Ravi Shripad

Roll no.58

House Price prediction using Linear Regression - SingleVariable

```
import pandas as pd
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt
```

```
C:\ProgramData\Anaconda3\lib\site-packages\scipy\__init__.py:146: UserWarning: A NumPy version >=1.16.5 and <1.23.0 is required for this
  warnings.warn(f"A NumPy version >={np_minversion} and <{np_maxversion}"
```

Load Dataset

```
Dataset = pd.read_csv('house dataset.csv')
Dataset.head()
```

	area	price
0	8450	208500
1	9600	181500
2	11250	223500
3	9550	140000
4	14260	250000

Load Summariz

```
print(Dataset.shape)
print(Dataset.head(5))
```

```
(1460, 2)
    area   price
0   8450  208500
1   9600  181500
2  11250  223500
3   9550  140000
4  14260  250000
```

```
Dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1460 entries, 0 to 1459
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   area    1460 non-null   int64
 1   price   1460 non-null   int64
dtypes: int64(2)
memory usage: 22.9 KB
```

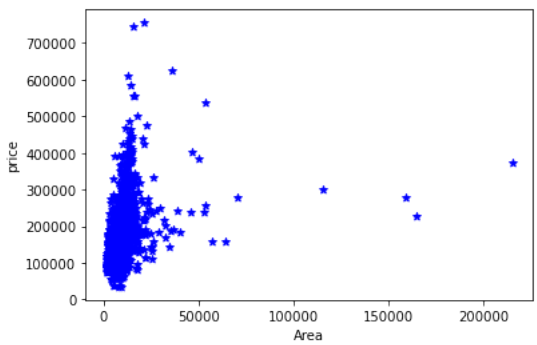
```
Dataset.describe()
```

	area	price
count	1460.000000	1460.000000
mean	10516.828082	180921.195890
std	9981.264932	79442.502883
min	1300.000000	34900.000000
25%	7553.500000	129975.000000
50%	9478.500000	163000.000000
75%	11601.500000	214000.000000
max	215245.000000	755000.000000

Visualize Dataset

```
plt.xlabel('Area')
plt.ylabel('price')
plt.scatter(Dataset.area,Dataset.price,color='blue', marker='*')
```

```
<matplotlib.collections.PathCollection at 0x14af2ff11f0>
```



Segreate Dataset into Input X & Output Y

```
X = Dataset.drop('price', axis='columns')
X
```

	area
0	8450
1	9600
2	11250
3	9550
4	14260
...	...
1455	7917
1456	13175
1457	9042
1458	9717
1459	9937

1460 rows × 1 columns

```
Y = Dataset.price
Y
```

```
0       208500
1       181500
```

```
2       223500
3       140000
4       250000
         ...
1455    175000
1456    210000
1457    266500
1458    142125
1459    147500
Name: price, Length: 1460, dtype: int64
```

Training Dataset using Linear Regression

```
model = LinearRegression()
model.fit(X,Y)
```

```
LinearRegression()
```

Predicted Price for Land sq.Feet of custom values

```
x=int(input('Enter house Squar fit'))
LandAreainSqFt=[[x]]
PredictedmodelResult = model.predict(LandAreainSqFt)
print(PredictedmodelResult)
```

```
Enter house Squar fit3000
[165136.067752]
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X does not have valid feature names, but LinearRegression w
  warnings.warn(
```

Checking model is right

Theory Calculation

Y=m*X+b(m is coefficient and b is intercept)

Coefficient -m

```
m=model.coef_
print(m)
```

```
[2.09997195]
```

intercept - b

```
b=model.intercept_
print(b)
```

```
158836.1518968766
```

Y=mx+b

x is independent variable- input - area

```
y = m*x + b
print("The price of {0} Squar feet Land is: {1}".format(x,y[0]))
```

```
The price of 3000 Squar feet Land is: 165136.06775199962
```

Part B-Exam marks

```
import pandas as pd
from sklearn.linear_model import LinearRegression
```

```
dataset = pd.read_csv('exam data.csv')
```

```
dataset.head(10)
```

	hours	age	internet	marks
0	6.83	15	1	78.50
1	6.56	16	0	76.74
2	NaN	17	1	78.68
3	5.67	18	0	71.82
4	8.67	19	1	84.19
5	7.55	20	0	81.18
6	6.67	15	0	76.99
7	8.99	16	0	85.46
8	5.19	17	1	70.66
9	6.75	18	0	77.82

```
print(dataset.shape)
print(dataset.head(5))
```

```
(201, 4)
   hours  age  internet  marks
0   6.83   15         1  78.50
1   6.56   16         0  76.74
2    NaN   17         1  78.68
3   5.67   18         0  71.82
4   8.67   19         1  84.19
```

```
X = dataset.iloc[:, :-1].values
print(X.shape)
X
```

```
(201, 3)
array([[ 6.83, 15.  ,  1.  ],
       [ 6.56, 16.  ,  0.  ],
       [  nan, 17.  ,  1.  ],
       [ 5.67, 18.  ,  0.  ],
       [ 8.67, 19.  ,  1.  ],
       [ 7.55, 20.  ,  0.  ],
       [ 6.67, 15.  ,  0.  ],
       [ 8.99, 16.  ,  0.  ],
       [ 5.19, 17.  ,  1.  ],
       [ 6.75, 18.  ,  0.  ],
       [ 6.59, 19.  ,  0.  ],
       [ 8.56, 20.  ,  1.  ],
       [ 7.75, 15.  ,  0.  ],
       [ 7.9 , 16.  ,  1.  ],
       [ 8.19, 17.  ,  0.  ],
       [ 6.55, 18.  ,  1.  ],
       [ 6.36, 19.  ,  0.  ],
       [ 8.44, 20.  ,  1.  ],
       [ 8.41, 15.  ,  0.  ],
       [ 7.67, 16.  ,  1.  ],
       [ 7.42, 17.  ,  1.  ],
       [ 8.16, 18.  ,  1.  ],
       [ 5.05, 19.  ,  1.  ],
       [ 5.85, 20.  ,  1.  ],
       [ 5.45, 15.  ,  0.  ],
       [ 7.96, 16.  ,  0.  ],
       [ 6.51, 17.  ,  0.  ],
```

```
       [ 6.73, 18.  ,  0.  ],
       [ 5.94, 19.  ,  1.  ],
       [ 7.48, 20.  ,  0.  ],
       [ 8.13, 15.  ,  1.  ],
       [  nan, 16.  ,  1.  ],
       [ 5.4 , 17.  ,  1.  ],
       [ 8.78, 18.  ,  0.  ],
       [ 8.72, 19.  ,  1.  ],
       [ 7.1 , 20.  ,  0.  ],
       [ 7.86, 15.  ,  1.  ],
       [ 7.19, 16.  ,  0.  ],
       [ 5.62, 17.  ,  1.  ],
       [ 7.88, 18.  ,  0.  ],
       [ 5.28, 19.  ,  1.  ],
       [ 8.92, 20.  ,  1.  ],
       [ 5.46, 15.  ,  0.  ],
       [ 8.3 , 16.  ,  1.  ],
       [ 8.09, 17.  ,  0.  ],
       [ 6.18, 18.  ,  1.  ],
       [ 7.01, 19.  ,  1.  ],
       [ 5.01, 20.  ,  0.  ],
       [ 5.54, 15.  ,  1.  ],
       [ 5.09, 16.  ,  1.  ],
       [ 5.09, 17.  ,  0.  ],
       [ 7.31, 18.  ,  1.  ],
       [ 8.71, 19.  ,  0.  ],
       [ 5.52, 20.  ,  1.  ],
       [ 8.76, 15.  ,  0.  ],
       [ 8.69, 16.  ,  1.  ],
```

```
dataset.columns[dataset.isna().any()]
```

```
Index(['hours'], dtype='object')
```

```
dataset.hours = dataset.hours.fillna(dataset.hours.mean())
```

```
X = dataset.iloc[:, :-1].values
print(X.shape)
X
```

```
(201, 3)
array([[ 6.83      , 15.        ,  1.        ],
       [ 6.56      , 16.        ,  0.        ],
       [ 6.98142857, 17.        ,  1.        ],
       [ 5.67      , 18.        ,  0.        ],
       [ 8.67      , 19.        ,  1.        ],
       [ 7.55      , 20.        ,  0.        ],
       [ 6.67      , 15.        ,  0.        ],
       [ 8.99      , 16.        ,  0.        ],
       [ 5.19      , 17.        ,  1.        ],
       [ 6.75      , 18.        ,  0.        ],
       [ 6.59      , 19.        ,  0.        ],
       [ 8.56      , 20.        ,  1.        ],
       [ 7.75      , 15.        ,  0.        ],
       [ 7.9       , 16.        ,  1.        ],
       [ 8.19      , 17.        ,  0.        ],
       [ 6.55      , 18.        ,  1.        ],
       [ 6.36      , 19.        ,  0.        ],
       [ 8.44      , 20.        ,  1.        ],
       [ 8.41      , 15.        ,  0.        ],
       [ 7.67      , 16.        ,  1.        ],
       [ 7.42      , 17.        ,  1.        ],
       [ 8.16      , 18.        ,  1.        ],
       [ 5.05      , 19.        ,  1.        ],
       [ 5.85      , 20.        ,  1.        ],
       [ 5.45      , 15.        ,  0.        ],
       [ 7.96      , 16.        ,  0.        ],
       [ 6.51      , 17.        ,  0.        ],
       [ 6.73      , 18.        ,  0.        ],
       [ 5.94      , 19.        ,  1.        ],
       [ 7.48      , 20.        ,  0.        ],
       [ 8.13      , 15.        ,  1.        ],
       [ 6.98142857, 16.        ,  1.        ],
       [ 5.4       , 17.        ,  1.        ],
       [ 8.78      , 18.        ,  0.        ],
       [ 8.72      , 19.        ,  1.        ],
       [ 7.1       , 20.        ,  0.        ],
       [ 7.86      , 15.        ,  1.        ],
       [ 7.19      , 16.        ,  0.        ],
       [ 5.62      , 17.        ,  1.        ],
       [ 7.88      , 18.        ,  0.        ],
```

```
       [ 5.28      , 19.        ,  1.        ],
       [ 8.92      , 20.        ,  1.        ],
       [ 5.46      , 15.        ,  0.        ],
       [ 8.3       , 16.        ,  1.        ],
       [ 8.09      , 17.        ,  0.        ],
       [ 6.18      , 18.        ,  1.        ],
       [ 7.01      , 19.        ,  1.        ],
       [ 5.01      , 20.        ,  0.        ],
       [ 5.54      , 15.        ,  1.        ],
       [ 5.09      , 16.        ,  1.        ],
       [ 5.09      , 17.        ,  0.        ],
       [ 7.31      , 18.        ,  1.        ],
       [ 8.71      , 19.        ,  0.        ],
       [ 5.52      , 20.        ,  1.        ],
       [ 8.76      , 15.        ,  0.        ],
       [ 8.69      , 16.        ,  1.        ],
       [ 5.75      , 17.        ,  1.        ],
```

```
dataset.hours
```

```
0      6.830000
1      6.560000
2      6.981429
3      5.670000
4      8.670000
         ...
196    8.560000
197    8.940000
198    6.600000
199    8.350000
200    4.150000
Name: hours, Length: 201, dtype: float64
```

```
Y = dataset.iloc[:,-1].values
Y
```

```
array([78.5 , 76.74, 78.68, 71.82, 84.19, 81.18, 76.99, 85.46, 70.66,
       77.82, 75.37, 83.88, 79.5 , 80.76, 83.08, 76.03, 76.04, 85.11,
       82.5 , 80.58, 82.18, 83.36, 70.67, 75.02, 70.96, 83.33, 74.75,
       75.65, 74.15, 80.17, 82.27, 76.14, 71.1 , 84.35, 83.08, 76.76,
       81.24, 78.21, 73.08, 83.23, 70.27, 86.41, 71.1 , 82.84, 82.38,
       72.96, 77.46, 70.11, 72.38, 71.41, 72.22, 77.77, 84.44, 71.45,
       82.21, 85.48, 75.03, 86.65, 70.9 , 71.7 , 73.61, 79.41, 76.19,
       80.43, 85.78, 70.06, 81.25, 81.7 , 69.27, 82.79, 71.8 , 71.79,
       74.97, 78.61, 77.59, 72.33, 72.08, 77.33, 70.05, 73.34, 84.  ,
       82.93, 76.63, 75.36, 77.29, 72.87, 73.4 , 81.74, 71.85, 84.6 ,
       79.56, 82.1 , 72.08, 79.1 , 81.01, 76.48, 75.39, 68.57, 83.64,
       82.3 , 75.18, 82.03, 82.99, 79.26, 77.55, 77.07, 72.1 , 73.25,
       74.25, 70.58, 81.08, 75.04, 76.38, 80.86, 78.42, 74.44, 70.34,
       85.04, 73.61, 75.55, 76.2 , 82.69, 76.83, 79.53, 83.57, 85.95,
       76.02, 77.65, 77.01, 74.49, 73.19, 71.86, 75.8 , 72.46, 78.39,
       83.48, 83.15, 71.22, 85.98, 83.91, 84.58, 80.31, 82.55, 75.52,
       83.82, 85.15, 82.75, 74.34, 82.02, 86.12, 71.87, 76.7 , 81.7 ,
       70.78, 78.45, 70.2 , 83.37, 75.52, 81.57, 80.72, 80.81, 79.49,
       79.17, 77.07, 82.04, 71.94, 81.6 , 70.79, 82.68, 83.08, 71.18,
       77.63, 77.78, 70.4 , 73.02, 71.11, 85.96, 73.64, 84.24, 78.17,
       77.19, 71.83, 86.99, 83.87, 71.5 , 79.63, 85.1 , 72.01, 77.27,
       79.87, 73.14, 70.51, 84.03, 79.64, 74.24, 81.67, 84.68, 86.75,
       78.05, 83.5 , 81.45])
```

```
model=LinearRegression()
model.fit(X,Y)
```

```
LinearRegression()
```

```
a=[[1,75,1]]
predicedModelresult = model.predict(a)
print(PredictedmodelResult)
```

```
[165136.067752]
```

Start coding or generate with AI.

Start coding or generate with AI.