```
import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

salary = pd.read_csv('https://github.com/YBI-Foundation/Dataset/raw/main/Salary%20Data.csv')

salary.head()
```

	Experience Years	Salary
0	1.1	39343
1	1.2	42774
2	1.3	46205
3	1.5	37731
4	2.0	43525

```
salary.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 40 entries, 0 to 39
Data columns (total 2 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   Experience Years  40 non-null     float64
 1   Salary            40 non-null     int64  
dtypes: float64(1), int64(1)
memory usage: 768.0 bytes
```

```
salary.describe()
```

	Experience Years	Salary
count	40.000000	40.000000
mean	5.152500	74743.625000
std	2.663715	25947.122885
min	1.100000	37731.000000
25%	3.200000	56878.250000
50%	4.600000	64472.500000
75%	6.875000	95023.250000
max	10.500000	122391.000000

Define y and x

```
y = salary['Salary']   #always use single square bracket
```

```
y.shape
```

```
(40,)
```

Split Data

```
from sklearn.model_selection import train_test_split
```

```
X = salary[['Experience Years']] #always use double square bracket
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size = 0.8, random_state = 2529)
```

```
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

```
((32, 1), (8, 1), (32,), (8,))
```

```
from sklearn.linear_model import LinearRegression
```

```
model = LinearRegression()
```

```
model.fit(X_train, y_train)
```

```
LinearRegression()
```

```
model.coef_
```

```
array([9398.19785815])
```

```
model.intercept_
```

```
26344.85810217697
```

```
y_pred = model.predict(X_test)
```

```
y_pred
```

```
array([ 90252.60353757,  59238.55060569, 106229.53989642,  63937.64953476,
        68636.74846383, 123146.29604108,  84613.68482268,  62997.82974895])
```

```
from sklearn.metrics import mean_absolute_percentage_error
```

```
mean_absolute_percentage_error(y_test, y_pred)
```

```
0.050339688879149136
```

Problem 1 : Predict SAT with GPA

```
sat = pd.read_csv('https://github.com/ybifoundation/Dataset/raw/main/SAT%20GPA.csv')
```

```
sat.head()
```

	SAT	GPA
0	1270	3.4
1	1220	4.0
2	1160	3.8
3	950	3.8
4	1070	4.0

```
sat.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   SAT     1000 non-null   int64  
 1   GPA     1000 non-null   float64
dtypes: float64(1), int64(1)
memory usage: 15.8 KB
```

```
sat.describe()
```

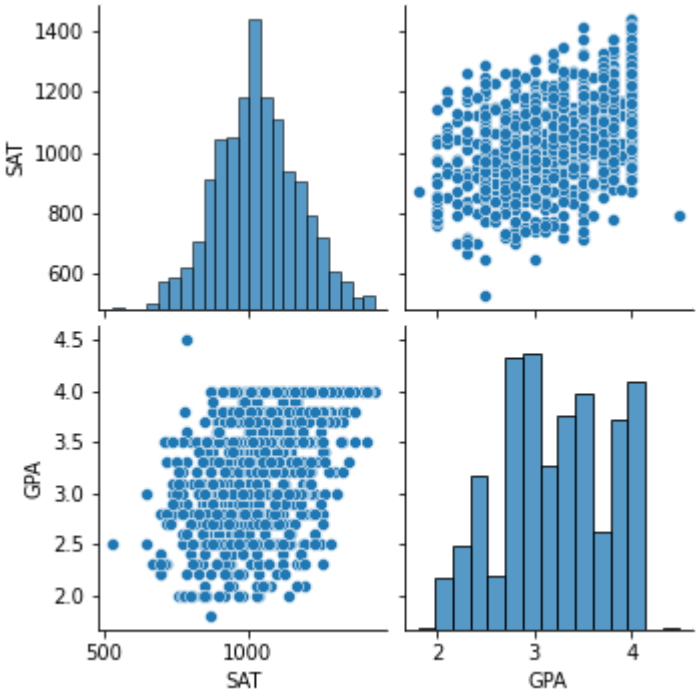
	SAT	GPA
count	1000.000000	1000.000000
mean	1033.290000	3.203700
std	142.873681	0.542541
min	530.000000	1.800000
25%	930.000000	2.800000
50%	1030.000000	3.200000
75%	1130.000000	3.700000
max	1440.000000	4.500000

```
sat.corr()
```

	SAT	GPA
SAT	1.000000	0.429649
GPA	0.429649	1.000000

```
sns.pairplot(sat)
```

```
<seaborn.axisgrid.PairGrid at 0x7fc6196ea810>
```




```
sat.columns
```

```
Index(['SAT', 'GPA'], dtype='object')
```

```
y = sat['SAT']
```

```
y.shape
```

```
(1000,)
```

```
X = sat[['GPA']]
```

```
X.shape
```

```
(1000, 1)
```

```
from sklearn.model_selection import train_test_split
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size = 0.7, random_state = 2529)
```

```
X_train.shape, X_test.shape, y_train.shape , y_test.shape
```

```
((700, 1), (300, 1), (700,), (300,))
```

```
X_train
```

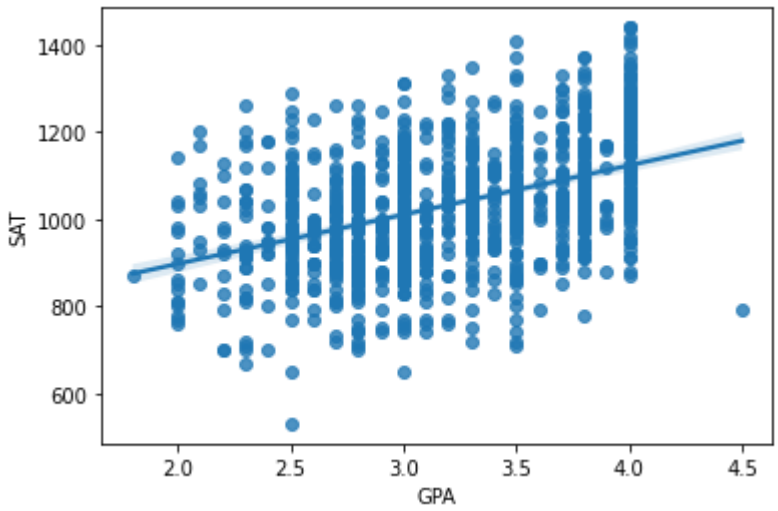
	GPA
669	3.7
583	3.7
688	2.8
422	3.9
825	4.0
...	...
740	2.5
399	2.6
828	3.2
562	2.7

```
from sklearn.linear_model import LinearRegression

reg = LinearRegression()

reg.fit(X_train, y_train)

     LinearRegression()

reg.intercept_

     673.2291896122774

reg.coef_

     array([111.01584994])

y_pred = reg.predict(X_test)

from sklearn.metrics import mean_absolute_error,mean_absolute_percentage_error, r2_score


mean_absolute_error(y_test, y_pred)

     105.93877473699905

mean_absolute_percentage_error(y_test, y_pred)

     0.10467104034918914

r2_score(y_test, y_pred)
```

```
0.18785383761597474
```

```
sns.regplot(x= 'GPA' , y = 'SAT' , data = sat)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc616be72d0>
```



Multiple Regression

```
df = pd.read_csv('https://github.com/ybifoundation/Dataset/raw/main/Boston.csv')
```

```
df.head()
```

	CRIM	ZN	INDUS	CHAS	NX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	M
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296.0	15.3	396.90	4.98	2
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242.0	17.8	396.90	9.14	2
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242.0	17.8	392.83	4.03	3
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222.0	18.7	394.63	2.94	3
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222.0	18.7	396.90	5.33	3

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 14 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   CRIM     506 non-null    float64
 1   ZN       506 non-null    float64
 2   INDUS    506 non-null    float64
 3   CHAS     506 non-null    int64  
 4   NX       506 non-null    float64
 5   RM       506 non-null    float64
 6   AGE      506 non-null    float64
 7   DIS      506 non-null    float64
```

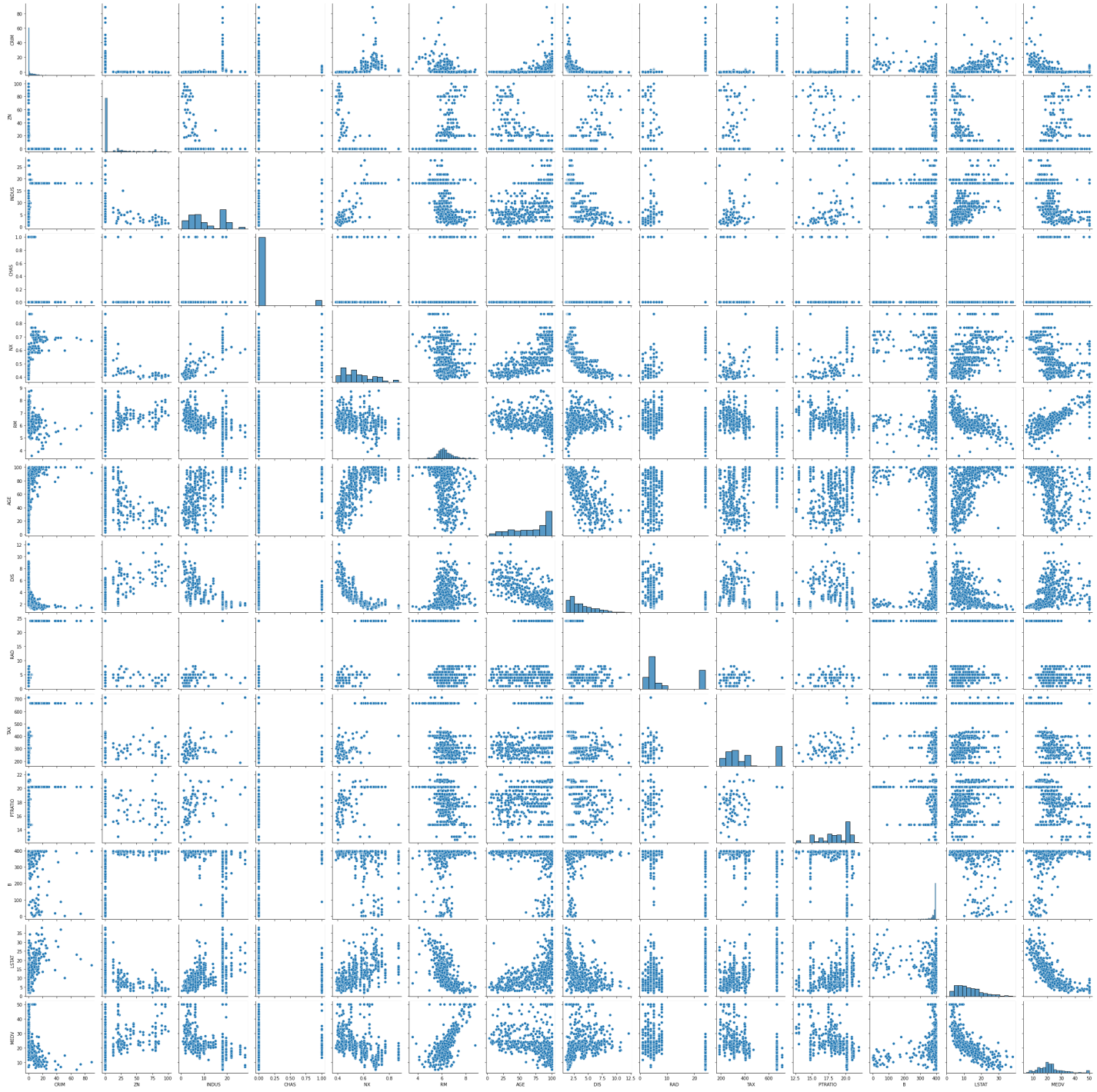
```
 8   RAD      506 non-null    int64
 9   TAX      506 non-null    float64
 10  PTRATIO  506 non-null    float64
 11  B        506 non-null    float64
 12  LSTAT    506 non-null    float64
 13  MEDV     506 non-null    float64
dtypes: float64(12), int64(2)
memory usage: 55.5 KB
```

```
df.describe()
```

	CRIM	ZN	INDUS	CHAS	NX	RM	AGE	
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.0
mean	3.613524	11.363636	11.136779	0.069170	0.554695	6.284634	68.574901	3.7
std	8.601545	23.322453	6.860353	0.253994	0.115878	0.702617	28.148861	2.1
min	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000	2.900000	1.1
25%	0.082045	0.000000	5.190000	0.000000	0.449000	5.885500	45.025000	2.1
50%	0.256510	0.000000	9.690000	0.000000	0.538000	6.208500	77.500000	3.2
75%	3.677083	12.500000	18.100000	0.000000	0.624000	6.623500	94.075000	5.1
max	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000	100.000000	12.1

```
sns.pairplot(df)
```

<seaborn.axisgrid.PairGrid at 0x7fc60f605fd0>



```
df.columns
```

```
Index(['CRIM', 'ZN', 'INDUS', 'CHAS', 'NX', 'RM', 'AGE', 'DIS', 'RAD', 'TAX',
       'PTRATIO', 'B', 'LSTAT', 'MEDV'],
      dtype='object')
```

```
y = df['MEDV']
```

```
X = df[['CRIM', 'ZN', 'INDUS', 'CHAS', 'NX', 'RM', 'AGE', 'DIS', 'RAD', 'TAX',
       'PTRATIO', 'B', 'LSTAT']]
```

```
X.shape
```

```
(506, 13)
```

```
from sklearn.model_selection import train_test_split
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size = 0.7, random_state = 2529)
```

```
from sklearn.preprocessing import StandardScaler
```

```
sc = StandardScaler()
```

```
X_train = sc.fit_transform(X_train)
```

```
X_test = sc.fit_transform(X_test)
```

```
X_train
```

```
array([[-0.14113619, -0.48175769, -0.19860022, ...,  0.00438903,
        -0.05084503, -0.01555641],
       [-0.42121529,  3.02166196, -1.33410259, ..., -1.68641979,
         0.42969249, -1.33650784],
       [-0.41266839, -0.48175769,  0.22414717, ...,  0.14148164,
         0.19739169, -0.10842497],
       ...,
       [-0.38944304, -0.48175769, -0.19860022, ...,  0.00438903,
         0.37963873,  0.77313338],
       [-0.41404001,  0.41002186, -0.81324318, ..., -0.72677154,
         0.43161763,  0.09671754],
       [-0.41578561,  2.06618387, -1.3831586 , ..., -0.04130851,
         0.39707198, -0.68781395]])
```

```
from sklearn.linear_model import LinearRegression
```

```
model = LinearRegression()
model.fit(X_train, y_train)
```

```
     LinearRegression()
```

```
model.intercept_
```

```
     22.83248587570622
```

```
model.coef_
```

```
     array([-1.20767891,  0.85995285,  0.1070255 ,  0.63555228, -2.43159195,
             3.08829222,  0.13082323, -3.31025945,  2.22711291, -1.65403572,
            -2.10989321,  0.94408913, -3.91890566])
```

```
df.columns
```

```
     Index(['CRIM', 'ZN', 'INDUS', 'CHAS', 'NX', 'RM', 'AGE', 'DIS', 'RAD', 'TAX',
            'PTRATIO', 'B', 'LSTAT', 'MEDV'],
           dtype='object')
```

```
y_pred = model.predict(X_test)
```

```
from sklearn.metrics import mean_absolute_percentage_error, r2_score
```

```
mean_absolute_percentage_error(y_test, y_pred)
```

```
     0.18900464575924525
```

```
r2_score(y_test, y_pred)
```

```
     0.5945114562128394
```