**Mini Project**

On

# BIKE SHARING PREDICTION

**Submitted in partial fulfillment of the requirements for the award of degree of**

**BACHELOR OF TECHNOLOGY**

**In**

**COMPUTER SCIENCE & ENGINEERING**

**BY**


| | |
|---|---|
| **18WH5A0506** | **SAMA NISCHALA** |
| **17WH1A0521** | **DANNAPANENI TRIPURA** |
| **18WH5A0510** | **KOBBA JAGADISHWARI** |


**Under the esteemed guidance of**


**Mr. U Chandrasekhar**

**Associate Professor**




**Department of Computer Science & Engineering**

**BVRIT HYDERABAD**

**COLLEGE OF ENGINEERING FOR WOMEN**

**(NBA Accredited EEE.ECE.CSE.IT B.Tech Courses)**

**(Approved by AICTE, New Delhi and Affiliated to JNTUH, Hyderabad)**

**Bachupally, Hyderabad – 500090**


**Jan, 2021**

**BVRIT HYDERABAD**

**COLLEGE OF ENGINEERING FOR WOMEN**

**(NBA Accredited EEE.ECE.CSE.IT B.Tech Courses)**

**(Approved by AICTE, New Delhi and Affiliated to JNTUH, Hyderabad)**

**Bachupally, Hyderabad – 500090**

**Department of Computer Science & Engineering**


## CERTIFICATE

This is to certify that the mini project entitled "**Bike Sharing Prediction**" is a bonafide work carried out by **Ms. SAMA NISCHALA (18WH5A0506), Ms. DANNAPANENI TRIPURA (17WH1A0521), Ms. KOBBA JAGADISHWARI (18WH5A0510)** in partial fulfillment for the award of B.Tech degree in **Computer Science & Engineering**, **BVRIT HYDERABAD College of Engineering for Women, Bachupally, Hyderabad**, affiliated to Jawaharlal Nehru Technological University Hyderabad, Hyderabad under my guidance and supervision. The results embodied in the project work have not been submitted to any other University or Institute for the award of any degree or diploma.

**Internal Guide**
**U. Chandrasekhar**
**Associate Professor, CSE**

**Head of the Department**
**Dr. K. Srinivasa Reddy**
**Professor, CSE**

## DECLARATION

We hereby declare that the work presented in this project entitled **"Bike Sharing Prediction"** submitted towards completion of Project work in IV Year of B.Tech of CSE at **BVRIT HYDERABAD College of Engineering for Women,** Hyderabad is an authentic record of our original work carried out under the guidance of **U Chandrasekhar, Associate Professor, Department of CSE.**

Sign with Date:
SAMA NISCHALA
(18wh5a0506)

Sign with Date:
DANNAPANENI TRIPURA
(17wh1a0521)

Sign with Date:
KOBBA JAGADISHWARI
(18wh5a0510)

## ACKNOWLEDGEMENT

We would like to express our sincere thanks to **Dr.K.V.N.Sunitha**, **Principal, and BVRIT HYDERABAD College of Engineering for Women,** for her support by providing the working facilities in the college.

Our sincere thanks and gratitude to **Dr. K. Srinivasa Reddy, Head, Department of CSE, BVRIT HYDERABAD College of Engineering for Women,** for all timely support and valuable suggestions during the period of our project.

We are extremely thankful to our Internal Guide, **U Chandrasekhar, Associate Professor, CSE, and BVRIT HYDERABAD College of Engineering for Women,** for his constant guidance and encouragement throughout the project.

Finally, we would like to thank our Mini Project Coordinator, all Faculty and Staff of CSE department who helped us directly or indirectly. Last but not least, we wish to acknowledge our **Parents** and **Friends** for giving moral strength and constant encouragement.

**SAMA NISCHALA (18wh5a0506)**

**DANNAPANENI TRIPURA (17wh1a0521)**

**KOBBA JAGADISHWARI (18wh5a0510)**

## ABSTRACT

In this project we tried to apply machine learning algorithm into a real world problem – "Bike Sharing Prediction". This dataset contains the hourly and daily count of rental bikes between years 2011 and 2012 in capital bike share system with the corresponding weather and seasonal information.

Bike Sharing System is an emerging mode of transport in the world and most of the developing countries are on the path of following the western model of Bike Sharing Systems. In India some entrepreneurs have tried to setup a bike share system and have failed in the past as they have failed to use data analytics properly.

There is a possibility that bike stations can be full or empty when a traveler comes to the station. Thus to predict the use of such system can be helpful for the users to plan their travels and also for the entrepreneurs to setup the system properly. This paper presents different ways to predict the number of bikes that can be rented in such a system, for case study purpose we have used a public data set. This dataset contains the hourly and daily count of rental bikes between years 2011 and 2012 in capital bike share system with the corresponding weather and seasonal information.

## LIST OF FIGURES

## LIST OF CONTENTS

## 1) INTRODUCTION

"Bike sharing" systems are new generation of traditional bike rentals where whole process from membership, rental and return back has become automatic. Through these systems, user is able to easily rent a bike from a particular position and return back at another position. Currently, there are about over 500 bike-sharing programs around the world which is composed of over 500 thousands bicycles. Today, there exists great interest in these systems due to their important role in traffic, environmental and health issues.

Apart from interesting real world applications of bike sharing systems, the characteristics of data being generated by these systems make them attractive for the research. Opposed to other transport services such as bus or subway, the duration of travel, departure and arrival position is explicitly recorded in these systems.

This feature turns bike sharing system into a virtual sensor network that can be used for sensing mobility in the city. Hence, it is expected that most of important events in the city could be detected via monitoring these data.

### 1.1) PROBLEM STATEMENT

To build a superior statistical model to predicate the number of bicycles that can be rented with availability of data. Prediction of bike rental count hourly or daily based on the environmental and seasonal settings.

### 1.2) OBJECTIVE

The main objective is to predict or classify the bike rental count hourly or daily based on the environmental and seasonal settings by using regression and classification.

For this purpose, we decided to treat the task as both regression and classification problem with each group member experimenting with a different algorithm for regression and classification.

## 2) REQUIREMENTS

### 2.1) SOFTWARE

- Google Collaborator

- Python 3

### 2.2) HARDWARE

- Intel core Pentium processor

- RAM 4GB

- Hard Disk Drive 1 TB

**3.) Design**:

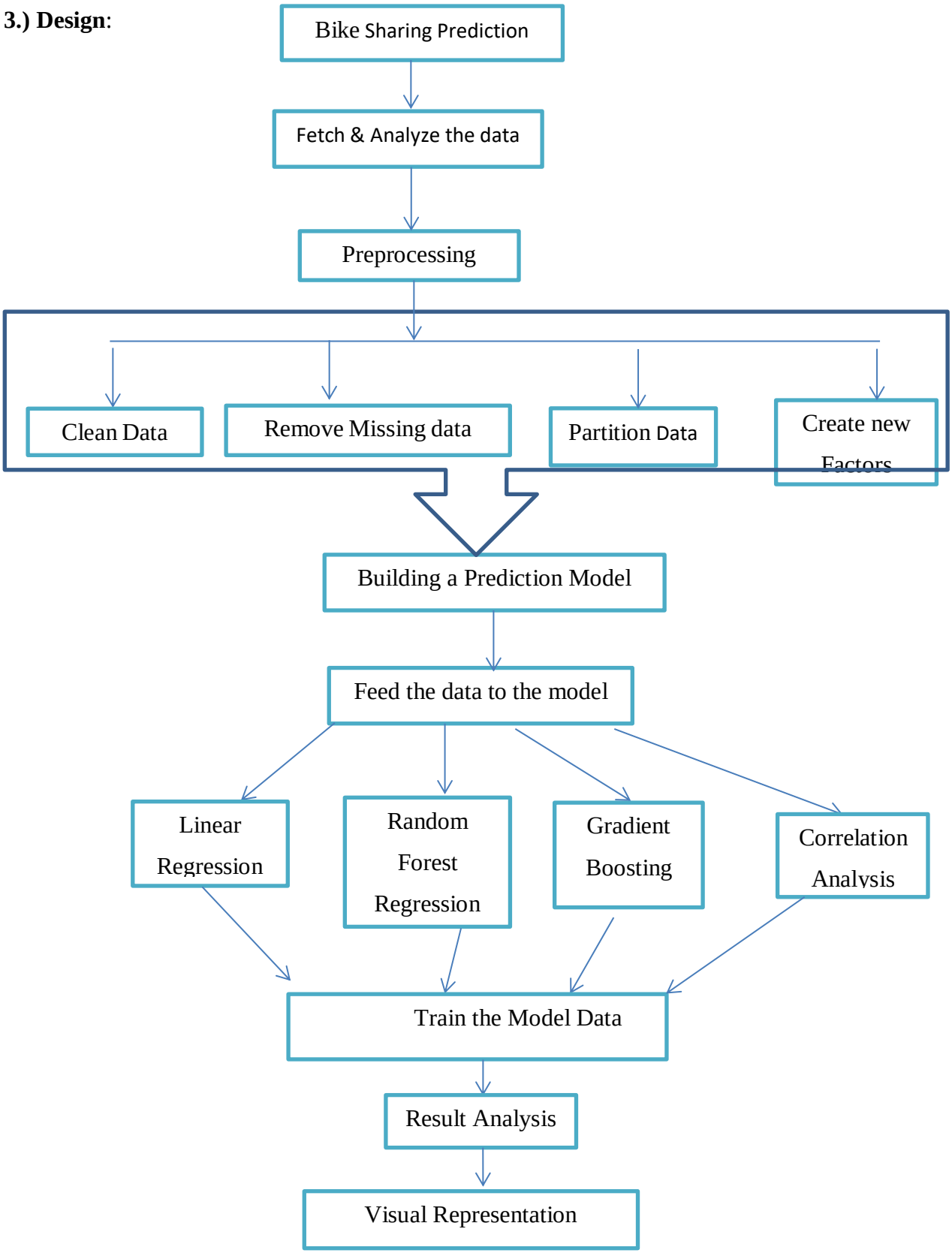



**Figure 1:** Flow chat of the model.

## 4) IMPLEMENTATION:

### 4.1) IMPORTS

In [2]:
```
import pandas as pd

# Read in data as pandas dataframe and display first 5 rows
features = pd.read_csv('/content/drive/My Drive/AI ML/Project_2/Project_2/Dataset/hour.csv')
features.head(5)
```

Out[2]:

| | instant | dteday | season | yr | mnth | hr | holiday | weekday | workingday | weathersit | temp | atemp | hum | windspeed | casual | registered | cnt |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2011-01-01 | 1 | 0 | 1 | 0 | 0 | 6 | 0 | 1 | 0.24 | 0.2879 | 0.81 | 0.0 | 3 | 13 | 16 |
| 1 | 2 | 2011-01-01 | 1 | 0 | 1 | 1 | 0 | 6 | 0 | 1 | 0.22 | 0.2727 | 0.80 | 0.0 | 8 | 32 | 40 |
| 2 | 3 | 2011-01-01 | 1 | 0 | 1 | 2 | 0 | 6 | 0 | 1 | 0.22 | 0.2727 | 0.80 | 0.0 | 5 | 27 | 32 |
| 3 | 4 | 2011-01-01 | 1 | 0 | 1 | 3 | 0 | 6 | 0 | 1 | 0.24 | 0.2879 | 0.75 | 0.0 | 3 | 10 | 13 |
| 4 | 5 | 2011-01-01 | 1 | 0 | 1 | 4 | 0 | 6 | 0 | 1 | 0.24 | 0.2879 | 0.75 | 0.0 | 0 | 1 | 1 |

In [3]:
```
print('The shape of our features is:', features.shape)
```

```
The shape of our features is: (17379, 17)
```

### 4.2) DATA ANALYSIS:

Data Analysis is a process of understanding the data. In this we find patterns and try to obtain inferences due to which the underlying patterns are observed. It is a process of inspecting, cleansing, transforming, and modeling data with the goal of discovering useful information, suggesting conclusions, and supporting decision-making.

In [4]:
```
features.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 17379 entries, 0 to 17378
Data columns (total 17 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   instant     17379 non-null  int64
 1   dteday      17379 non-null  object
 2   season      17379 non-null  int64
 3   yr          17379 non-null  int64
 4   mnth        17379 non-null  int64
 5   hr          17379 non-null  int64
 6   holiday     17379 non-null  int64
 7   weekday     17379 non-null  int64
 8   workingday  17379 non-null  int64
 9   weathersit  17379 non-null  int64
 10  temp        17379 non-null  float64
 11  atemp       17379 non-null  float64
 12  hum         17379 non-null  float64
 13  windspeed   17379 non-null  float64
 14  casual      17379 non-null  int64
 15  registered  17379 non-null  int64
 16  cnt         17379 non-null  int64
dtypes: float64(4), int64(12), object(1)
memory usage: 2.3+ MB
```

Now, Here we check for any null values that are present or not.

**Null Value**: A **NULL value** is a special marker used in SQL to indicate that a data **value** does not exist in the database.

```
In [5]: print(features.isnull().any())
```

```
instant       False
dteday        False
season        False
yr            False
mnth          False
hr            False
holiday       False
weekday       False
workingday    False
weathersit    False
temp          False
atemp         False
hum           False
windspeed     False
casual        False
registered    False
cnt           False
dtype: bool
```

Here we can see that they is no null values are present in this dataset.

**BOX PLOTS:**

A boxplot is used below to analyze the relationship between a categorical feature and a continuous feature. Here we use various plots to analyze the data.

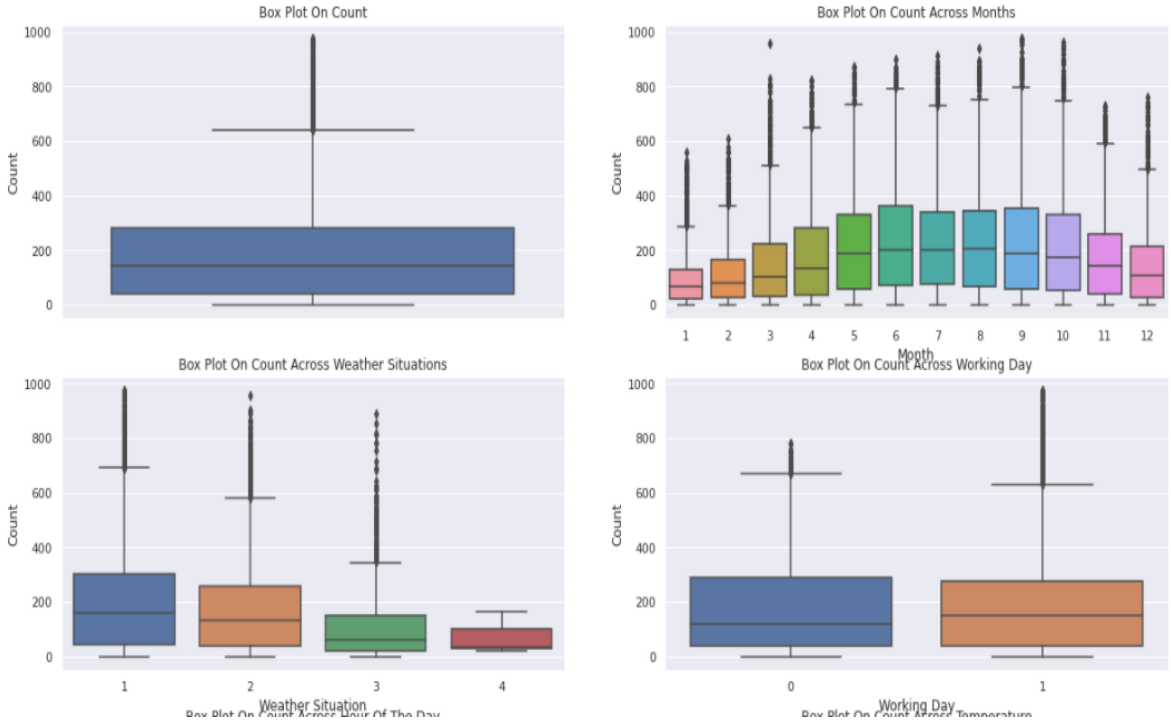
In [6]:
```
import seaborn as sns
import matplotlib.pyplot as plt

sns.set(font_scale=1.0)
fig, axes = plt.subplots(nrows=3,ncols=2)
fig.set_size_inches(20, 15)
sns.boxplot(data=features,y="cnt",orient="v",ax=axes[0][0])
sns.boxplot(data=features,y="cnt",x="mnth",orient="v",ax=axes[0][1])
sns.boxplot(data=features,y="cnt",x="weathersit",orient="v",ax=axes[1][0])
sns.boxplot(data=features,y="cnt",x="workingday",orient="v",ax=axes[1][1])
sns.boxplot(data=features,y="cnt",x="hr",orient="v",ax=axes[2][0])
sns.boxplot(data=features,y="cnt",x="temp",orient="v",ax=axes[2][1])

axes[0][0].set(ylabel='Count',title="Box Plot On Count")
axes[0][1].set(xlabel='Month', ylabel='Count',title="Box Plot On Count Across Months")
axes[1][0].set(xlabel='Weather Situation', ylabel='Count',title="Box Plot On Count Across Weather Situations")
axes[1][1].set(xlabel='Working Day', ylabel='Count',title="Box Plot On Count Across Working Day")
axes[2][0].set(xlabel='Hour Of The Day', ylabel='Count',title="Box Plot On Count Across Hour Of The Day")
axes[2][1].set(xlabel='Temperature', ylabel='Count',title="Box Plot On Count Across Temperature")
```

```
/usr/local/lib/python3.6/dist-packages/statsmodels/tools/_testing.py:19: FutureWarning: pandas.util.testing is deprecated. Use the functi
ons in the public API at pandas.testing instead.
  import pandas.util.testing as tm
```

Out[6]:
```
[Text(0, 0.5, 'Count'),
 Text(0.5, 0, 'Temperature'),
 Text(0.5, 1.0, 'Box Plot On Count Across Temperature')]
```



@

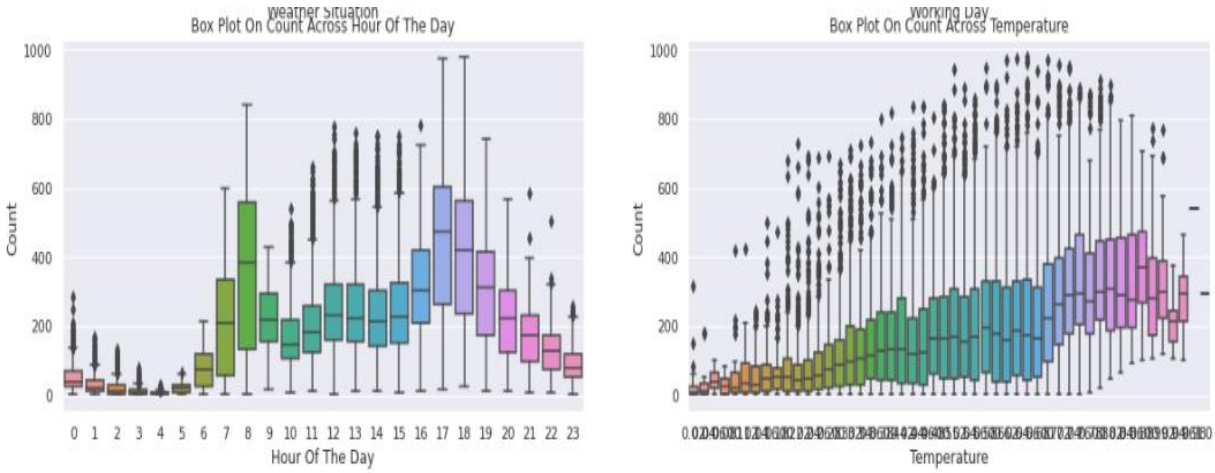



**Figure 2:** Blot plots on various attributes

The working day and holiday box plot indicates that more bicycles are rent during normal working days than on weekends or holidays. The hourly box plots show a local maximum at 8 am in the morning and 5pm and 6 pm in the evening. Another important factor seems to be the temperature: higher temperatures lead to an increasing number of bike rents and lower temperatures not only decrease the average number of rents but also shows more outliers in the data.

```
In [7]: features[:24*10].plot(x='dteday', y='cnt')
Out[7]: <matplotlib.axes._subplots.AxesSubplot at 0x7ff928f9d7b8>
```

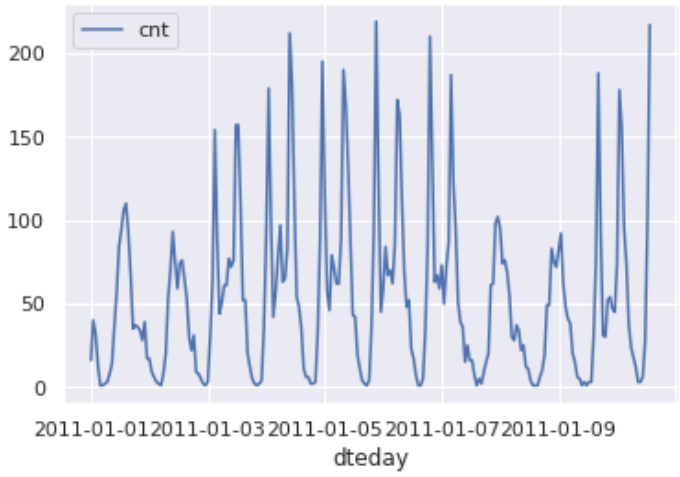



**Figure 3:** Graph according to monthly wise count

```
In [8]: fig,ax = plt.subplots()
        fig.set_size_inches(10, 6)
        sns.pointplot(data=features[['hr', 'cnt', 'season']],
                      x='hr',y='cnt',
                      hue='season',ax=ax)
        ax.set(title="Season wise hourly distribution of counts")
```

```
Out[8]: [Text(0.5, 1.0, 'Season wise hourly distribution of counts')]
```

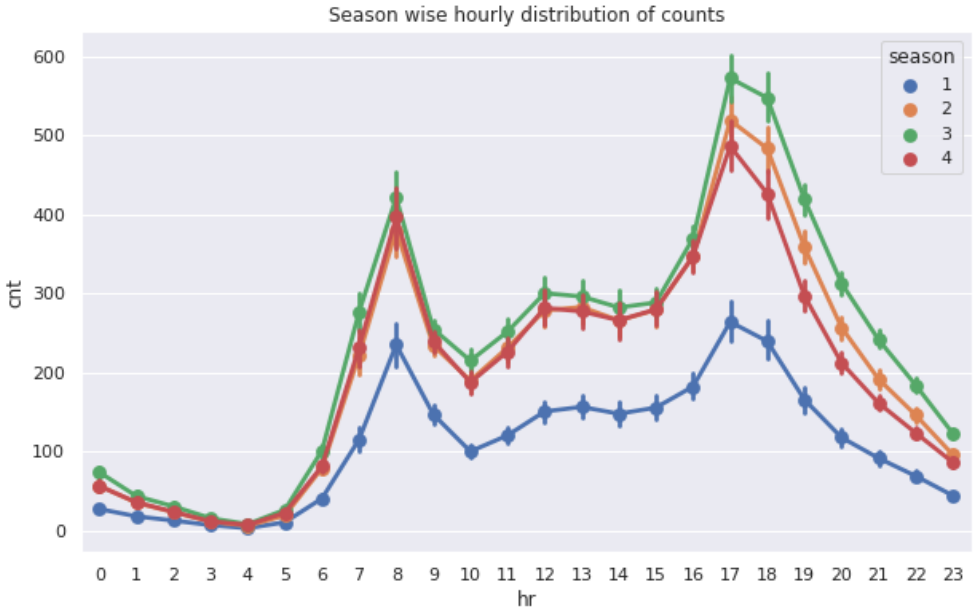



**Figure 4:** Season wise hourly distribution of counts.

- Here, blue colour indicates the “spring” season.
- Orange colour indicates the “summer” season.
- Green colour indicates the “Rainy” season.
- Red colour indicates the “winter” season.

Hence, we observe that spring has the lowest count throughout the year.

```
In [9]: fig,ax = plt.subplots()
        fig.set_size_inches(10, 6)
        sns.pointplot(data=features[['hr', 'cnt', 'weekday']],
                      x='hr',y='cnt',
                      hue='weekday',ax=ax)
        ax.set(title="Weekday wise hourly distribution of counts")
Out[9]: [Text(0.5, 1.0, 'Weekday wise hourly distribution of counts')]
```

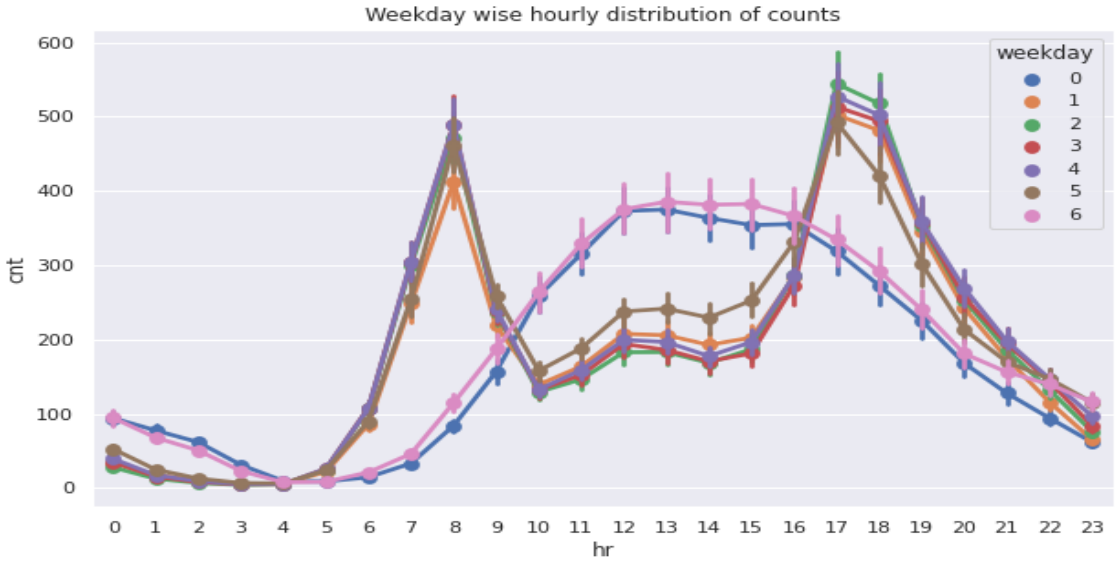



**Figure 5:** Weekday wise hourly distribution of counts.

- Here blue colour indicates “Sunday”.
- Orange colour indicates “Monday”.
- Green colour indicates “Tuesday”.
- Red colour indicates “Wednesday”.
- Blue colour indicates “Thursday”.
- Violet colour indicates “Friday”.
- Pink colour indicates “Saturday”.

  Here, the days 0 and 6 have similar trend.

```
In [10]: fig,ax = plt.subplots()
         sns.barplot(data=features[['mnth',
                                    'cnt']],
                     x="mnth",y="cnt")
         ax.set(title="Monthly distribution of counts")
Out[10]: [Text(0.5, 1.0, 'Monthly distribution of counts')]
```

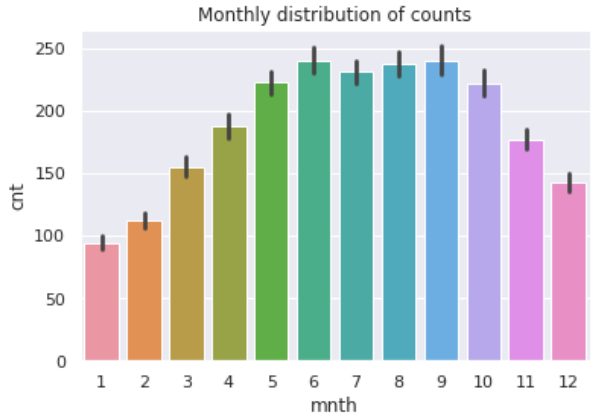



**Figure 6:** Box plot on Monthly distribution of counts.

We observed that from above figure June and September see the highest bike users.

```
In [11]: sns.violinplot(data=features[['yr','cnt']],
                        x="yr",y="cnt")
Out[11]: <matplotlib.axes._subplots.AxesSubplot at 0x7ff9248b7e80>
```

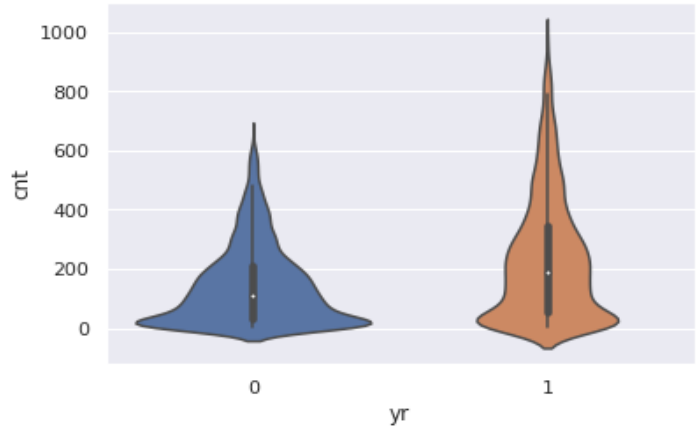



**Figure 7:** Violin plots

- A Violin Plot is used to visualise the distribution of the data and its probability density. The violin plots are a method of plotting numeric data and can be

considered a combination of the "box plot" with a "kernel density plot". In the violin plot, we can find the same information as in the box plots:

- median (a white dot on the violin plot)
- interquartile range (the black bar in the centre of violin)
- The lower/upper adjacent values (the black lines stretched from the bar).

### 4.4) CORRELATION ANALYSIS

Correlation Analysis is a statistical method that is used to discover if there is a relationship between two variables and how strong that relationship may be.

```
In [12]: import numpy as np
         matrix = features.corr()
         heat = np.array(matrix)
         heat[np.tril_indices_from(heat)] = False
         fig,ax= plt.subplots()
         fig.set_size_inches(30,20)
         sns.set(font_scale=1.0)
         sns.heatmap(matrix, mask=heat,vmax=1.0, vmin=0.0, square=True,annot=True, cmap="Reds")

Out[12]: <matplotlib.axes._subplots.AxesSubplot at 0x7ff9244e80b8>
```

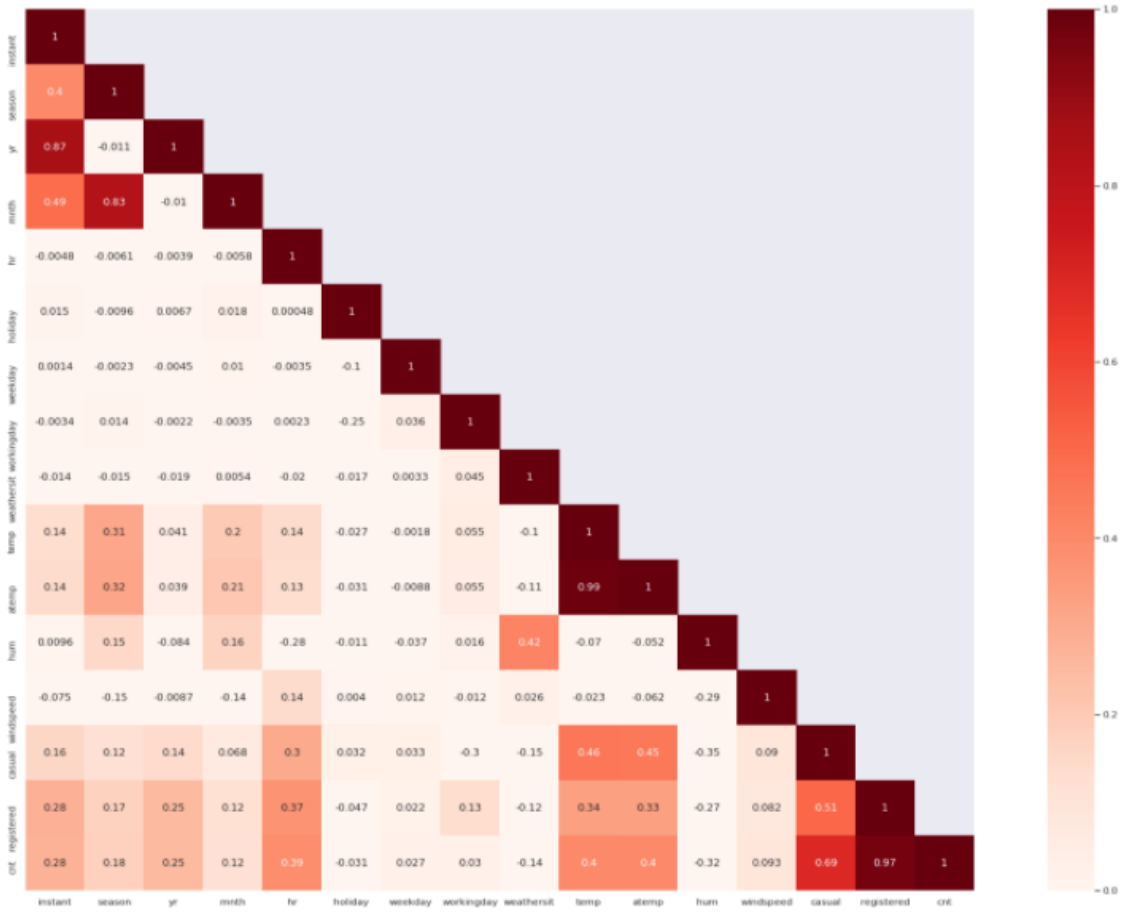



**Figure 8:** Correlation analysis.

### 4.5) DATA SPLIT:

In [0]:
```
# Use numpy to convert to arrays


# Labels are the values we want to predict
labels = np.array(features['cnt'])

# Remove the labels from the features
# axis 1 refers to the columns
features= features.drop(['cnt', 'instant', 'dteday', 'casual', 'registered'], axis = 1)

# Saving feature names for later use
feature_list = list(features.columns)

# Convert to numpy array
features = np.array(features)
```

In [14]:
```
features
```

Out[14]:
```
array([[ 1.    ,  0.    ,  1.    , ...,  0.2879,  0.81  ,  0.    ],
       [ 1.    ,  0.    ,  1.    , ...,  0.2727,  0.8   ,  0.    ],
       [ 1.    ,  0.    ,  1.    , ...,  0.2727,  0.8   ,  0.    ],
       ...,
       [ 1.    ,  1.    , 12.    , ...,  0.2576,  0.6   ,  0.1642],
       [ 1.    ,  1.    , 12.    , ...,  0.2727,  0.56  ,  0.1343],
       [ 1.    ,  1.    , 12.    , ...,  0.2727,  0.65  ,  0.1343]])
```

**train_test_split ():** It method from the sklearn package is used to split training and testing data.

**test_size** — this parameter decides the size of the data that has to be split as the test dataset. This is given as a fraction. For example, if you pass 0.5 as the value, the dataset will be split 50% as the test dataset. If you're specifying this parameter, you can ignore the next parameter.

**Random_state** — here you pass an integer, which will act as the seed for the random number generator during the split.

```
In [0]: # Using Skicit-learn to split data into training and testing sets
        from sklearn.model_selection import train_test_split

        # Split the data into training and testing sets
        x_train, x_test, y_train, y_test = train_test_split(features, labels, test_size = 0.25,
                                                            random_state = 42)
```

Hence, we take 25% data for test data with a random state 42.

### 4.6) TRAIN MODEL

1. You configure a model, by choosing a particular type of algorithm, and defining its parameters or hyper parameters. Choose any of the following model types:
    - Classification models, based on neural networks, decision trees, and decision forests, and other algorithms.
    - Regression models, which can include standard linear regression, or which use other algorithms, including neural networks and Bayesian regression.
2. Provide a dataset that is labelled, and has data compatible with the algorithm. Connect both the data and the model to Train Model.
3. After training is completed, use the trained model with one of the scoring modules, to make predictions on new data.

- Here, we use regression model as 'cnt' is a continuous output variables.

In [16]:

```
# Sort validation set for plots

#x_test = test[features].values
from prettytable import PrettyTable
# Sklearn metrics
from sklearn.metrics import mean_squared_error, mean_absolute_error, mean_squared_log_error, accuracy_score
# Sklearn models
from sklearn.linear_model import Lasso, ElasticNet, Ridge, SGDRegressor
from sklearn.svm import SVR, NuSVR
from sklearn.ensemble import BaggingRegressor, RandomForestRegressor, AdaBoostRegressor
from sklearn.neighbors import KNeighborsClassifier
from sklearn.cluster import KMeans

from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier

from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor


table = PrettyTable()
table.field_names = ["Model", "Mean Squared Error", "Score"]

models = [
    SGDRegressor(max_iter=1000, tol=1e-3),
    Lasso(alpha=0.1),
    ElasticNet(random_state=0),
    Ridge(alpha=.5),
    SVR(gamma='auto', kernel='linear'),
    SVR(gamma='auto', kernel='rbf'),
    BaggingRegressor(),
    BaggingRegressor(KNeighborsClassifier(), max_samples=0.5, max_features=0.5),
    NuSVR(gamma='auto'),
    RandomForestRegressor(random_state=0, n_estimators=300),
    AdaBoostRegressor(DecisionTreeRegressor(max_depth=40),
                      n_estimators=50, random_state=0)
]

for model in models:
    model.fit(x_train, y_train)
    y_res = model.predict(x_test)

    mse = mean_squared_error(y_test, y_res)
    score = model.score(x_test, y_test)

    table.add_row([type(model).__name__, format(mse, '.2f'), format(score, '.2f')])

print(table)
```

```
+-----------------------+--------------------+-------+
|         Model         | Mean Squared Error | Score |
+-----------------------+--------------------+-------+
|      SGDRegressor     |       24145.81     |  0.25 |
|         Lasso         |       19842.91     |  0.38 |
|       ElasticNet      |       24172.61     |  0.25 |
|         Ridge         |       19836.38     |  0.38 |
|          SVR          |       21536.58     |  0.33 |
|          SVR          |       13836.69     |  0.57 |
|    BaggingRegressor   |       2135.96      |  0.93 |
|    BaggingRegressor   |       12718.40     |  0.60 |
|         NuSVR         |       14086.19     |  0.56 |
| RandomForestRegressor |       1809.67      |  0.94 |
|   AdaBoostRegressor   |       2030.11      |  0.94 |
+-----------------------+--------------------+-------+
```

**Figure 9:** Different Algorithm Metrics

In the above table we notice that ensemble Regression models (RandomForestRegressor, AdaBoostRegressor and BaggingRegressor) have highest accuracy score. So, we choose RandomForestRegressor as it has least mean square error when compared to other regression models.

## MODEL WITH RANDOM FOREST REGRESSOR

Random Forest is a learning method that operates by constructing multiple decision trees. The final decision is made based on the majority of the trees and is chosen by the random forest.

```
In [17]: table = PrettyTable()
         table.field_names = ["Model", "Dataset", "MSE", "MAE", 'RMSLE', "Score"]
         # Model training
         model = RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=40,
                    max_features='auto', max_leaf_nodes=None,
                    min_impurity_decrease=0.0, min_impurity_split=None,
                    min_samples_leaf=1, min_samples_split=4,
                    min_weight_fraction_leaf=0.0, n_estimators=300, n_jobs=None,
                    oob_score=False, random_state=None, verbose=0, warm_start=True)
         model.fit(x_train, y_train)

         def evaluate(x, y, dataset):
             pred = model.predict(x)

             mse = mean_squared_error(y, pred)
             mae = mean_absolute_error(y, pred)
             score = model.score(x, y)
             rmsle = np.sqrt(mean_squared_log_error(y, pred))

             table.add_row([type(model).__name__, dataset, format(mse, '.2f'), format(mae, '.2f'), format(rmsle, '.2f'), format(score, '.2f')])


         evaluate(x_train, y_train, 'training')
         evaluate(x_test, y_test, 'validation')

         print(table)
+----------------------+------------+---------+-------+-------+-------+
|        Model         |  Dataset   |   MSE   |  MAE  | RMSLE | Score |
+----------------------+------------+---------+-------+-------+-------+
| RandomForestRegressor |  training  |  374.69 | 11.34 |  0.18 |  0.99 |
| RandomForestRegressor | validation | 1821.14 | 25.57 |  0.36 |  0.94 |
+----------------------+------------+---------+-------+-------+-------+
```

**Figure 10:** Random Forest Metrics Table

### Feature Importance:

Feature importances for non-working days are also similar to working days. So, **Time, Humidity, Temp and Month** are the most **important features** in all the cases.

From that all cases we can easily find nearby bike, we can lock and unlock the bikes anywhere through bike number, there is an ability to check-out trip details, transactions, advances booking system etc.

In [18]:
```
# Get numerical feature importances
importances = list(model.feature_importances_)

# List of tuples with variable and importance
feature_importances = [(feature, round(importance, 2)) for feature, importance in zip(feature_list, importances)]

# Sort the feature importances by most important first
feature_importances = sorted(feature_importances, key = lambda x: x[1], reverse = True)

# Print out the feature and importances
[print('Variable: {:20} Importance: {}'.format(*pair)) for pair in feature_importances];
```

```
Variable: hr                   Importance: 0.62
Variable: temp                 Importance: 0.12
Variable: yr                   Importance: 0.08
Variable: workingday           Importance: 0.06
Variable: hum                  Importance: 0.03
Variable: season               Importance: 0.02
Variable: mnth                 Importance: 0.02
Variable: weathersit           Importance: 0.02
Variable: atemp                Importance: 0.02
Variable: weekday              Importance: 0.01
Variable: windspeed            Importance: 0.01
Variable: holiday              Importance: 0.0
```

In [19]:
```
%matplotlib inline

# Set the style
plt.style.use('fivethirtyeight')

# list of x locations for plotting
x_values = list(range(len(importances)))

# Make a bar chart
plt.bar(x_values, importances, orientation = 'vertical')

# Tick labels for x axis
plt.xticks(x_values, feature_list, rotation='vertical')

# Axis labels and title
plt.ylabel('Importance'); plt.xlabel('Variable'); plt.title('Variable Importances');
```

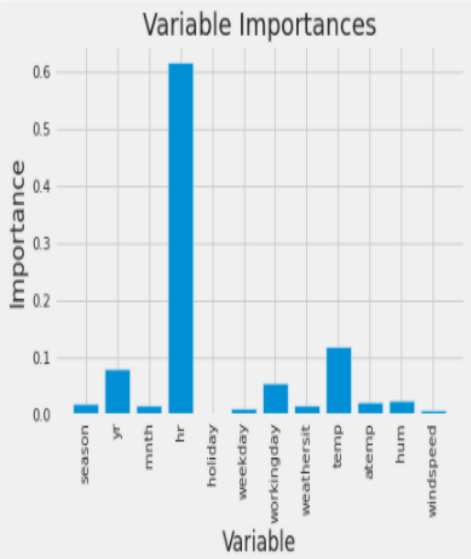



**Figure 8:** For a working day, the important features for predicting registered.

## CONCLUSION:

The result corresponds to the high correlation of the hour and temperature variable with the bicycle sharing count in the feature correlation matrix.

Hence, here hr and temp have highest correlation with hr having a importance of 0.62 and temp having a importance 0.12 of respectively.

**REFERENCES:**


- Bike-Sharing-Demand: https://www.kaggle.com/c/bike-sharing-demand/notebooks
- https://medium.com/@limavallantin/analysing-bike-sharing-trends-with-python-a9f574c596b9
- Research-papers: https://www.researchgate.net/publication/337062461_Regression_Model_for_Bike-Sharing_Service_by_Using_Machine_Learning
- Data set link: http://archive.ics.uci.edu/ml/datasets/Bike+Sharing+Dataset


**PROJECT LINK:**

https://github.com/d-b-tripura/Bike_Sharing/blob/master/BikeShare.ipynb