In [1]:
```
import pandas as pd
import os
from warnings import filterwarnings
filterwarnings("ignore")

os.chdir(r'C:\Users\HP\Downloads\Bike Sharing Demand')
```

In [2]:
```
df = pd.read_csv('Bike Sharing Demand dataset.csv')
```

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

Out[3]:

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	cou
0	2011-01-01 00:00:00	1	0	0	1	9.84	14.395	81	0.0	3	13	
1	2011-01-01 01:00:00	1	0	0	1	9.02	13.635	80	0.0	8	32	
2	2011-01-01 02:00:00	1	0	0	1	9.02	13.635	80	0.0	5	27	
3	2011-01-01 03:00:00	1	0	0	1	9.84	14.395	75	0.0	3	10	
4	2011-01-01 04:00:00	1	0	0	1	9.84	14.395	75	0.0	0	1	

In [4]:
```
df.tail()
```

Out[4]:

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered
10881	2012-12-19 19:00:00	4	0	1	1	15.58	19.695	50	26.0027	7	329
10882	2012-12-19 20:00:00	4	0	1	1	14.76	17.425	57	15.0013	10	231
10883	2012-12-19 21:00:00	4	0	1	1	13.94	15.910	61	15.0013	4	164
10884	2012-12-19 22:00:00	4	0	1	1	13.94	17.425	61	6.0032	12	117
10885	2012-12-19 23:00:00	4	0	1	1	13.12	16.665	66	8.9981	4	84

In [5]:
```
df.shape
```

Out[5]:
```
(10886, 12)
```

In [6]:
```
df.columns
```

Out[6]:
```
Index(['datetime', 'season', 'holiday', 'workingday', 'weather', 'temp',
       'atemp', 'humidity', 'windspeed', 'casual', 'registered', 'count'],
      dtype='object')
```

In [7]:
```
#check if there is are missing values in dataset and take sum of all missing values
df.isnull().sum()
```

Out[7]:
```
datetime      0
season        0
holiday       0
workingday    0
weather       0
temp          0
atemp         0
humidity      0
windspeed     0
casual        0
registered    0
count         0
dtype: int64
```

In [8]:
```
df.dropna(inplace=True) #if you want to delete missing values in data
```

In [9]:
```
#describtive statistics
df.describe()
```

Out[9]:

	season	holiday	workingday	weather	temp	atemp	humidity	winds
count	10886.000000	10886.000000	10886.000000	10886.000000	10886.00000	10886.000000	10886.000000	10886.00
mean	2.506614	0.028569	0.680875	1.418427	20.23086	23.655084	61.886460	12.79
std	1.116174	0.166599	0.466159	0.633839	7.79159	8.474601	19.245033	8.16
min	1.000000	0.000000	0.000000	1.000000	0.82000	0.760000	0.000000	0.00
25%	2.000000	0.000000	0.000000	1.000000	13.94000	16.665000	47.000000	7.00
50%	3.000000	0.000000	1.000000	1.000000	20.50000	24.240000	62.000000	12.99
75%	4.000000	0.000000	1.000000	2.000000	26.24000	31.060000	77.000000	16.99
max	4.000000	1.000000	1.000000	4.000000	41.00000	45.455000	100.000000	56.99

In [10]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10886 entries, 0 to 10885
Data columns (total 12 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   datetime    10886 non-null  object 
 1   season      10886 non-null  int64  
 2   holiday     10886 non-null  int64  
 3   workingday  10886 non-null  int64  
 4   weather     10886 non-null  int64  
 5   temp        10886 non-null  float64
 6   atemp       10886 non-null  float64
 7   humidity    10886 non-null  int64  
 8   windspeed   10886 non-null  float64
 9   casual      10886 non-null  int64  
 10  registered  10886 non-null  int64  
 11  count       10886 non-null  int64  
dtypes: float64(3), int64(8), object(1)
memory usage: 1020.7+ KB
```

In [11]:
```
import datetime
df["datetime"] = pd.to_datetime(df["datetime"])

# Another way
# df["datetime"].apply(pd.to_datetime, infer_datetime_format=True,errors="coerce")
```

In [12]:
```
df = df.drop("datetime", axis=1)
```

In [13]:
```
# split data
x = df.iloc[:, :-1] # all row and all col except last col
y = df.iloc[:, -1]  # just last col
```

In [14]:
```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x,y,random_state=100)
```

In [15]:
```
from xgboost import XGBRegressor
```

In [16]:
```
xg_reg = XGBRegressor()
```

In [17]:
```
xg_reg.fit(x_train, y_train)
```

Out[17]:
```
XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1,
             colsample_bynode=1, colsample_bytree=1, enable_categorical=False,
             gamma=0, gpu_id=-1, importance_type=None,
             interaction_constraints='', learning_rate=0.300000012,
             max_delta_step=0, max_depth=6, min_child_weight=1, missing=nan,
             monotone_constraints='()', n_estimators=100, n_jobs=8,
             num_parallel_tree=1, predictor='auto', random_state=0, reg_alpha=0,
             reg_lambda=1, scale_pos_weight=1, subsample=1, tree_method='exact',
             validate_parameters=1, verbosity=None)
```

In [18]:
```
# make prediction
y_pred = xg_reg.predict(x_test)
```

In [19]:
```
y_pred
```

Out[19]:
```
array([504.5948 , 147.46849, 277.1645 , ..., 370.84723, 447.6926 ,
       327.49417], dtype=float32)
```

In [20]:
```
from sklearn.metrics import mean_squared_error
```

In [21]:
```
mse = mean_squared_error(y_test, y_pred)
print("MSE Value", mse)
```

```
MSE Value 14.767791694756506
```

In [22]:
```
# cross validation to improve accuracy
from sklearn.model_selection import cross_val_score
```

In [23]:
```
model = XGBRegressor(objective = "reg:squarederror")
```

In [24]:
```
scores  =cross_val_score(model, x, y, scoring= "neg_mean_squared_error", cv=10)
```

In [25]:
```
scores
```

Out[25]:
```
array([ -1.96497946, -14.45037442, -11.59824682, -11.24511197,
         -4.50380405,  -4.59030974, -33.65379986, -18.28033932,
        -51.17683587, -13.25094222])
```

In [ ]:
```

```