```
import pandas as pd
import gspread as gs
import numpy as np
from matplotlib import pyplot as plt
from sklearn.linear_model import Lasso
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import AdaBoostRegressor
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.svm import SVR
from sklearn import metrics
from sklearn.model_selection import train_test_split
```

```
SHEET_ID = '1WEM8wpCBgtW1SY2jTf7VS1rrDdFbBnOkxahsi0UH6tE'

SHEET_NAME = 'laptop_data_set'

url = f'https://docs.google.com/spreadsheets/d/{SHEET_ID}/gviz/tq?tqx=out:csv&sheet={SHEET_NAME}'

df = pd.read_csv(url)
```

```
TypeName = {'2 in 1 Convertible': 68326.36006611567, 'Workstation': 121497.52568275864, 'Netbook': 33898.44096000001, 'Notebook': 41703

ScreenResolution = {'IPS Panel Quad HD+ 2560x1440': 100231.04639999999, 'IPS Panel Full HD 1920x1080': 71593.77153537121, 'Touchscreen 

Cpu = {'Intel Core i3 6006U 2.0GHz': 25756.2846, 'Intel Core M m7-6Y75 1.2GHz': 69210.72, 'Intel Celeron Quad Core N3710 1.6GHz': 17529

Ram = {'4GB': 30651.055161497323, '16GB': 103191.166872, '8GB': 62913.71345525031, '6GB': 32778.19387317073, '12GB': 66037.27795199999,

Memory = {'2TB HDD': 34563.30210000001, '512GB Flash Storage': 65108.159999999996, '1TB HDD': 35918.83146188338, '500GB HDD': 33585.470

OpSys = {'macOS': 93220.32738461539, 'Android': 23123.52, 'Linux': 32877.42944516129, 'Windows 10': 62098.83595294104, 'No OS': 31555.8

Weight = {'1.26kg': 77946.18912, '2.54kg': 26053.92, '1.99kg': 70784.6112, '1.36kg': 91250.37257142858, '2.26kg': 58554.72, '4.33kg': 6

Gpu = {'Nvidia Quadro M620': 104929.63200000001, 'AMD Radeon R4 Graphics': 21477.0348, 'Nvidia GeForce GTX 1050M': 65303.52, 'AMD Radeo
```

```
df.replace({'TypeName':TypeName,'ScreenResolution ':ScreenResolution,'Cpu':Cpu,'Ram':Ram,'Memory':Memory,'Gpu':Gpu,'OpSys':OpSys,'Weigh

display(df)
```

	Company	TypeName	Inches	ScreenResolution	Cpu	Ram
0	Apple	82489.713429	13.3	89508.091200	76143.513600	62913.713455
1	Apple	82489.713429	13.3	55339.804800	54815.529600	62913.713455
2	HP	41703.867610	15.6	61619.932062	48935.966627	62913.713455
3	Apple	82489.713429	15.4	132872.194800	135195.336000	103191.166872
4	Apple	82489.713429	13.3	89508.091200	102393.504000	62913.713455
...	...	...	...	...	...	...
1287	Lenovo	68326.360066	14.0	63995.593177	68139.911608	30651.055161
1288	Lenovo	68326.360066	13.3	84004.800000	68139.911608	103191.166872
1289	Lenovo	41703.867610	14.0	28933.351971	17286.938182	13552.857818
1290	HP	41703.867610	15.6	28933.351971	68139.911608	32778.193873
1291	Asus	41703.867610	15.6	28933.351971	17286.938182	30651.055161

1292 rows × 11 columns

Linear Regression

```
X = df.drop(['Company','Price','OpSys','Inches'],axis = 1)
Y = df['Price']
```

```
X_train,X_test,Y_train,Y_test= train_test_split(X,Y,test_size=0.1,random_state = 1)


lin_reg_model = LinearRegression()
lin_reg_model.fit(X_train,Y_train)

pred=lin_reg_model.predict(X_test)
error_score = metrics.r2_score(Y_test,pred)
print(error_score)

lin_reg_model.coef_
```

```
    0.824698902217682
    array([0.11021991, 0.18641233, 0.2000966 , 0.15486066, 0.26713546,
           0.22741172, 0.34230716])
```

Lasso Regression

```
X = df.drop(['Company','Price','OpSys','Inches'],axis = 1)
Y = df['Price']

X_train,X_test,Y_train,Y_test= train_test_split(X,Y,test_size=0.1,random_state = 1)

lasso_reg = Lasso()
lasso_reg.fit(X_train,Y_train)

#y_pred_lass =lasso_reg.predict(X_test)

# print(y_pred_lass)

df1 = pd.DataFrame(np.concatenate((X_test,Y_test.values.reshape(-1,1)), axis=1))

display(df1)

accuracy = lasso_reg.score(X_test, Y_test)

print('The Accuray Score is: ',accuracy*100,'%')
```

	0	1	2	3	4	
0	68326.360066	63995.593177	94935.369600	62913.713455	65501.763414	79191.2
1	91661.979600	71593.771535	127769.169600	181849.215812	99144.010983	213510.7
2	41703.867610	61619.932062	64284.451200	103191.166872	81833.497920	46059.4
3	41703.867610	61619.932062	55637.013176	62913.713455	65501.763414	55998.3
4	68326.360066	63995.593177	71262.616299	62913.713455	65501.763414	60797.2
...	...	...	...	...	...	
125	68326.360066	73303.496885	48935.966627	30651.055161	65501.763414	60797.2
126	41703.867610	71593.771535	71262.616299	62913.713455	65501.763414	49853.1
127	91661.979600	71593.771535	58273.902109	62913.713455	65501.763414	64161.0
128	82489.713429	137542.320000	71262.616299	62913.713455	99144.010983	60797.2
129	41703.867610	28933.351971	22310.467200	30651.055161	33585.470986	21197.1

130 rows × 8 columns

```
X_train,X_test,Y_train,Y_test= train_test_split(X,Y,test_size=0.1,random_state = 1)
df1 = pd.DataFrame(np.concatenate((X_test,Y_test.values.reshape(-1,1)), axis=1))

print(df.var())
```

```
    TypeName            5.171171e+08
    Inches              2.017853e+00
    ScreenResolution    5.443265e+08
    Cpu                 7.804959e+08
```

```
Ram                 7.341892e+08
Memory              7.209689e+08
Gpu                 7.314869e+08
OpSys               1.427899e+08
Weight              6.896977e+08
Price               1.152259e+09
dtype: float64
<ipython-input-12-cb9e81acc888>:4: FutureWarning: The default value of numeric_only in DataFrame.var is deprecated. In a future ve
  print(df.var())
```

Gradien Boost Regressor(Descision Tree)

```
X = df.drop(['Company','Price'],axis = 1)
Y = df['Price']

X_train,X_test,Y_train,Y_test= train_test_split(X,Y,test_size=0.1,random_state = 5)

# df1 = pd.DataFrame(np.concatenate((X_test,Y_test.values.reshape(-1,1)), axis=1))

# display(df1)

regressor = DecisionTreeRegressor(min_samples_split=10, max_depth=6, criterion="absolute_error")

reg = GradientBoostingRegressor(random_state=0)

boostmodel = reg.fit(X_train,Y_train)

y_pred = boostmodel.predict(X_test)

error_score = metrics.r2_score(Y_test,y_pred)

print(error_score)

print(boostmodel.predict([[91661.979600,    15.6,   61619.932062,   93794.560866,   103191.166872,  67496.316970,   72708.323657,   620

# regressor.fit(X_train, Y_train.values.reshape(-1,1))

# y_pred = regressor.predict(X_test)

# error_score = metrics.r2_score(Y_test,y_pred)

# print(error_score)
```

```
0.9221378950089215
[72984.76055269]
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but GradientBoostin
  warnings.warn(
```

```
# import joblib

# joblib.dump(boostmodel,'model.pkl')
```

```
['model.pkl']
```

```
# from google.colab import files
# files.download('model.pkl')
```

RandomForestRegressor

```
X = df.drop(['Company','Price'],axis = 1)
Y = df['Price']

X_train,X_test,Y_train,Y_test= train_test_split(X,Y,test_size=0.1,random_state = 5)

rf = RandomForestRegressor(n_estimators = 9, random_state = 42,min_samples_split=10,
                           max_depth=6, criterion="absolute_error",min_samples_leaf = 6,max_samples = 800)
```

```
rf.fit(X_train, Y_train);

y_pred = rf.predict(X_test)

score = metrics.r2_score(Y_test,y_pred)
print(score)

print(rf.predict([[91661.979600,    17.3,   61619.932062,   93794.560866,   103191.166872,  98895.788778,   96793.676100,   62098.83595
```

```
0.9165954392553686
[95693.1888]
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but RandomForestReg
  warnings.warn(
```

KNN

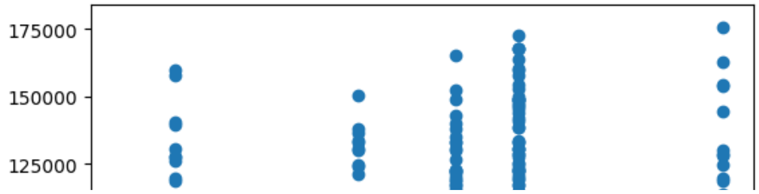
```
X = df.drop(['Company','Price'],axis = 1)
Y = df['Price']
X_train,X_test,Y_train,Y_test= train_test_split(X,Y,test_size=0.1,random_state = 5)

clf = KNeighborsRegressor(9)
clf.fit(X_train,Y_train)

y_pred = clf.predict(X_test)

score = metrics.r2_score(Y_test,y_pred)
print(score)

print(clf.predict([[91661.979600,   15.6,   61619.932062,   93794.560866,   103191.166872,  67496.316970,   72708.323657,   62098.83595
```

```
0.8867964331181687
[78498.9632]
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but KNeighborsRegre
  warnings.warn(
```

```
X = df.drop(['Company','Price'],axis = 1)
Y = df['Price']
X_train,X_test,Y_train,Y_test= train_test_split(X,Y,test_size=0.1,random_state = 5)

regressor = SVR(kernel='linear')
regressor.fit(X_train,Y_train)

y_pred = regressor.predict(X_test)

print(y_pred)

score = metrics.r2_score(Y_test,y_pred)
print(score)
#accuracy = 86.6%
```

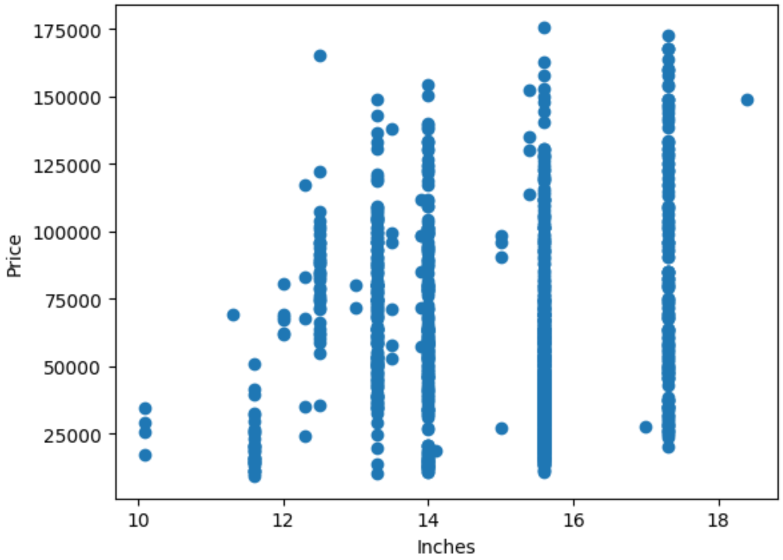
```
plt.scatter(df['TypeName'],df['Price'])
plt.xlabel('Typename')
plt.ylabel('Price')
```

```
Text(0, 0.5, 'Price')
```



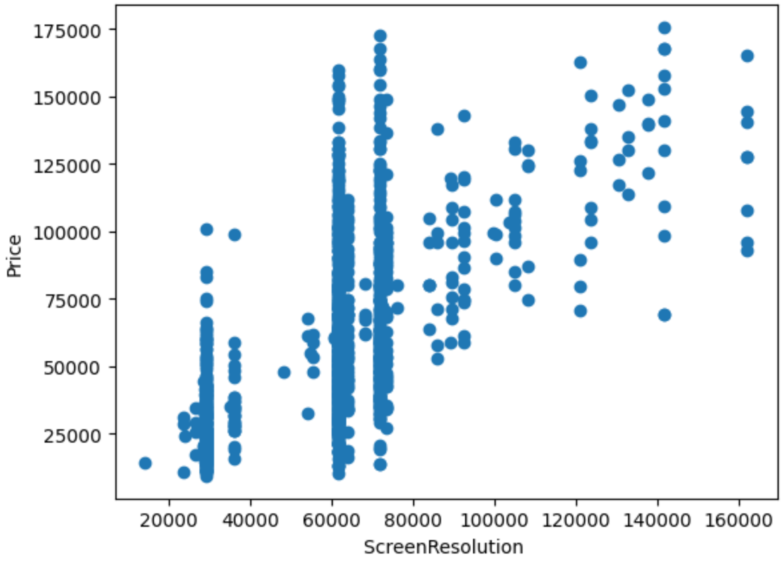
```
plt.scatter(df['Inches'],df['Price'])
plt.xlabel('Inches')
plt.ylabel('Price')
```

```
Text(0, 0.5, 'Price')
```



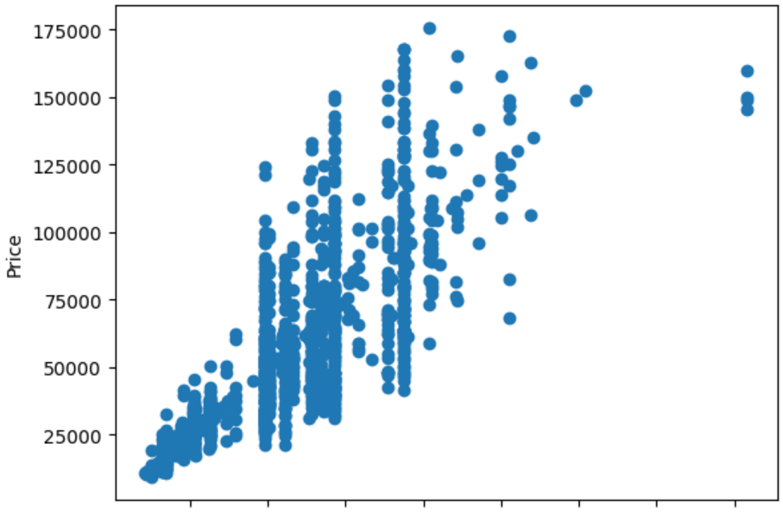
```
plt.scatter(df['ScreenResolution '],df['Price'])
plt.xlabel('ScreenResolution ')
plt.ylabel('Price')
```

```
Text(0, 0.5, 'Price')
```



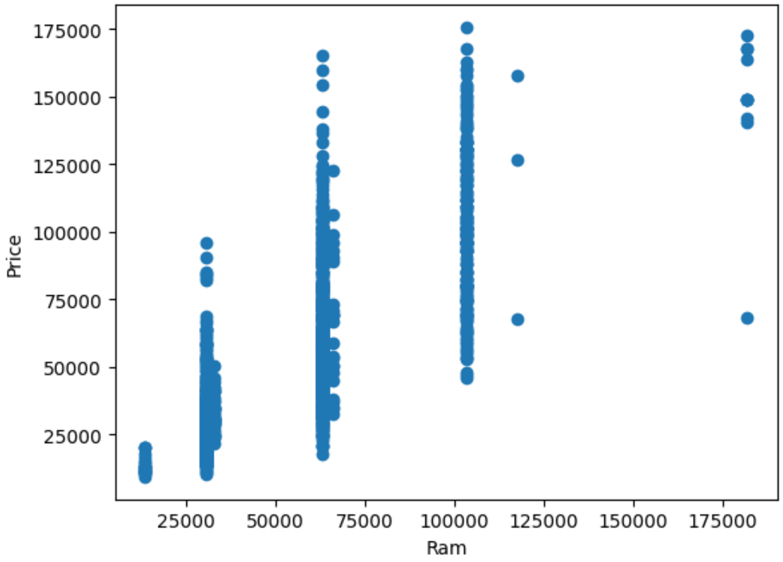
```
plt.scatter(df['Cpu'],df['Price'])
plt.xlabel('Cpu')
plt.ylabel('Price')
```

```
Text(0, 0.5, 'Price')
```



```
plt.scatter(df['Ram'],df['Price'])
plt.xlabel('Ram')
plt.ylabel('Price')
```

```
Text(0, 0.5, 'Price')
```

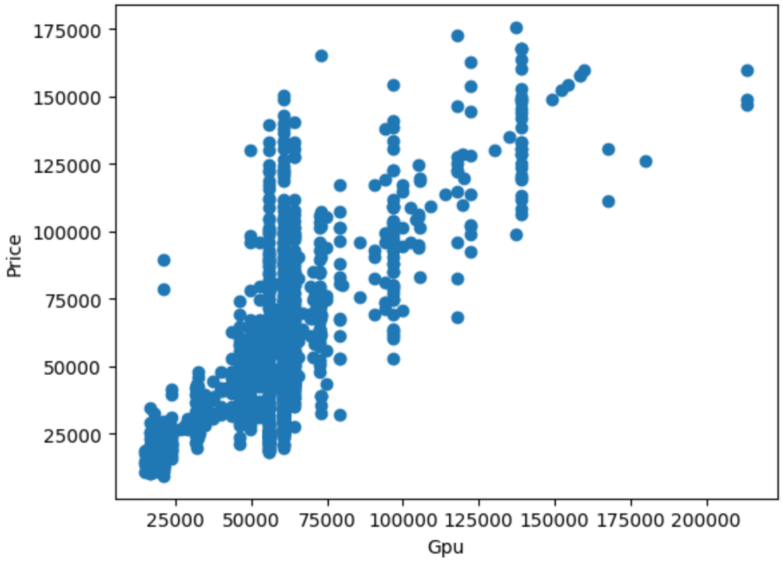


```
plt.scatter(df['Memory'],df['Price'])
plt.xlabel('Memory')
plt.ylabel('Price')
```

```
Text(0, 0.5, 'Price')
```

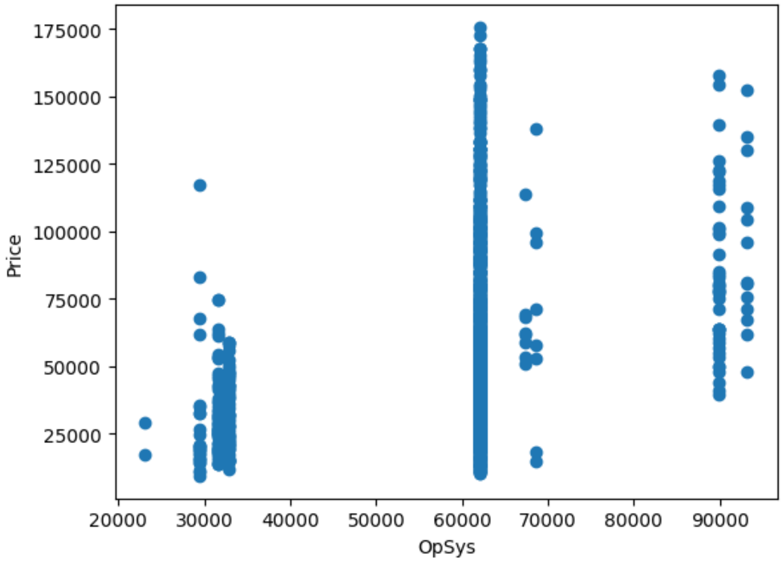
```
plt.scatter(df['Gpu'],df['Price'])
plt.xlabel('Gpu')
plt.ylabel('Price')
```

```
Text(0, 0.5, 'Price')
```



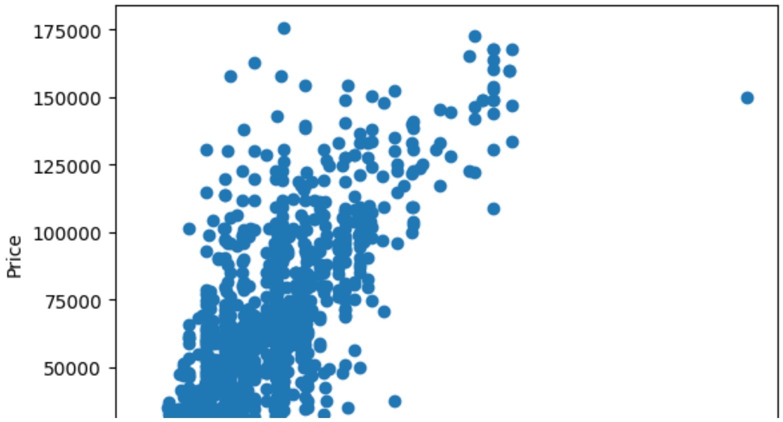
```
plt.scatter(df['OpSys'],df['Price'])
plt.xlabel('OpSys')
plt.ylabel('Price')
```

```
Text(0, 0.5, 'Price')
```



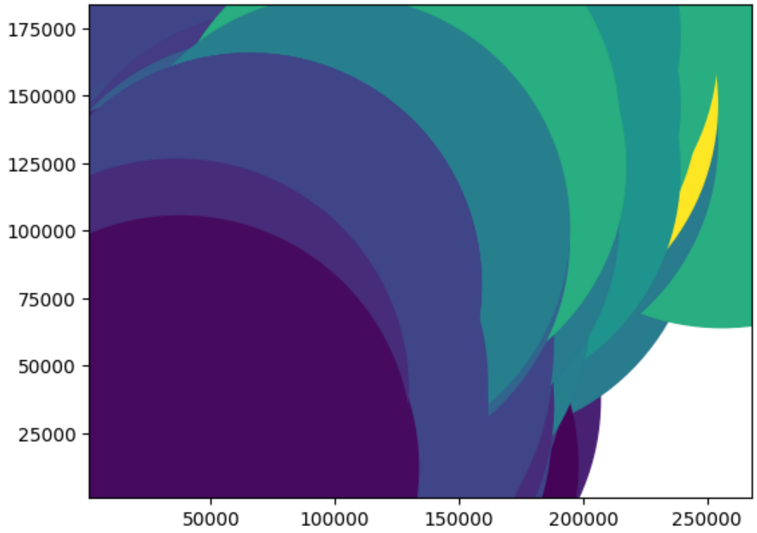
```
plt.scatter(df['Weight'],df['Price'])
plt.xlabel('Weights')
plt.ylabel('Price')
```

```
Text(0, 0.5, 'Price')
```



```
plt.scatter(df['Weight'],df['Price'],df['OpSys'],df['Gpu'])
```

```
<matplotlib.collections.PathCollection at 0x7efcf46becb0>
```



✓ 0s completed at 1:32 PM