CH : - 5 Regression Analysis ¶

1.Supervised Learning

Regression(integer or float)

- Linear
- 1.)simple
- 2.)multiple
- polynomial
- 1.)simple
- 2.)multiple

Classification(categorical data)

- KNN
- Decision Tree

equation

- y = mx+b (linear)
- y = a+bx+cx1+dx2+ex3 (linear)
- y = a+bx+cx^2+dx^3 (polynomial)
- y = a+bx1+cx2^2+dx3^3 (polynomial)

Simple Linear Regression

scikit library

- preprocessing

In [2]:

```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
df = pd.read_csv("Book1.csv")
df.head()
```

Out[2]:

	cgpa	package
0	6.89	3.26
1	5.12	1.98
2	7.82	3.25
3	7.42	3.67
4	6.94	3.57

In [3]:

```
x = df.iloc[:,0:1]
y = df.iloc[:,-1]
# x = df.drop("package",axis=1)
# x = df[['cgpa','x1','x2']]
# x = df['cgpa']
# y = df['package']
```

In [4]:

```
x.shape
```

Out[4]:

```
(200, 1)
```

In [5]:

```
y.shape
```

Out[5]:

```
(200,)
```

- processing

In [6]:

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2)
x_train.shape
```

Out[6]:

```
(160, 1)
```

In [7]:

```
help(train_test_split)
```

```
Help on function train_test_split in module sklearn.model_selection._split:

train_test_split(*arrays, **options)
    Split arrays or matrices into random train and test subsets

    Quick utility that wraps input validation and
    ``next(ShuffleSplit().split(X, y))`` and application to input data
    into a single call for splitting (and optionally subsampling) data in a
    oneliner.

    Read more in the :ref:`User Guide <cross_validation>`.

    Parameters
    ----------
    *arrays : sequence of indexables with same length / shape[0]
        Allowed inputs are lists, numpy arrays, scipy-sparse
        matrices or pandas dataframes.

    test_size : float or int, default=None
        If float, should be between 0.0 and 1.0 and represent the proportion
        of the dataset to include in the test split. If int, represents the
        absolute number of test samples. If None, the value is set to the
        complement of the train size. If ``train_size`` is also None, it wil
l
        be set to 0.25.

    train_size : float or int, default=None
        If float, should be between 0.0 and 1.0 and represent the
        proportion of the dataset to include in the train split. If
        int, represents the absolute number of train samples. If None,
        the value is automatically set to the complement of the test size.

    random_state : int or RandomState instance, default=None
        Controls the shuffling applied to the data before applying the spli
t.
        Pass an int for reproducible output across multiple function calls.
        See :term:`Glossary <random_state>`.


    shuffle : bool, default=True
        Whether or not to shuffle the data before splitting. If shuffle=Fals
e
        then stratify must be None.

    stratify : array-like, default=None
        If not None, data is split in a stratified fashion, using this as
        the class labels.

    Returns
    -------
    splitting : list, length=2 * len(arrays)
        List containing train-test split of inputs.

        .. versionadded:: 0.16
            If the input is sparse, the output will be a
            ``scipy.sparse.csr_matrix``. Else, output type is the same as th
e
            input type.

    Examples
    --------
```

```
>>> import numpy as np
>>> from sklearn.model_selection import train_test_split
>>> X, y = np.arange(10).reshape((5, 2)), range(5)
>>> X
array([[0, 1],
       [2, 3],
       [4, 5],
       [6, 7],
       [8, 9]])
>>> list(y)
[0, 1, 2, 3, 4]

>>> X_train, X_test, y_train, y_test = train_test_split(
...     X, y, test_size=0.33, random_state=42)
...
>>> X_train
array([[4, 5],
       [0, 1],
       [6, 7]])
>>> y_train
[2, 0, 3]
>>> X_test
array([[2, 3],
       [8, 9]])
>>> y_test
[1, 4]

>>> train_test_split(y, shuffle=False)
[[0, 1, 2], [3, 4]]
```

In [8]:

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2)
x_test.shape
```

Out[8]:

```
(40, 1)
```

In [9]:

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2)
y_train.shape
```

Out[9]:

```
(160,)
```

In [10]:

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2)
y_test.shape
```

Out[10]:

```
(40,)
```

In [11]:

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2)
lr=LinearRegression()
model = lr.fit(x_train,y_train) # creating a model or equation
lr.coef_
```

Out[11]:

```
array([0.56087409])
```

In [12]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2)
lr=LinearRegression()
model = lr.fit(x_train,y_train) # creating a model or equation
lr.intercept_
```

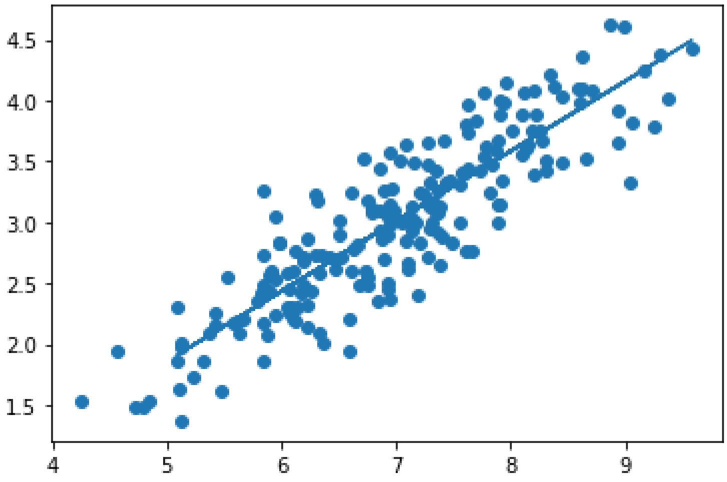
Out[12]:
```
-0.9923344509732703
```

- posting

In [13]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2)
lr=LinearRegression()
model = lr.fit(x_train,y_train) # creating a model or equation
y_pred = lr.predict(x_test)
y_pred
```

Out[13]:
```
array([2.0881033 , 3.27231514, 2.37569761, 2.93960801, 3.41329274,
       2.51667521, 2.67457012, 2.96216442, 2.31930657, 2.34750209,
       3.90389479, 3.18772858, 2.91141249, 4.12381985, 3.76291719,
       3.92081211, 2.42644954, 2.96780353, 3.31742798, 2.34750209,
       1.78923079, 2.61253997, 1.45652365, 2.86066055, 3.31742798,
       3.50915751, 2.54487073, 2.6463746 , 4.21968462, 2.05426868,
       2.80990862, 2.3700585 , 2.97908174, 3.44148826, 3.96592494,
       3.0749465 , 4.45652699, 2.34750209, 3.74036078, 2.11065972])
```

In [14]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2)
lr=LinearRegression()
model = lr.fit(x_train,y_train) # creating a model or equation
y_pred = lr.predict(x_test)
plt.scatter(df['cgpa'],df['package'])
plt.plot(x_test,y_pred)
plt.show()
```



In [15]:

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn import metrics
import math
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2)
lr=LinearRegression()
model = lr.fit(x_train,y_train)
y_pred = lr.predict(x_test)
print("MAE",metrics.mean_absolute_error(y_test,y_pred))
print("MSE",metrics.mean_squared_error(y_test,y_pred))
print("RMSE",math.sqrt(metrics.mean_squared_error(y_test,y_pred)))
print("R2score",metrics.r2_score(y_test,y_pred)) # x depend on y
```

```
MAE 0.28640979648873444
MSE 0.1297678341291693
RMSE 0.3602330275379665
R2score 0.6884288307682702
```

In [16]:

```
import numpy as np
test_data = np.array([9.3,7.8])
print(test_data)
print(test_data.shape)
```

```
[9.3 7.8]
(2,)
```

In [17]:

```
import numpy as np
test_data = test_data.reshape(-1,1)
print(test_data)
print(test_data.shape)
```

```
[[9.3]
 [7.8]]
(2, 1)
```

In [18]:

```
lr.predict(test_data)
```

Out[18]:

```
array([4.35342271, 3.48035551])
```

Multiple Linear Regression

In [19]:

```
# Target is selling_price.
import pandas as pd
df = pd.read_csv("car data.csv")
df.isna().sum()
```

Out[19]:

```
Car_Name         0
Year             0
Selling_Price    0
Present_Price    0
Kms_Driven       0
Fuel_Type        0
Seller_Type      0
Transmission     0
Owner            0
dtype: int64
```

In [20]:

```
import pandas as pd
df = pd.read_csv("car data.csv")
df = df.drop("Car_Name",axis=1)
df
```

Out[20]:

	Year	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission	Ow
0	2014	3.35	5.59	27000	Petrol	Dealer	Manual	
1	2013	4.75	9.54	43000	Diesel	Dealer	Manual	
2	2017	7.25	9.85	6900	Petrol	Dealer	Manual	
3	2011	2.85	4.15	5200	Petrol	Dealer	Manual	
4	2014	4.60	6.87	42450	Diesel	Dealer	Manual	
...	...	...	...	...	...	...	...	
296	2016	9.50	11.60	33988	Diesel	Dealer	Manual	
297	2015	4.00	5.90	60000	Petrol	Dealer	Manual	
298	2009	3.35	11.00	87934	Petrol	Dealer	Manual	
299	2017	11.50	12.50	9000	Diesel	Dealer	Manual	
300	2016	5.30	5.90	5464	Petrol	Dealer	Manual	

301 rows × 8 columns

In [21]:

```
df["Age"] = 2025-df["Year"]
df
```

Out[21]:

	Year	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission	Ow
0	2014	3.35	5.59	27000	Petrol	Dealer	Manual	
1	2013	4.75	9.54	43000	Diesel	Dealer	Manual	
2	2017	7.25	9.85	6900	Petrol	Dealer	Manual	
3	2011	2.85	4.15	5200	Petrol	Dealer	Manual	
4	2014	4.60	6.87	42450	Diesel	Dealer	Manual	
...	...	...	...	...	...	...	...	
296	2016	9.50	11.60	33988	Diesel	Dealer	Manual	
297	2015	4.00	5.90	60000	Petrol	Dealer	Manual	
298	2009	3.35	11.00	87934	Petrol	Dealer	Manual	
299	2017	11.50	12.50	9000	Diesel	Dealer	Manual	
300	2016	5.30	5.90	5464	Petrol	Dealer	Manual	

301 rows × 9 columns

In [22]:

```
df = df.drop("Year",axis=1)
df
```

Out[22]:

	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission	Owner	A
0	3.35	5.59	27000	Petrol	Dealer	Manual	0	
1	4.75	9.54	43000	Diesel	Dealer	Manual	0	
2	7.25	9.85	6900	Petrol	Dealer	Manual	0	
3	2.85	4.15	5200	Petrol	Dealer	Manual	0	
4	4.60	6.87	42450	Diesel	Dealer	Manual	0	
...	...	...	...	...	...	...	...	
296	9.50	11.60	33988	Diesel	Dealer	Manual	0	
297	4.00	5.90	60000	Petrol	Dealer	Manual	0	
298	3.35	11.00	87934	Petrol	Dealer	Manual	0	
299	11.50	12.50	9000	Diesel	Dealer	Manual	0	
300	5.30	5.90	5464	Petrol	Dealer	Manual	0	

301 rows × 8 columns

In [23]:

```
pd.get_dummies(df)
```

Out[23]:

	Selling_Price	Present_Price	Kms_Driven	Owner	Age	Fuel_Type_CNG	Fuel_Type_Diesel
0	3.35	5.59	27000	0	11	0	0
1	4.75	9.54	43000	0	12	0	1
2	7.25	9.85	6900	0	8	0	0
3	2.85	4.15	5200	0	14	0	0
4	4.60	6.87	42450	0	11	0	1
...	...	...	...	...	...	...	...
296	9.50	11.60	33988	0	9	0	1
297	4.00	5.90	60000	0	10	0	0
298	3.35	11.00	87934	0	16	0	0
299	11.50	12.50	9000	0	8	0	1
300	5.30	5.90	5464	0	9	0	0

301 rows × 12 columns

In [24]:
```
df = pd.get_dummies(df,drop_first=True) # only one column remove from each cat
df
```

Out[24]:

	Selling_Price	Present_Price	Kms_Driven	Owner	Age	Fuel_Type_Diesel	Fuel_Type_Petrol
0	3.35	5.59	27000	0	11	0	1
1	4.75	9.54	43000	0	12	1	0
2	7.25	9.85	6900	0	8	0	1
3	2.85	4.15	5200	0	14	0	1
4	4.60	6.87	42450	0	11	1	0
...	...	...	...	...	...	...	...
296	9.50	11.60	33988	0	9	1	0
297	4.00	5.90	60000	0	10	0	1
298	3.35	11.00	87934	0	16	0	1
299	11.50	12.50	9000	0	8	1	0
300	5.30	5.90	5464	0	9	0	1

301 rows × 9 columns

In [25]:
```
# target is selling price
y = df['Selling_Price']
x = df.drop('Selling_Price',axis=1)
x.shape
```

Out[25]:
```
(301, 8)
```

In [26]:
```
y = df['Selling_Price']
x = df.drop('Selling_Price',axis=1)
y.shape
```

Out[26]:
```
(301,)
```

In [27]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2,random_stat
x_train.shape
```

Out[27]:
```
(240, 8)
```

In [28]:
```
x_test.shape
```

Out[28]:
```
(61, 8)
```

In [29]:
```
y_train.shape
```

Out[29]:
```
(240,)
```

In [30]:
```
y_test.shape
```

Out[30]:
```
(61,)
```

In [31]:
```
from sklearn.linear_model import LinearRegression
lr = LinearRegression()
model = lr.fit(x_train,y_train)
lr.coef_
```

Out[31]:
```
array([ 4.28802981e-01, -5.92316903e-06, -8.49009727e-01, -4.06411225e-01,
        2.86056285e+00,  7.03103550e-01, -1.07477347e+00, -1.50897975e+00])
```

In [32]:
```
print(x.columns)
```

```
Index(['Present_Price', 'Kms_Driven', 'Owner', 'Age', 'Fuel_Type_Diesel',
       'Fuel_Type_Petrol', 'Seller_Type_Individual', 'Transmission_Manual'],
      dtype='object')
```

In [33]:
```
from sklearn.linear_model import LinearRegression
lr = LinearRegression()
model = lr.fit(x_train,y_train)
lr.intercept_
```

Out[33]:
```
6.850757321843104
```

In [34]:
```
from sklearn import metrics
import math
y_pred = lr.predict(x_test)
y_pred
```

Out[34]:
```
array([10.57889241,  0.71754255,  4.23613904,  5.17855444,  9.75533583,
        4.20615652,  2.6748089 ,  7.63221623,  0.17032381,  5.13283747,
        6.15727726,  6.44545506,  2.11900126,  7.6644102 ,  1.91710055,
        1.71680788,  2.02672159,  1.85294787,  9.56717323,  4.23076317,
        1.48724915,  9.36189904,  1.46103587,  9.54572517,  0.82954545,
        8.32266051,  1.53698563, -3.16293717,  4.2127377 ,  2.10385587,
        3.42279444,  3.71284702,  5.54538889,  7.6834828 , -1.89757368,
        7.06591847,  8.46027656,  5.77704136,  6.10531922,  6.52919752,
       16.08564344,  2.07410694,  1.05462465, -0.44710144,  7.06940454,
        6.73548589,  0.98609923,  7.08961114, 14.16420523,  3.00434598,
        8.2871215 , -0.87862345,  8.8814273 ,  1.15099681,  2.11848057,
       -0.82641923,  0.69287067,  9.88583992, -0.45444272, -2.41051663,
       10.28627971])
```

In [35]:
```
from sklearn import metrics
import math
y_pred = lr.predict(x_test)
print("MAE : ",metrics.mean_absolute_error(y_test,y_pred))
print("MSE : ",metrics.mean_squared_error(y_test,y_pred))
print("RMSE : ",math.sqrt(metrics.mean_squared_error(y_test,y_pred)))
print("R2score : ",metrics.r2_score(y_test,y_pred))
```

```
MAE :  1.2453565634870642
MSE :  2.750689262139753
RMSE :  1.658520202511791
R2score :  0.8502332355855595
```

In [36]:
```
# when column value is given
import numpy as np
test_data = np.array([6.7, 10000,0, 7, 1,0, 0, 1])
test_data = test_data.reshape(-1,8)
test_data
```

Out[36]:
```
array([[6.7e+00, 1.0e+04, 0.0e+00, 7.0e+00, 1.0e+00, 0.0e+00, 0.0e+00,
        1.0e+00]])
```

In [37]:
```
import numpy as np
test_data = np.array([6.7, 10000,0, 7, 1,0, 0, 1])
test_data = test_data.reshape(-1,8)
lr.predict(test_data)
```

Out[37]:
```
array([8.17121013])
```

Advertising csv

In [38]:
```
import pandas as pd
df = pd.read_csv("advertising.csv")
df.isna().sum()
```

Out[38]:
```
TV           0
Radio        0
Newspaper    0
Sales        0
dtype: int64
```

In [39]:
```
y = df["Sales"]
x = df.drop("Sales",axis=1)
```

In [40]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2)
x_train.shape
```

Out[40]:
```
(160, 3)
```

In [41]:
```
x_test.shape
```

Out[41]:
```
(40, 3)
```

In [42]:
```
y_train.shape
```

Out[42]:
```
(160,)
```

In [43]:
```
y_test.shape
```

Out[43]:
```
(40,)
```

In [44]:

```
from sklearn.linear_model import LinearRegression
lr = LinearRegression()
model = lr.fit(x_train,y_train)
lr.coef_
```

Out[44]:

```
array([0.05313408, 0.11357858, 0.00333475])
```

In [45]:

```
from sklearn.linear_model import LinearRegression
lr = LinearRegression()
model = lr.fit(x_train,y_train)
lr.intercept_
```

Out[45]:

```
4.645434610871224
```

In [46]:

```
from sklearn.linear_model import LinearRegression
lr = LinearRegression()
model = lr.fit(x_train,y_train)
y_pred = lr.predict(x_test)
y_pred
```

Out[46]:

```
array([ 7.32219292,  9.44780358, 16.08001839, 10.91179974,  9.94024746,
       16.95992676, 10.11639368, 12.36888929, 11.62393635, 12.57834914,
       11.08245536, 14.11836139,  8.3597048 , 21.14862847, 12.56036115,
       20.98615338, 10.46624475, 10.35001541, 17.97571054,  9.65169244,
       19.07693595,  9.04246403, 15.35125093,  8.93746723, 16.92844   ,
       14.69215049,  9.60888352,  8.02648269, 11.81179103, 17.05220234,
       13.75124131, 17.83846248, 20.65012944, 16.66301181,  9.64682542,
        7.78662847, 24.85375342,  9.88188164, 17.58112532, 12.78679528])
```

In [47]:

```
from sklearn import metrics
import math
y_pred = lr.predict(x_test)
print("MAE : ",metrics.mean_absolute_error(y_test,y_pred))
print("MSE : ",metrics.mean_squared_error(y_test,y_pred))
print("RMSE : ",math.sqrt(metrics.mean_squared_error(y_test,y_pred)))
print("R2score : ",metrics.r2_score(y_test,y_pred))
```

```
MAE :  1.220206245120477
MSE :  2.7003495910559043
RMSE :  1.643274046243019
R2score :  0.8808076842140988
```

Simple Ploynomial regression

In [48]:

```
import pandas as pd
df = pd.read_csv("temp.csv")
df
```

Out[48]:

	sno	Temperature	Pressure
0	1	0	0.0002
1	2	20	0.0012
2	3	40	0.0060
3	4	60	0.0300
4	5	80	0.0900
5	6	100	0.2700

In [49]:

```
df = df.drop("sno",axis=1)
df
```

Out[49]:

	Temperature	Pressure
0	0	0.0002
1	20	0.0012
2	40	0.0060
3	60	0.0300
4	80	0.0900
5	100	0.2700

In [50]:

```
y = df["Pressure"]
x = df.drop("Pressure",axis=1)
```

In [51]:

```
from sklearn.preprocessing import PolynomialFeatures
poly = PolynomialFeatures(degree=2)
x_poly = poly.fit_transform(x)
x_poly
```

Out[51]:

```
array([[1.0e+00, 0.0e+00, 0.0e+00],
       [1.0e+00, 2.0e+01, 4.0e+02],
       [1.0e+00, 4.0e+01, 1.6e+03],
       [1.0e+00, 6.0e+01, 3.6e+03],
       [1.0e+00, 8.0e+01, 6.4e+03],
       [1.0e+00, 1.0e+02, 1.0e+04]])
```

In [65]:

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x_poly,y,test_size=0.2,randon
In [66]: from sklearn.linear_model import LinearRegression
lr = LinearRegression()
lr.fit(x_train,y_train)
lr.coef_
```

```
Out[66]: array([ 0.00000000e+00, -2.23878205e-03,  4.87916667e-05])
```

```
In [67]: lr.intercept_
```

```
Out[67]: 0.0030807692307686374
```

```
In [68]: y_pred = lr.predict(x_test)
         y_pred
```

```
Out[68]: array([ 0.13624487, -0.02217821])
```

```
In [69]: import numpy as np
         test_data = np.array([54,67])
         test_data = test_data.reshape(2,1)
         test_data = poly.fit_transform(test_data)
         lr.predict(test_data)
```

```
Out[69]: array([0.02446304, 0.07210816])
```

Multiple Polynomial Regression

```
In [99]: import pandas as pd
         df = pd.read_csv("Real estate.csv")
         df
```

Out[99]:

	No	X1 transaction date	X2 house age	X3 distance to the nearest MRT station	X4 number of convenience stores	X5 latitude	X6 longitude	Y house price of unit area
0	1	2012.917	32.0	84.87882	10	24.98298	121.54024	37.9
1	2	2012.917	19.5	306.59470	9	24.98034	121.53951	42.2
2	3	2013.583	13.3	561.98450	5	24.98746	121.54391	47.3
3	4	2013.500	13.3	561.98450	5	24.98746	121.54391	54.8
4	5	2012.833	5.0	390.56840	5	24.97937	121.54245	43.1
...	...	...	...	...	...	...	...	...
409	410	2013.000	13.7	4082.01500	0	24.94155	121.50381	15.4
410	411	2012.667	5.6	90.45606	9	24.97433	121.54310	50.0
411	412	2013.250	18.8	390.96960	7	24.97923	121.53986	40.6
412	413	2013.000	8.1	104.81010	5	24.96674	121.54067	52.5
413	414	2013.500	6.5	90.45606	9	24.97433	121.54310	63.9

414 rows × 8 columns

In [100]:
```
df = df.drop("No",axis=1)
df
```

Out[100]:

	X1 transaction date	X2 house age	X3 distance to the nearest MRT station	X4 number of convenience stores	X5 latitude	X6 longitude	Y house price of unit area
0	2012.917	32.0	84.87882	10	24.98298	121.54024	37.9
1	2012.917	19.5	306.59470	9	24.98034	121.53951	42.2
2	2013.583	13.3	561.98450	5	24.98746	121.54391	47.3
3	2013.500	13.3	561.98450	5	24.98746	121.54391	54.8
4	2012.833	5.0	390.56840	5	24.97937	121.54245	43.1
...	...	...	...	...	...	...	...
409	2013.000	13.7	4082.01500	0	24.94155	121.50381	15.4
410	2012.667	5.6	90.45606	9	24.97433	121.54310	50.0
411	2013.250	18.8	390.96960	7	24.97923	121.53986	40.6
412	2013.000	8.1	104.81010	5	24.96674	121.54067	52.5
413	2013.500	6.5	90.45606	9	24.97433	121.54310	63.9

414 rows × 7 columns

In [101]:
```
y = df["Y house price of unit area"]
x = df.drop("Y house price of unit area",axis=1)
```

In [102]:
```
from sklearn.preprocessing import PolynomialFeatures
poly = PolynomialFeatures(degree=2)
x_poly = poly.fit_transform(x)
x_poly
```

Out[102]:
```
array([[1.00000000e+00, 2.01291700e+03, 3.20000000e+01, ...,
        6.24149290e+02, 3.03643739e+03, 1.47720299e+04],
       [1.00000000e+00, 2.01291700e+03, 1.95000000e+01, ...,
        6.24017387e+02, 3.03609828e+03, 1.47718525e+04],
       [1.00000000e+00, 2.01358300e+03, 1.33000000e+01, ...,
        6.24373157e+02, 3.03707359e+03, 1.47729221e+04],
       ...,
       [1.00000000e+00, 2.01325000e+03, 1.88000000e+01, ...,
        6.23961931e+02, 3.03597212e+03, 1.47719376e+04],
       [1.00000000e+00, 2.01300000e+03, 8.10000000e+00, ...,
        6.23338106e+02, 3.03447431e+03, 1.47721345e+04],
       [1.00000000e+00, 2.01350000e+03, 6.50000000e+00, ...,
        6.23717159e+02, 3.03545749e+03, 1.47727252e+04]])
```

In [107]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x_poly,y,test_size=0.2,random
```

In [108]:

```
from sklearn.linear_model import LinearRegression
lr = LinearRegression()
model = lr.fit(x_train,y_train)
model.coef_
# lr.coef_
```

Out[108]:

```
array([-2.30201635e+02, -1.88045458e+04,  9.28403731e+02,  2.61016682e+01,
        1.05269906e+04,  1.00715255e+05, -4.97732053e+05,  5.31841968e+00,
        1.41151382e-01, -1.48819301e-03,  3.58924294e-02, -1.25377084e+02,
        4.33398978e+00,  2.23482207e-02, -8.54009205e-05,  1.05267264e-02,
       -9.54992785e+00, -8.02419068e+00, -1.48516558e-06, -1.89678872e-03,
       -2.89866581e-01, -1.30560016e-01, -1.33071674e-02, -1.27599274e+02,
       -6.09794087e+01,  8.53750188e+03, -2.24954222e+03,  2.24478069e+03])
```

In [105]:

```
# lr.intercept_
model.intercept_
```

Out[105]:

```
47867918.24478324
```

In [106]:

```
y_pred = lr.predict(x_test)
y_pred
```

Out[106]:

```
array([47.737425  , 39.02338394, 22.23622888, 34.67978097, 24.77320855,
       33.37398518, 37.99670412, 32.80283378, 42.67046385, 57.77061178,
       45.22143111, 24.95019238, 40.79493067, 52.53829317, 50.74464512,
       27.85910624, 37.64700218, 28.56401196, 18.76919939, 18.3504373 ,
       43.32510749, 15.97532092, 39.9409853 , 38.53202164, 26.94980829,
       28.80667563, 29.62812249, 53.64500886, 15.9352848 , 44.73877761,
       55.76121162, 43.98363999, 35.69946508, 49.21053347, 29.9582713 ,
       19.17678588, 49.94324894, 28.45277803, 32.23210655, 15.33233763,
       22.01546432, 30.02814393, 52.37805844, 39.46631756, 38.20469537,
       50.88411851, 45.81583012, 17.46285062, 45.22183378, 34.75537521,
       31.80412726, 42.46314272, 50.5586628 , 40.45696788, 48.82210195,
       45.57401511, 47.02995715,  8.1405787 , 44.21885642, 42.22999367,
       18.06767014, 55.08323424, 19.82581403, 47.24241443, 36.43434843,
       47.75834337, 59.97113802, 26.3289122 , 16.66986829, 18.48499367,
       25.28848647, 30.33666265, 21.15872747, 19.01363268, 19.32715356,
       21.47406061, 42.43209426, 49.52050402, 47.35496488, 30.59281138,
       19.17439804, 36.55856462, 52.7026816 ])
```

In [96]:

```
from sklearn import metrics
import math
y_pred = lr.predict(x_test)
print("MAE : ",metrics.mean_absolute_error(y_test,y_pred))
print("MSE : ",metrics.mean_squared_error(y_test,y_pred))
print("RMSE : ",math.sqrt(metrics.mean_squared_error(y_test,y_pred)))
print("R2score : ",metrics.r2_score(y_test,y_pred))
```

```
MAE :  5.226824101554342
MSE :  96.73983431451933
RMSE :  9.835641022044234
R2score :  0.5636304149032404
```

In [110]:

```
import numpy as np
test_data = np.array([54,67,69,34,14,22])
test_data = test_data.reshape(1,6)
test_data = poly.fit_transform(test_data)
lr.predict(test_data)
```

Out[110]:

```
array([39599971.72038367])
```