```
import pandas as pd
```

```
import numpy as np
```

```
import matplotlib.pyplot as plt
```

```
import seaborn as sb
```

```
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()
```

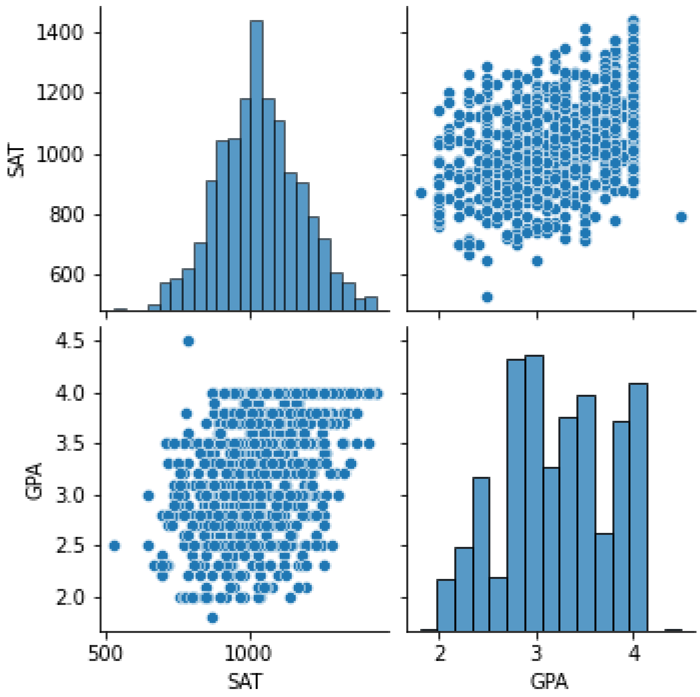
	SAT	GPA
count	1000.000000	1000.000000
mean	1033.290000	3.203700
std	142.873681	0.542541
min	530.000000	1.800000

```
sat.corr()
```

	SAT	GPA
SAT	1.000000	0.429649
GPA	0.429649	1.000000

```
from seaborn.axisgrid import pairplot
sb.pairplot(sat)
```

```
<seaborn.axisgrid.PairGrid at 0x7f7d268badd0>
```



```
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
352	3.0

700 rows × 1 columns

```
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_percentage_error(y_test,y_pred)
```

```
    0.1046927663671282
```

```
mean_absolute_error(y_test,y_pred)
```

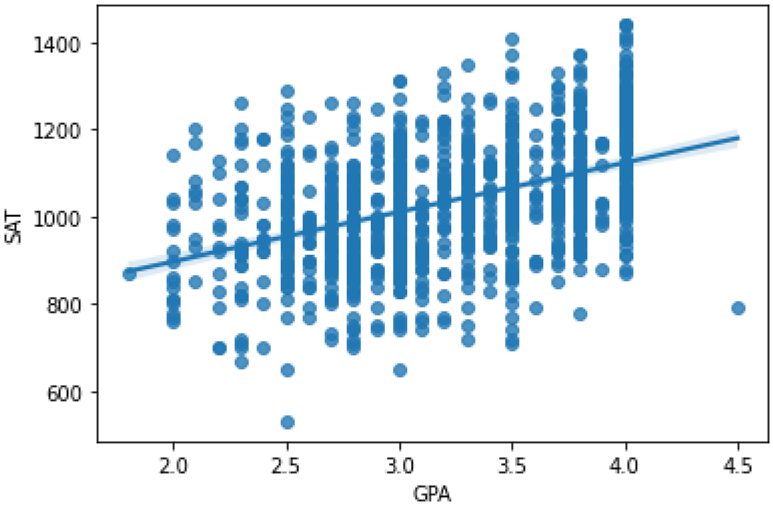
```
    105.93877473699905
```

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

```
    0.18785383761597474
```

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

```
    <matplotlib.axes._subplots.AxesSubplot at 0x7f7d1f5c5610>
```



```
import matplotlib.pyplot as plt
```

```
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
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296.0	15.3	396.90
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242.0	17.8	396.90
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242.0	17.8	392.83
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222.0	18.7	394.63
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222.0	18.7	396.90

```
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
```

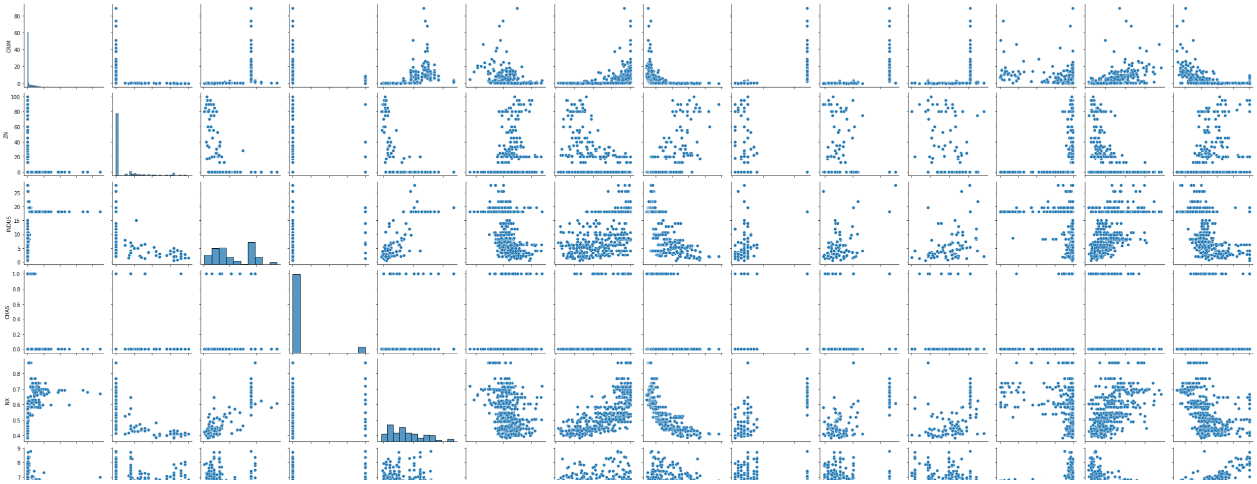
```
df.describe()
```

	CRIM	ZN	INDUS	CHAS	NX	RM	
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.00
mean	3.613524	11.363636	11.136779	0.069170	0.554695	6.284634	68.57
std	8.601545	23.322453	6.860353	0.253994	0.115878	0.702617	28.14
min	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000	2.90

```
import seaborn as so
```

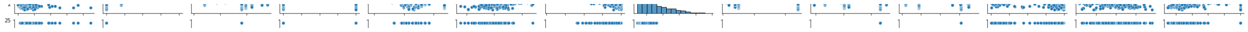
50%	0.256510	0.000000	9.690000	0.000000	0.538000	6.208500	77.50

```
so.pairplot(df)
```



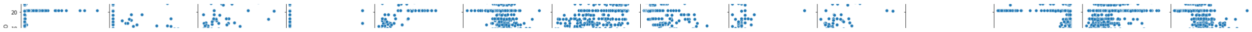
```
df.columns
```

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



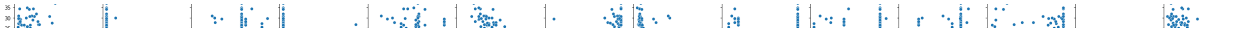
```
y=df['MEDV']
```

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

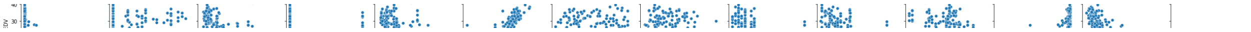


```
x.shape
```

```
(506, 11)
```



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



```
sc=StandardScaler()
```

```
x_train=sc.fit_transform(x_train)
```

```
x_test=sc.fit_transform(x_test)
```

```
x_train
```

```
array([[ 0.9146784 ],
       [ 0.9146784 ],
       [-0.75077649],
       [ 1.28477948],
       [ 1.46983002],
       [ 0.17447623],
       [ 1.09972894],
       [-0.3806754 ],
       [ 1.09972894],
       [-2.23118083],
       [-2.04613029],
       [-2.04613029],
       [ 0.17447623],
       [ 0.9146784 ],
       [-2.60128191],
       [-0.3806754 ],
       [-0.75077649],
       [ 0.72962785],
       [ 1.46983002],
       [ 0.9146784 ],
       [ 1.28477948],
       [ 0.17447623],
       [ 1.28477948],
       [-0.01057432],
       [ 0.9146784 ],
       [-0.56572594],
       [ 1.09972894],
       [-1.6760292 ],
       [ 0.17447623],
       [ 1.09972894],
       [-0.3806754 ],
       [-0.3806754 ],
       [ 1.09972894],
       [ 1.09972894],
       [-0.93582703],
       [ 0.54457731],
       [-0.01057432],
       [ 0.35952677],
       [ 0.17447623],
       [ 0.9146784 ],
       [ 0.54457731],
       [ 1.09972894],
       [-1.30592812],
       [ 0.9146784 ],
       [-1.6760292 ],
       [-0.93582703],
       [-1.30592812],
       [ 1.46983002],
       [-0.19562486],
       [ 0.54457731],
       [-0.3806754 ],
       [ 1.46983002],
       [ 1.46983002],
       [ 1.09972894],
       [-0.3806754 ],
       [ 0.54457731],
       [-1.86107974],
       [ 0.35952677],
```

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

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

```
LinearRegression()
```

```
model.intercept_
```

```
     1029.1142857142856
```

```
model.coef_
```

```
     array([59.99217745])
```

```
model=LinearRegression()
```

```
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_error,mean_absolute_percentage_error,r2_score
```

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

```
     0.1046927663671282
```

```
mean_absolute_error(y_test,y_pred)
```

```
     105.8772997993962
```

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

```
     0.18858714592440917
```

```
df1=pd.read_csv('https://github.com/ybifoundation/Dataset/raw/main/Fish.csv')
```

```
df1.head()
```

	Category	Species	Weight	Height	Width	Length1	Length2	Length3
0	1	Bream	242.0	11.5200	4.0200	23.2	25.4	30.0

```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 159 entries, 0 to 158
Data columns (total 8 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Category  159 non-null    int64
 1   Species   159 non-null    object
 2   Weight    159 non-null    float64
 3   Height    159 non-null    float64
 4   Width     159 non-null    float64
 5   Length1   159 non-null    float64
 6   Length2   159 non-null    float64
 7   Length3   159 non-null    float64
dtypes: float64(6), int64(1), object(1)
memory usage: 10.1+ KB
```

```
df1.describe()
```

	Category	Weight	Height	Width	Length1	Length2	Le
count	159.000000	159.000000	159.000000	159.000000	159.000000	159.000000	159.0
mean	3.264151	398.326415	8.970994	4.417486	26.247170	28.415723	31.2
std	1.704249	357.978317	4.286208	1.685804	9.996441	10.716328	11.6
min	1.000000	0.000000	1.728400	1.047600	7.500000	8.400000	8.8
25%	2.000000	120.000000	5.944800	3.385650	19.050000	21.000000	23.1
50%	3.000000	273.000000	7.786000	4.248500	25.200000	27.300000	29.4
75%	4.500000	650.000000	12.365900	5.584500	32.700000	35.500000	39.6
max	7.000000	1650.000000	18.957000	8.142000	59.000000	63.400000	68.0

```
df1.columns
```

```
Index(['Category', 'Species', 'Weight', 'Height', 'Width', 'Length1',
       'Length2', 'Length3'],
      dtype='object')
```

```
df1.shape
```

```
(159, 8)
```

```
y=df1['Weight']
```

```
y.shape
```

```
(159,)
```

```
y
```

```
0      242.0
1      290.0
2      340.0
3      363.0
4      430.0
       ...
154     12.2
155     13.4
156     12.2
157     19.7
158     19.9
Name: Weight, Length: 159, dtype: float64
```

```
x=[['Weight', 'Height', 'Width', 'Length1',
       'Length2', 'Length3']]
```

```
x=df1.drop(['Species','Weight'], axis=1)
```

```
x.shape
```

```
(159, 6)
```

```
x
```

	Category	Height	Width	Length1	Length2	Length3
0	1	11.5200	4.0200	23.2	25.4	30.0
1	1	12.4800	4.3056	24.0	26.3	31.2

```
df_new=df1.sample(1)
```

```
df_new
```

	Category	Species	Weight	Height	Width	Length1	Length2	Length3
41	5	Roach	110.0	6.1677	3.3957	19.1	20.8	23.1

```
x_new=df_new[['Weight', 'Height', 'Length1',
       'Length2', 'Length3']]
```

157	6	2.8728	2.0672	13.2	14.3	15.2

```
x_new.shape
```

```
(1, 5)
```

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

```
model1=LinearRegression()
```

```
model1.fit(x_test,y_test)
```

```
LinearRegression()
```

```
x_new
```

	Weight	Height	Length1	Length2	Length3
41	110.0	6.1677	19.1	20.8	23.1

```
model1.coef_
```

```
array([64.62382312])
```

```
model1.intercept_
```

```
1043.0333333333333
```

```
import statsmodels.api as sm
```

```
/usr/local/lib/python3.7/dist-packages/statsmodels/tools/_testing.py:19: FutureWarning:
  import pandas.util.testing as tm
```

```
x=sm.add_constant(x)
```

```
/usr/local/lib/python3.7/dist-packages/statsmodels/tsa/tsatools.py:117: FutureWarning:
  x = pd.concat(x[::order], 1)
```

```
x.head()
```

	const	Category	Height	Width	Length1	Length2	Length3
0	1.0	1	11.5200	4.0200	23.2	25.4	30.0
1	1.0	1	12.4800	4.3056	24.0	26.3	31.2
2	1.0	1	12.3778	4.6961	23.9	26.5	31.1
3	1.0	1	12.7300	4.4555	26.3	29.0	33.5
4	1.0	1	12.4440	5.1340	26.5	29.0	34.0

```
sample=sm.OLS(y_test,x_test).fit()
```

```
y_pred=sample.predict(y_train)
```

```
y_pred
```

```
669    55192.803252
583    63591.708095
688    62391.864546
422    70190.847615
825    76789.987134
           ...
740    55192.803252
399    53992.959704
828    55192.803252
562    50393.429057
352    64791.551644
Length: 700, dtype: float64
```

```
y_pred.shape
```

```
(700,)
```

```
from sklearn.metrics import mean_squared_error
```

```
mean_squared_error(y_test,y_pred)
```

```
1104767.546960686
```

```
print(sample.summary())
```

```
                                 OLS Regression Results                                
=======================================================================================
Dep. Variable:                    SAT   R-squared (uncentered):                   0.003
Model:                            OLS   Adj. R-squared (uncentered):              0.002
Method:                 Least Squares   F-statistic:                              2.339
Date:                Tue, 19 Apr 2022   Prob (F-statistic):                       0.127
Time:                        15:46:45   Log-Likelihood:                         -5854.2
No. Observations:                 700   AIC:                                  1.171e+04
Df Residuals:                     699   BIC:                                  1.171e+04
Df Model:                           1                                                  
Covariance Type:            nonrobust                                                  
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
x1            59.9922     39.226      1.529      0.127     -17.023     137.008
==============================================================================
Omnibus:                        0.718   Durbin-Watson:                   0.032
Prob(Omnibus):                  0.698   Jarque-Bera (JB):                0.796
Skew:                           0.071   Prob(JB):                        0.672
Kurtosis:                       2.914   Cond. No.                         1.00
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specif
```

```
sample
```

```
<statsmodels.regression.linear_model.RegressionResultsWrapper at 0x7f7d1d5e12d0>
```

●