PROJECT-1

Predication of bike rental count on daily based on the environmental and seasonal settings

Jitendra Singh
jeetustar@gmail.com

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION

1.1 PROBLEM STATEMENT

The objective of this Case is to Predication of bike rental count on daily based on 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, Uber Bikes, Rapido etc. And these bike rentingcompanies deliver services to lakhs of customers daily. Now it becomes really important to manage theirdata properly to come up with new business ideas to get best results. In this case we have to identify in which days there can be most demand, such that we have enough strategies met to deal with such demand.

1.2 DATA

The goal is to build regression models which will predict the number of bikes used based on the environmental and season behavior. Given below is a sample of the data set that we are using to predict the number of bikes. The given dataset contains 16 variables and 731 observations. The "cnt" is the target variable and remaining all other variables are the independent variables. See the detail of the data in following image.

	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 1.1: Bike Count Sample Data

As you can see in the below table we have the following 13 variables, which we used to predict the count of bikes:

S.No	**Variables**
1	Instant
2	Dteday
3	Session
4	Yr
5	Mnth
6	Holiday
7	Weekday
8	Workingday
9	Weathersit
10	Temp
11	Atemp
12	Hum
13	Windspeed

Table 1.2: Predictor variables

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 data exploration.
- Modeling: It includes identifying suitable Machine Learning Algorithms and applying those algorithms in given dataset.

2.1: Pre-processing

Here, we will use techniques like missing value analysis, outlier analysis, feature selection etc.These techniques are used to structure our data. Basically, pre-processing is done because and the model asks for structured data and preprocessing is used to structure the data we have got. As, normally the data we get can be messy i.e.: it can include many missing values, inconsistent values etc. And these 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.

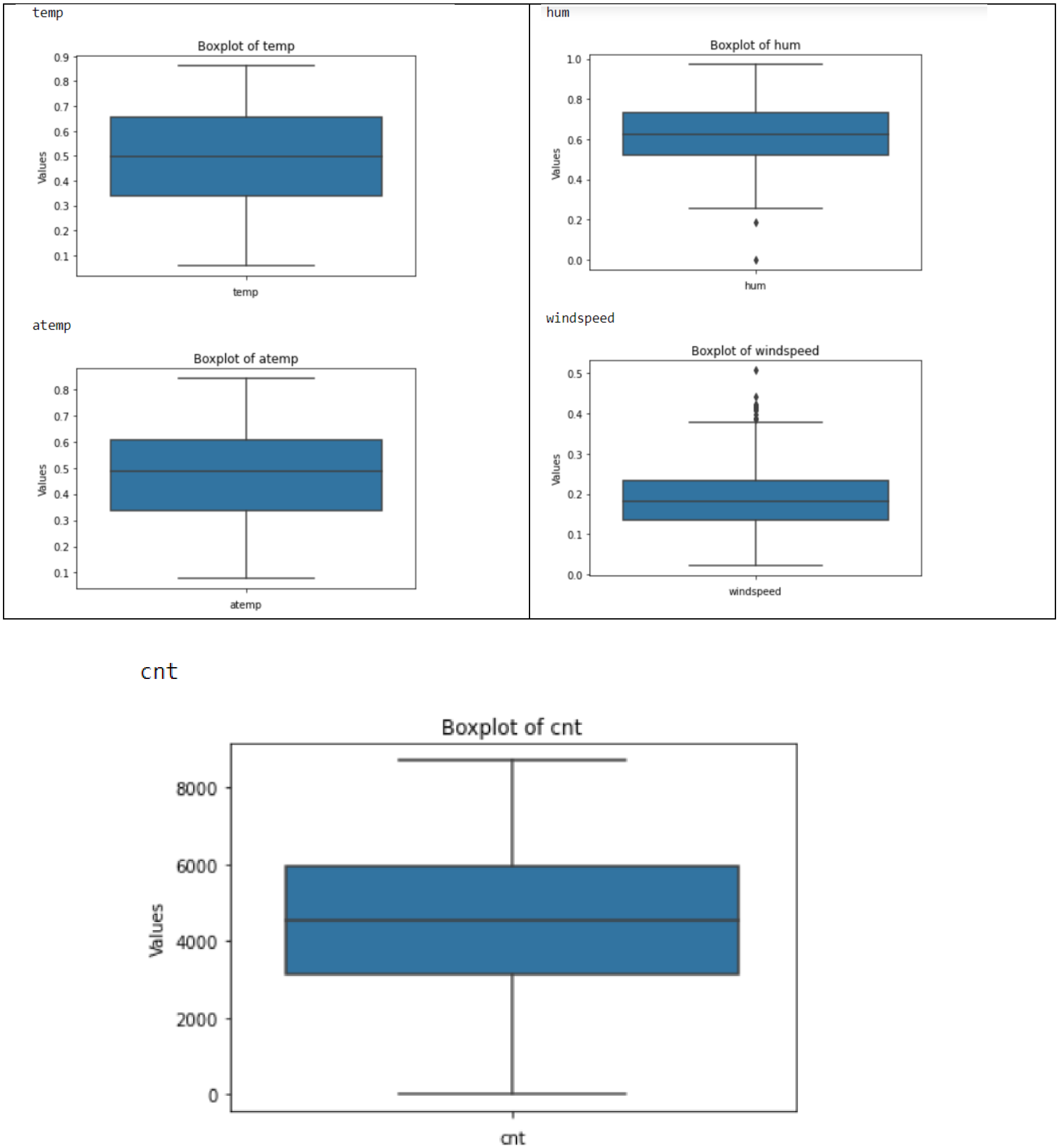
after checking the data, it is found that the data doesn't consist any missing values.

```
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 value found
```

2.1.1: Missing Value

2.1.2: Outlier Analysis

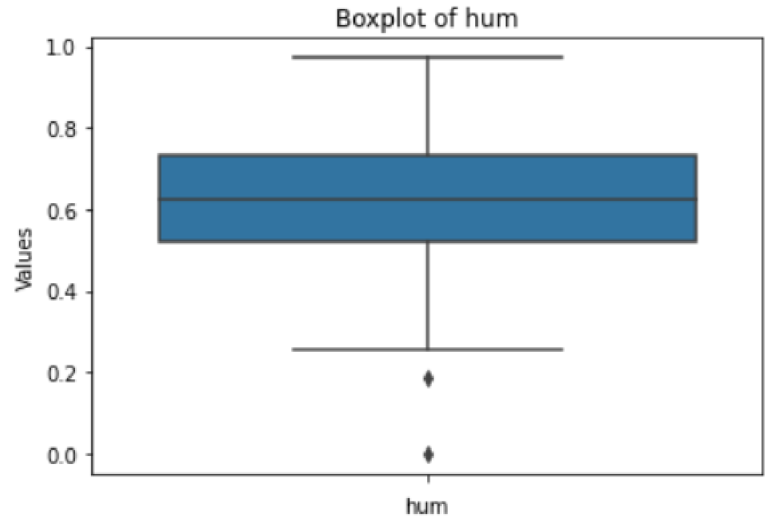
Outlier is an abnormal observation that stands or deviates away from other observations. These happensbecause of manual error, poor quality of data and it is correct but exceptional data. But, it can cause an error in predicting the target variables. So we have to check for outliers in our data set and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



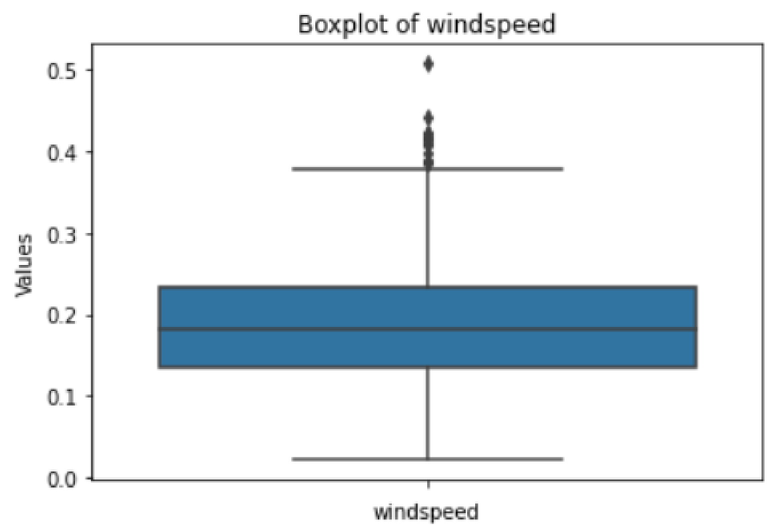
2.1.2: Boxplot for all variables

So as we can see outliers are found in only two variables these are Humidity and windspeed, following are the box plots for both the variables and dots outside the quartile ranges are outliers.

hum



windspeed



Plot: Outliers

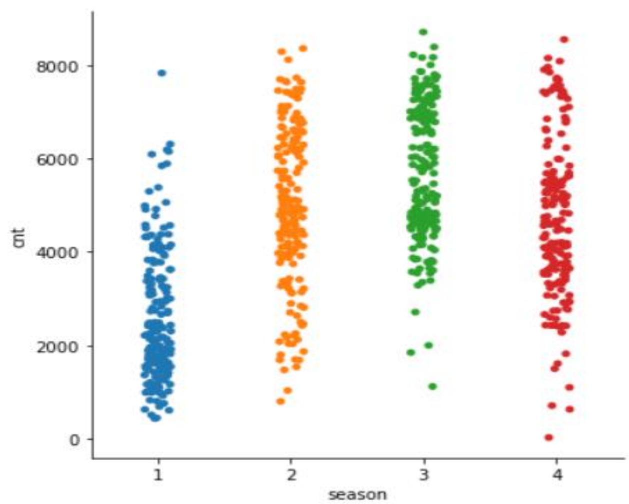
All these outliers mentioned above happened because of manual error, or interchange of data, or may be correct data but exceptional. But all these outliers can hamper our data model. So there is a requirement to eliminate or replace such outliers, and impute with proper methods to get better accuracy of the model.In this project, I used median method to impute the outliers in windspeed and humidity variables.

2.1.3: Categorical Plots

The catplot function provides a new framework giving access to several types of plots that show relationship between numerical variable and one or more categorical variables. Catplot can handle 8 different plots currently available in Seaborn. catplot function can do all these types of plots and one can specify the type of plot one needs with the kind parameter.
Categorical Plots 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 somecases three variables are plotted together for our better understanding and visualization.

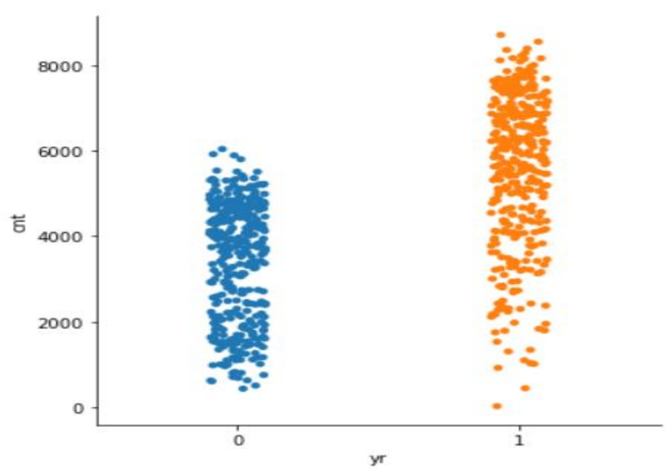
Categorical variables is as shown in the below figures:

A) Session



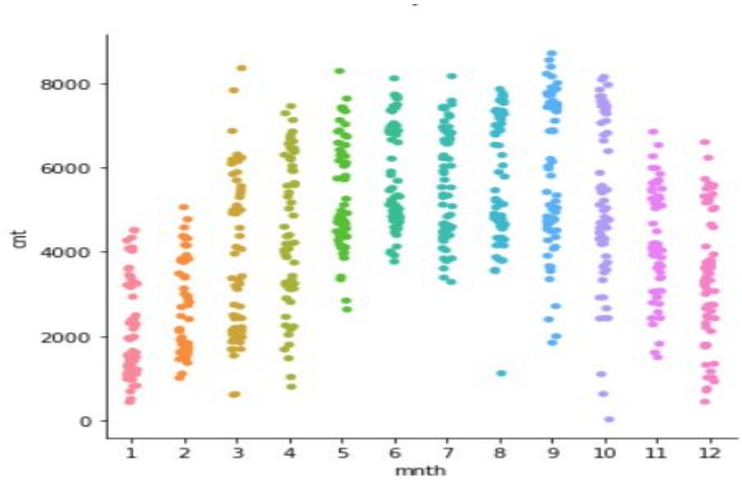
We have highest count in Season 2, 3 and 4

B) Year



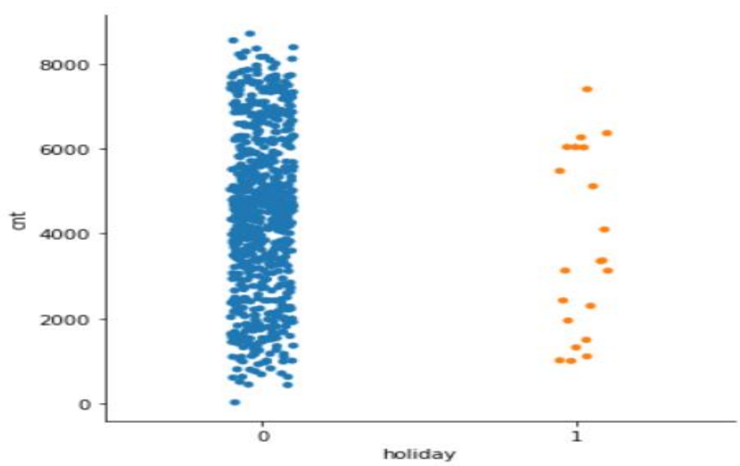
We have high count in year 1

C) Month



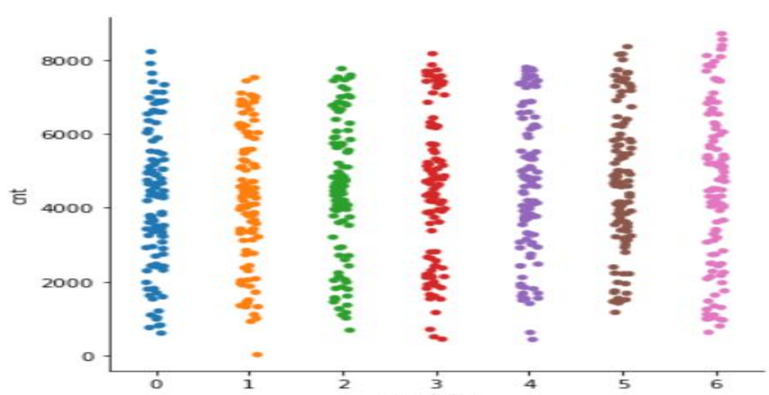
We have good count in month 3-10

D) Holiday



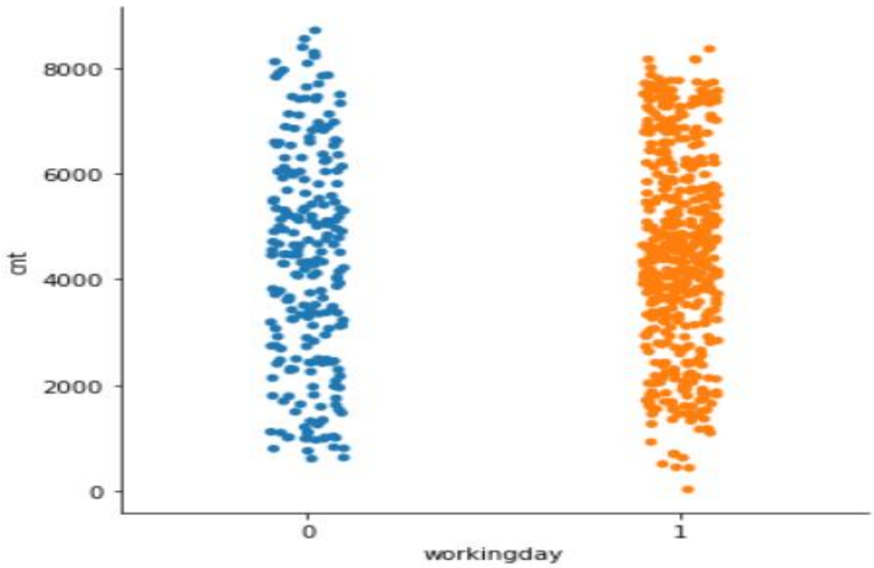
We have high count in holidays than non-holidays

E) Weekday



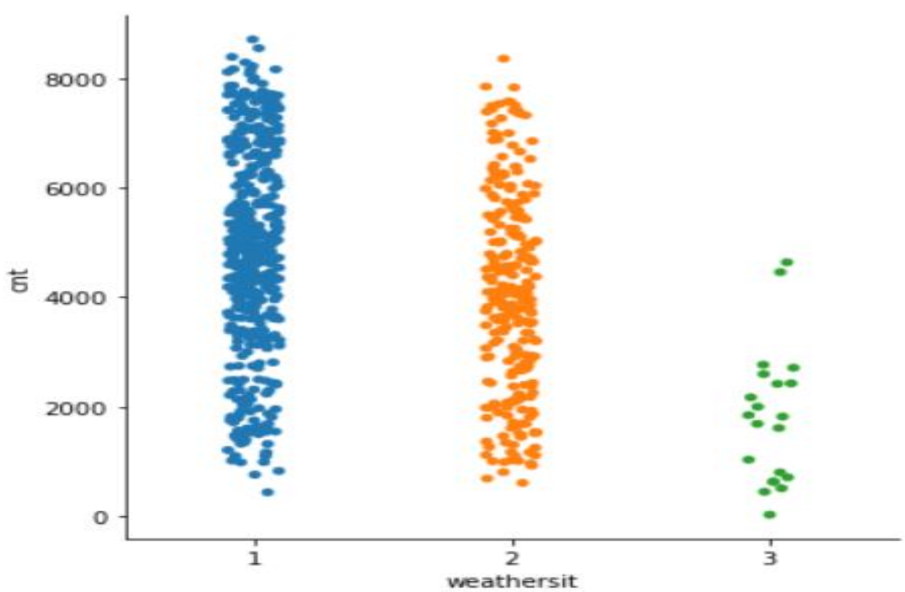
We have highest count in weekdays 0-6

F) Workingday



We have highest count in workingday 1

G) Weathersit



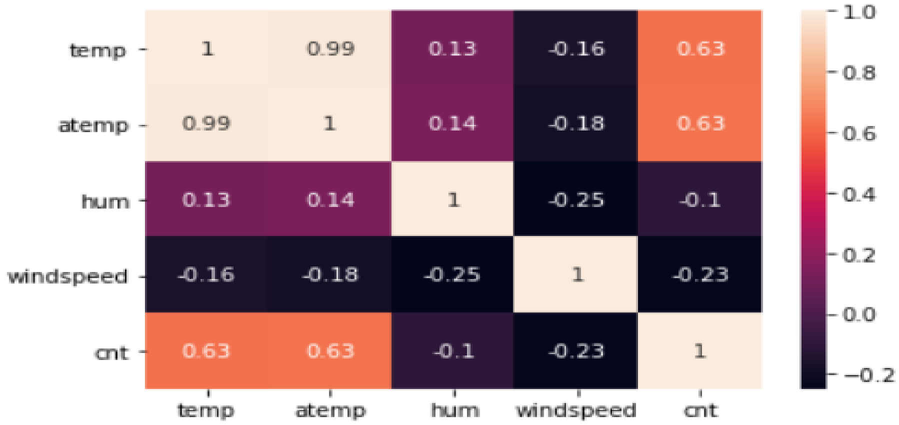
We have highest count in weather 1

2.1.4: Feature Selection

Feature Selection reduces the complexity of a model and makes it easier to interpret. It also reduces overfitting. Features are selected based on their scores in various statistical tests for their correlation with the outcome variable. Correlation plot is used to find out if there is any multicollinearity between variables.

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

A) Correlation Analysis



Plot: Correlation Analysis

Values which are close to 1 are highly correlated, so temp & temp are highly correlated with each other. So, we need to drop atemp as it is similar to temperature.

B) ANOVA Test

```
                  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. these variables have no dependency over target variable. So, in furtherprocesses these variables can be dropped before modeling.

2.2: Modeling

After Data Analysis and Data Pre-Processing, next step is Modeling. 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 haveto choose our model. So, the first step in Modeling is selection of model.

2.2.1: Model Selection

In this case we have to predict the count of bike renting according to environmental and seasonal condition. So the target variable here is a continuous variable. For Continuous we can use various Regression models. Model having less error rate and more accuracy will be our final model.

Models built are:-

2.2.2: Linear Regression

Linear regression 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 estimatethe value of the response *Y*, when only the predictors (*X- Values*) are known.

Linear Regression is applied in Python; details are described following.

```
from sklearn.linear_model import LinearRegression
lin = LinearRegression()
lin.fit(X_train, y_train)
ylin_pred = lin.predict(X_test)
```

```
from sklearn import metrics

print("Mean absolute error:",metrics.mean_absolute_error(y_test, ylin_pred))
print("Mean Squared Error:",metrics.mean_squared_error(y_test, ylin_pred))
print("Root Mean Squared Error:",np.sqrt(metrics.mean_squared_error(y_test, ylin_pred)))
print("R2 score:{:0.2f}".format(metrics.r2_score(y_test, ylin_pred)*100),"%")
```

```
Mean absolute error: 569.74077815677
Mean Squared Error: 648801.3757133388
Root Mean Squared Error: 805.4820765934763
R2 score:82.44 %
```

So with Linear Regression model in python we got R2 score 82.44% .

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
import random
random.seed(1)
classifier = RandomForestRegressor(n_estimators = 50)
classifier.fit(X_train, y_train.values.ravel())
yRand_pred = classifier.predict(X_test)
```

```
print('Mean absolute error:',metrics.mean_absolute_error(y_test, yRand_pred))
print('Mean Squared Error:',metrics.mean_squared_error(y_test, yRand_pred))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, yRand_pred)))
print("R2 score:{:0.2f}".format(metrics.r2_score(y_test, yRand_pred)*100),"%")
```

```
Mean absolute error: 411.155238095238
Mean Squared Error: 370310.8203020408
Root Mean Squared Error: 608.53169210982
R2 score:89.98 %
```

So with Random Forest model in python we got R2 score 89.98%.
Like the above 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.

2.2.4: Decision Tree

Decision Tree is a supervised learning predictive model that uses a set of binary rules to calculate the target value/dependent variable.
A tree has many analogies in real life, and turns out that it has influenced a wide area of machine learning, covering both classification and regression. In decision analysis, a decision tree can be used to visually and explicitly represent decisions and decision making. As the name goes, it uses a tree-like model of decisions.

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

```python
from sklearn import tree
clf = tree.DecisionTreeRegressor()
clf.fit(X_train, y_train)
yDec_pred = clf.predict(X_test)
```

```python
print('Mean absolute error:',metrics.mean_absolute_error(y_test, yDec_pred))
print('Mean Squared Error:',metrics.mean_squared_error(y_test, yDec_pred))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, yDec_pred)))
print('R2 score:{:0.2f}'.format(metrics.r2_score(y_test, yDec_pred)*100),"%")
```

```
Mean absolute error: 538.2925170068028
Mean Squared Error: 614816.7551020408
Root Mean Squared Error: 784.1025156840404
R2 score:83.36 %
```

So with Decision Tree model in python we got R2 score 83.36%.

The above fit plot shows the criteria that is used in developing the decision tree in Python. To develop the model in python, during modeling I have kept all the attributes at default. 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.

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 modelis fit into it to test data to predict the target variable. After predicting the target variable in test data, theactual and predicted values of target variable are compare to get the error and accuracy. And looking overthe 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

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 this, error metrics are used.

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

Method	MAE Error
Linear Regression	569.74077815677
Random Forest	411.155238095238
Decision Tree	538.2925170068028

If we observe the above tables, we choose the model with lowest MAE as a suitable Model. Here, we get Random Forest as a better model. So following this we can conclude that Both Random Forest and Decision Tree can be used asmodel for this data, if you evaluate on the basis of MAE. But we need more error metrics to cross check this.

3.2: Accuracy

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

Method	Accuracy (in Percentage)
Linear Regression	82.44
Random Forest	89.98
Decision Tree	83.36

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

3.3: Root Mean Squared Error

Root Mean Squared Error is another metric that helps us to know about the predicted values.

Method	R – Square
Linear Regression	805.4820765934763
Random Forest	608.53169210982
Decision Tree	784.1025156840404

Root Mean Squared Error 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, it is found that Random Forest is a best fit model for the given data.

Conclusion: -

Here we found that Random Forest is a best fit model for the given data.

APPENDIX

Python Code

Importing the libraries

In [1]:
```
import pandas as pd
import os
import seaborn as sns
import numpy as np
from random import randrange,uniform
from sklearn.metrics import r2_score
from scipy import stats
import matplotlib.pyplot as plt
from sklearn import preprocessing
from sklearn.preprocessing import StandardScaler
from sklearn import linear_model
from sklearn.metrics import mean_squared_error
```

Some utility functions ¶

In [2]:
```
def set_day(df):
    '''
    This function assigns day names to each of the
    rows in the dataset.
    '''
    ## Assumes the first day of the dataset is Saturday
    days = ["Sat", "Sun", "Mon", "Tue", "Wed", "Thr", "Fri"]
    temp = ['d']*df.shape[0]
    i = 0
    indx = 0
    cur_day = df.weekday[0]
    for day in df.weekday:
        temp[indx] = days[(day-cur_day+7)%7]
        indx += 1
    df['dayWeek'] = temp
    return df

# Function that takes in a dataframe with yr and mnth attribute and calculates an array denoting the month number from the start
def mnth_cnt(df):
    '''
    Compute the count of months from the start of
    the time series.
    '''
    import itertools
    yr = df['yr'].tolist()
    mnth = df['mnth'].tolist()
    out = [0] * df.shape[0]
    indx = 0
    for x, y in zip(mnth, yr):
        out[indx] = x + 12 * y
        indx += 1
    return out
```

In [3]:
```
# Working directory
os.chdir("C:/Users/prashant/OneDrive/Desktop/Edwisor/Projects/Project1 (Bike Rental-Jitendra)")
```

In [4]:
```
os.getcwd()
```

Out[4]:
```
'C:\\Users\\prashant\\OneDrive\\Desktop\\Edwisor\\Projects\\Project1 (Bike Rental-Jitendra)'
```

In [5]:
```
#Importing Data

dir = 'C:\\Users\\prashant\\OneDrive\\Desktop\\Edwisor\\Projects\\Project1 (Bike Rental-Jitendra)'
df = pd.read_csv(os.path.join(dir, 'day.csv'))
print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 731 entries, 0 to 730
Data columns (total 16 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   instant     731 non-null    int64
 1   dteday      731 non-null    object
 2   season      731 non-null    int64
 3   yr          731 non-null    int64
 4   mnth        731 non-null    int64
 5   holiday     731 non-null    int64
 6   weekday     731 non-null    int64
 7   workingday  731 non-null    int64
 8   weathersit  731 non-null    int64
 9   temp        731 non-null    float64
 10  atemp       731 non-null    float64
 11  hum         731 non-null    float64
 12  windspeed   731 non-null    float64
 13  casual      731 non-null    int64
 14  registered  731 non-null    int64
 15  cnt         731 non-null    int64
dtypes: float64(4), int64(11), object(1)
memory usage: 91.5+ KB
None
```

In [6]:
```
# Load Data
bikesData = pd.read_csv("day.csv")
```

In [7]:
```
bikesData.head()
```

Out[7]:

	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

In [8]:
```
#Type of DataFrame
print(type(bikesData))
```

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

In [9]:
```
#Varaibles
print(bikesData.dtypes)
```

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

In [10]:
```
print(bikesData.shape)
```

```
(731, 16)
```

In [11]:
```
# Index range
print(bikesData.index)
```

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

In [12]:
```python
#columns
print("Columns", bikesData.columns)
```

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

In [13]:
```python
#unique values in dataset
bikesData.nunique()
```

Out[13]:
```
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 [14]:
```python
#As we observe,some of the attributes are not required as per the requirement.
# So we can dropped :['instant','casual','registered','dteday'].
columnsToDrop = ['instant','casual','registered','dteday']

print(bikesData.shape)
```

```
(731, 16)
```

In [15]:
```python
numeric_var = ['temp', 'atemp', 'hum', 'windspeed', 'cnt']

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

Analyzing the dataset

In [16]:
```
# dataset with null value
bikesData.isnull().sum()
```

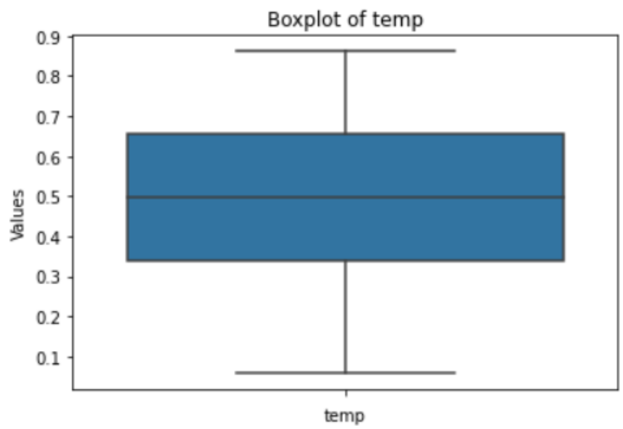
Out[16]:
```
instant       0
dteday        0
season        0
yr            0
mnth          0
holiday       0
weekday       0
workingday    0
weathersit    0
temp          0
atemp         0
hum           0
windspeed     0
casual        0
registered    0
cnt           0
dtype: int64
```

No missing value found

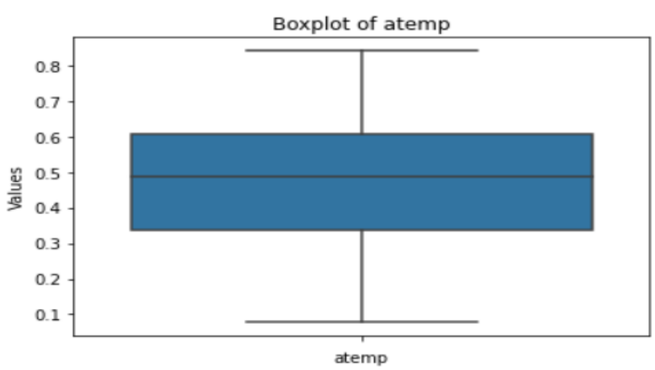
Outlier Analysis

In [17]:
```
for i in numeric_var:
    print(i)
    sns.boxplot(y = bikesData[i])
    plt.xlabel(i)
    plt.ylabel("Values")
    plt.title("Boxplot of " + i)
    plt.show()
```

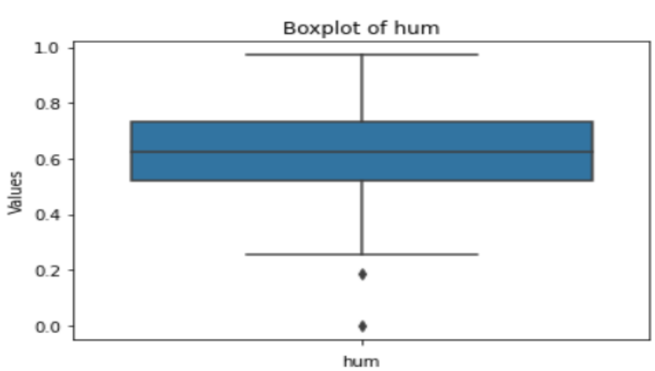
temp



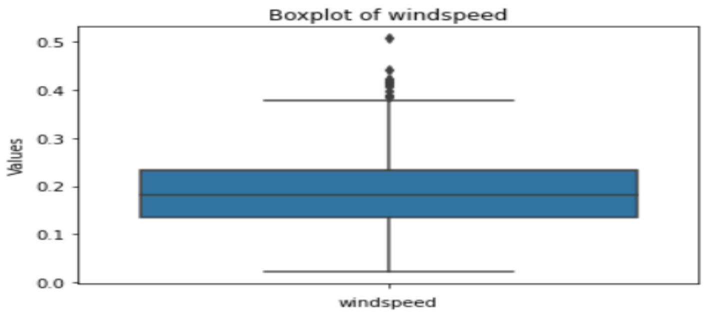
atemp



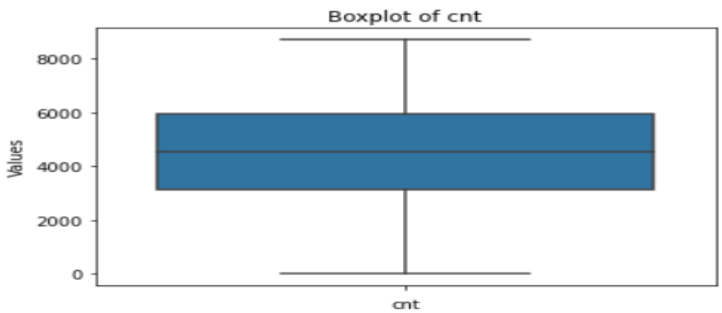
hum



windspeed



cnt



outliers are in windspeed and humidity.

In [18]:
```
for i in numeric_var:
    print(i)
    q75, q25 = np.percentile(bikesData.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))
```

```
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 [19]:
```
bikesData.loc[bikesData[i]<Innerfence, i] = np.nan
bikesData.loc[bikesData[i]>Upperfence, i] = np.nan
```

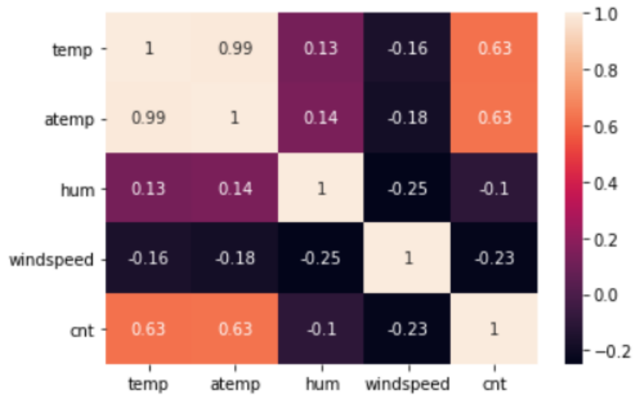
```
In [20]: # Dropping original date column
train=bikesData.drop(['dteday','instant','casual','registered','dteday'], axis=1)
```

FEATURE SELECTION

```
In [21]: # Correlation Analysis and Anova test # we have correlation value in  between temp ,atemp, hum, windspeed, cnt
bikesData_cor = bikesData.loc[:, numeric_var]
correlation_result = bikesData_cor.corr()
print(correlation_result)
```

```
               temp     atemp       hum  windspeed       cnt
temp       1.000000  0.991702  0.126963  -0.157944  0.627494
atemp      0.991702  1.000000  0.139988  -0.183643  0.631066
hum        0.126963  0.139988  1.000000  -0.248489 -0.100659
windspeed -0.157944 -0.183643 -0.248489   1.000000 -0.234545
cnt        0.627494  0.631066 -0.100659  -0.234545  1.000000
```

```
In [22]: heatmap = sns.heatmap(correlation_result, annot=True)
```



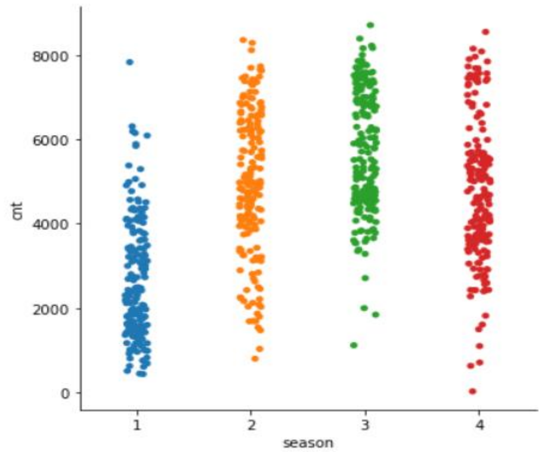
Values which are close to 1 are highly correlated, so temp & atemp are highly correlated with each other

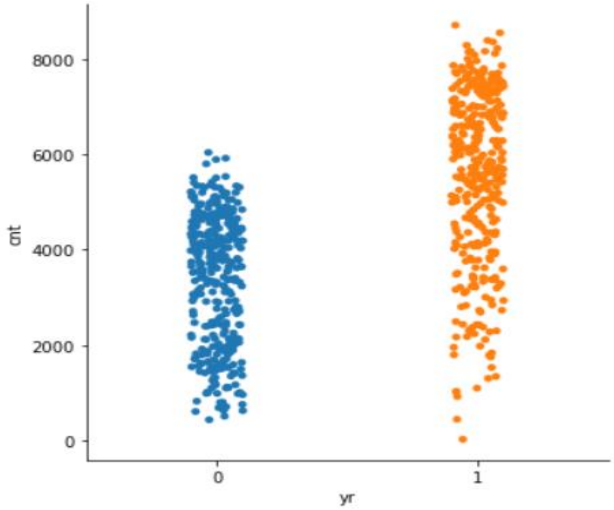
```
In [23]: # Anova Test

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

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

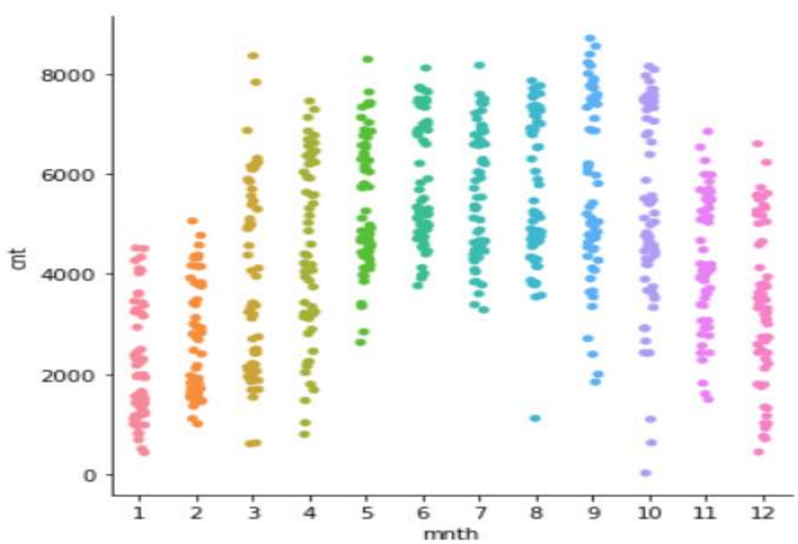
```
                sum_sq     df           F        PR(>F)
season    4.517974e+08    1.0  143.967653  2.133997e-30
Residual  2.287738e+09  729.0         NaN           NaN
                sum_sq     df           F        PR(>F)
yr        8.798289e+08    1.0  344.890586  2.483540e-63
Residual  1.859706e+09  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
```

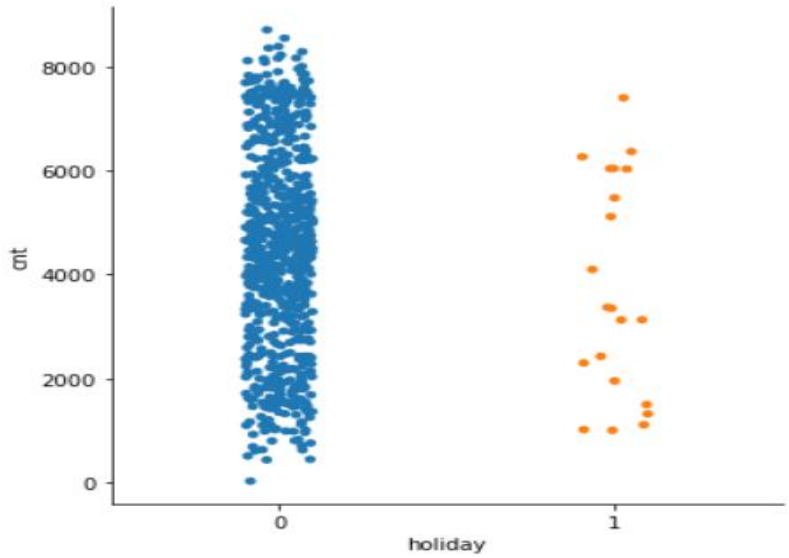
Categorical Plots ¶

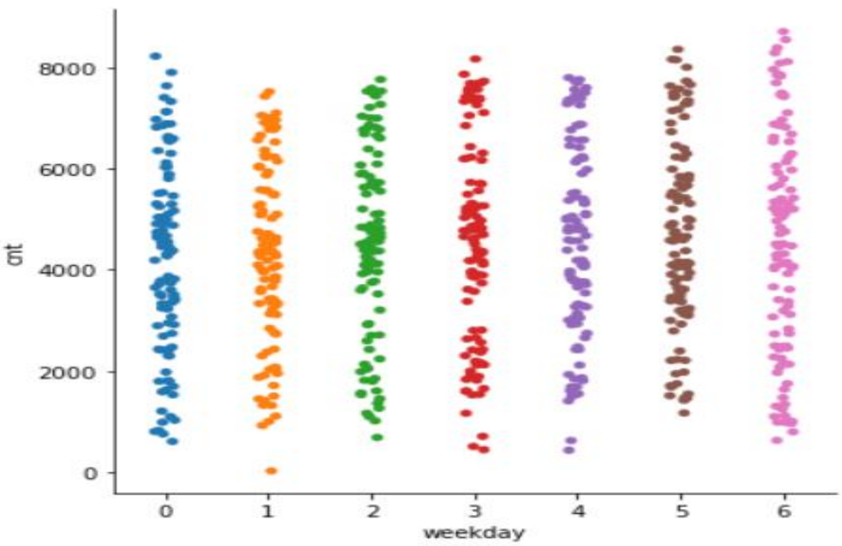
In [24]:
```
for i in categorical_var:
    sns.catplot(x = i, y = "cnt", data=bikesData)
```

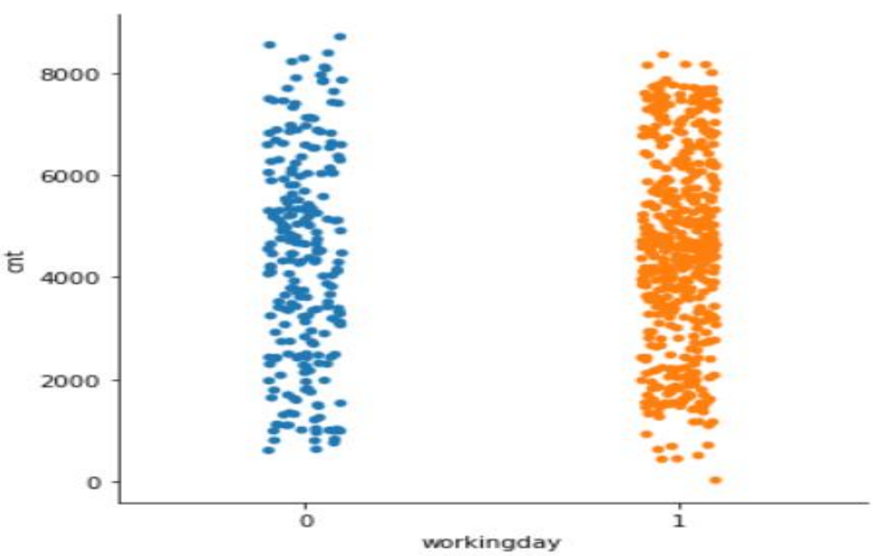


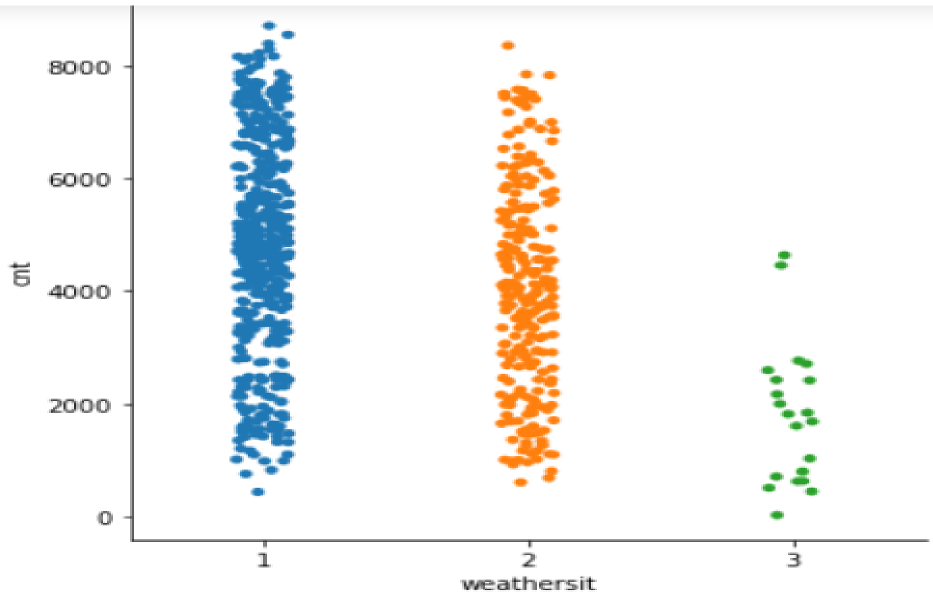





8000
6000
4000
2000
0
cnt
0
1
holiday


8000
6000
4000
2000
0
cnt
0
1
2
3
4
5
6
weekday


8000
6000
4000
2000
0
cnt
0
1
workingday



Results we got -

We have highest count in Season 2, 3 and 4

we have high count in year 1

we have good count in month 3-10

we have high count in holidays than non holidays

we have highest count in weekdays 0-6

we have highest count in workingday 1

we have highest count in weather 1

In [25]:
```
#Removing variables atemp beacuse it is highly correlated with temp,
train=bikesData.drop(['atemp'], axis=1)
```

Modeling & Model Evaluation

In [26]:
```
X = bikesData[['season', 'yr', 'mnth', 'holiday', 'weekday',
       'workingday', 'weathersit', 'temp', 'hum', 'windspeed']]
y = bikesData[['cnt']]
```

In [27]:
```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

# split into train test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=1)
print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)
```

```
(584, 10) (147, 10) (584, 1) (147, 1)
```

Linear Regression

In [28]:
```
from sklearn.linear_model import LinearRegression
lin = LinearRegression()
lin.fit(X_train, y_train)
ylin_pred = lin.predict(X_test)
```

In [29]:
```
from sklearn import metrics

print("Mean absolute error:",metrics.mean_absolute_error(y_test, ylin_pred))
print("Mean Squared Error:",metrics.mean_squared_error(y_test, ylin_pred))
print("Root Mean Squared Error:",np.sqrt(metrics.mean_squared_error(y_test, ylin_pred)))
print("R2 score:{:0.2f}".format(metrics.r2_score(y_test, ylin_pred)*100),"%")
```

```
Mean absolute error: 569.74077815677
Mean Squared Error: 648801.3757133388
Root Mean Squared Error: 805.4820765934763
R2 score:82.44 %
```

Random Forest

In [30]:
```
from sklearn.ensemble import RandomForestRegressor
import random
random.seed(1)
classifier = RandomForestRegressor(n_estimators = 50)
classifier.fit(X_train, y_train.values.ravel())
yRand_pred = classifier.predict(X_test)
```

In [31]:
```
print('Mean absolute error:',metrics.mean_absolute_error(y_test, yRand_pred))
print('Mean Squared Error:',metrics.mean_squared_error(y_test, yRand_pred))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, yRand_pred)))
print("R2 score:{:0.2f}".format(metrics.r2_score(y_test, yRand_pred)*100),"%")
```

```
Mean absolute error: 411.155238095238
Mean Squared Error: 370310.8203020408
Root Mean Squared Error: 608.53169210982
R2 score:89.98 %
```

Decision Tree

In [32]:
```
from sklearn import tree
clf = tree.DecisionTreeRegressor()
clf.fit(X_train, y_train)
yDec_pred = clf.predict(X_test)
```

In [33]:
```
print('Mean absolute error:',metrics.mean_absolute_error(y_test, yDec_pred))
print('Mean Squared Error:',metrics.mean_squared_error(y_test, yDec_pred))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, yDec_pred)))
print('R2 score:{:0.2f}'.format(metrics.r2_score(y_test, yDec_pred)*100),"%")
```

```
Mean absolute error: 538.2925170068028
Mean Squared Error: 614816.7551020408
Root Mean Squared Error: 784.1025156840404
R2 score:83.36 %
```

In [34]:
```
#Conclusion
##Model has been developed and result calculated.
```

In [ ]:

References

- www.edwisor.com
- www.youtube.com
- www.wikipedia.com
- Machine Learning Yearning By Andrew NG

NOTE- I have completed the project only in python not R as per got confirmation from Support Team Member-Arjun in query session. We got instruction that we don't need to work on R, we need to work only in python so I have completed in Python.