# Problem statement

predicting the house price in USA.To create a model to help him estimate of what the house would sell for.

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
df=pd.read_csv("Sleep_health")
```

# To display top 10 rows

In [3]:
```
df.head(10)
```
Out[3]:

| | Person ID | Gender | Age | Occupation | Sleep Duration | Quality of Sleep | Physical Activity Level | Stress Level | BMI Category | Blood Pressure | Heart Rate | Daily Steps | Sleep Disorder |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | Male | 27 | Software Engineer | 6.1 | 6 | 42 | 6 | Overweight | 126/83 | 77 | 4200 | None |
| **1** | 2 | Male | 28 | Doctor | 6.2 | 6 | 60 | 8 | Normal | 125/80 | 75 | 10000 | None |
| **2** | 3 | Male | 28 | Doctor | 6.2 | 6 | 60 | 8 | Normal | 125/80 | 75 | 10000 | None |
| **3** | 4 | Male | 28 | Sales Representative | 5.9 | 4 | 30 | 8 | Obese | 140/90 | 85 | 3000 | Sleep Apnea |
| **4** | 5 | Male | 28 | Sales Representative | 5.9 | 4 | 30 | 8 | Obese | 140/90 | 85 | 3000 | Sleep Apnea |
| **5** | 6 | Male | 28 | Software Engineer | 5.9 | 4 | 30 | 8 | Obese | 140/90 | 85 | 3000 | Insomnia |
| **6** | 7 | Male | 29 | Teacher | 6.3 | 6 | 40 | 7 | Obese | 140/90 | 82 | 3500 | Insomnia |
| **7** | 8 | Male | 29 | Doctor | 7.8 | 7 | 75 | 6 | Normal | 120/80 | 70 | 8000 | None |
| **8** | 9 | Male | 29 | Doctor | 7.8 | 7 | 75 | 6 | Normal | 120/80 | 70 | 8000 | None |
| **9** | 10 | Male | 29 | Doctor | 7.8 | 7 | 75 | 6 | Normal | 120/80 | 70 | 8000 | None |

# Data Cleaning And Pre-Processing

In [4]:
```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 374 entries, 0 to 373
Data columns (total 13 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   Person ID                374 non-null    int64  
 1   Gender                   374 non-null    object 
 2   Age                      374 non-null    int64  
 3   Occupation               374 non-null    object 
 4   Sleep Duration           374 non-null    float64
 5   Quality of Sleep         374 non-null    int64  
 6   Physical Activity Level  374 non-null    int64  
 7   Stress Level             374 non-null    int64  
 8   BMI Category             374 non-null    object 
 9   Blood Pressure           374 non-null    object 
 10  Heart Rate               374 non-null    int64  
 11  Daily Steps              374 non-null    int64  
 12  Sleep Disorder           374 non-null    object 
dtypes: float64(1), int64(7), object(5)
memory usage: 38.1+ KB
```

In [5]:
```python
# Display the statistical summary
df.describe()
```

Out[5]:

| | Person ID | Age | Sleep Duration | Quality of Sleep | Physical Activity Level | Stress Level | Heart Rate | Daily Steps |
|---|---|---|---|---|---|---|---|---|
| **count** | 374.000000 | 374.000000 | 374.000000 | 374.000000 | 374.000000 | 374.000000 | 374.000000 | 374.000000 |
| **mean** | 187.500000 | 42.184492 | 7.132086 | 7.312834 | 59.171123 | 5.385027 | 70.165775 | 6816.844920 |
| **std** | 108.108742 | 8.673133 | 0.795657 | 1.196956 | 20.830804 | 1.774526 | 4.135676 | 1617.915679 |
| **min** | 1.000000 | 27.000000 | 5.800000 | 4.000000 | 30.000000 | 3.000000 | 65.000000 | 3000.000000 |
| **25%** | 94.250000 | 35.250000 | 6.400000 | 6.000000 | 45.000000 | 4.000000 | 68.000000 | 5600.000000 |
| **50%** | 187.500000 | 43.000000 | 7.200000 | 7.000000 | 60.000000 | 5.000000 | 70.000000 | 7000.000000 |
| **75%** | 280.750000 | 50.000000 | 7.800000 | 8.000000 | 75.000000 | 7.000000 | 72.000000 | 8000.000000 |
| **max** | 374.000000 | 59.000000 | 8.500000 | 9.000000 | 90.000000 | 8.000000 | 86.000000 | 10000.000000 |

In [6]:
```
# To display the col headings
df.columns
```

Out[6]:
```
Index(['Person ID', 'Gender', 'Age', 'Occupation', 'Sleep Duration',
       'Quality of Sleep', 'Physical Activity Level', 'Stress Level',
       'BMI Category', 'Blood Pressure', 'Heart Rate', 'Daily Steps',
       'Sleep Disorder'],
      dtype='object')
```

In [7]:
```
cols=df.dropna(axis=1)
```

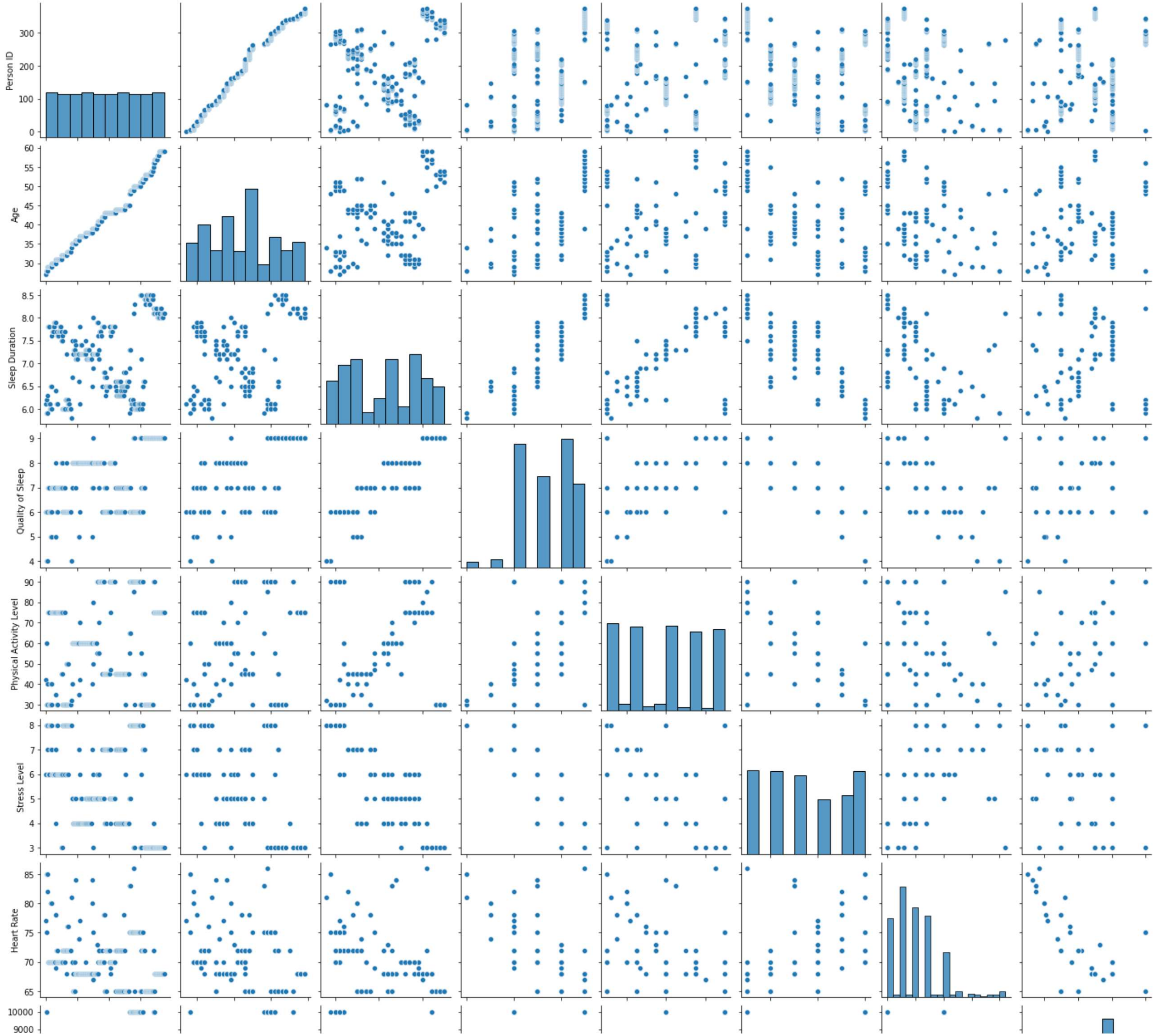
In [8]:
```
cols.columns
```

Out[8]:
```
Index(['Person ID', 'Gender', 'Age', 'Occupation', 'Sleep Duration',
       'Quality of Sleep', 'Physical Activity Level', 'Stress Level',
       'BMI Category', 'Blood Pressure', 'Heart Rate', 'Daily Steps',
       'Sleep Disorder'],
      dtype='object')
```

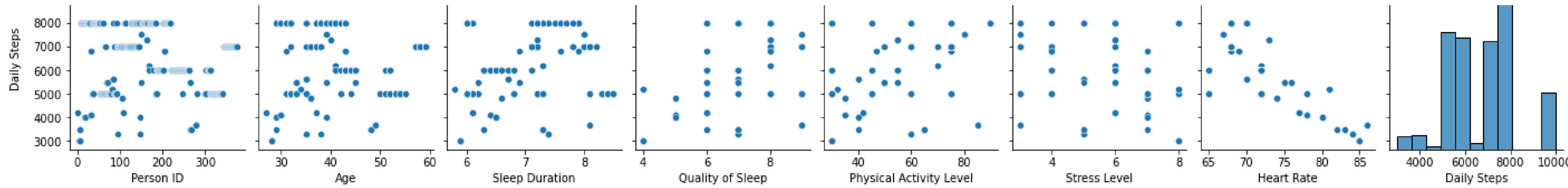
# EDA and Visualization

In [9]:
```
sns.pairplot(cols)
```

Out[9]:
```
<seaborn.axisgrid.PairGrid at 0x20092c3d610>
```

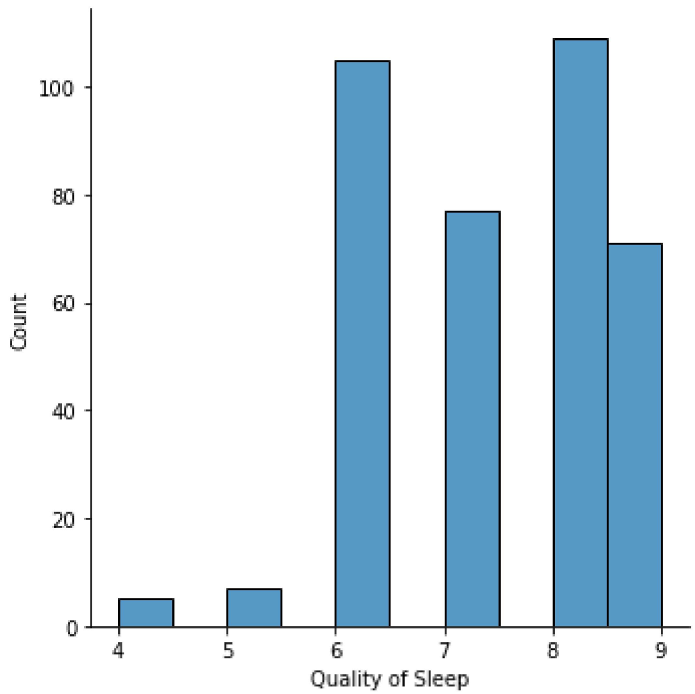


Person ID
Age
Sleep Duration
Quality of Sleep
Physical Activity Level
Stress Level
Heart Rate




In [10]:
```
1 sns.displot(df['Quality of Sleep'])
```

Out[10]: <seaborn.axisgrid.FacetGrid at 0x20095ead3a0>



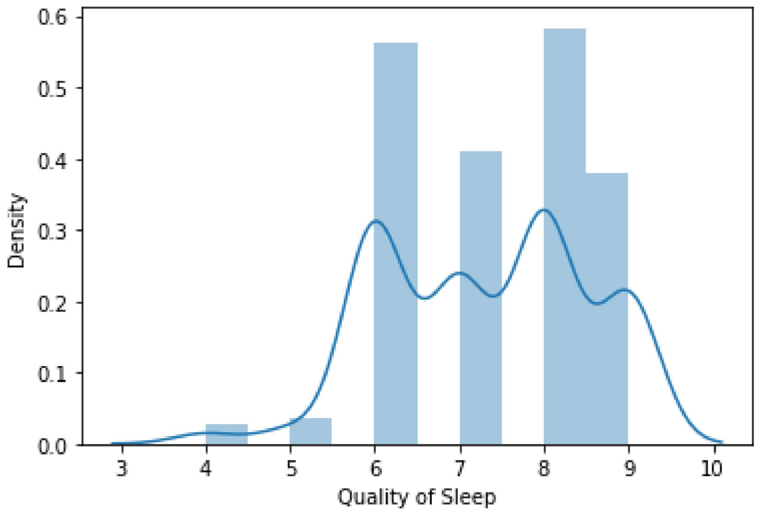
In [11]:

```
# We use displot in older version we get distplot use displot
sns.distplot(df['Quality of Sleep'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a dep
recated function and will be removed in a future version. Please adapt your code to use either `displot` (a
figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[11]:

```
<AxesSubplot:xlabel='Quality of Sleep', ylabel='Density'>
```



In [12]:

```
df1=cols[['Quality of Sleep', 'Physical Activity Level', 'Stress Level']]
df1
```

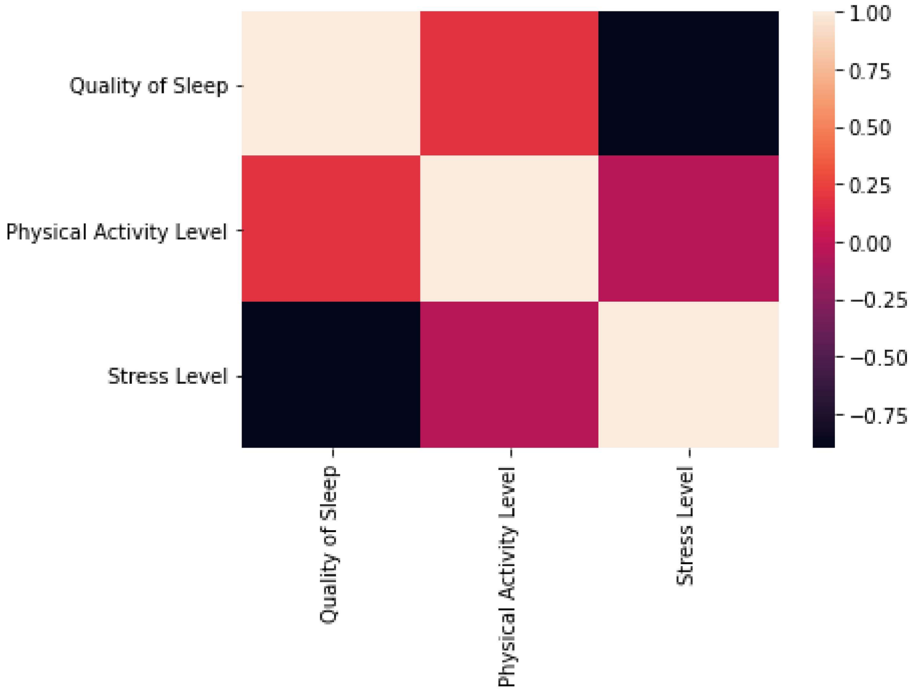
Out[12]:

| | Quality of Sleep | Physical Activity Level | Stress Level |
|---|---|---|---|
| **0** | 6 | 42 | 6 |
| **1** | 6 | 60 | 8 |
| **2** | 6 | 60 | 8 |
| **3** | 4 | 30 | 8 |
| **4** | 4 | 30 | 8 |
| **...** | ... | ... | ... |
| **369** | 9 | 75 | 3 |
| **370** | 9 | 75 | 3 |
| **371** | 9 | 75 | 3 |
| **372** | 9 | 75 | 3 |
| **373** | 9 | 75 | 3 |

374 rows × 3 columns

```
In [13]: sns.heatmap(df1.corr())
```

Out[13]: <AxesSubplot:>




# To train the model - MODEL BUILD

Going to train linear regression model;We split our data into 2 variables x and y where x is independent var(input) and y is dependent on x(output), we could ignore address col as it is not required for our model

```
In [14]: x=df1[['Quality of Sleep', 'Physical Activity Level', 'Stress Level']]
         y=df1[['Quality of Sleep']]
```

# To split the dataset into test data

In [15]:
```
# importing lib for splitting test data
from sklearn.model_selection import train_test_split
```

In [16]:
```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.3)
```

In [17]:
```
from sklearn.linear_model import LinearRegression

lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[17]: LinearRegression()

In [18]:
```
print(lr.intercept_)
```

```
[-1.77635684e-15]
```

In [19]:
```
print(lr.score(x_test,y_test))
```

```
1.0
```

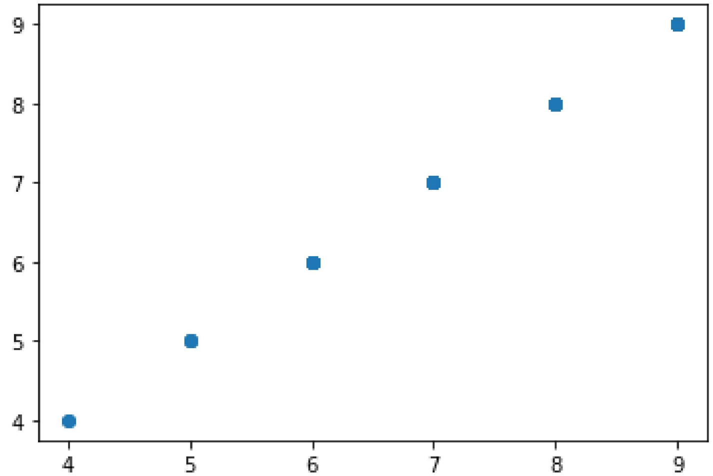
In [20]:
```
coeff=pd.DataFrame(lr.coef_)
coeff
```

Out[20]:

| | 0 | 1 | 2 |
|---|---|---|---|
| 0 | 1.0 | 8.111872e-17 | -1.707167e-16 |

In [21]:
```
pred = lr.predict(x_test)
plt.scatter(y_test,pred)
```

Out[21]: <matplotlib.collections.PathCollection at 0x2009828f6d0>




In [22]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [23]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[23]: Ridge(alpha=10)

In [24]:
```
rr.score(x_test,y_test)
```

Out[24]: 0.9969900659175175

In [25]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[25]: Lasso(alpha=10)

In [26]:
```
la.score(x_test,y_test)
```

Out[26]: -0.0032932238920933

# ELASTIC NET

In [27]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[27]:
```
ElasticNet()
```

In [28]:
```
print(en.coef_)
```

```
[ 0.17198479  0.00581034 -0.28979061]
```

In [29]:
```
print(en.intercept_)
```

```
[7.25607742]
```

In [30]:
```
prediction=en.predict(x_test)
prediction
```

Out[30]:
```
array([7.531623  , 8.37034408, 7.04079592, 7.70593316, 6.14397145,
       7.82141361, 8.37034408, 8.37034408, 6.31988066, 8.37034408,
       5.81162255, 7.531623  , 7.56227374, 8.37034408, 7.82141361,
       7.82141361, 7.15700269, 6.49259177, 7.18372901, 6.14397145,
       6.52091714, 6.31988066, 8.10887884, 8.37034408, 6.14397145,
       8.37034408, 7.82141361, 6.14397145, 7.531623  , 8.37034408,
       7.15700269, 7.82141361, 7.70593316, 7.15700269, 8.45749916,
       6.52091714, 8.42844747, 7.531623  , 8.10887884, 7.82141361,
       7.56227374, 7.70593316, 7.70593316, 8.10887884, 6.49259177,
       7.531623  , 8.37034408, 8.37034408, 6.49259177, 6.69290192,
       7.15700269, 8.02404914, 7.04079592, 6.29082896, 6.14397145,
       7.82141361, 8.37034408, 6.14397145, 6.14397145, 8.37034408,
       7.56227374, 7.15700269, 6.14397145, 7.56227374, 7.15700269,
       8.37034408, 7.82141361, 6.14397145, 8.10887884, 8.10887884,
       8.10887884, 7.15700269, 7.531623  , 7.82141361, 6.14397145,
       7.531623  , 7.56227374, 7.2127807 , 7.531623  , 7.70593316,
       6.29082896, 6.31828161, 6.49259177, 7.15700269, 7.70593316,
       6.52091714, 6.52091714, 7.70593316, 8.37034408, 6.14397145,
       7.35963822, 5.80000188, 6.14397145, 7.531623  , 7.15700269,
       6.69290192, 7.70593316, 7.70593316, 7.15700269, 7.15700269,
       7.38868991, 6.49259177, 7.33058652, 6.31828161, 8.37034408,
       8.10887884, 7.531623  , 6.49259177, 6.89553745, 6.14397145,
       7.15700269, 6.49259177, 8.39939578])
```

In [31]:
```
print(en.score(x_test,y_test))
```

```
0.7950486692003498
```

# EVALUATION METRICS

In [32]:
```
from sklearn import metrics
```

In [33]:

```
print("Mean Absolute Error:",metrics.mean_absolute_error(y_test,prediction))
```

```
Mean Absolute Error: 0.45495704210174887
```

In [34]:

```
print("Mean Squared Error:",metrics.mean_squared_error(y_test,prediction))
```

```
Mean Squared Error: 0.31944094177888893
```

In [35]:

```
print("Root Mean Squared Error:",np.sqrt(metrics.mean_squared_error(y_test,prediction)))
```

```
Root Mean Squared Error: 0.5651910666127777
```

In [ ]:

```

```