MULTIPLE LINEAR REGRESSION

ABOUT THE DATASET 'Student's Performance'

Variable Notes

1. Hours Studied: The total number of hours spent studying by each student.
2. Previous Scores: The scores obtained by students in previous tests.
3. Extracurricular Activities: Whether the student participates in extracurricular activities (Yes or No).
4. Sleep Hours: The average number of hours of sleep the student had per day.
5. Sample Question Papers Practiced: The number of sample question papers the student practiced.

DATA COLLECTION AND EXPLORATION

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

In [2]:
```
# LOAD THE DATASET
data = pd.read_csv('Student_Performance.csv')
data
```

Out[2]:

	Hours Studied	Previous Scores	Extracurricular Activities	Sleep Hours	Sample Question Papers Practiced	Performance Index
0	7	99	Yes	9	1	91.0
1	4	82	No	4	2	65.0
2	8	51	Yes	7	2	45.0
3	5	52	Yes	5	2	36.0
4	7	75	No	8	5	66.0
...	...	...	...	...	...	...
9995	1	49	Yes	4	2	23.0
9996	7	64	Yes	8	5	58.0
9997	6	83	Yes	8	5	74.0
9998	9	97	Yes	7	0	95.0
9999	7	74	No	8	1	64.0

10000 rows × 6 columns

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

Out[3]:

	Hours Studied	Previous Scores	Extracurricular Activities	Sleep Hours	Sample Question Papers Practiced	Performance Index
0	7	99	Yes	9	1	91.0
1	4	82	No	4	2	65.0
2	8	51	Yes	7	2	45.0
3	5	52	Yes	5	2	36.0
4	7	75	No	8	5	66.0

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

Out[4]:

	Hours Studied	Previous Scores	Extracurricular Activities	Sleep Hours	Sample Question Papers Practiced	Performance Index
9995	1	49	Yes	4	2	23.0
9996	7	64	Yes	8	5	58.0
9997	6	83	Yes	8	5	74.0
9998	9	97	Yes	7	0	95.0
9999	7	74	No	8	1	64.0

In [5]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 6 columns):
 #   Column                          Non-Null Count  Dtype  
---  ------                          --------------  -----  
 0   Hours Studied                   10000 non-null  int64  
 1   Previous Scores                 10000 non-null  int64  
 2   Extracurricular Activities      10000 non-null  object 
 3   Sleep Hours                     10000 non-null  int64  
 4   Sample Question Papers Practiced  10000 non-null  int64  
 5   Performance Index               10000 non-null  float64
dtypes: float64(1), int64(4), object(1)
memory usage: 468.9+ KB
```

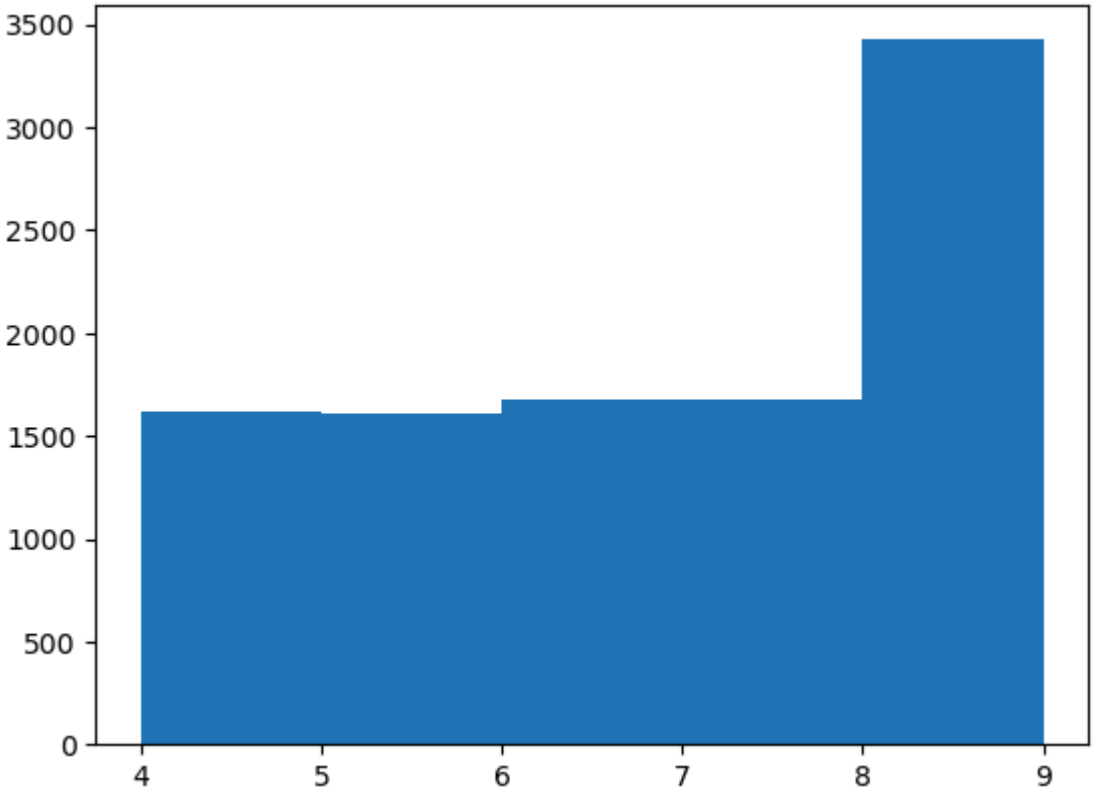
In [6]:
```
data.describe()
```

Out[6]:

	Hours Studied	Previous Scores	Sleep Hours	Sample Question Papers Practiced	Performance Index
count	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000
mean	4.992900	69.445700	6.530600	4.583300	55.224800
std	2.589309	17.343152	1.695863	2.867348	19.212558
min	1.000000	40.000000	4.000000	0.000000	10.000000
25%	3.000000	54.000000	5.000000	2.000000	40.000000
50%	5.000000	69.000000	7.000000	5.000000	55.000000
75%	7.000000	85.000000	8.000000	7.000000	71.000000
max	9.000000	99.000000	9.000000	9.000000	100.000000

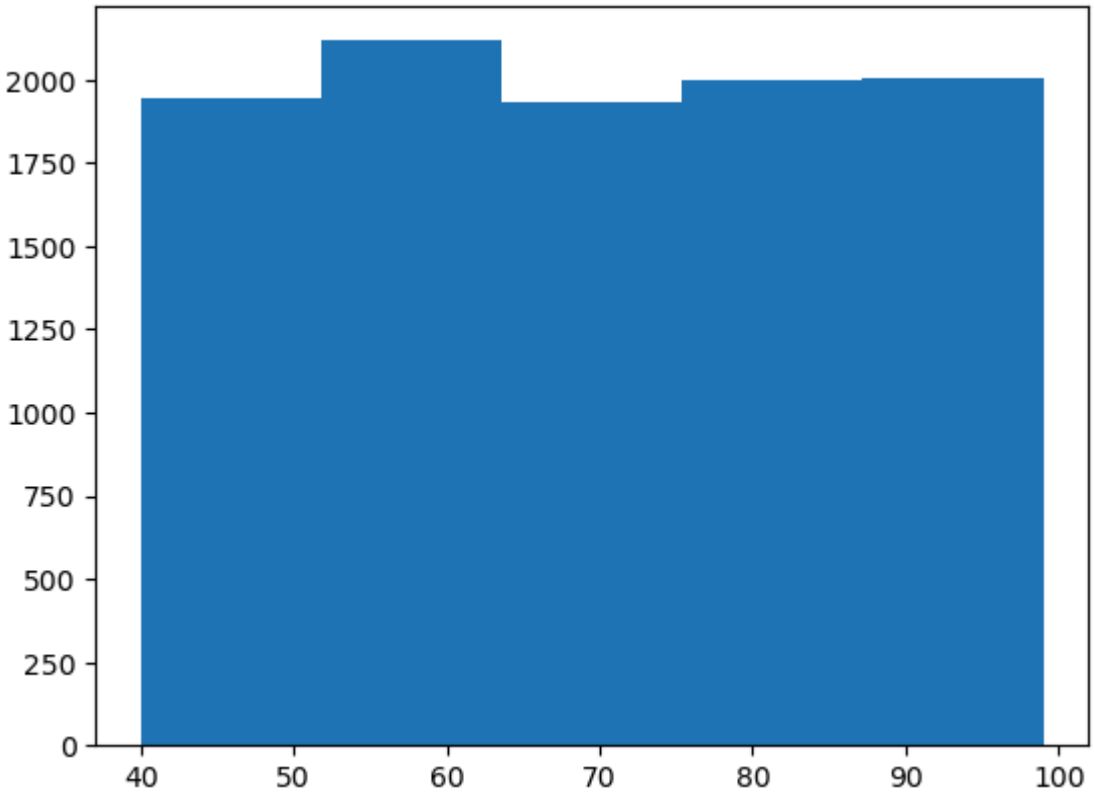
In [46]:
```
plt.hist(data['Sleep Hours'],bins=5)
```

Out[46]:
```
(array([1619., 1606., 1673., 1676., 3426.]),
 array([4., 5., 6., 7., 8., 9.]),
 <BarContainer object of 5 artists>)
```



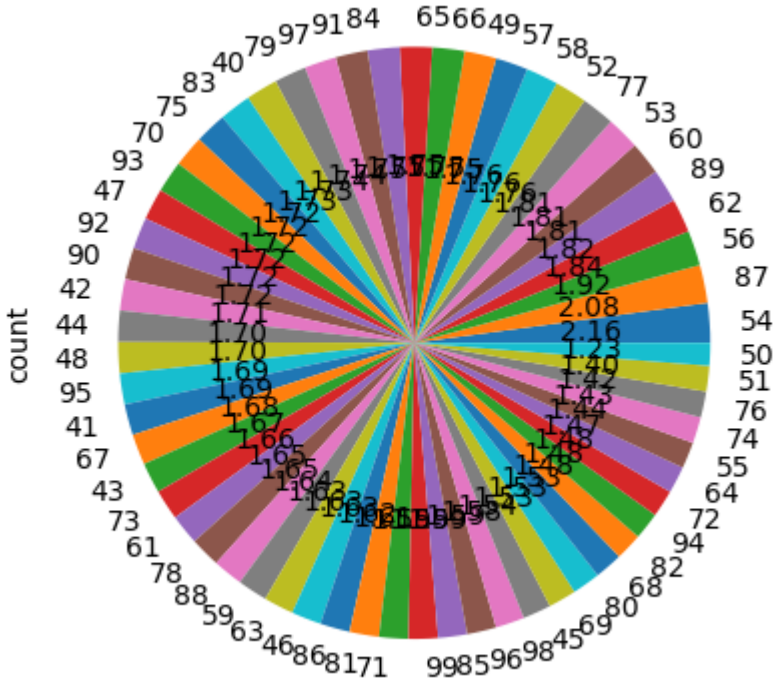
In [48]:
```
plt.hist(data['Previous Scores'],bins=5)
```

Out[48]:
```
(array([1947., 2117., 1931., 2001., 2004.]),
 array([40. , 51.8, 63.6, 75.4, 87.2, 99. ]),
 <BarContainer object of 5 artists>)
```



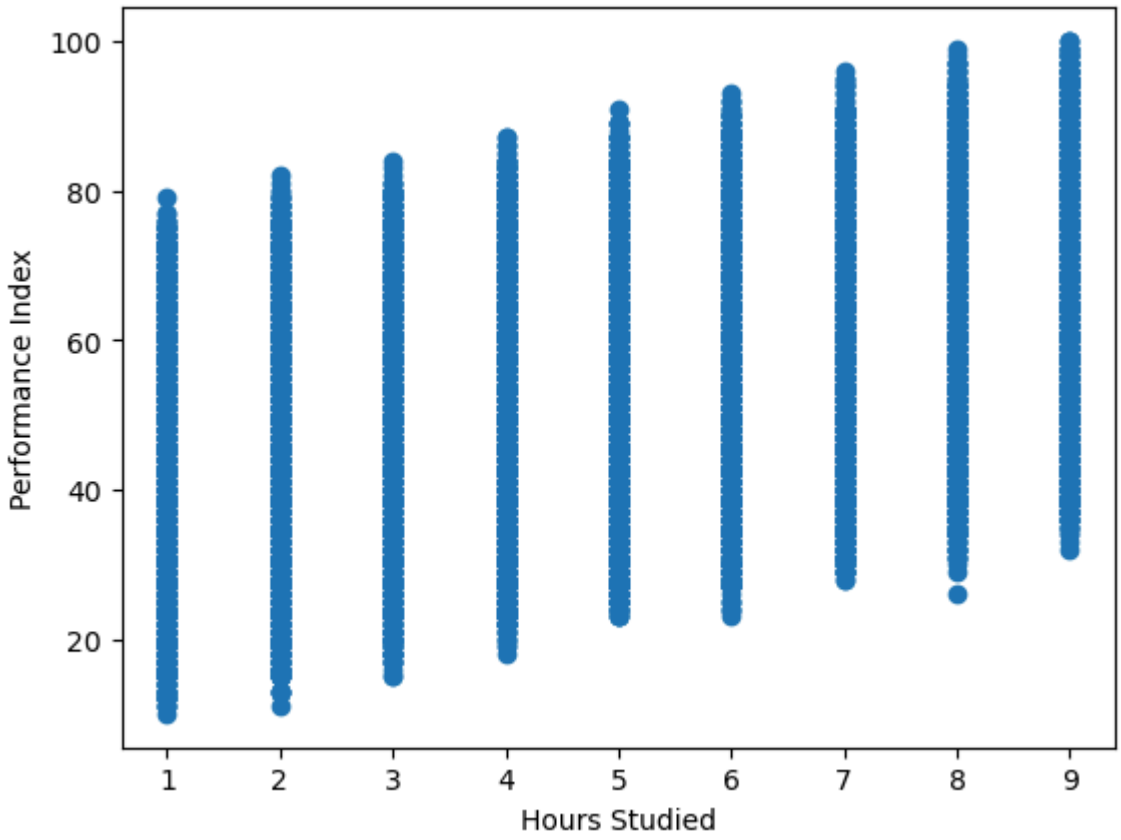
In [49]:
```
data['Previous Scores'].value_counts().plot(kind='pie', autopct='%.2f')
```

Out[49]:
```
<Axes: ylabel='count'>
```



In [51]:
```
plt.scatter(data['Hours Studied'],data['Performance Index'])
plt.xlabel("Hours Studied")
plt.ylabel("Performance Index")
```

Out[51]:
```
Text(0, 0.5, 'Performance Index')
```

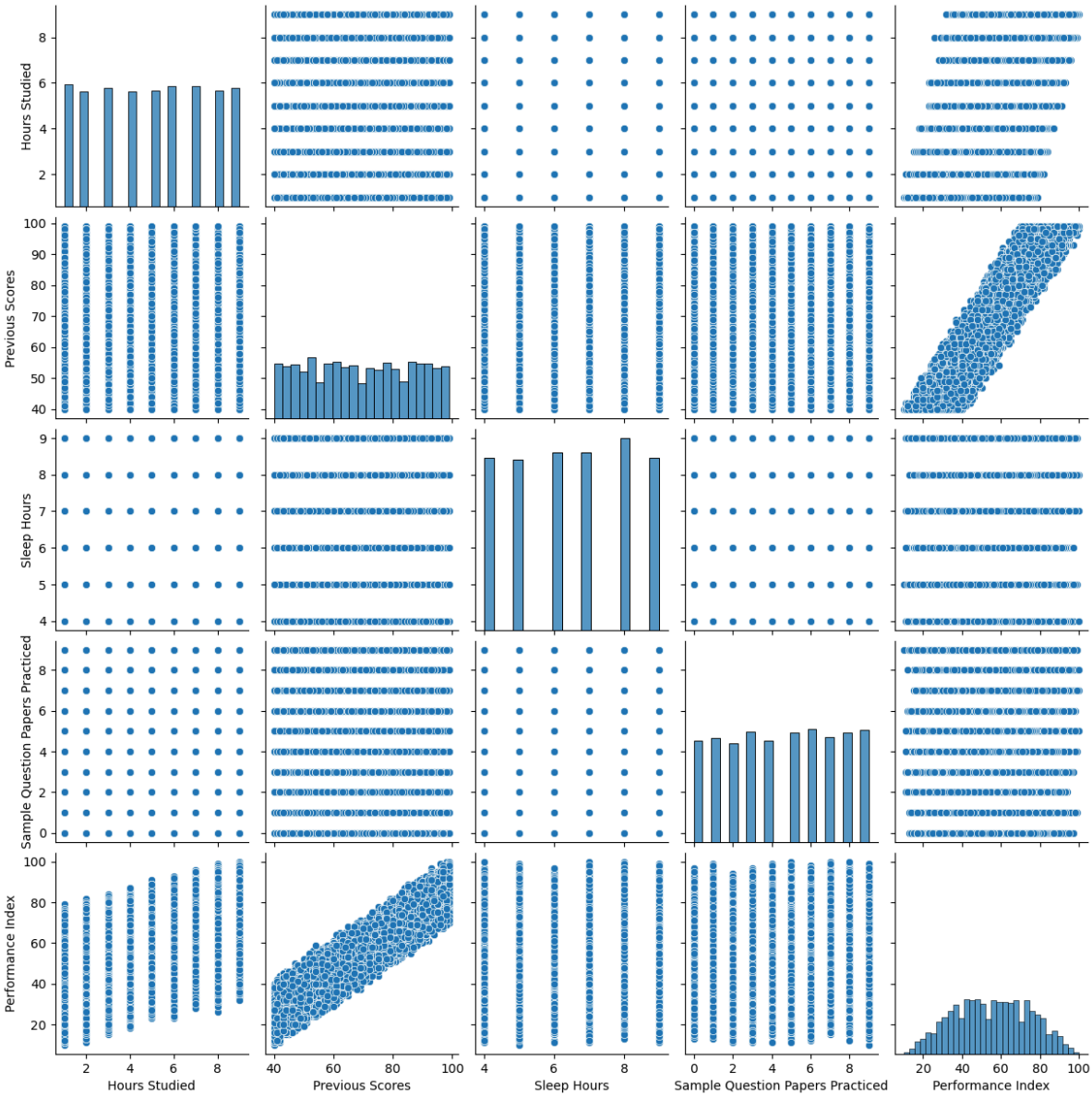


Visualization

In [8]:
```
import seaborn as sns
sns.pairplot(data)
```

```
C:\Users\Nishita Bala\anaconda3\Lib\site-packages\seaborn\axisgrid.py:118: UserWar
ning: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```

Out[8]:
```
<seaborn.axisgrid.PairGrid at 0x1d00a1cc250>
```



In [11]:
```
data.drop(['Extracurricular Activities'], axis=1, inplace=True)
data.corr()
```

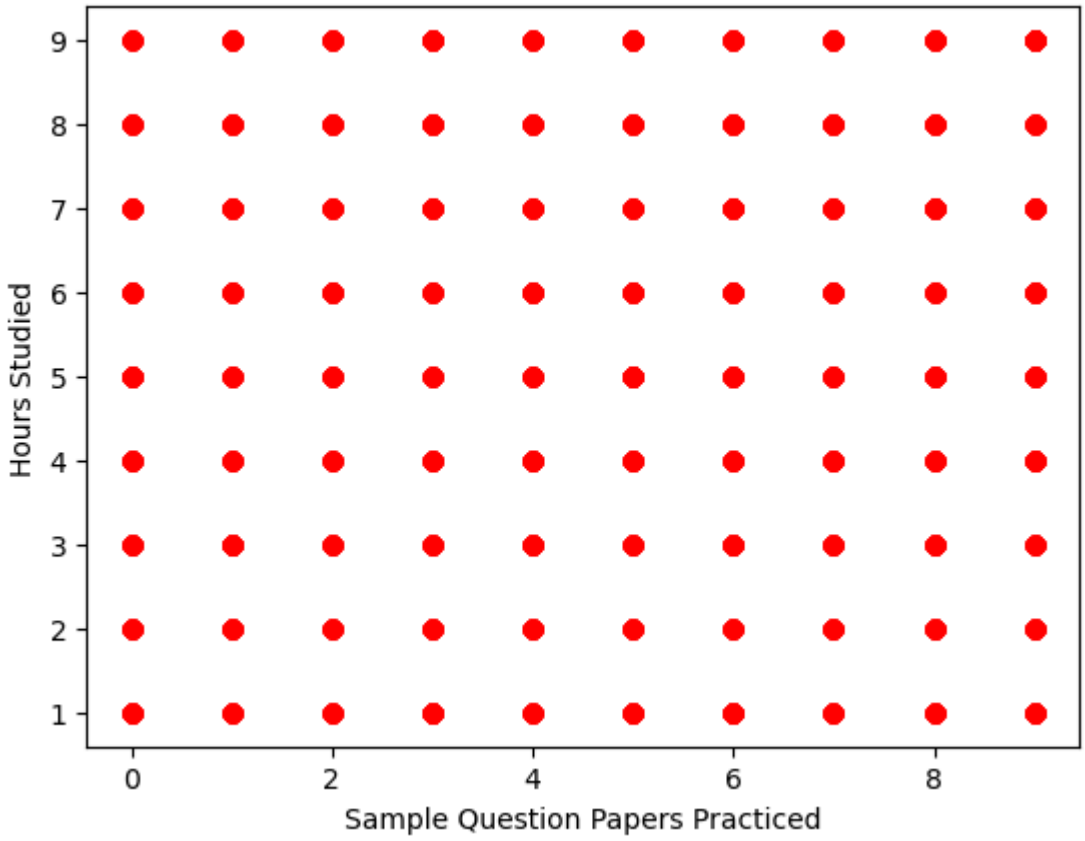
Out[11]:

	Hours Studied	Previous Scores	Sleep Hours	Sample Question Papers Practiced	Performance Index
Hours Studied	1.000000	-0.012390	0.001245	0.017463	0.373730
Previous Scores	-0.012390	1.000000	0.005944	0.007888	0.915189
Sleep Hours	0.001245	0.005944	1.000000	0.003990	0.048106
Sample Question Papers Practiced	0.017463	0.007888	0.003990	1.000000	0.043268
Performance Index	0.373730	0.915189	0.048106	0.043268	1.000000

VISUALIZE THE DATAPOINTS MORE CLOSELY

In [14]:
```
plt.scatter(data['Sample Question Papers Practiced'],data['Hours Studied'],color='r
plt.xlabel("Sample Question Papers Practiced")
plt.ylabel("Hours Studied")
```

Out[14]:
```
Text(0, 0.5, 'Hours Studied')
```



In [15]:
```
## INDEPENDENT AND DEPENDENT
X = data.iloc[:,:-1]
Y = data.iloc[:,-1]
```

In [17]:
```
X.head()
```

Out[17]:

	Hours Studied	Previous Scores	Sleep Hours	Sample Question Papers Practiced
0	7	99	9	1
1	4	82	4	2
2	8	51	7	2
3	5	52	5	2
4	7	75	8	5

In [18]:
```
Y.head()
```

Out[18]:
```
0    91.0
1    65.0
2    45.0
3    36.0
4    66.0
Name: Performance Index, dtype: float64
```

TRAIN TEST SPLIT

The code uses train_test_split from scikit-learn to split dataset X and Y into training and testing sets. It assigns 25% to testing (test_size=0.25) and ensures reproducibility with random_state=42

In [20]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, Y_train, Y_test = train_test_split(X,Y,test_size=0.25, random_stat
```

In [22]:
```
X_train.shape
```

Out[22]:
```
(7500, 4)
```

In [23]:
```
X_test.shape
```

Out[23]:
```
(2500, 4)
```

In [24]:
```
Y_train.shape
```

Out[24]:
```
(7500,)
```

In [26]:
```
Y_test.shape
```

Out[26]:
```
(2500,)
```

STANDARDIZATION

The code initializes a StandardScaler to normalize data. It fits and transforms X_train, scaling its features to zero mean and unit variance, and applies the same transformation to X_test.

In [29]:
```
from sklearn.preprocessing import StandardScaler
```

In [31]:
```
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.fit_transform(X_test)
```

In [32]:
```
X_train
```

Out[32]:
```
array([[ 0.00796272, -1.19062085, -0.90874945,  0.13314392],
       [ 0.77854894,  1.05902612, -1.49731775,  1.53122497],
       [-0.76262349,  1.40512565,  0.26838714, -1.26493713],
       ...,
       [ 1.54913515, -1.24830411,  0.26838714,  0.48266418],
       [-1.5332097 , -1.30598737,  1.44552374, -1.6144574 ],
       [-1.14791659, -1.36367062, -0.32018115,  0.48266418]])
```

APPLY LINEAR REGRESSION

The code imports the LinearRegression model from sklearn.linear_model, initializes it with n_jobs=-1 for parallel processing, and fits the model to training data X_train and Y_train.

In [34]:
```
from sklearn.linear_model import LinearRegression
regression = LinearRegression()
```

In [35]:
```
regression.fit(X_train,Y_train)
```

Out[35]:
```
▾ LinearRegression
LinearRegression()
```

In [43]:
```
print("Coefficient or slope (Beta1):",regression.coef_)
print("Intercept (Beta0):",regression.intercept_)
```

```
Coefficient or slope (Beta1): [ 7.41203862 17.6216476   0.8042706   0.5448865 ]
Intercept (Beta0): 55.4184
```

CROSS VALIDATION

In [36]:
```
from sklearn.model_selection import cross_val_score
validation_score = cross_val_score(regression, X_train, Y_train, scoring='neg_mean_
```

In [38]:
```
np.mean(validation_score)
```

Out[38]:
```
-4.286929433512026
```

PREDICTION FOR TEST DATA

In [40]:
```
Y_pred = regression.predict(X_test)
Y_pred
```

Out[40]:
```
array([55.71531998, 22.92497608, 48.36823638, ..., 69.03877563,
       54.63586201, 55.34841244])
```

PERFORMANCE MATRICS

In [41]:
```
from sklearn.metrics import mean_absolute_error, mean_squared_error
```

In [42]:
```
mse = mean_squared_error(Y_test,Y_pred)
mae = mean_absolute_error(Y_test,Y_pred)
rmse = np.sqrt(mse)

print("Mean Square Error",mse)
print("Mean Absolute",mae)
print("Root Mean Square Error",rmse)
```

```
Mean Square Error 4.7576433734708345
Mean Absolute 1.7429353096711198
Root Mean Square Error 2.1812022770643797
```