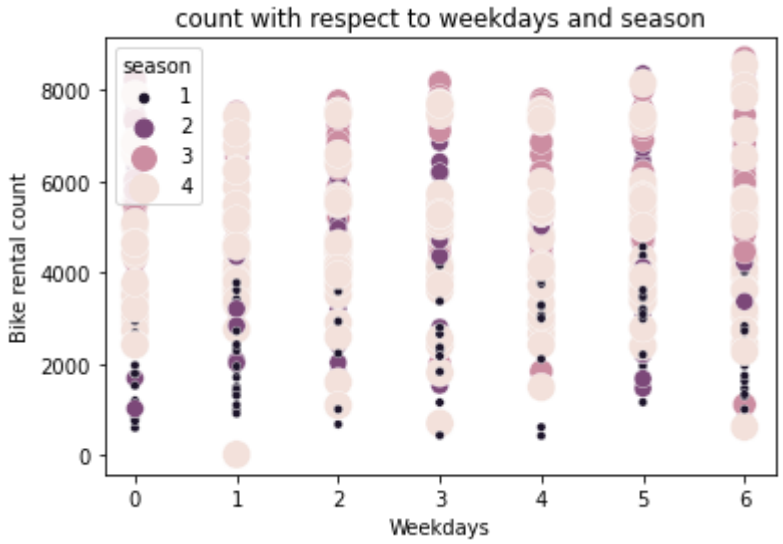
```
cmap = sns.cubehelix_palette(dark=.9, light=.1, as_cmap=True)

scatter_plot2 = sns.scatterplot(x="weekday", y="cnt", hue="season", size="season", sizes =

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

Out[17]:

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



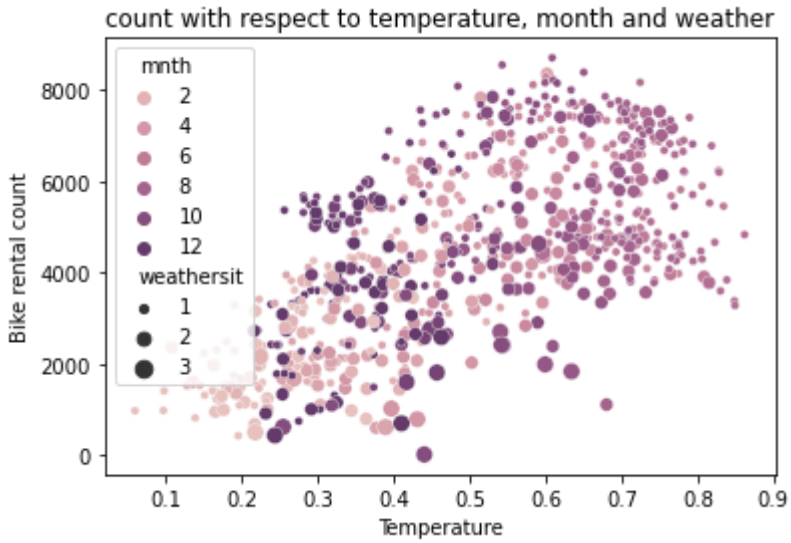
count vs weekdays and season, Count is high in 1st season and 4th and 6th weekdays

In [18]:

```
cmap2 = sns.cubehelix_palette(dark=.3, light=.8, as_cmap=True)
scatter_plot3 = sns.scatterplot(x="temp", y="cnt", hue="mnth", size="weathersit", palette=c
plt.title("count with respect to temperature, month and weather")
plt.ylabel("Bike rental count")
plt.xlabel("Temperature")
```

Out[18]:

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



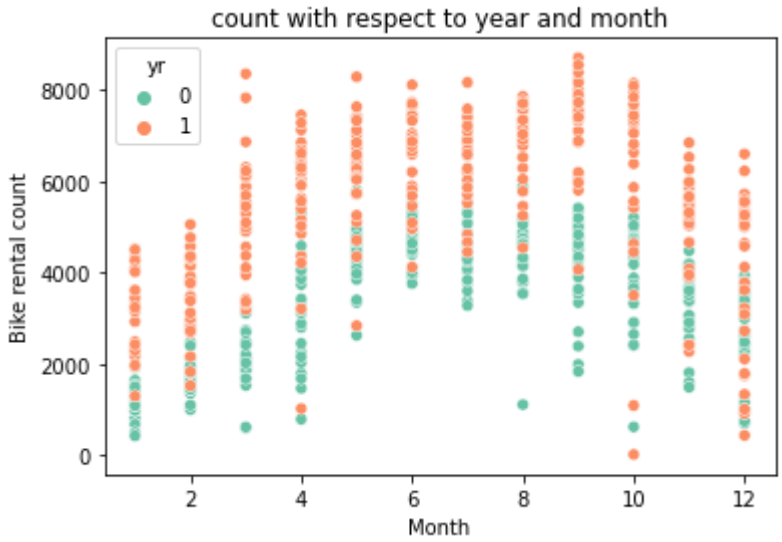
count vs temperature, month and weather, Count is high in range temperature 0.5 to 0.8, in 8th month and weather is 0.

In [19]:

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

Out[19]:

```
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

Correlation Analysis

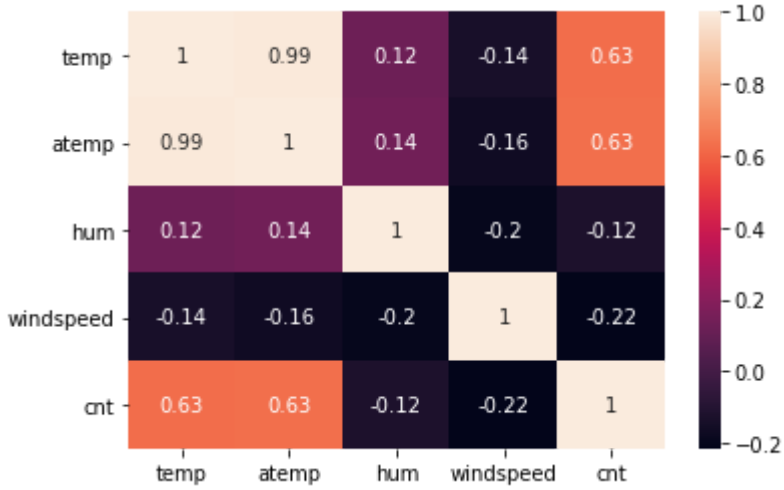
In [20]:

```
Data_cor = Data.loc[:, numeric_var]
correlation_result = Data_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 [21]:

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



temp and atemp highly correlated with each other

ANOVA Test

In [22]:

```
import statsmodels.api as sm
from statsmodels.formula.api import ols
for i in categorical_var:
 mod = ols('cnt' + '~' + i, data = Data).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
```

holiday, weekday and workingday has p value > 0.05, by which, we accept null hypothesis.

In [23]:

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

```
(731, 8)
```

Final variable

In [24]:

```
numeric_var = ["temp","hum","windspeed","cnt"] # numeric variables
categorical_var = ["season", "yr", "mnth", "weathersit"] # categorical variables
```

Feature Scaling

In [25]:

```
Final_Data.describe()
```

Out[25]:

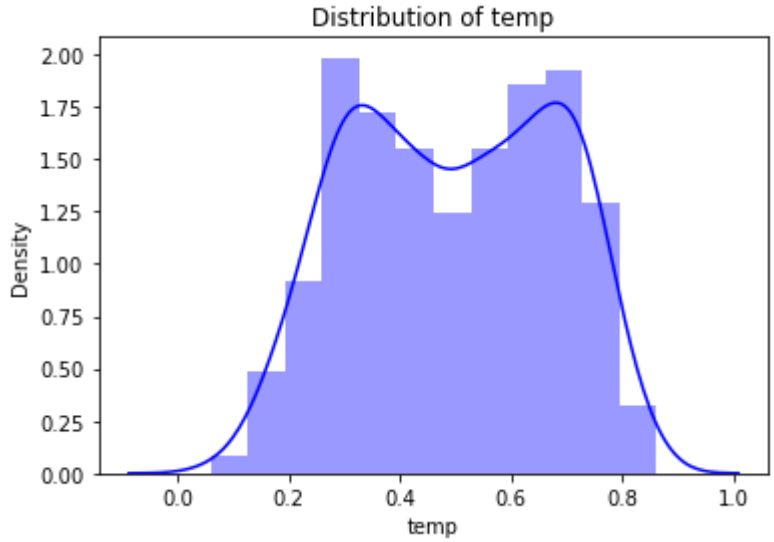
	season	yr	mnth	weathersit	temp	hum	windspeed	
count	731.000000	731.000000	731.000000	731.000000	731.000000	731.000000	731.000000	7
mean	2.496580	0.500684	6.519836	1.395349	0.495385	0.629354	0.186257	45
std	1.110807	0.500342	3.451913	0.544894	0.183051	0.139566	0.071156	19
min	1.000000	0.000000	1.000000	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	0.229785	59
max	4.000000	1.000000	12.000000	3.000000	0.861667	0.972500	0.378108	87

In [26]:

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

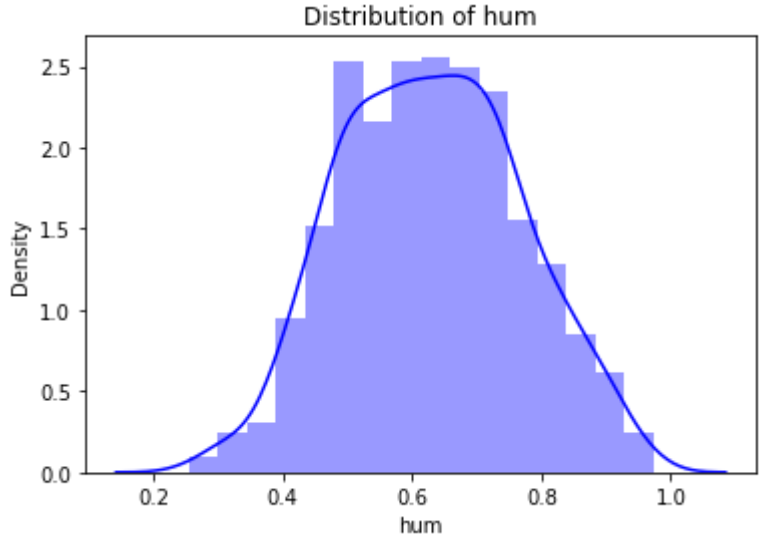
```
temp

E:\anaconda\lib\site-packages\seaborn\distributions.py:2557: FutureWarning:
`distplot` is a deprecated function and will be removed in a future version.
Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



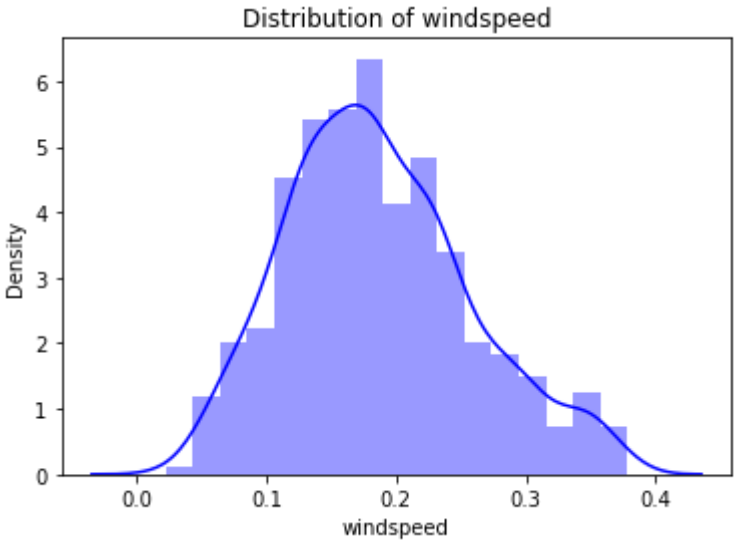

```
hum

E:\anaconda\lib\site-packages\seaborn\distributions.py:2557: FutureWarning:
`distplot` is a deprecated function and will be removed in a future version.
Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



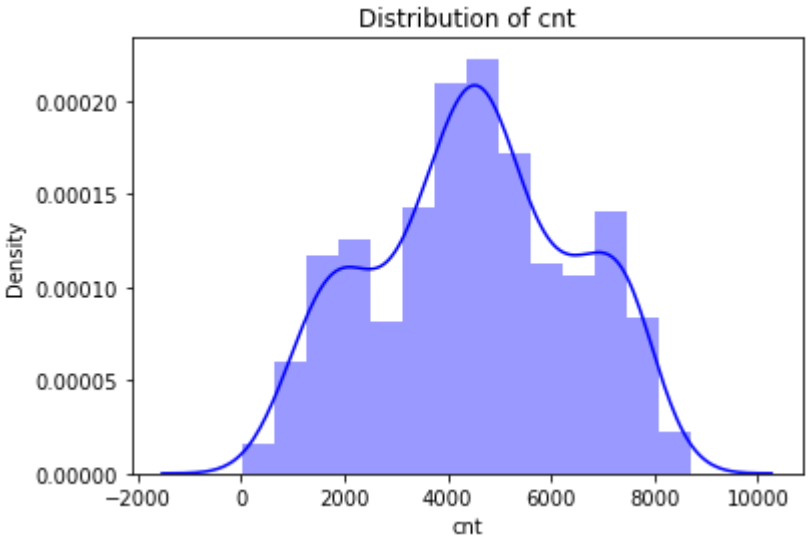

```
windspeed
```

```
E:\anaconda\lib\site-packages\seaborn\distributions.py:2557: FutureWarnin
g: `distplot` is a deprecated function and will be removed in a future ver
sion. Please adapt your code to use either `displot` (a figure-level funct
ion with similar flexibility) or `histplot` (an axes-level function for hi
stograms).
  warnings.warn(msg, FutureWarning)
```




```
E:\anaconda\lib\site-packages\seaborn\distributions.py:2557: FutureWarning:
`distplot` is a deprecated function and will be removed in a future version.
Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

cnt



Each variable is normalized

No need of feature scaling

In [27]:

```
Data_Final = Final_Data
```

Generating Dummy Variables

In [28]:

```
Data_Final = pd.get_dummies(Data_Final, columns = categorical_var)
Data_Final.shape
```

Out[28]:

```
(731, 25)
```

Model Development

Error Metric

In [29]:

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

In [30]:

```
def Rsquare(y_actual, y_predicted):
    mean_y_actual = np.mean(y_actual)
    numerator = 0
    denominator = 0
    for y_actual, y_predicted in zip(y_actual, y_predicted):
        numerator += (y_actual - y_predicted)**2
        denominator += (y_actual - mean_y_actual)**2
        ratio = numerator / denominator

    return 1 - ratio
```

In [31]:

```
X = Data_Final.drop(['cnt'], axis = "columns")
Y = Data_Final['cnt']
```

In [32]:

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

Decision Tree Model

In [33]:

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

In [34]:

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

In [35]:

```
D_test = Tree_Model.predict(X_test)

Mape_tree = MAPE(y_test, D_test)

R2_tree = Rsquare(y_test, D_test)

Acc_tree = 100 - Mape_tree
```

In [36]:

```
print("MAPE:-"+str(Mape_tree))
print("ACCURACY:-"+str(100 - Mape_tree))
print("Rsquare:-"+str(R2_tree))
```

```
MAPE:-36.94809301452646
ACCURACY:-63.05190698547354
Rsquare:-0.6464697716428665
```

Random Forest Model

In [37]:

```
from sklearn.ensemble import RandomForestRegressor
RFModel = RandomForestRegressor(n_estimators=100).fit(X_train,y_train)
```

In [38]:

```
RFTest = RFModel.predict(X_test)
```

In [39]:

```
Mape_rf = MAPE(y_test, RFTest)

R2_rf = Rsquare(y_test, RFTest)

Acc_rf = 100 - Mape_rf
```

In [40]:

```
print("MAPE:-"+str(Mape_rf))
print("ACCURACY:-"+str(100 - Mape_rf))
print("Rsquare:-"+str(R2_rf))
```

```
MAPE:-20.447167232476453
ACCURACY:-79.55283276752354
Rsquare:-0.8824755831274108
```

Linear Regression Model

In [41]:

```
from sklearn.linear_model import LinearRegression
LRModel = LinearRegression().fit(X_train,y_train)
```

In [42]:

```
LRTest = LRModel.predict(X_test)
```

In [43]:

```
Mape_lr = MAPE(y_test, LRTest)

R2_lr = Rsquare(y_test, LRTest)

Acc_lr = 100 - Mape_lr
```

In [44]:

```
print("MAPE:-"+str(Mape_lr))
print("ACCURACY:-"+str(100 - Mape_lr))
print("Rsquare:-"+str(R2_lr))
```

```
MAPE:-18.782083030717057
ACCURACY:-81.21791696928294
Rsquare:-0.8410468277222658
```

Comparison of models' accuracy

In [45]:

```
models = ['DesicionTree','RandomForest',  'Liner Regression']
col = [Acc_tree, Acc_rf, Acc_lr]
data = {'Models':models,'Accuracy':col}
graph_df = pd.DataFrame(data)
graph_df
```

Out[45]:

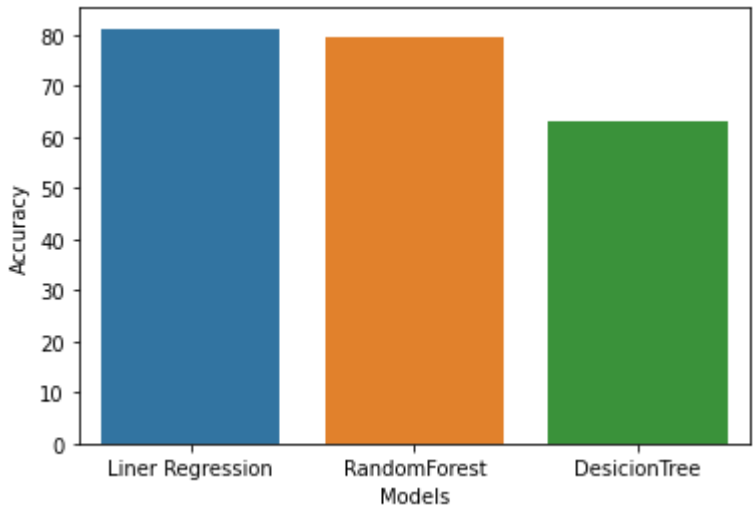
	Models	Accuracy
0	DesicionTree	63.051907
1	RandomForest	79.552833
2	Liner Regression	81.217917

In [46]:

```
graph_df = graph_df.sort_values(by=['Accuracy'], axis = 0, ascending=False)
```

In [47]:

```
fig, ax = plt.subplots()
sns.barplot(x=graph_df['Models'], y=graph_df['Accuracy'], data=graph_df);
```



Comparison of models' MAPE

In [48]:

```
models = ['DesicionTree','RandomForest',  'Liner Regression']
col = [Mape_tree, Mape_rf, Mape_lr]
data = {'Models':models,'MAPE':col}
graph_df = pd.DataFrame(data)
graph_df
```

Out[48]:

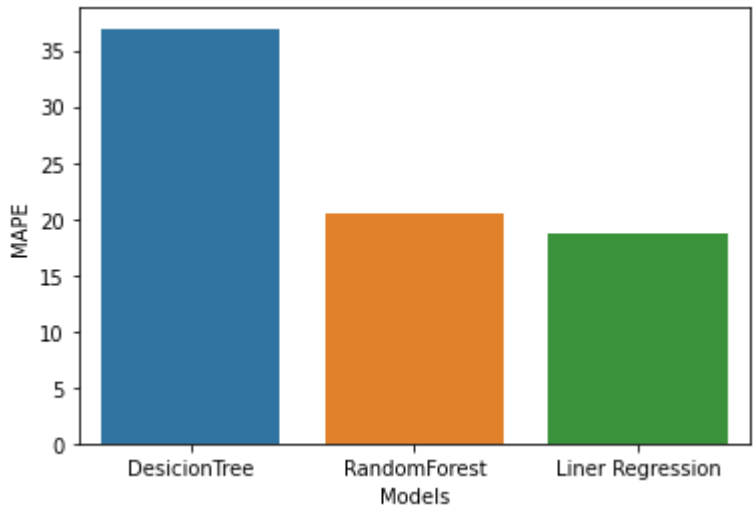
	Models	MAPE
0	DesicionTree	36.948093
1	RandomForest	20.447167
2	Liner Regression	18.782083

In [49]:

```
graph_df = graph_df.sort_values(by=['MAPE'], axis = 0, ascending=False)
```

In [50]:

```
fig, ax = plt.subplots()
sns.barplot(x=graph_df['Models'], y=graph_df['MAPE'], data=graph_df);
```



Comparison of models' Rsquare

In [51]:

```
models = ['DesicionTree','RandomForest',  'Liner Regression']
col = [R2_tree, R2_rf, R2_lr]
data = {'Models':models,'R2':col}
graph_df = pd.DataFrame(data)
graph_df
```

Out[51]:

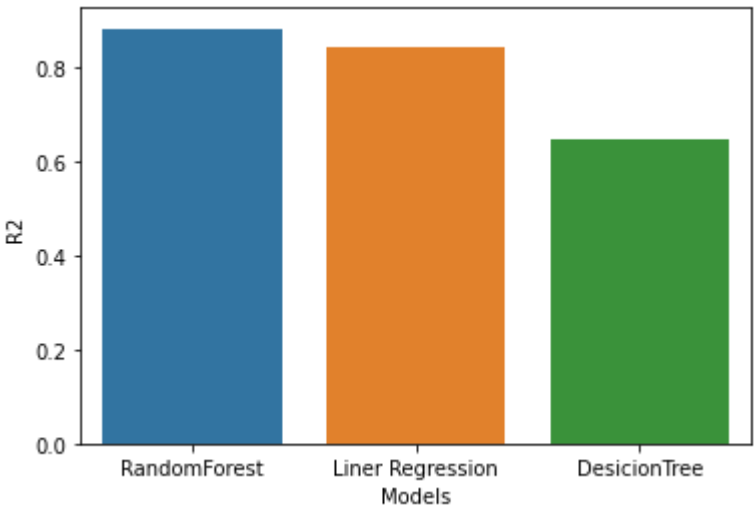
	Models	R2
0	DesicionTree	0.646470
1	RandomForest	0.882476
2	Liner Regression	0.841047

In [52]:

```
graph_df = graph_df.sort_values(by=['R2'], axis = 0, ascending=False)
```

In [53]:

```
fig, ax = plt.subplots()
sns.barplot(x=graph_df['Models'], y=graph_df['R2'], data=graph_df);
```



In [ ]:

```
```

In [ ]:

```
```