Prediction of Bike Rental based on Environmental and Seasonal Factors

- Mohammed Shadir (Shadhir99)

TABLE OF CONTENTS

1) Introduction
    a. Problem Statement
    b. Methodology
    c. Data Exploration
2) Descriptive Analysis
    a. Univariate Analysis
        i. Normality Check for Numerical Variables
        ii. Frequency table for Categorical Variables`
    b. Central Tendency Calculation
        i. Mean, Median and Mode
    c. Bivariate Analysis
        i. Numerical - Categorical
        ii. Categorical - Categorical
        iii. Numerical – Numerical
3) Data Pre-Processing
    a. Missing Value Analysis
    b. Feature Extraction
        i. Feature Selection
        ii. Feature Scaling
        iii. Dimension Reduction
    c. Outlier Analysis
        i. Outlier Detection
        ii. Outlier Treatment with Imputation
    d. Encoding Categorical Variables
4) Modelling and Prediction
    a. Regression Models
        i. Decision Tree Regression
        ii. Random Forest Regression
        iii. Multiple Linear Regression
        iv. XGBoost Regression

Introduction

Rental bike systems provide an alternative type of transport to many people. However, it is largely believed that the regular use of them is largely influenced by the environmental factors. People usually blame the bad weather conditions for not getting to work on a bike. The weather conditions and seasonal effect can contribute to the reduction in using these systems while in some cases; the extensive use of them due to good weather conditions could create issues for terrific management authorities in large cities.

In this project, a solution is proposed to model the number of bike users for a rental bike system based on the environmental and seasonal factors. The relationships between environmental variables such as temperature, humidity, wind speed, holidays, weekdays, etc. which can play part in a user's decision on whether or not to use a bike are investigate. To analyse these variable and their relationships, extensive use of analysis and visualisation tools such as R, Rattle and Weka were used. It is then a linear regression model produced to predict the number of bike users based on weather conditions and seasons and etc.

After performing several tests to determine the quality of the model, it is found that the proposed model can explain around 80% of variations in the target value (number of bike users).

Problem Statement

The aim of this project is to analyse the relationship between the factors which could somehow affect the number of rented bikes and their mobility in a city. It also introduces a model for predication of a bike rental system that can count hourly or daily usage based on the environmental and seasonal settings.

Data Exploration

The data set under study is related to 2-year usage log of a bike sharing system namely Capital Bike Sharing (CBS) at Washington, D.C., USA. The use of two whole sequential years (2011, 2012) enables us to investigate the effect of seasonal setting on the bike rental system. Dataset covers about records for 731 days and has 15 set of features for describing the data with 'cnt' as the total count of bikes rented (target variable).

Data Columns

Total Columns: 15, Rows: 731

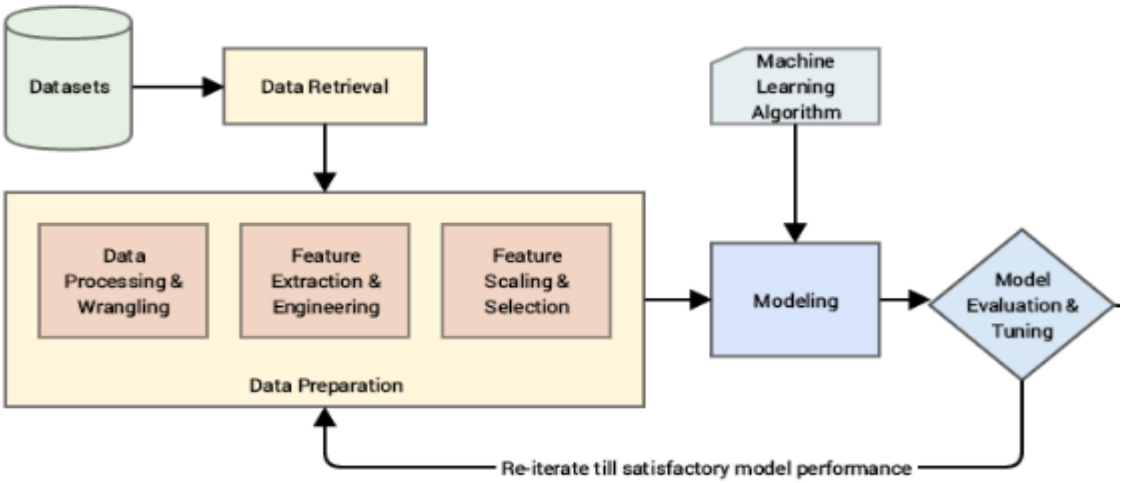
Name	Type	Description
dteday	datetime64[ns]	Date
season	Categorical	Season (1:springer, 2:summer, 3:fall, 4:winter)
yr	Categorical	Year (0: 2011, 1:2012)
mnth	Categorical	Month (1 to 12)
holiday	Categorical	whether day is holiday or not(1 or 0)
weekday	Categorical	Day of the week(0 to 6)
workingday	Categorical	If neither weekend nor holiday is 1, otherwise is 0.
weathersit	Categorical	1:Clear 2:Misty 3:LightRain 4:HeavyRain
temp	Numerical	Normalized temperature in Celsius
atemp	Numerical	Normalized Feeling temperature in Celsius
hum	Numerical	Normalized humidity
windspeed	Numerical	Normalized wind speed
casual	Numerical	count of casual users
registered	Numerical	count of registered users
cnt	Numerical	count of total rental bikes including both casual and registered

Top 5 Records:

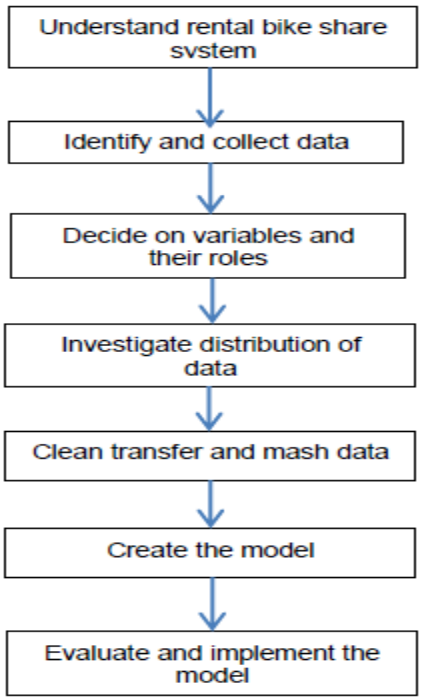
instant	dteday	season	yr	mnth	holiday	weekday	workingday	weathersit	temp	atemp	hum	windspeed	casual	registered	cnt
1	2011-01-01	1	0	1	0	6	0	2	0.344167	0.363625	0.805833	0.160446	331	654	985
2	2011-01-02	1	0	1	0	0	0	2	0.363478	0.353739	0.696087	0.248539	131	670	801
3	2011-01-03	1	0	1	0	1	1	1	0.196364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

Methodology

For implementation of this project, we're following the traditional CRISP-DM Process and has been divided into three major sections. The first section involves the data preparations. In the second section some exploratory data analysis is conducted and finally third section provides solution to a predictive model, and fits the model.

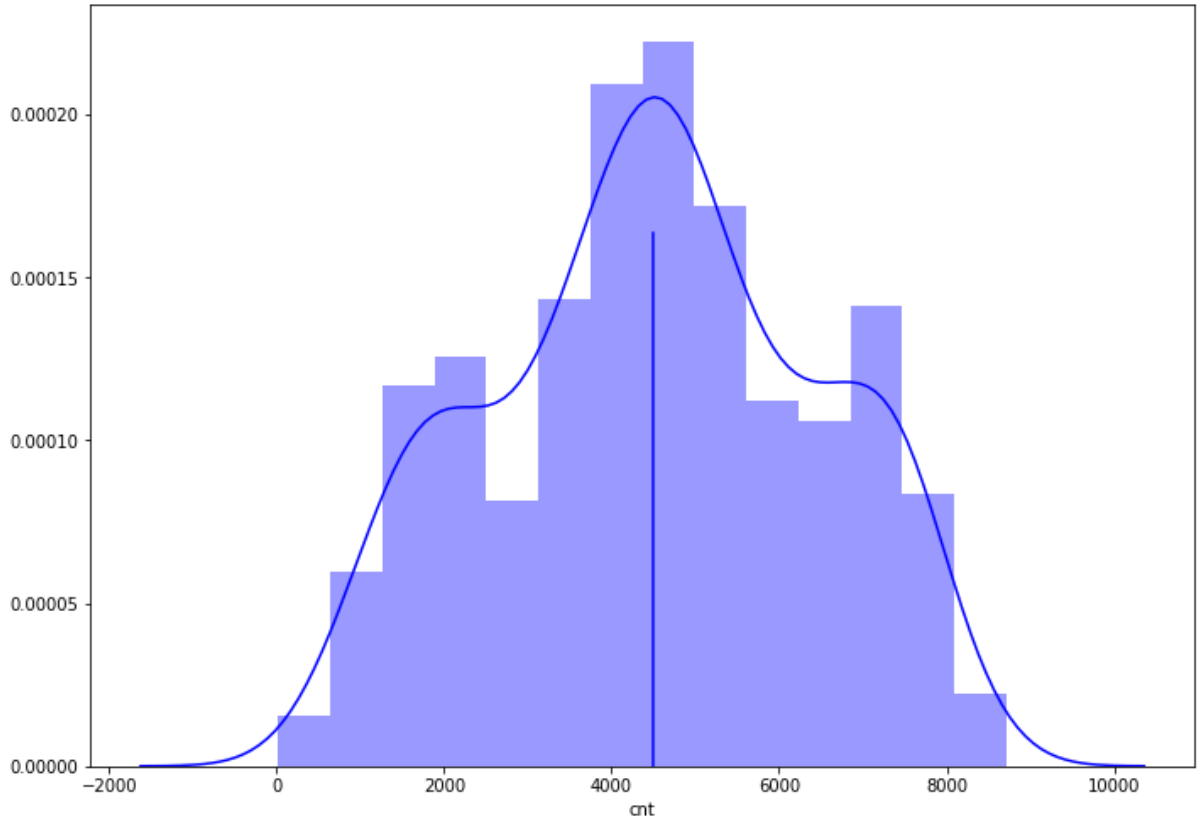


However the steps proposed for the completion of the project as a whole involves seven distinctive phase which are described in the next flowchart.



Univariate Analysis

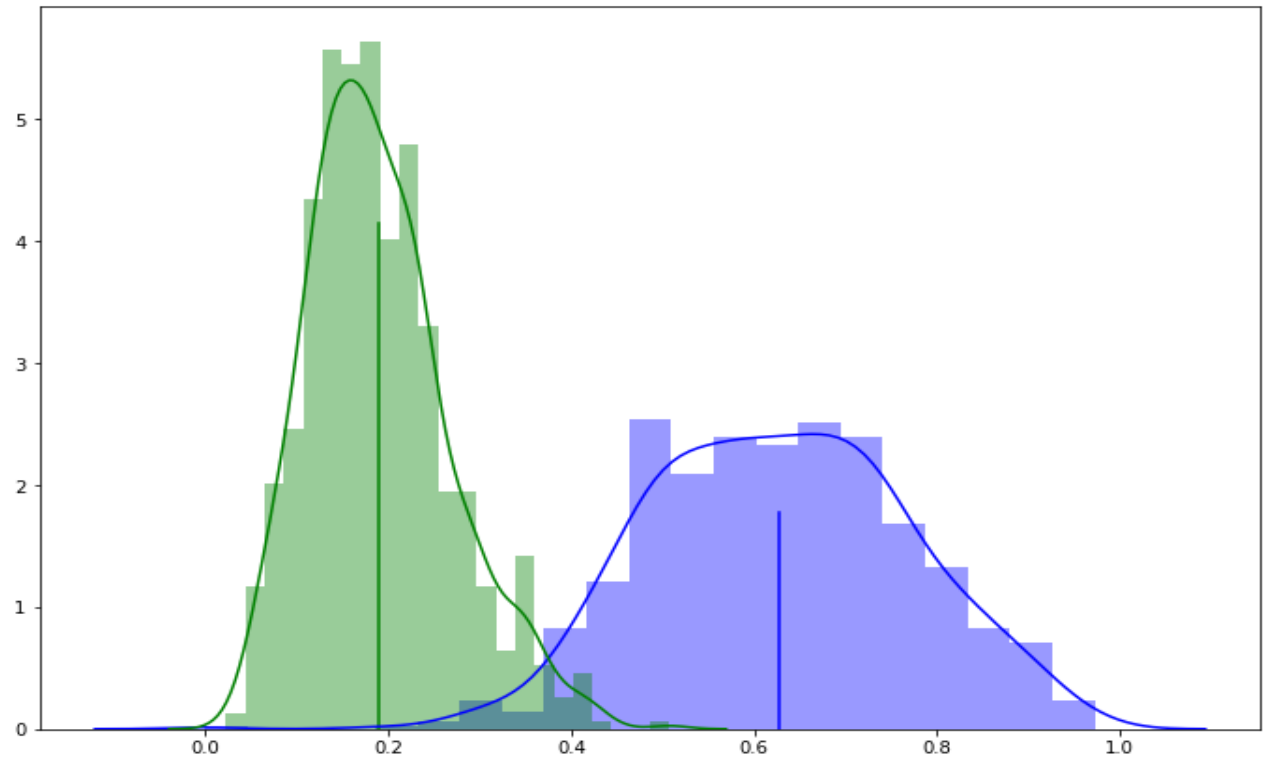
Distribution of Target Variable ‘Cnt’:



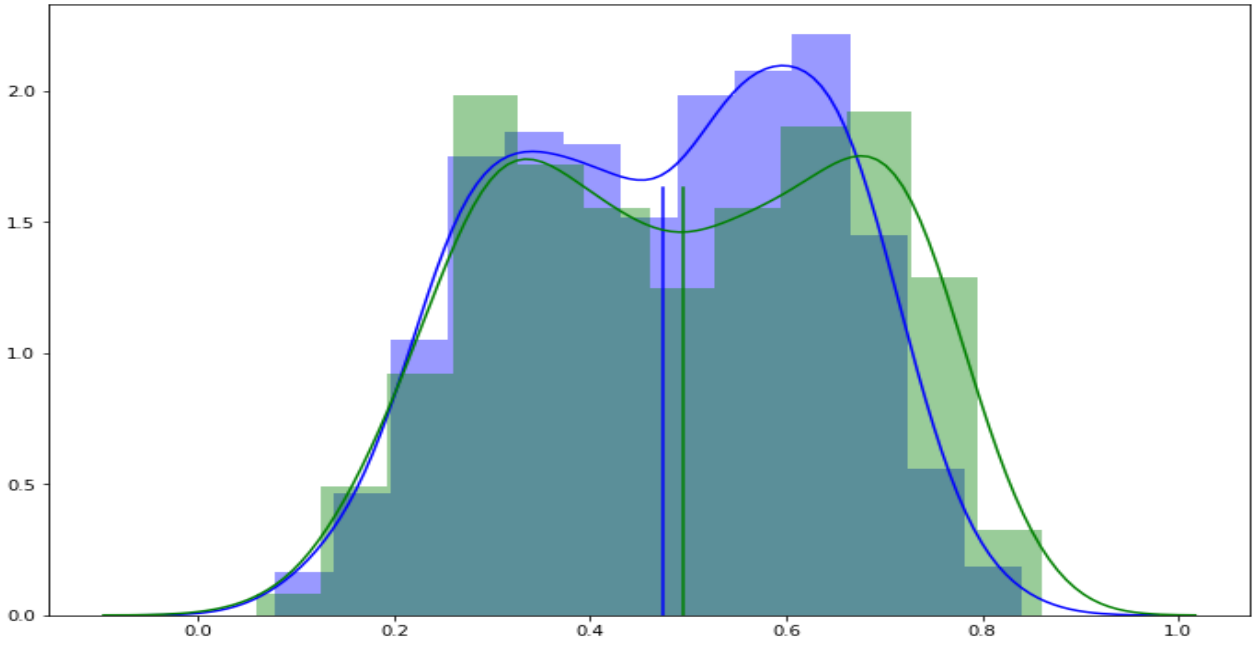
As per above histogram plot along with mean, ‘cnt’ target variable follows normal distribution approximately which tells that most of the value lies close to mean. It also indicates that ‘cnt’ variable has platykurtic curve with thin tail ends

Distribution of Independent Variable ‘hum’ and ‘windspeed’:

Above histogram indicates distribution for ‘hum’ in blue colour and ‘windspeed’ in green colour. Hum is approximately normally distributed with platykurtic curve while windspeed is rightly skewed with leptokurtic curve with heavy tail end distribution.



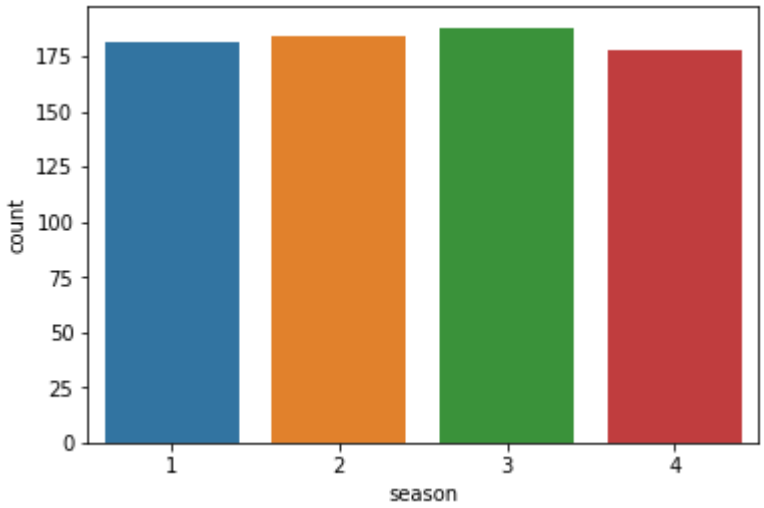
Distribution of Variable 'atemp', 'temp':



Above histogram indicates distribution for ‘atemp’ in blue colour and ‘temp’ in green colour. Temp follows bimodal normally distribution with platykurtic curve while windspeed is slightly left skewed with platykurtic curve with thin tail end distribution.

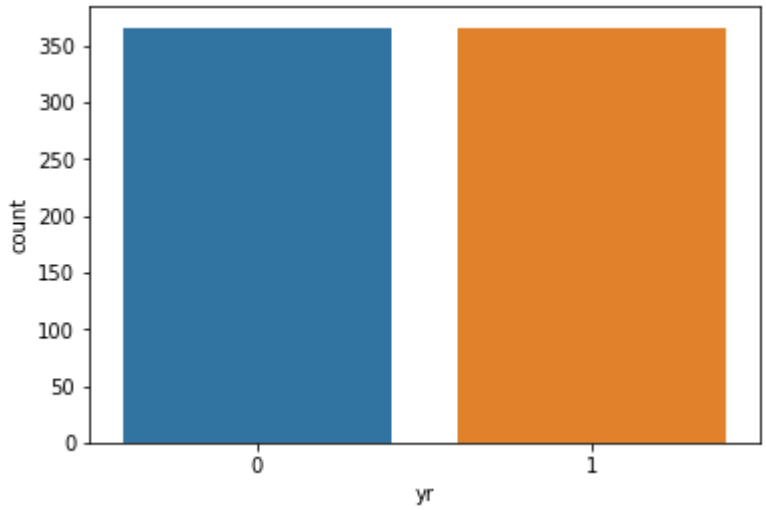
Frequency plot for Categorical Variables

Frequency Countplot for ‘season’ variable



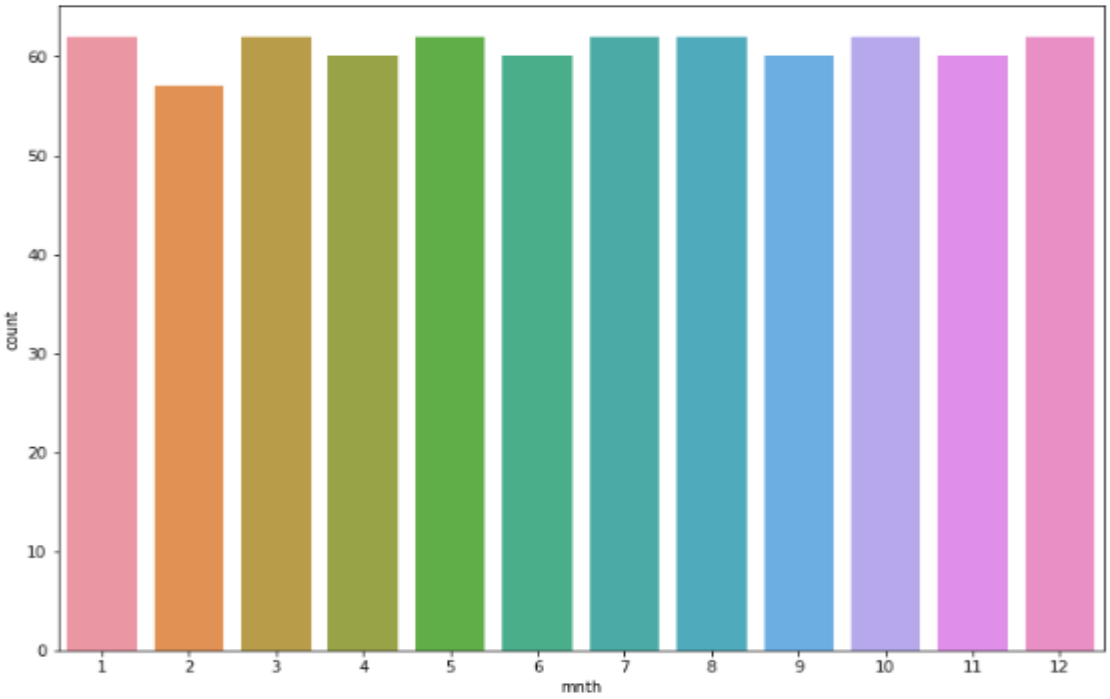
Above Bar plot indicates that Season variable is equally distributed among 4 categories i.e. Season (1: Spring, 2: Summer, 3: Fall, 4: Winter) has approximately similar number of records for every type. Thus, Bike rentals were approximately the same for irrespective of any season.

Frequency Countplot for yr variable



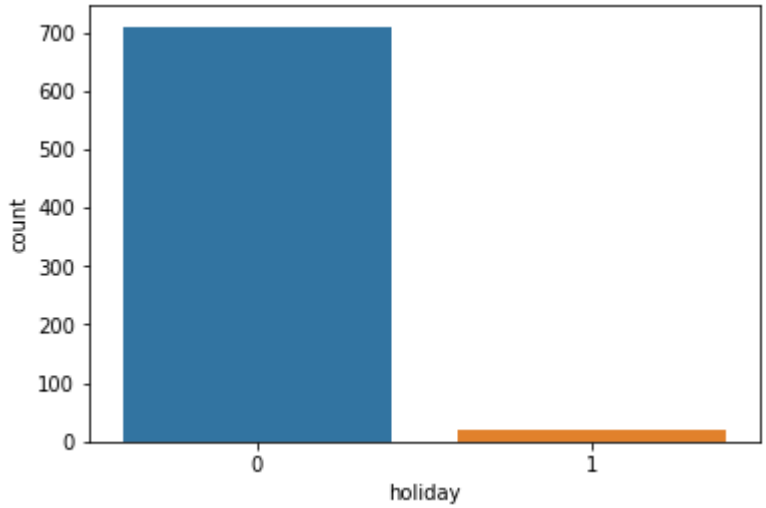
Above Bar plot indicates that yr variable is equally distributed among 2 categories i.e. Year (0: 2011, 1:2012) has equal number of records for every type. Thus, both the years experienced almost same number of bike rentals.

Frequency Countplot for mnth variable:



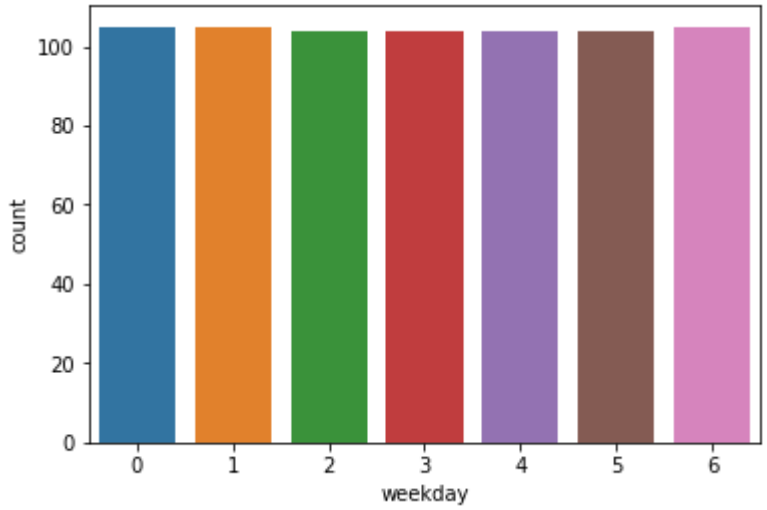
Above Bar plot indicates that month variable is equally distributed among 12 categories i.e. Month (1:12, JAN: DEC) has approximately similar number of records for every type. Thus, Bike rentals was similar for every month.

Frequency Countplot for holiday variable:



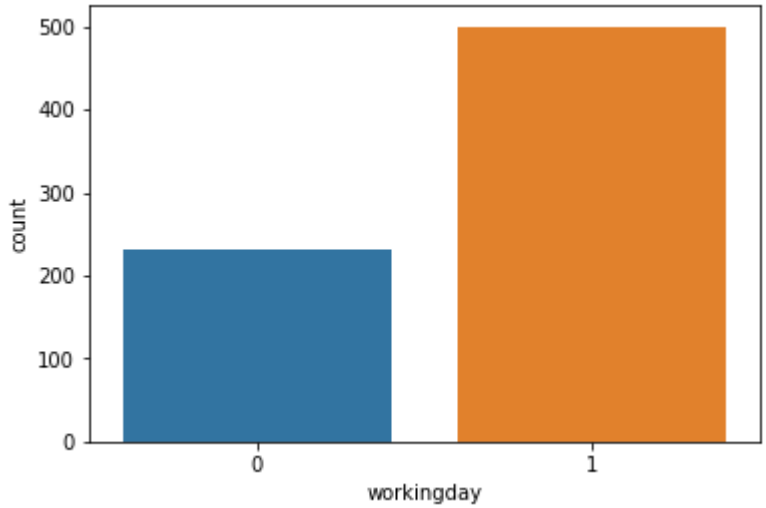
Above Bar plot indicates that our dataset contains almost 97% records with holiday variable as 0 (not holiday) and has very less records for days which are holiday.

Frequency Countplot for weekday variable:



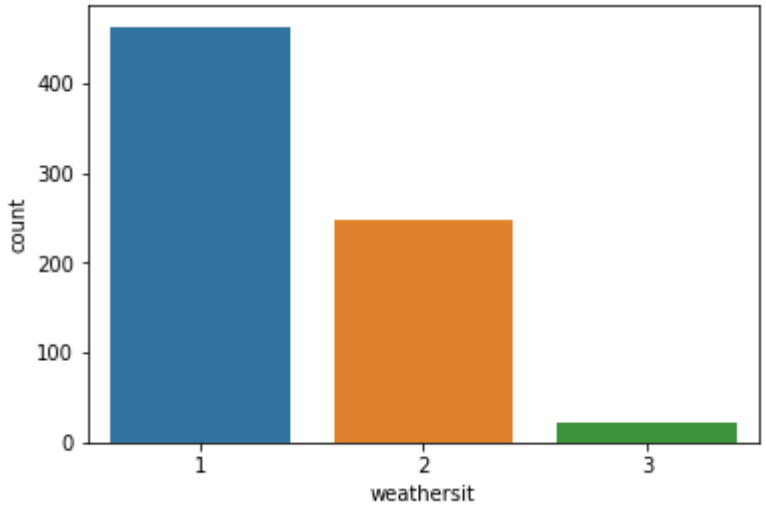
Above Bar plot indicates that weekday variable is equally distributed among 6 categories i.e. 0 to 6 (SUN to SAT) has almost equal number of records for every type. Thus, dataset consists of equal records for all days of the week for both years.

Frequency Countplot for workingday variable:



Above Bar plot indicates that our dataset contains almost 67% records with workingday variable as 1 (working day) and has very less records for days which are neither weekend nor holiday.

Frequency Countplot for weathersit variable:



Above Bar plot illustrates that dataset contains records mostly with weather type as 1 (Clear Sky) while weather type 2 (Misty) is the second most common weather type found in the dataset. Records having Weather Type 3 (Light Rain) are rarely found while Weather Type 4 (Heavy Rain) are not present which indicates that there were no bikes rented when there was heavy rain given any point of time.

Central Tendency Calculation

Mean and Median for Numerical Variables:

Given below is the mean and median for variables ‘hum’, ’windspeed’, ’temp’, ’atemp’, ’cnt’ which acts as a central measurement which can summarize the entire variable with a single number.

```
hum
Mean :  0.6278940629274962
Median :  0.626667

windspeed
Mean :  0.1904862116279068
Median :  0.180975

temp
Mean :  0.49538478850889184
Median :  0.49833299999999997

atemp
Mean :  0.47435398864569067
Median :  0.48673299999999997

cnt
Mean :  4504.3488372093025
Median :  4548.0
```

Mode for Categorical Variables:

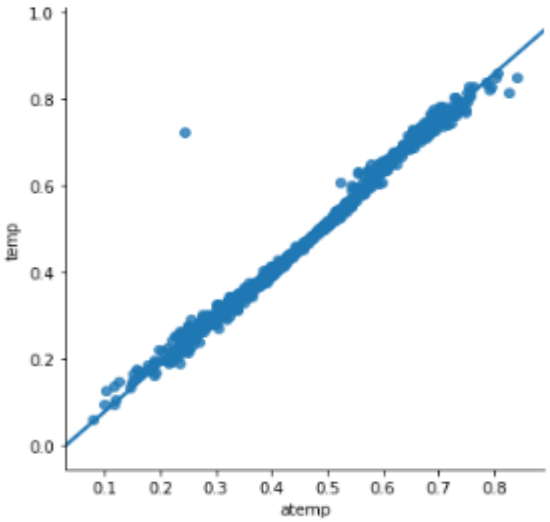
Given below is the mode for variables ‘season, yr, mnth, holiday, workingday, weathersit’ which is the most repeated value found in the entire variable distribution.

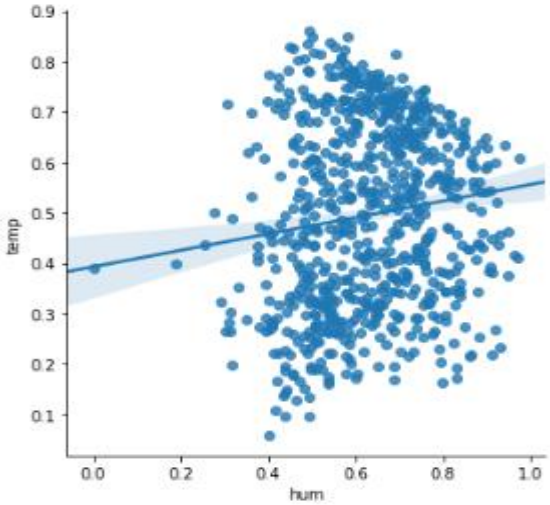
```
season
Mode :  3

yr
Mode :  1

mnth
Mode :  1

holiday
Mode :  0

weekday
Mode :  0

workingday
Mode :  1

weathersit
Mode :  1
```

Bivariate Analysis

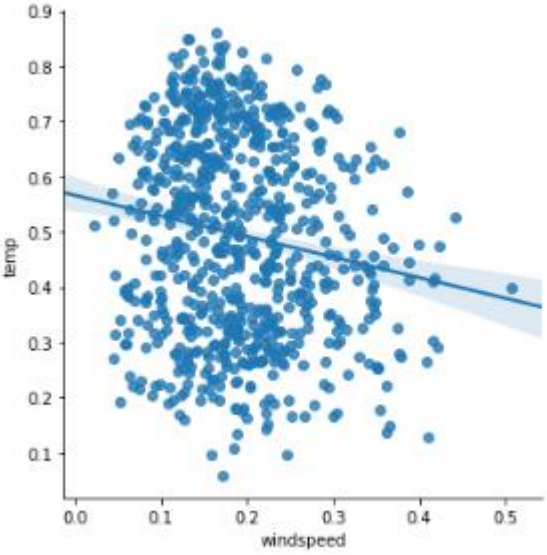
a) Numerical – Numerical

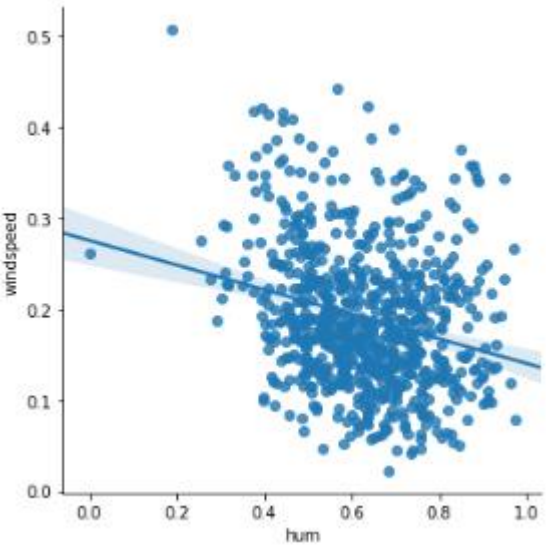
Given below is the scatter plot for numerical variables (‘temp’, ’atemp’, ’hum’, ’windspeed’) for bivariate numerical analysis which represents the relationship and correlation between both the numerical variables examined.





Variable ‘atemp’ and ‘temp’ display a strong positive correlation with each other and also their distribution is also similar, hence we need to check for multicollinearity problems among variables before modelling our linear regression equation for predicting bike rental counts.

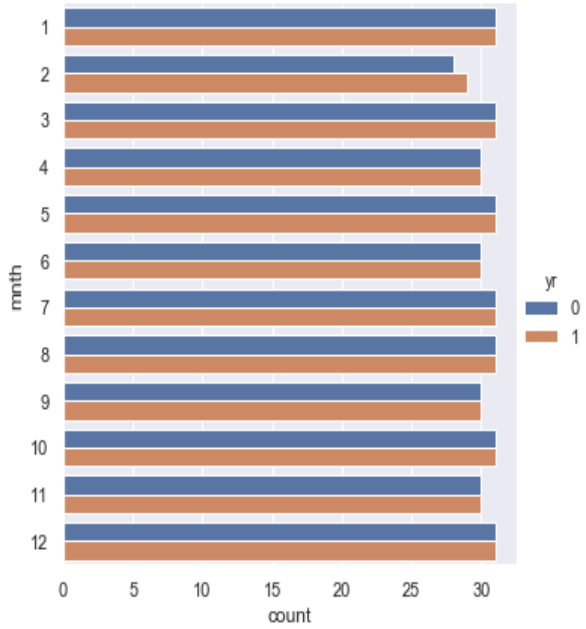


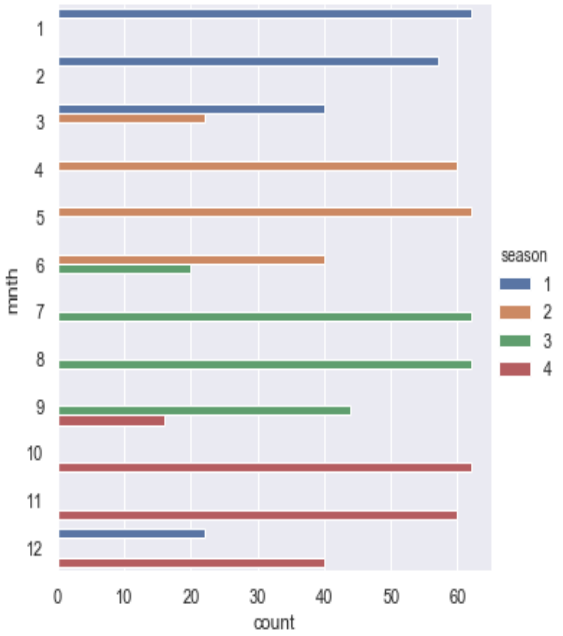


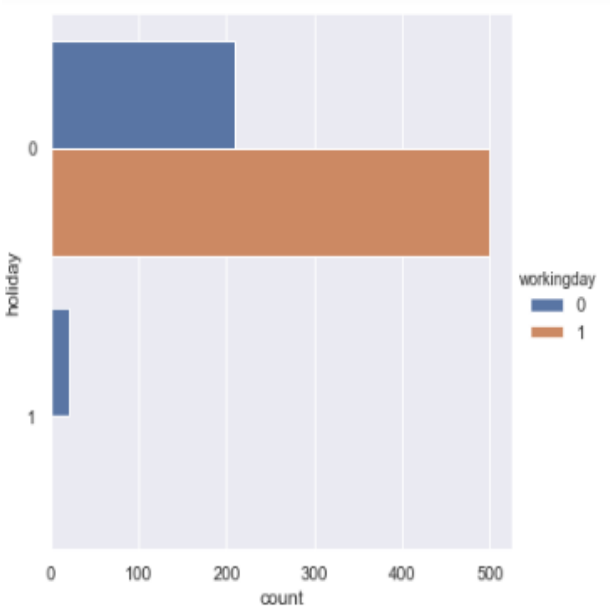
Other variables doesn’t show much correlation with each other which can affect the regression model and hence are independent of each other.

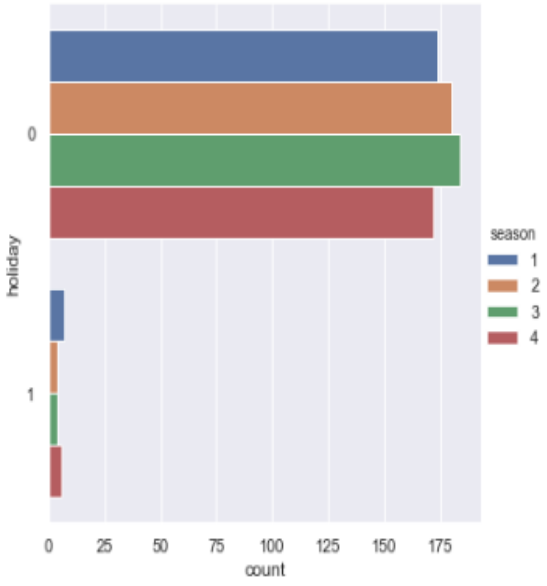
b) Categorical – Categorical

Given below is the Frequency Count plot for categorical variables (‘season, ’yr’, ’mnth’, ’weekday’, ’workingday’, ’holiday’) for bivariate categorical analysis which represents the frequency of categories distributed between both the categorical variables examined.

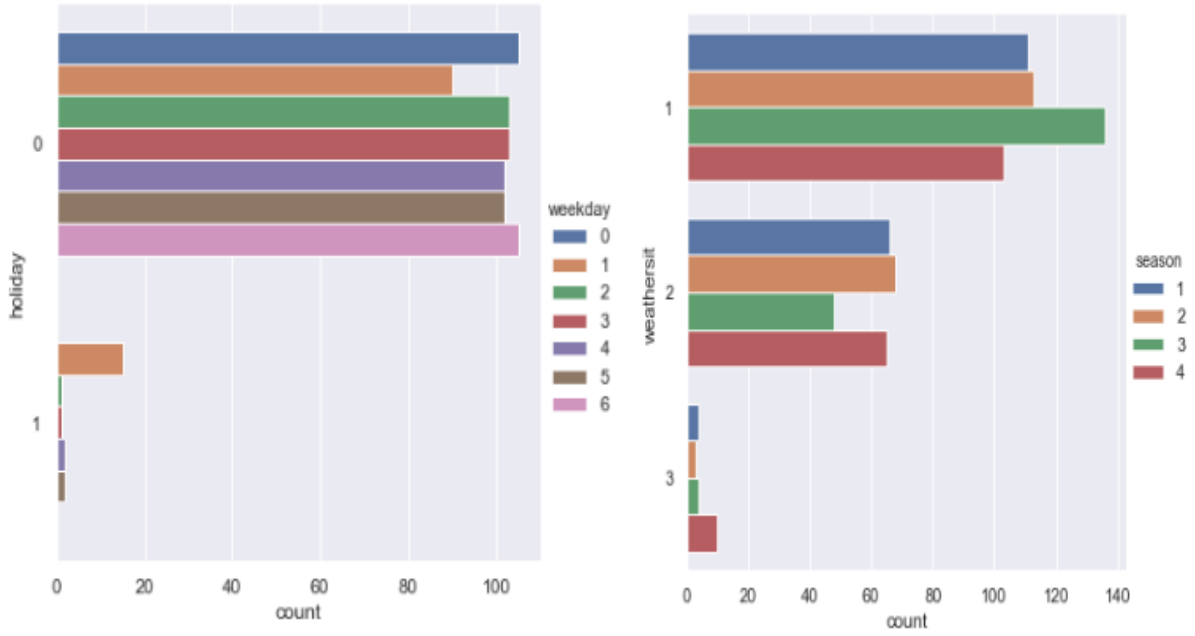








From above bar plots, we can get a clear idea that the bike rental frequency was similar for both the years (2011 and 2012) as well as for their months while there are very less records with holiday as 1 (i.e holidays for both the years) with respect to working day as well as season. The records are also normally distributed among seasons and also follow the monthwise pattern of seasons to months.



Here also, Records with holiday's as 1 are almost empty and doesn't show their presence in the nature. Hence we can make a assumption that the probability of bikes being rented on holidays is almost 0.1.

Also we come to know that there are no records present with weather type 4 (heavy rain) and also very less significance is given to the records with weather type 3 (Light Rain). We can conclude that the weather type 1 has most of the records in which the bikes are rented the most and also the probability assumption is far more superior for weather type 1 and 2 rather than 3 and 4.

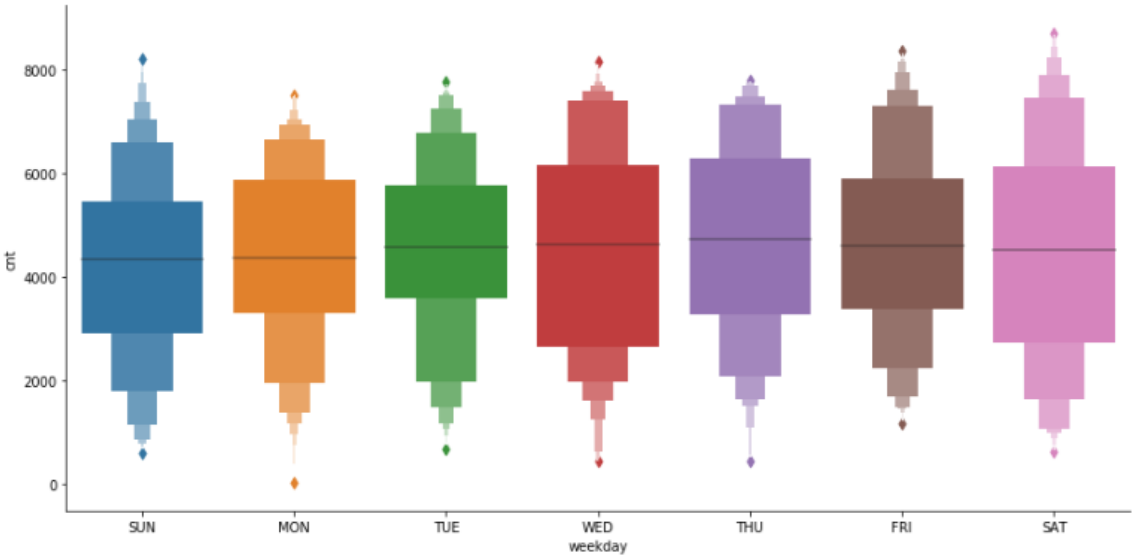
Conclusion : Weekday variable is clearly associated with holiday and working day while season variable is associated with mnth.

c) Numerical – Categorical

Given below is the combination of plots for examining relationship between categorical and numerical variables with our target dependant variable ‘cnt’ so that we can analyse which variables contribute the most in the prediction of our final target variable ‘cnt’ and can be used as important identifier in our modelling techniques.

Rental Count per Day (SUN to SAT):

In the below given boxen plot, we are analysing the count of bikes rented for every day of the week.

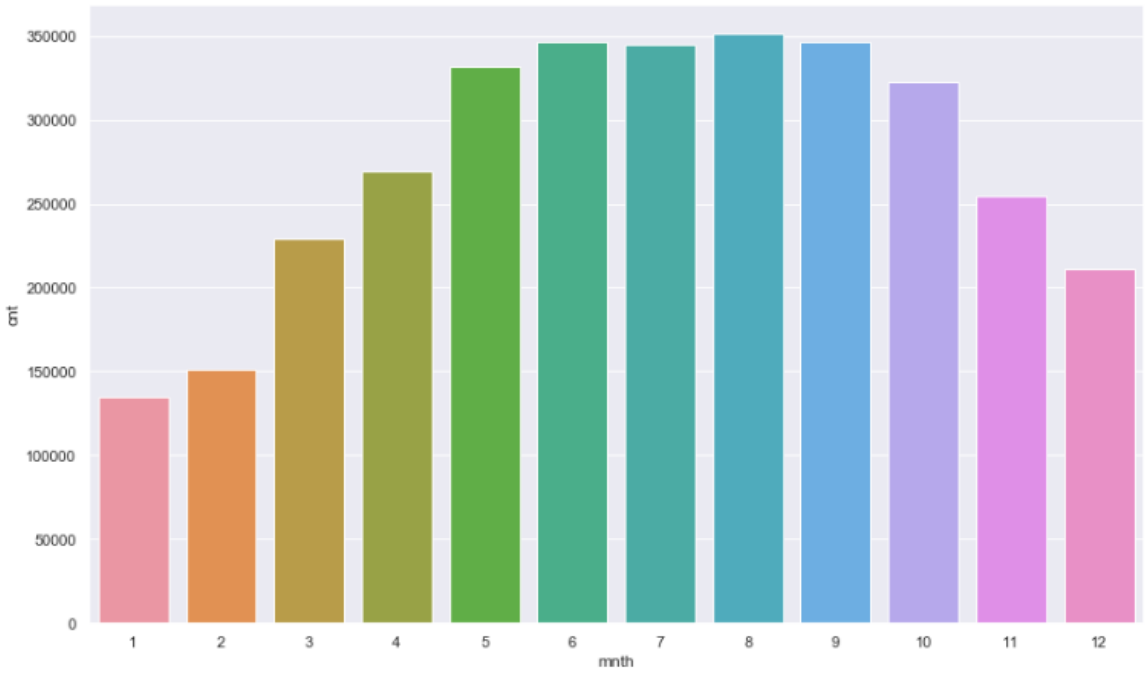


Observation: For all Days, The count is almost similar and uniform (4500~ approx.).

Conclusion: There were no unequal distributions of bikes rented with respect to any day of the week for both the years (2011 and 2012). Be it any day, we can safely assume that the bike rental counts were the same for that particular day.

Bike Rental Count on Monthly Basis (JAN to DEC) for both years:

In the below given bar plot, we are analysing the count of bikes rented in every month for years 2012 and 2011.

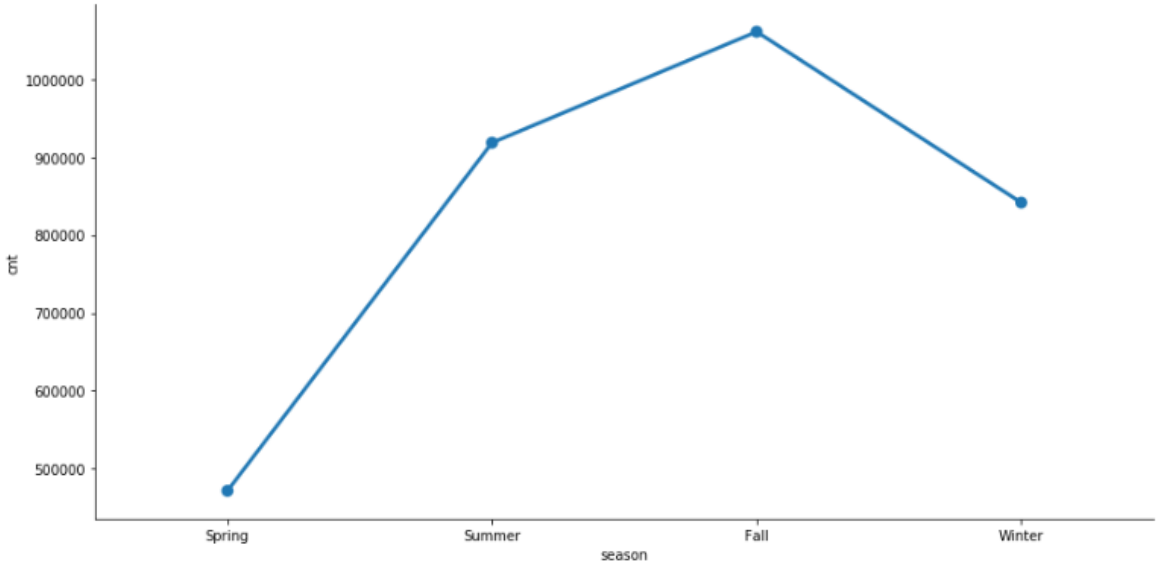


Observation: We can see that there were less bikes rented on the month of Jan, Feb (Start of year) and also the sales peak during the mid-year

Conclusion: As per above diagram, there is a difference in count of bikes rented with respect to the month we're analysing and as per monthwise, there is unequal distribution of cnt variable and the maximum number of bikes rented are observed during the months 'June, July, August and September' for both the years 2011 and 2012.

Bike Rental Count on Seasonal Basis:

In the below given line plot, we are analysing the count of bikes rented in every season for years 2012 and 2011.

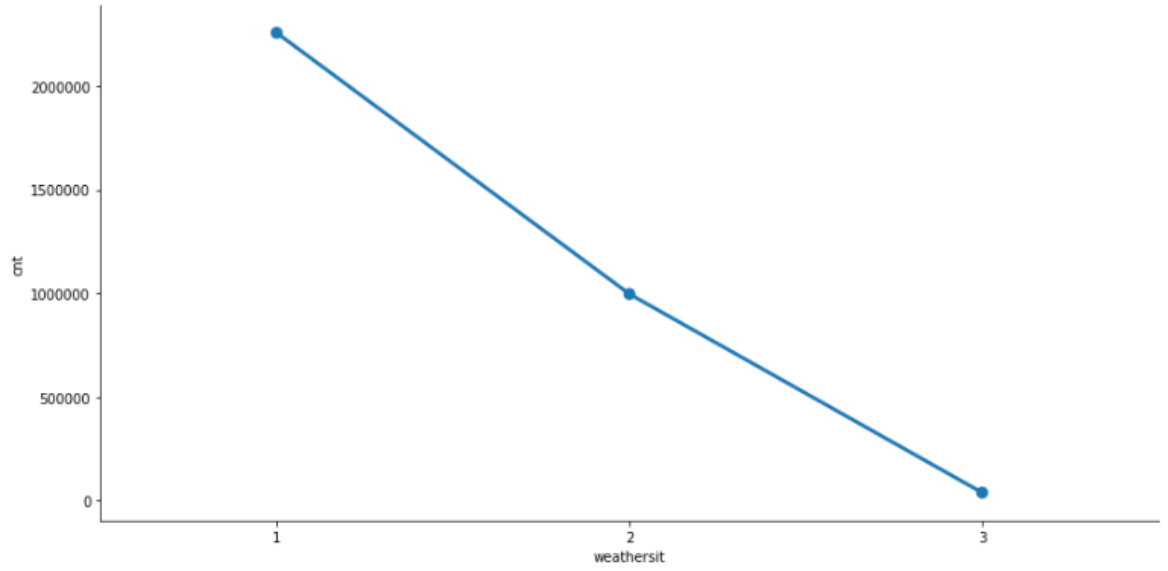


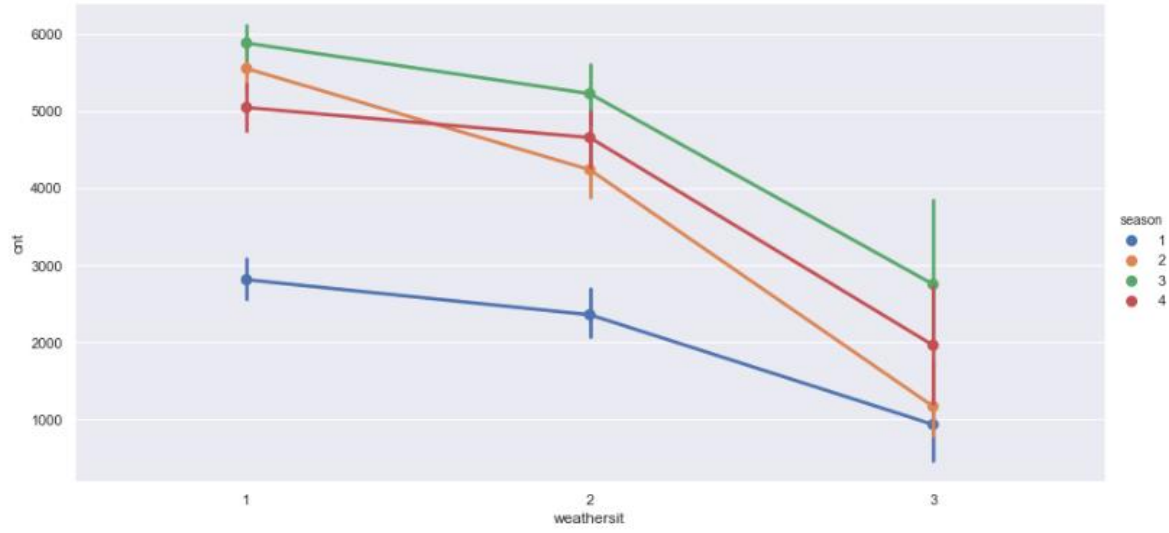
Observation: Bike Rental counts are very high during Fall Season and very low during Spring Season.

Conclusion: Counts for Bikes rented are significantly very high in Fall season accompanied by Summer and Winter which showcases comparatively less number of records, Also, count for Spring season is very lowest. Hence, we can safely assume that during middle of year (2011 and 2012) in fall season, maximum number of bike rentals were experienced by the system.

Bike Rental Count on basis of Weather Type:

In the below given line plot, we are analysing the count of bikes rented for every weather type for years 2012 and 2011.



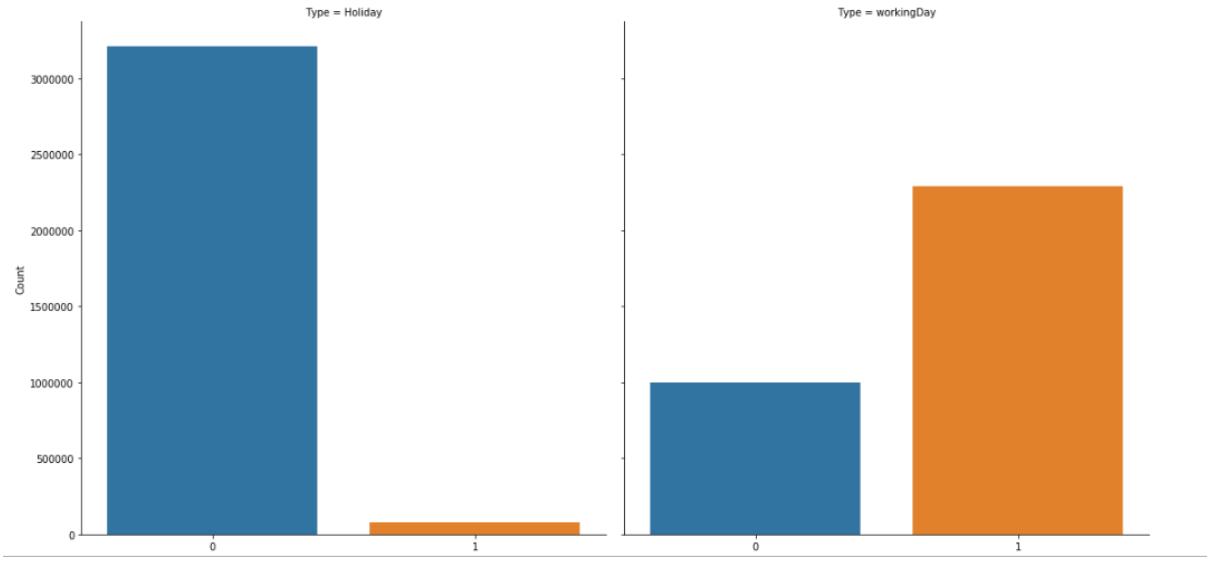


Observation: Bike Rental Counts are very high with Clear, Few clouds, partly cloudy, partly cloudy weather type and almost negligible with Light Snow, Light Rain +Thunderstorm. There are no bike rentals during heavy rain. Weather type 1 i.e. Clear Sky is majorly observed in fall season.

Conclusion: Bikes rented for Weather type 3 (Light Rain) are almost rarely found whereas for weather type 1 (Clear Sky) are mostly found. We can make an assumption that most bikes will be rented when there is clear sky or at least misty weather type. Also, Since Clear Sky and Misty Weather type is majorly observed in fall, summer and winter Season, number of bikes rented are also high in these seasons.

Bike Rental Count on basis of Day Type (Holiday/Working Day):

In the below given bar plots, we are analysing the count of bikes rented for every day (Holiday/Working) type for years 2012 and 2011.



Observation: Bikes are rented more on Non-Holiday's and comparatively more on working day (excluding weekends)

Conclusion: Bike rental counts are more observed on days which are neither weekend nor holiday, hence it is safe to assume that mostly bike rentals are observed during the non-holidays.

Data Pre-Processing

Missing Value Analysis

Missing values in data is a common phenomenon in real world problems. Knowing how to handle missing values effectively is a required step to reduce bias and to produce powerful models.

	Variables	Missing_percentage
0	dteday	0.0
1	season	0.0
2	yr	0.0
3	mnth	0.0
4	holiday	0.0
5	weekday	0.0
6	workingday	0.0
7	weathersit	0.0
8	temp	0.0
9	atemp	0.0
10	hum	0.0
11	windspeed	0.0
12	casual	0.0
13	registered	0.0
14	cnt	0.0

Our Dataset doesn’t have presence of any missing or null values as shown below, hence no need of Missing Value Imputation.

Feature Extraction

Feature Selection

The features in our data are important to the predictive models we use and will influence the results we are going to achieve. The quality and quantity of the features will have great influence on whether the model is good or not.

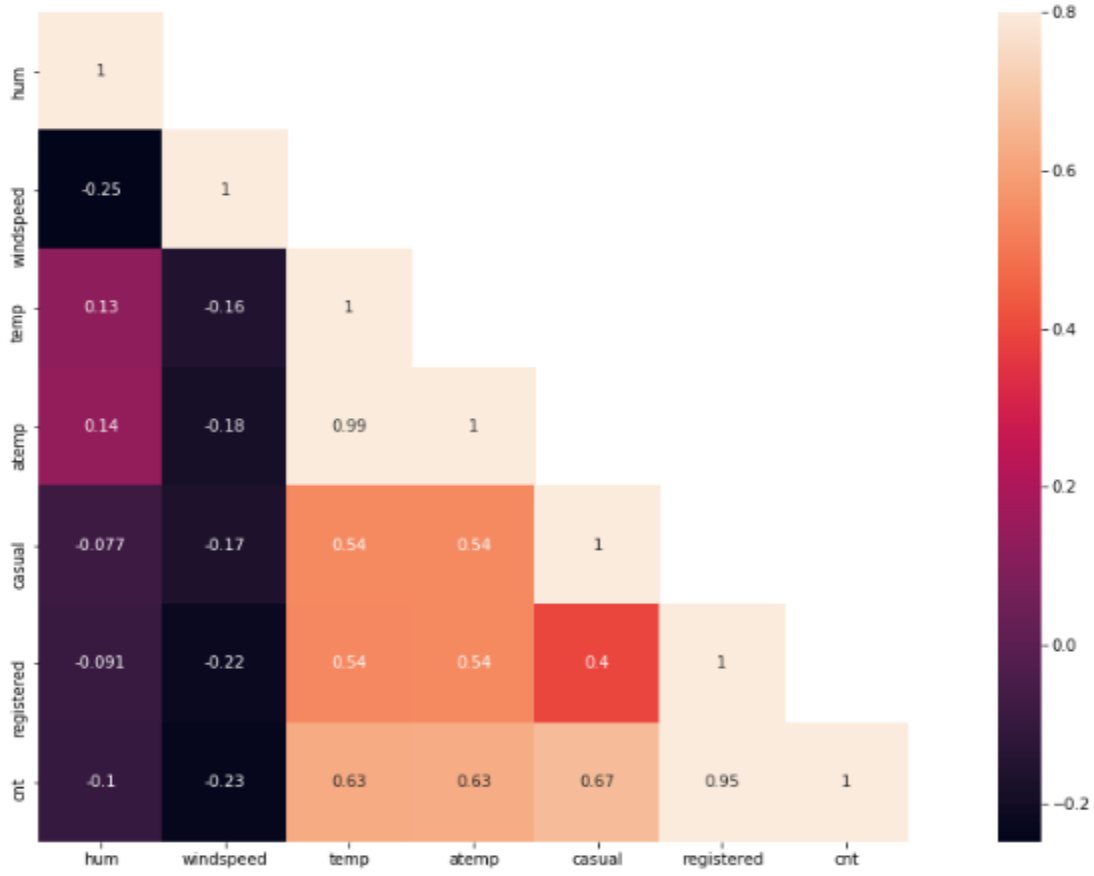
This becomes even more important when the number of features are very large. You need not use every feature at your disposal for creating an algorithm. You can assist your algorithm by feeding in only those features that are really important.

Since we're about to implement a Regression model, we've keep an eye for the below criteria for our features:

i. The relationship between two independent variable should be less and

ii. The relationship between Independent and Target variables should be high.

We can check for this criteria using Correlation plot given below:

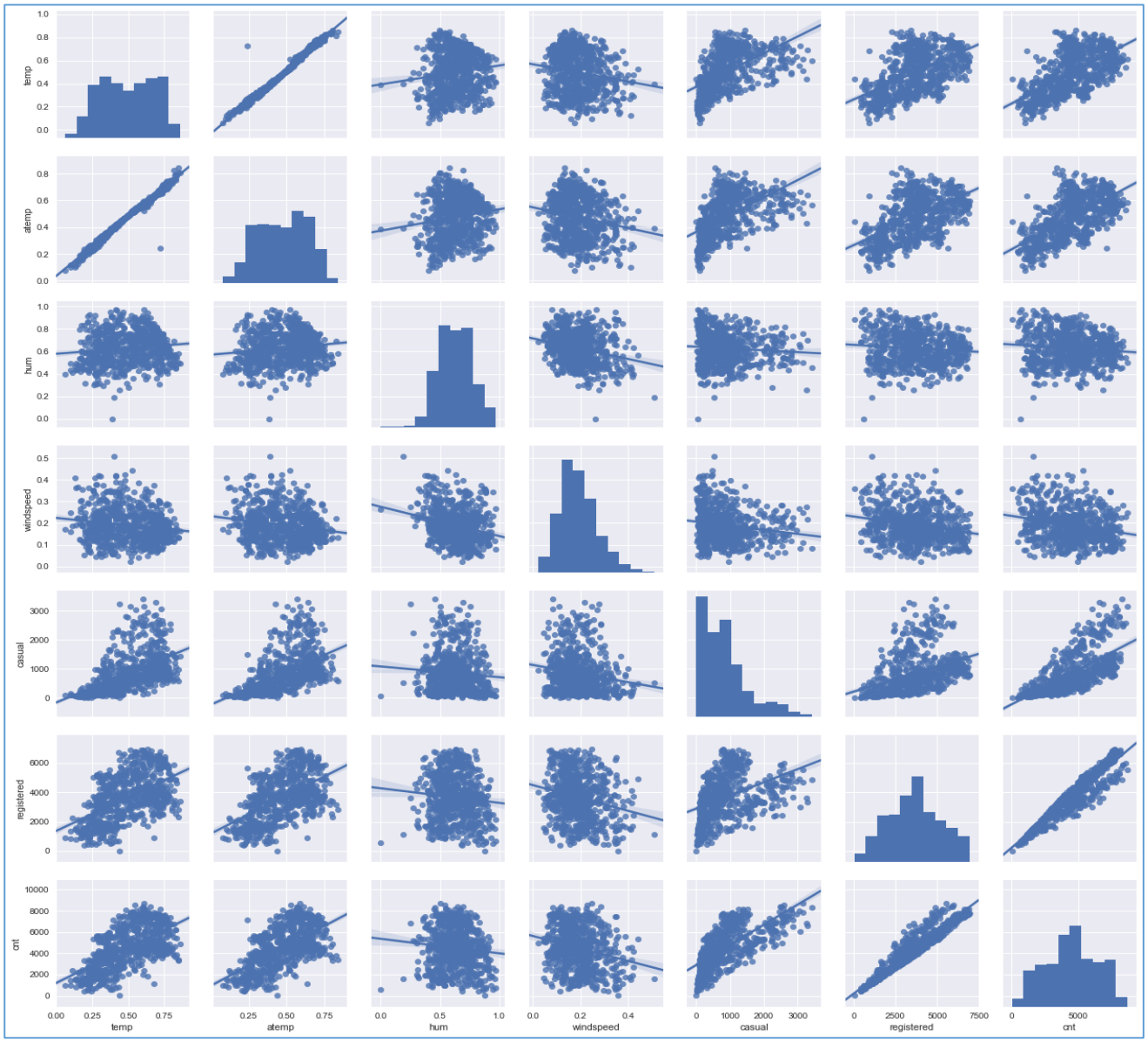


The above plot clearly shows that there is strong positive correlation among variables ‘temp’ and ‘atemp’ i.e. only one variable is enough for explaining both of them, hence we can safely remove one of them.

Also, it is also noted that variable ‘temp’ shows high correlation (0.63) with target variable ‘cnt’ which can be an important identifier while building the predictive model.

We will discard variables ‘casual’ and ‘registered’ from our analysis since we’re trying to predict values of those variables only (Since target variable ‘cnt’ is actually the sum of both of these variables).

For further deep understanding along with visualizations, we can use the below graph for a good grasp of which variables show strong correlation with each other:



Feature Scaling

Normalization is the process of reducing unwanted variation within or between variables.

Apart from the variables ‘casual’ and ‘registered’, every other variable is already in normalized form i.e. every variable is in a common scale of 0 to 1.

Since we are not dealing with independent variables ‘casual’ and ‘registered’, we will skip the feature scaling for our dataset as every variable is already scaled and is present in normalized form.

Dimension Reduction

As explained above, we will remove ‘atemp’ since it shows a lot of correlation with 'temp' variable.

Also, we will remove 'casual' and 'registered' since its aggregation sum results to 'cnt' and we have to predict 'cnt' using other important aspects.

Also, we do not need indexing column ‘instant’ for building our prediction model as it does not have any significance with respect to our dataset.

We also do not need Datetime variable 'dteday' since it doesn’t help in prediction and also we have covered all the necessary factors in 'season', 'yr', 'mnth'.

In all, we will be deleting the below listed columns as per our analysis so that we can build a better predictive model otherwise as per a basic Machine Learning rule, if we put garbage IN, we will get only garbage OUT. By garbage here, we mean noise in Data.

Columns to be dropped from dataset:

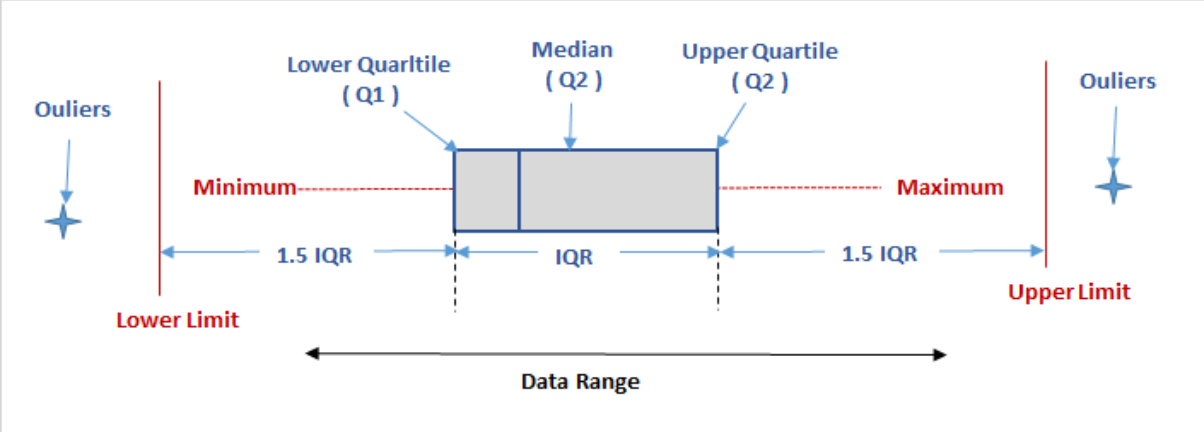
1) Dteday
2) Season
3) Yr
4) Mnth

Outlier Analysis

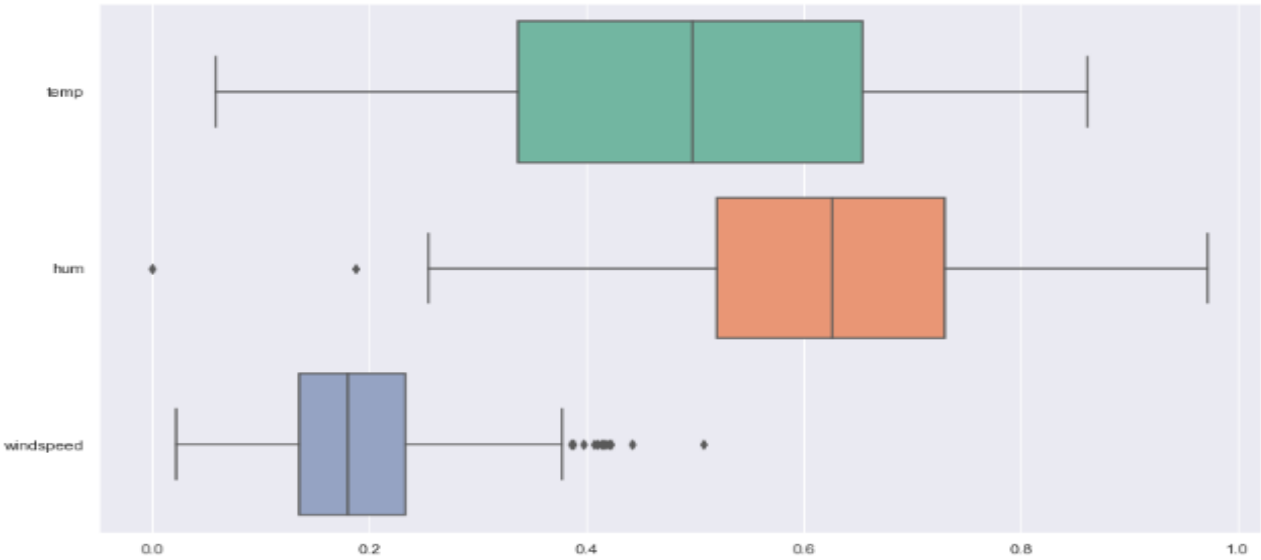
Outliers are extreme values that deviate from other observations on data, they may indicate a variability in a measurement, experimental errors or a novelty. In other words, an outlier is an observation that diverges from an overall pattern on a sample.

Outlier Detection

A data point that is distinctly separate from the rest of the data. One definition of outlier is any data point more than 1.5 interquartile ranges (IQRs) below the first quartile or above the third quartile.



We will now try to detect outliers for all numerical variables in our dataset using boxplot-IQR method:



Observation: There are outliers present in the 'hum' and 'windspeed' variable and hence we need to treat them.

Outlier Treatment with KNN Imputation

We can either delete these observations which contains outlier or we can replace them with null values and then impute with KNN so that we do not lose important data from which we can build a better predictive model.

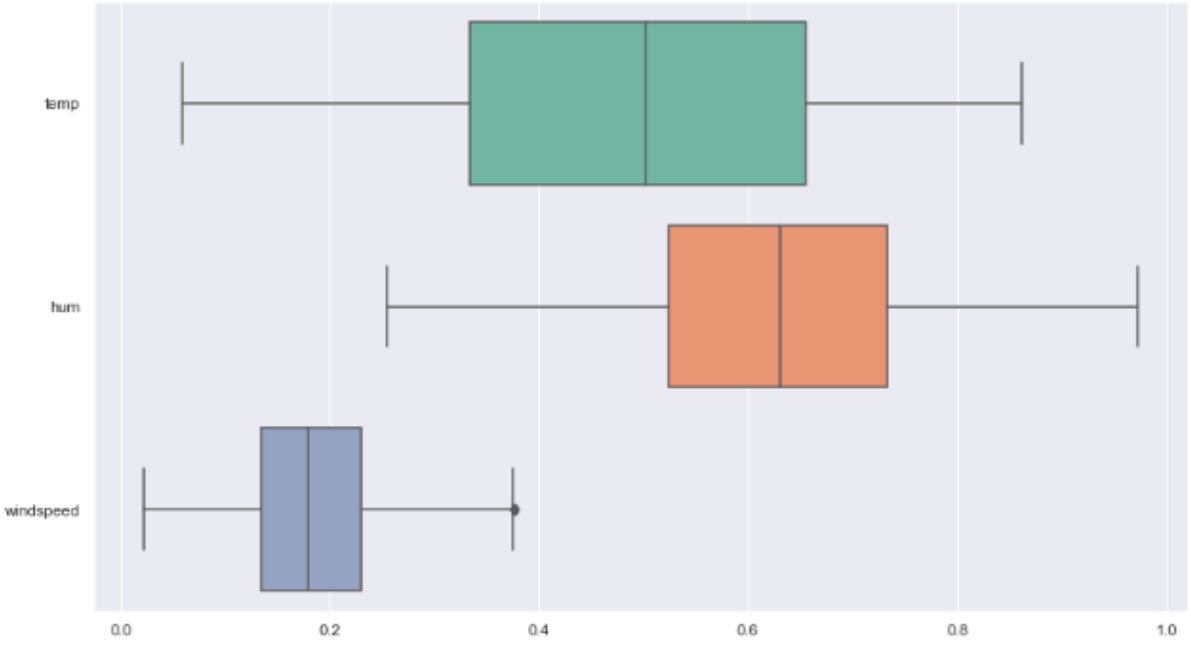
Thus, we will impute all the observations which contains outliers

```
# Columns to Check for outliers
cnames = ['hum','windspeed','temp']

for i in cnames:
    q75, q25 = np.percentile(bike_df.loc[:,i], [75,25])

    iqr = q75 - q25

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

    bike_df.loc[bike_df.loc[:,i] < min ,:i ] = np.nan
    bike_df.loc[bike_df.loc[:,i] > max ,:i ] = np.nan

## Calculating Missing values
#missing_val_boxplot = pd.DataFrame(bike_df.isnull().sum())
#print(missing_val_boxplot)

## KNN Apply for the dataset
bike_df = pd.DataFrame(KNN(k=3).complete(bike_df), columns=bike_df.columns)
```

KNN Output:

```
Imputing row 1/731 with 0 missing, elapsed time: 0.274
Imputing row 101/731 with 0 missing, elapsed time: 0.277
Imputing row 201/731 with 0 missing, elapsed time: 0.277
Imputing row 301/731 with 0 missing, elapsed time: 0.278
Imputing row 401/731 with 0 missing, elapsed time: 0.279
Imputing row 501/731 with 0 missing, elapsed time: 0.281
Imputing row 601/731 with 0 missing, elapsed time: 0.282
Imputing row 701/731 with 0 missing, elapsed time: 0.283
```

Boxplot Analysis after outliers are removed:



Encoding Categorical Variables

Encoding categorical variables is a must before we implement any machine learning model since many machine learning algorithms cannot operate on label data directly. They require all input variables and output variables to be numeric.

In general, this is mostly a constraint of the efficient implementation of machine learning algorithms rather than hard limitations on the algorithms themselves.

We will encode all our categorical variables for better performance and accuracy from our predictive model.

Columns after one hot encoding:

	yr	holiday	workingday	temp	hum	windspeed	season_1	season_2	season_3	season_4	...	mnth_9	mnth_10	mnth_11	mnth_12	weekday_1	weekday_2	weekday_3	weekday_4	weekday_5	weekday_6
0	0	0	0	0.344167	0.805833	0.160446	1	0	0	0	...	0	0	0	0	0	0	0	0	0	1
1	0	0	0	0.363478	0.696087	0.248539	1	0	0	0	...	0	0	0	0	0	0	0	0	0	0
2	0	0	1	0.196364	0.437273	0.248309	1	0	0	0	...	0	0	0	0	1	0	0	0	0	0
3	0	0	1	0.200000	0.590435	0.160296	1	0	0	0	...	0	0	0	0	0	1	0	0	0	0
4	0	0	1	0.226957	0.436957	0.186900	1	0	0	0	...	0	0	0	0	0	0	1	0	0	0
5	0	0	1	0.204348	0.518261	0.089565	1	0	0	0	...	0	0	0	0	0	0	0	1	0	0
6	0	0	1	0.196522	0.498696	0.168726	1	0	0	0	...	0	0	0	0	0	0	0	0	1	0
7	0	0	0	0.165000	0.535833	0.266804	1	0	0	0	...	0	0	0	0	0	0	0	0	0	1
8	0	0	0	0.138333	0.434167	0.361950	1	0	0	0	...	0	0	0	0	0	0	0	0	0	0
9	0	0	1	0.150833	0.482917	0.223267	1	0	0	0	...	0	0	0	0	1	0	0	0	0	0

As shown above, all the categorical variables are one hot encoded i.e. all the categorical feature columns are converted into binary class.

This binary transformation of categorical variables with unique column names will help our predictive model for better prediction and accuracy.

Modelling and Prediction

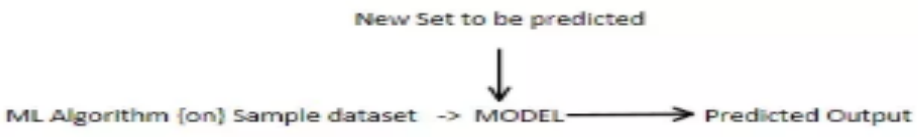
A machine learning model can be a mathematical representation of a real-world process. To generate a machine learning model you will need to provide training data to a machine learning algorithm to learn from.

In order to generate ML Model, we need:

1. Sample Data with target attribute given.
2. ML Algorithm chosen according to the nature of target attribute.

Process:

- Input the training dataset.
- Let the machine learning algorithm run on the data. [The algorithm now learns and captures the pattern in the data]
- Tune the parameters to control the learning of the algorithm. [To facilitate accuracy]
- After the algorithm finishes learning, the model is finally built.



Now, when a new dataset comes in for prediction, it is passed to the model. The model that is built by learning the past sample data, thus predicts the output.

Regression Models

Regression analysis is a form of predictive modelling technique which investigates the relationship between a **dependant** and **independent variable (s)**.

We've chosen regression since our dependant variable is a continuous and our independent variables are either continuous or categorical.

Also, we have to take into consideration that with respect to the analysis we have done until now, it's very clear that '**temp'** most contributing variable for our dependant variable 'cnt' as it has strong positive correlation of 0.63 with it. Hence it will play a key role in our model development.

In this project, we have implemented 4 different Machine learning Regression techniques to obtain the best model and accuracy.

Decision Tree

Decision tree is a flowchart like tree structure, where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (terminal node) holds a class label.

Random Forest

Random Forest is an ensemble technique capable of performing both regression and classification tasks with the use of multiple decision trees and a technique called **Bootstrap Aggregation**, commonly known as **bagging**.

Linear Regression

Linear regression performs the task to predict a dependent variable value (y) based on a given independent variable (x). So, this regression technique finds out a linear relationship between x (input) and y (output)

XGBoost

XGBoost is an implementation of gradient boosted decision trees designed for speed and performance that is dominative competitive machine learning.

Before implementing any Machine Learning module, we have to first divide our dataset into training set (used for training the model) and validation set (used for testing) so we can evaluate our model and find metrics like:

RMSE

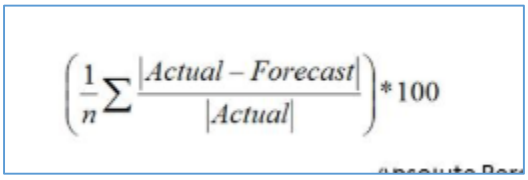
Root-Mean-Squared-Error (RMSE)
The most commonly used metric for regression tasks is RMSE (Root Mean Square Error). This is defined as the square root of the average squared distance between the actual score and the predicted score:

$$\text{rmse} = \sqrt{\frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n}}$$

Here, yi denotes the true score for the i-th data point, and ŷi^ denotes the predicted value.

MAPE

(Mean Absolute Percent Error) measures the size of the error in percentage terms. It is calculated as the average of the unsigned percentage error.



ACCURACY

Accuracy is the percentage of correct predictions to the total number of input samples.

Accuracy = 100 – MAPE

Decision Tree Regression

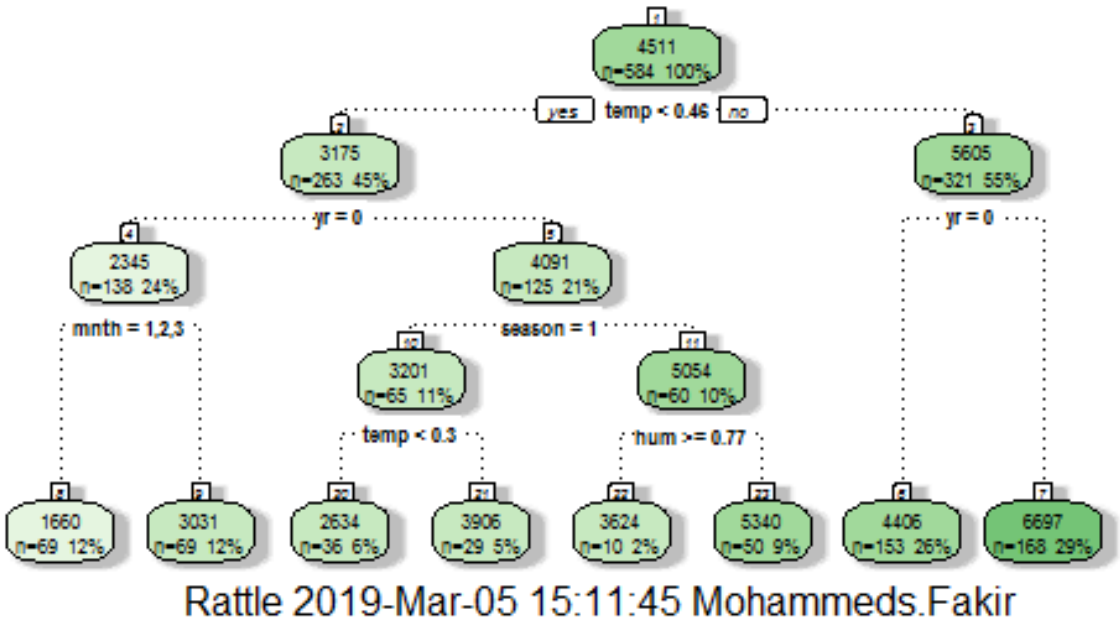
Decision tree regression observes features of an object and trains a model in the structure of a tree to predict data in the future to produce meaningful continuous output. Continuous output means that the output/result is not discrete, i.e., it is not represented just by a discrete, known set of numbers or values.

Decision Tree Algorithm:

```
In [156]: ## Build the Decision Tree
          from sklearn.metrics import accuracy_score
          C50_model = DecisionTreeRegressor(max_depth = 8, min_samples_split = 4).fit(X_train, Y_train)

In [157]: C50_model

Out[157]: DecisionTreeRegressor(criterion='mse', max_depth=None, max_features=None,
                    max_leaf_nodes=None, min_impurity_decrease=0.0,
                    min_impurity_split=None, min_samples_leaf=1,
                    min_samples_split=2, min_weight_fraction_leaf=0.0,
                    presort=False, random_state=None, splitter='best')
```

We've set the parameters like max_depth (The maximum depth of the tree) and min_samples_split (The minimum number of samples required to split an internal node) for decision tree for better accuracy.

Decision Tree Image:



As per the image above, the root node of the decision tree is the most important variable with highest correlation with target variable i.e. 'temp' and

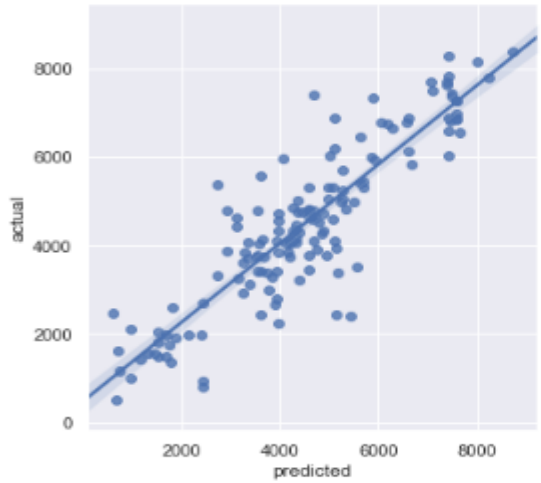
then later on conditional branches are constructed using variables like ‘yr’, ‘season’, ‘mnth’ and ‘hum’.

Using this tree, we will predict values for our testing set and validate the predicted outputs against the real values so that we can check evaluation metrics for our model.

Evaluation of Decision Tree Model:

```
#################### Error Metrics for Decision Tree ##############################
# Calcuate and display MAPE
mape_decision_tree = MAPE(Y_test, C50_reg_predictions)
print("MAPE : ",mape_decision_tree)

# Calcuate and display RMSE
rmse_decision_tree = RMSE(Y_test, C50_reg_predictions)
print("RMSE : ", rmse_decision_tree)

# Calculate and display accuracy
accuracy = ACCURACY(mape_decision_tree)
print('Accuracy:', round(accuracy, 2), '%.')

#Comparison Dataframe
decision_tree_eval = pd.DataFrame({'predicted':C50_reg_predictions, 'actual': Y_test})

#Scatter plot of Actual vs Predicted for Decision Tree
ax = sns.lmplot(x="predicted", y="actual", data=decision_tree_eval);
```

```
MAPE :  18.469143638840336
RMSE :  868.3361836818482
Accuracy: 81.53 %.
```



Evaluation:

Our Decision Tree Model has RMSE as 868 which is quite high and MAPE is also high with 18.46 which is clearly stating that our Decision Tree Model is over fitted and it’s working well for training data but won’t predict good for new set of data. To overcome this overfit we have to tune the model using Random Forest.

Accuracy of our decision Tree model is 81.53% which can also be increased if we obtain a more generalized and less over fitted model.

Random Forest Regression

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

Random forest functions in below way

i. Draws a bootstrap sample from training data.

ii. For each sample grow a decision tree and at each node of the tree

   a. Randomly draws a subset of mtry variable and p total of features that are available

   b. Picks the best variable and best split from the subset of mtry variable

   c. Continues until the tree is fully grown.

Since Decision tree Model is overfitting and its accuracy, MAPE and RMSE is also poor we will develop model using Random Forest which will correct the overfitting by using maximum variables for predicting the count values.

Random Forest Algorithm:

```
################################ Random Forest Implementation ###################################
from sklearn.ensemble import RandomForestRegressor
RF_model = RandomForestRegressor(n_estimators = 1000)
RF_model.fit(X_train, Y_train);
```

```
RF_model
```

```
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=1000, n_jobs=1,
           oob_score=False, random_state=None, verbose=0, warm_start=False)
```

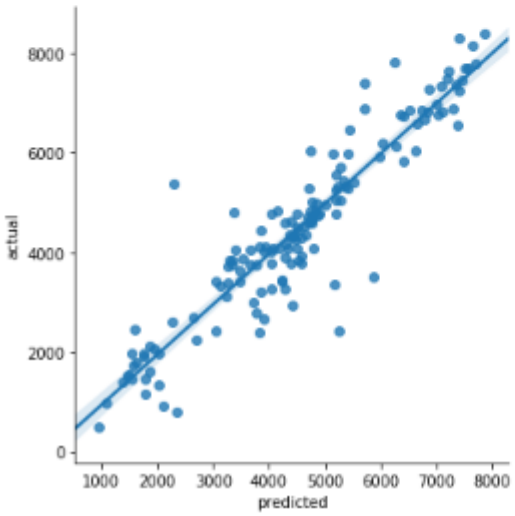
We've used 1000 Decision trees for predicting the final count of bikes rented using Random Forest which is a combination of all Decision trees created using random subset of dataset variables.

Random Forests use all the possible combinations of variables present in the dataset for building more accurate, generalized and less over fitted model.

Evaluation of Error Metrics for Random Forest:

```
############################ Error Metrics for Random Forest ################
# Calculate and display MAPE
mape_random_forest = MAPE(Y_test, RF_predictions)
print("MAPE : ",mape_random_forest)

# Calculate and display RMSE
rmse_random_forest = RMSE(Y_test, RF_predictions)
print("RMSE : ", rmse_random_forest)

#Calculate and display accuracy
accuracy = ACCURACY(mape_random_forest)
print('Accuracy:', round(accuracy, 2), '%.')

#Comparison Dataframe for Random Forest
randomforest_eval = pd.DataFrame({'predicted':RF_predictions, 'actual': Y_test})

#Scatter plot of Actual vs Predicted for Decision Tree
ax = sns.lmplot(x="predicted", y="actual", data=randomforest_eval);
```

```
MAPE :  14.212747895064357
RMSE :  681.269717488902
Accuracy: 85.79 %.
```



Evaluation:

As expected, Our Random Forest model has less MAPE (14.21) and RMSE (681.26) as compared to Decision Tree model as well as it has increased the accuracy (85.79%) of the prediction model by 4% which is quite remarkable.

With that said, random forests are a strong modelling technique and much more robust than a single decision tree. They aggregate many decision trees to limit overfitting as well as error due to bias and therefore yield useful results.

Multiple Linear Regression

Linear Regression establishes a relationship between **dependent variable (Y)** and one or more **independent variables (X)** using a **best fit straight line** (also known as regression line).

It is represented by an equation **Y = a + b*X + e**, where a is intercept, b is slope of the line and e is error term. This equation can be used to predict the value of target variable based on given predictor variable(s).

Important Points for MLR:

- There must be **linear relationship** between independent and dependent variables
- Multiple regression suffers from **multicollinearity, autocorrelation, heteroscedasticity**.
- Linear Regression is very sensitive to **Outliers**. It can terribly affect the regression line and eventually the forecasted values.
- Multicollinearity can increase the variance of the coefficient estimates and make the estimates very sensitive to minor changes in the model. The result is that the coefficient estimates are unstable
- In case of multiple independent variables, we can go with **forward selection**, **backward elimination** and **step wise approach** for selection of most significant independent variables.

Multicollinearity Check for Numerical Variables:

In statistics, **multicollinearity** (also collinearity) is a phenomenon in which one predictor variable in a multiple regression model can be linearly predicted from the others with a substantial degree of accuracy.

```
No variable from the 4 input variables has collinearity problem.

The linear correlation coefficients ranges between:
min correlation ( hum ~ temp ):  0.1228359
max correlation ( cnt ~ temp ):  0.627494

---------- VIFs of the remained variables --------
  Variables      VIF
1      temp 1.775098
2       hum 1.163715
3 windspeed 1.124723
4       cnt 1.872105
```

Observation: After removal of ‘atemp’ variable, our dataset’s numerical variable doesn’t show any multicollinearity problems.

Multiple Linear Regression Algorithm:

```
############################### Linear Regression ##########################
# Fitting Multiple Linear Regression in the Training Set
from sklearn.linear_model import LinearRegression
linear_regressor = LinearRegression()
linear_regressor = linear_regressor.fit(X_train , Y_train) #Linear_Regression.
```

```
linear_regressor
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)
```

Below given are the coefficients of Independent variables (slope (m) of the regression line) which are used for predicting the target variable.

	Names	Coefficients
0	yr	1988.038039
1	holiday	-752.820463
2	workingday	-127.216356
3	temp	4729.747114
4	hum	-1530.268933
5	windspeed	-3332.685656
6	season_1	-630.640945
7	season_2	237.523009
8	season_3	-125.538537
9	season_4	721.790501
10	mnth_2	277.832837
11	mnth_3	570.225908
12	mnth_4	536.366498
13	mnth_5	708.165094
14	mnth_6	624.771598
15	mnth_7	216.974519
16	mnth_8	728.334124
17	mnth_9	1234.086741
18	mnth_10	842.233027
19	mnth_11	225.884524
20	mnth_12	177.178660
21	weekday_1	317.986214
22	weekday_2	375.891621
23	weekday_3	426.639023
24	weekday_4	492.190627
25	weekday_5	477.462529
26	weekday_6	444.802041
27	weathersit_1	960.419832
28	weathersit_2	492.937258
29	weathersit_3	-1250.223063

Linear Regression Summary:

```
############################## Tuning Linear Regression ##################################

### New Ordianry Least Squared Object for creating Linear Regressor from statsmodels
import statsmodels.formula.api as asm
## Building Model with all predictors


X_opt = X
X_opt = np.array(X_opt, dtype=float)
Y = np.array(Y, dtype=float)

regressor_OLS = asm.OLS(endog = Y, exog=X_opt).fit()
regressor_OLS.summary()
```

OLS Regression Results

Dep. Variable:	y	**R-squared:**	0.976
Model:	OLS	**Adj. R-squared:**	0.975
Method:	Least Squares	**F-statistic:**	981.4
Date:	Tue, 05 Mar 2019	**Prob (F-statistic):**	0.00
Time:	21:58:57	**Log-Likelihood:**	-5886.8
No. Observations:	731	**AIC:**	1.183e+04
Df Residuals:	702	**BIC:**	1.196e+04
Df Model:	29		
Covariance Type:	nonrobust		

Linear Regression Evaluation:

```
#Predict using Linear Regressor
OLS_predictions = regressor_OLS.predict(X_test)

# Calculate and display MAPE
mape_OLS = MAPE(Y_test, OLS_predictions)
print("MAPE : ",mape_linear)

# Calculate and display RMSE
rmse_OLS = RMSE(Y_test, OLS_predictions)
print("RMSE : ", rmse_linear)

# Calculate and display accuracy
accuracy = 100 - np.mean(mape_OLS)
print('Accuracy:', round(accuracy, 2), '%.')
```

```
MAPE :  20.749633184576798
RMSE :  872.0697751160301
Accuracy: 80.97 %.
```

Linear Regression model displays slightly less accuracy of approximately 80.97% as compared to Decision Tree and Random Forest. Also, RMSE and MAPE is also higher as compared to both the Algorithms.

XGBoost Algorithm

XGBoost is a scalable and accurate implementation of gradient boosting machines and it has proven to push the limits of computing power for boosted trees algorithms as it was built and developed for the sole purpose of model performance and computational speed. Specifically, it was engineered to exploit every bit of memory and hardware resources for tree boosting algorithms.

XGBoost is capable to build random forests very fast than traditional methods and is memory efficient and gives high accuracy.

The implementation of XGBoost offers several advanced features for model tuning, computing environments and algorithm enhancement. It is capable of performing the three main forms of gradient boosting (Gradient Boosting (GB), Stochastic GB and Regularized GB) and it is robust enough to support fine tuning and addition of regularization parameters.

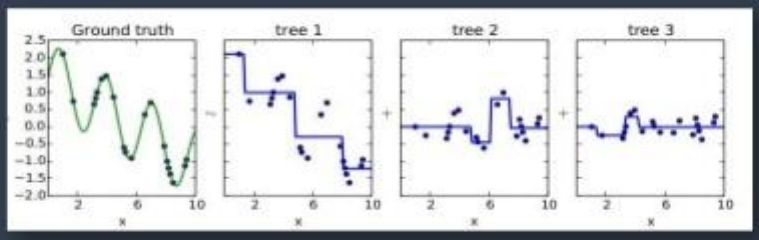
How XGBoost Works:

XGBoost is an ensemble method that seeks to create a strong classifier (model) based on “weak” classifiers. In this context, weak and strong refer to a measure of how correlated are the learners to the actual target variable. By adding models on top of each other iteratively, the errors of the previous model are corrected by the next predictor, until the training data is accurately predicted or reproduced by the model.

Now, gradient boosting also comprises an ensemble method that sequentially adds predictors and corrects previous models. However, instead of assigning different weights to the classifiers after every iteration, this method fits the new model to new residuals of the previous prediction and then minimizes the loss when adding the latest prediction. So, in the end, you are updating your model using gradient descent and hence the name, gradient boosting. This is supported for both regression and classification problems. XGBoost specifically, implements this algorithm for decision tree boosting with an additional custom regularization term in the objective function.

XGBoost explained in 2 pics (2/2)

Gradient boosting on CART



- One more tree = loss mean decreases = more data explained
- Each tree captures some parts of the model
- Original data points in tree 1 are replaced by the loss points for tree 2 and 3

XGBoost explained in 2 pics (1/2)

Classification And Regression Tree (CART)

Decision tree is about learning a set of rules:

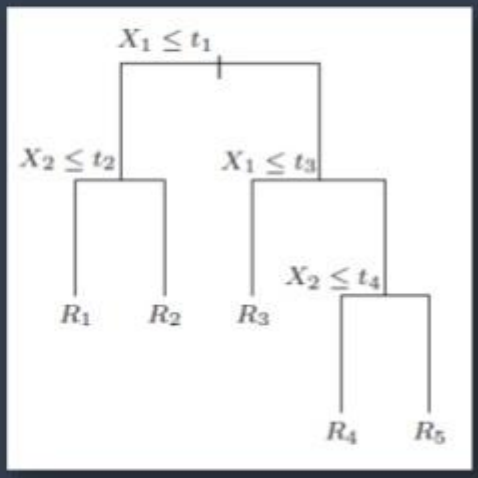
if$(X_1 \leq t_1)$ & if$(X_2 \leq t_2)$ then R_1
if$(X_1 \leq t_1)$ & if$(X_2 > t_2)$ then R_2
...

Advantages:

- Interpretable
- Robust
- Non linear link

Drawbacks:

- Weak Learner ☹
- High variance



Gradient Boosting

Iteratively learning weak classifiers with respect to a distribution and adding them to a final strong classifier.

- Each round we learn a new tree to approximate the negative gradient and minimize the loss

Whole model prediction $\hat{y}_i^{(t)} = \hat{y}_i^{(t-1)} + f_t(x_i)$ Tree t prediction

- Loss:

$$Obj^{(t)} = \left(\sum_{i=1}^{n} \ell(y_i, \hat{y}^{(t-1)} + f_t(x_i))\right) + \Omega(f_t)$$

Complexity cost by introducing additional tree

Friedman, J. H. (March 1999) *Stochastic Gradient Boosting.*

XGBoost Algorithm Implementation:

```
# Let's try XGboost algorithm to see if we can get better results
import xgboost
xgb = xgboost.XGBRegressor(n_estimators=1000, booster='gbtree', n_jobs=-1, learning_rate=0.05, gamma=0, subsample=0.75,
                           colsample_bytree=1, max_depth=7, nthread=3)
```

```
xgb.fit(X_train,Y_train)
```

```
XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1,
       colsample_bytree=1, gamma=0, learning_rate=0.05, max_delta_step=0,
       max_depth=7, min_child_weight=1, missing=None, n_estimators=1000,
       n_jobs=-1, nthread=3, objective='reg:linear', random_state=0,
       reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=None,
       silent=True, subsample=0.75)
```

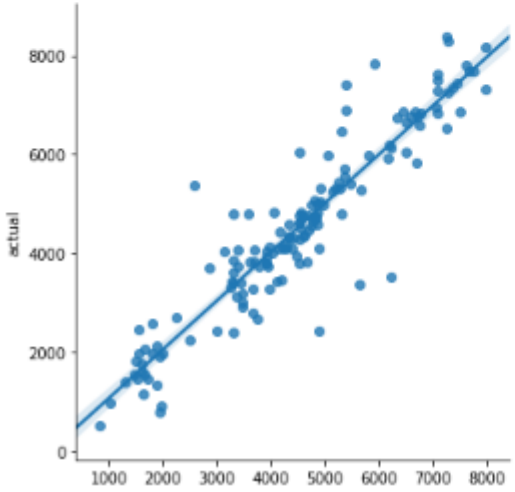
Parameter Explanation:

- Gamma = A node is split only when the resulting split gives a positive reduction in the loss function. Gamma specifies the minimum loss reduction required to make a split.
- N_Estimators = number of trees to be constructed.
- Booster = type of model to run at each iteration
- Subsample = Denotes the fraction of observations to be randomly samples for each tree. Lower values make the algorithm more conservative and prevents overfitting but too small values might lead to under-fitting.
- Learning_Rate = Rate at which model learns from data. Makes the model more robust by shrinking the weights on each step
- Max_Depth = maximum depth of a tree, Used to control over-fitting as higher depth will allow model to learn relations very specific to a particular sample.
- ColSample_by_tree = Denotes the fraction of columns to be randomly samples for each tree.
- nThread = This is used for parallel processing and number of cores in the system should be entered
- objective = type of algorithm to be used, ‘reg:linear’ in our case

XGboost Evaluation Metrics:

```
############################## Error Metrics for XGBoost ##########################
# Calculate and display MAPE
mape_xgb = MAPE(Y_test, xgb_predictions)
print("MAPE : ",mape_xgb)

# Calculate and display RMSE
rmse_xgb = RMSE(Y_test, xgb_predictions)
print("RMSE : ", rmse_xgb)

# Calculate and display accuracy
accuracy_xgb = 100 - np.mean(mape_xgb)
print('Accuracy:', round(accuracy_xgb, 2), '%.')

#Comparison Dataframe for MultiLinear OLS regression
xgboost_eval = pd.DataFrame({'predicted':xgb_predictions, 'actual': Y_test})

#Scatter plot of Actual vs Predicted for MultiLinear OLS regression
ax = sns.lmplot(x="predicted", y="actual", data=xgboost_eval)
```

```
MAPE :  12.927487078677105
RMSE :  682.2968442684705
Accuracy: 87.07 %.
```



As expected, XGBoost Gives the Highest Accuracy with 87.07% and also MAPE and RMSE is also the lowest among all the previously used Algorithms.

Conclusion:

We have performed Exploratory Data Analysis, Feature Engineering on our dataset and analyzed for relationship among dependent and independent variables of bikes rented dataset.

As per our analysis and predictive model, we have successfully examined all four regression models and among them we can deploy XGBoost Model which showcases highest accuracy and lowest MAPE, RMSE for predicting future bike rental count.