Introduction and Problem Definition: Machine learning is vastly growing with applications in every field of life. In this work, we have used Machine learning Linear regression & Gradient Boosting Regression model for the predition of bike rental count based on envirnomental and seasonal settings. The data set depicts bike sharing counts aggregated on hourly basis under different circumstances, weather condtitions and other variables that can effect the rental count prediction. This dataset is a mix of multi class discrete & contionous data,and the variable count which is a dependent variable. As the variable count do vary and is not a categorical variable, it can be predicted using regression models.

Importing Libraries:

In [1]:
```
#Importing Necessary Liberaries
import pandas
import numpy
import sklearn
from sklearn import metrics
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.metrics import accuracy_score
from sklearn.metrics import r2_score
```

Data Ingestion: This dataset consist of different variable, data types of which are given below: instant: int dteday : objec season : int yr : int mnth : int hr : int holiday : int weekday : int workingday : object weathersit : int temp : float atemp: float hum: float windspeed: float casual: int registered: int cnt: int

In [2]:
```
#Loading Data
Bike = pandas.read_csv(r"F:\Fiverr Work\bike-dataset\bike-dataset hour.csv")
Bike.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  object
 9   weathersit  17379 non-null  int64
 10  temp        15595 non-null  float64
 11  atemp       15595 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(11), object(2)
memory usage: 2.3+ MB
```

Data Preparation: In this section I have filled the missing values by means methods, which closesly mimicks the actual value. I changed the workingday variable data type, created two new columns, one for peak time, and other for whether it is night time or not. And also some of the variables were removed like holiday, month, instant, dteday, yr, casual, and registered. As there is no variable that require data binning in this case becasue the continous variable vary and could not be grouped in to smaller bins. Also the workingday variable was encoded.

In [3]:
```
#Converting datatype from string/object to boolean/integer

Bike['workingday'] = Bike['workingday'].map({'Yes': 1, 'No': 0})

Bike.info()

# Finding null values in data frame

print(Bike.isnull().values.any())
print(Bike.isnull().sum())

# Filling missing values with mean

Bike.temp.fillna(Bike.temp.mean(), inplace=True)
Bike.atemp.fillna(Bike.atemp.mean(), inplace=True)


#Removing unwanted columns
Bike.drop('holiday', axis = 1, inplace=True)
Bike.drop('mnth', axis = 1, inplace=True)
Bike.drop('instant', axis = 1, inplace=True)
Bike.drop('dteday', axis = 1, inplace=True)
Bike.drop('yr', axis = 1, inplace=True)
Bike.drop('casual', axis = 1, inplace=True)
Bike.drop('registered', axis = 1, inplace=True)

#Conditions or creating new column

conditions = [
    (Bike['hr'] >= 7) & (Bike['hr'] <= 9) & (Bike['workingday']==1) ,
    (Bike['hr'] >= 16) & (Bike['hr'] <= 19) & (Bike['workingday']==1),
    (Bike['hr'] >= 10) & (Bike['hr'] <= 16) & (Bike['workingday']==0)]
values =[1,1,1]
Bike['pt'] = numpy.select(conditions, values)

Bike['nighttime']= numpy.where((Bike['hr']>= 22) & (Bike['hr']<=4), 1, 0)
```

```
<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        15595 non-null  float64
 11  atemp       15595 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
True
instant          0
dteday           0
season           0
yr               0
mnth             0
hr               0
holiday          0
weekday          0
workingday       0
weathersit       0
temp          1784
atemp         1784
hum              0
windspeed        0
casual           0
registered       0
cnt              0
dtype: int64
```

Data Segregation: In this work, dataset was split in to train test with 80/20.

In [4]:

```
## Data segregation - test and train

x = Bike.drop('cnt',axis=1)
y = Bike['cnt']

xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size = 0.2, random_state
```

Model Training: Linear Regression and Gradient Boosting Regression models were used to train the data. As these both are well know for performing well on regression problems, both of the algorithm performed well on this dataset. And auto optimization of the model was used.

In [5]:
```
## LinearRegression Regression

clf =LinearRegression()
clf= clf.fit(xtrain, ytrain)
ypred_lr=clf.predict(xtest)

#Gradient Booster Regressor

gbr = GradientBoostingRegressor().fit(xtrain,ytrain)
ypred_gbr=gbr.predict(xtest)
```

Model Evaluation: Accuracy score was used for both the models. Also r-square was used for linear regression.

In [6]:
```
#R-Squared for linear regression
r_squared= clf.score(xtrain, ytrain)
print("R-Squared")
print(r_squared)

LL= r2_score(ytest, ypred_lr,multioutput='variance_weighted')

print(clf.score(xtest, ytest))
print(LL)
# Accuracy score for BGR
print(gbr.score(xtest, ytest))
```

```
R-Squared
0.624415826016074
0.6171332078677134
0.6171332078677134
0.8082490074787397
```

Conclusion: Evaluating the performance of both models, it seems that gradient boosting regression models worked well with an accuracy score of 80%. Gradient boosting regression model is better in making preditions of rental count based on the data.