```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
import  pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns


from sklearn import linear_model
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error
from sklearn.preprocessing import PolynomialFeatures

from sklearn.model_selection import train_test_split
```

```
df = pd.read_csv(r'/content/drive/MyDrive/Amit NootBook/Projects/AMIT.ML .sesion..4/house price.csv')
```

```
df.shape
```

```
(414, 8)
```

```
df.columns
```

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

```
df.head()
```

	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

```
# Rename Columns

df = df.rename(columns={'No':'No' , 'X1 transaction date':'transaction date' , 'X2 house age':'house age' , 'X3 distance to the neares
```

```
df.head()
```

	No	transaction date	house age	distance to MRT	no of cov stores	latitude	longitude	house price
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

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 414 entries, 0 to 413
Data columns (total 8 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   No                414 non-null    int64
 1   transaction date  414 non-null    float64
 2   house age         414 non-null    float64
 3   distance to MRT   414 non-null    float64
 4   no of cov stores  414 non-null    int64
 5   latitude          414 non-null    float64
 6   longitude         414 non-null    float64
 7   house price       414 non-null    float64
```

```
dtypes: float64(6), int64(2)
memory usage: 26.0 KB
```

```
df.isna().sum()
```

```
No                  0
transaction date    0
house age           0
distance to MRT     0
no of cov stores    0
latitude            0
longitude           0
house price         0
dtype: int64
```

```
df['transaction date'] = df['transaction date'].astype(str)
```

```
df['transaction date'] = df['transaction date'].str.split('.')
2
3
```

```
df['transaction date'] = df['transaction date'].astype(str).str.split('.').str[0]
df['transaction date']
```

```
0      ['2012', '917']
1      ['2012', '917']
2      ['2013', '583']
3        ['2013', '5']
4      ['2012', '833']
           ...
409      ['2013', '0']
410    ['2012', '667']
411     ['2013', '25']
412      ['2013', '0']
413      ['2013', '5']
Name: transaction date, Length: 414, dtype: object
```

```
df['transaction year'] = df['transaction date'].str[2:6]
df['transaction year']
```

```
0      2012
1      2012
2      2013
3      2013
4      2012
       ...
409    2013
410    2012
411    2013
412    2013
413    2013
Name: transaction year, Length: 414, dtype: object
```

```
df['transaction month'] = df['transaction date'].str[10:11]
df['transaction month']
```

```
0      9
1      9
2      5
3      5
4      8
      ..
409    0
410    6
411    2
412    0
413    5
Name: transaction month, Length: 414, dtype: object
```

```
df['transaction day'] = df['transaction date'].str[11:13]
df['transaction day']
```

```
0      17
1      17
2      83
3      ']
4      33
       ..
409    ']
410    67
411    5'
412    ']
```

```
413    ']
Name: transaction day, Length: 414, dtype: object
```

```
df.head()
```

	No	transaction date	house age	distance to MRT	no of cov stores	latitude	longitude	house price	transacti ye
0	1	['2012', '917']	32.0	84.87882	10	24.98298	121.54024	37.9	20
1	2	['2012', '917']	19.5	306.59470	9	24.98034	121.53951	42.2	20
2	3	['2013', '583']	13.3	561.98450	5	24.98746	121.54391	47.3	20
3	4	['2013', '5']	13.3	561.98450	5	24.98746	121.54391	54.8	20

```
# Reorder the columns

df = df.reindex(columns = ['No' , 'transaction date' , 'transaction year' , 'transaction month' , 'transaction day' ,'house age' ,'dis
```

```
df
```

	No	transaction date	transaction year	transaction month	transaction day	house age	distance to MRT	st
0	1	['2012', '917']	2012	9	17	32.0	84.87882	
1	2	['2012', '917']	2012	9	17	19.5	306.59470	
2	3	['2013', '583']	2013	5	83	13.3	561.98450	
3	4	['2013', '5']	2013	5	']	13.3	561.98450	
4	5	['2012', '833']	2012	8	33	5.0	390.56840	
...	...	...	...	...	...	...	...	
409	410	['2013', '0']	2013	0	']	13.7	4082.01500	
410	411	['2012', '667']	2012	6	67	5.6	90.45606	
411	412	['2013', '25']	2013	2	5'	18.8	390.96960	
412	413	['2013', '0']	2013	0	']	8.1	104.81010	
413	414	['2013', '5']	2013	5	']	6.5	90.45606	

```
# Will drop Transaction date , Transaction day
df = df.drop(['transaction date','transaction day'] , axis =1)
```

```
df
```

	No	transaction year	transaction month	house age	distance to MRT	no of cov stores	latitude	longitude
0	1	2012	9	32.0	84.87882	10	24.98298	121.54024
1	2	2012	9	19.5	306.59470	9	24.98034	121.53951
2	3	2013	5	13.3	561.98450	5	24.98746	121.54391
3	4	2013	5	13.3	561.98450	5	24.98746	121.54391
4	5	2012	8	5.0	390.56840	5	24.97937	121.54245
...	...	...	...	...	...	...	...	...
409	410	2013	0	13.7	4082.01500	0	24.94155	121.50381
410	411	2012	6	5.6	90.45606	9	24.97433	121.54310
411	412	2013	2	18.8	390.96960	7	24.97923	121.53986
412	413	2013	0	8.1	104.81010	5	24.96674	121.54067
413	414	2013	5	6.5	90.45606	9	24.97433	121.54310

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 414 entries, 0 to 413
Data columns (total 9 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
```

```
 0   No                  414 non-null    int64
 1   transaction year    414 non-null    object
 2   transaction month   414 non-null    object
 3   house age           414 non-null    float64
 4   distance to MRT     414 non-null    float64
 5   no of cov stores    414 non-null    int64
 6   latitude            414 non-null    float64
 7   longitude           414 non-null    float64
 8   house price         414 non-null    float64
dtypes: float64(5), int64(2), object(2)
memory usage: 29.2+ KB
```

```
df['transaction year']  = df['transaction year'].astype('int64')
df['transaction month'] = df['transaction month'].astype('int64')
```

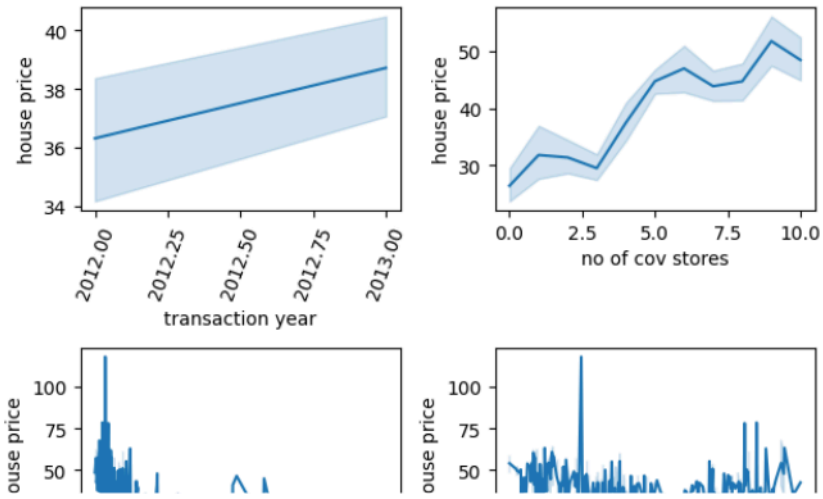
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 414 entries, 0 to 413
Data columns (total 9 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   No                  414 non-null    int64
 1   transaction year    414 non-null    int64
 2   transaction month   414 non-null    int64
 3   house age           414 non-null    float64
 4   distance to MRT     414 non-null    float64
 5   no of cov stores    414 non-null    int64
 6   latitude            414 non-null    float64
 7   longitude           414 non-null    float64
 8   house price         414 non-null    float64
dtypes: float64(5), int64(4)
memory usage: 29.2 KB
```

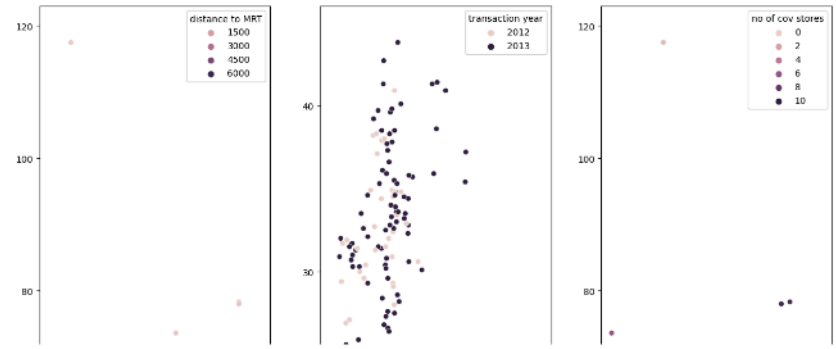
```
df
```

	No	transaction year	transaction month	house age	distance to MRT	no of cov stores	latitude	longitude
0	1	2012	9	32.0	84.87882	10	24.98298	121.54024
1	2	2012	9	19.5	306.59470	9	24.98034	121.53951
2	3	2013	5	13.3	561.98450	5	24.98746	121.54391
3	4	2013	5	13.3	561.98450	5	24.98746	121.54391
4	5	2012	8	5.0	390.56840	5	24.97937	121.54245
...	...	...	...	...	...	...	...	...
409	410	2013	0	13.7	4082.01500	0	24.94155	121.50381
410	411	2012	6	5.6	90.45606	9	24.97433	121.54310
411	412	2013	2	18.8	390.96960	7	24.97923	121.53986
412	413	2013	0	8.1	104.81010	5	24.96674	121.54067
413	414	2013	5	6.5	90.45606	9	24.97433	121.54310

```
plt.subplot(2,2,1)
sns.lineplot(data = df , x = df['transaction year'] , y = df['house price'] )
plt.xticks(rotation=70)


plt.subplot(2,2,2)
sns.lineplot(data = df , x = df['no of cov stores'] , y = df['house price'] )


plt.subplot(2,2,3)
sns.lineplot(data = df , x = df['distance to MRT'] , y = df['house price'] )


plt.subplot(2,2,4)
sns.lineplot(data = df , x = df['house age'] , y = df['house price'] )

plt.tight_layout()
```

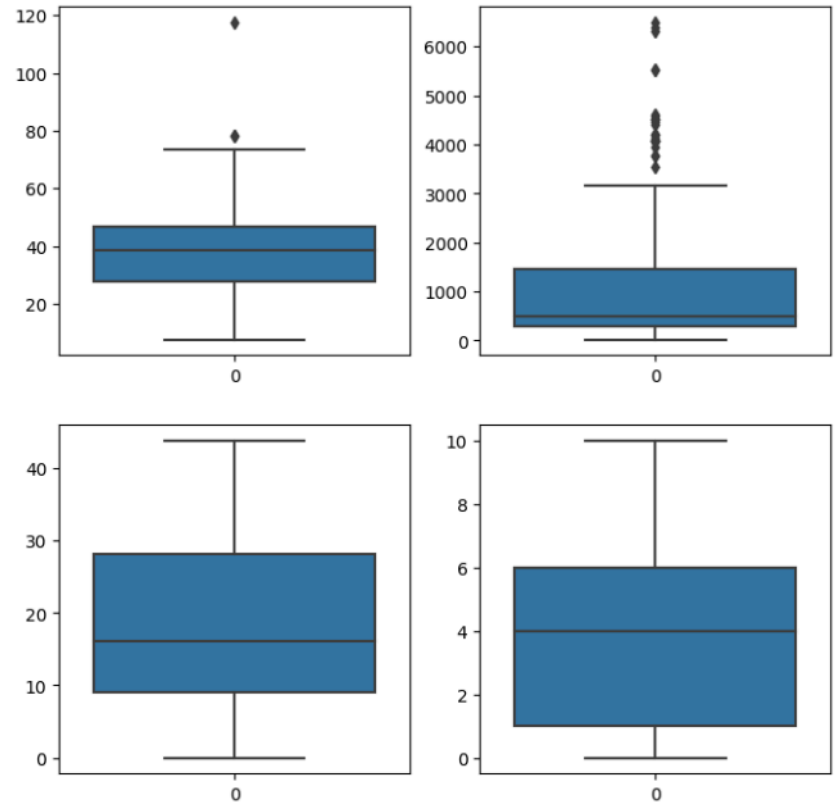



```
1 fig , axes = plt.subplots(1,3 ,  figsize=(13,13))
2
3 sns.scatterplot(data = df ,  x = df['no of cov stores'] , hue = df ['distance to MRT'] , y= df['house price'] ,  ax = axes[0])
4 sns.scatterplot(data = df ,  x = df['house price'] , y = df ['house age'] , hue = df['transaction year'] , ax = axes[1])
5 sns.scatterplot(data = df ,  x = df['house age'] , y = df ['house price'] , hue= df['no of cov stores'] ,  ax = axes[2])
6 plt.tight_layout()
```




```
fig , axes = plt.subplots(2,2 , figsize=(8,8))
sns.boxplot(df['house price'] , ax = axes[0][0])
sns.boxplot(df['distance to MRT'] , ax = axes[0][1])
sns.boxplot(df['house age'] , ax = axes[1][0])
sns.boxplot(df['no of cov stores'] , ax = axes[1][1])
```

```
<Axes: >
```



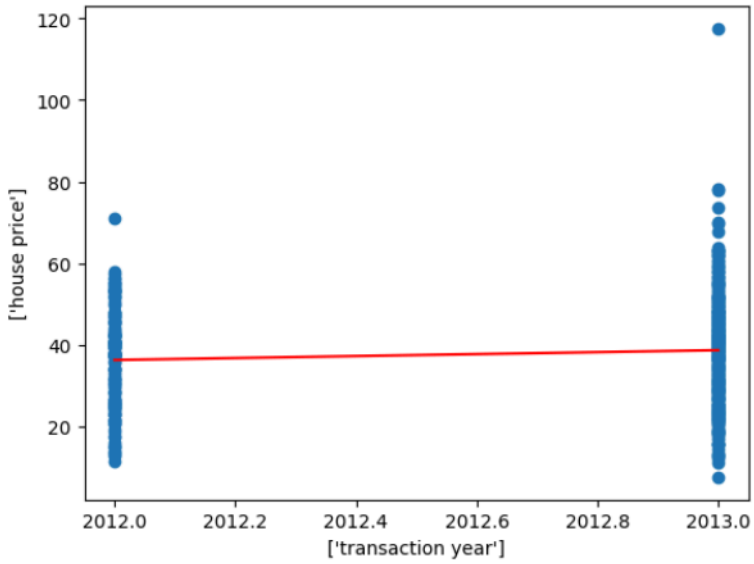
Regression

```
def linear_regression(x , y , x_label , y_label):
  model = linear_model.LinearRegression().fit(x,y)
  # Get the intercept (a_0) and coefficient (a_1)
  a_0 = model.intercept_
  a_1 = model.coef_

  # Scoring the prediction accuracy
  y_pred = model.predict(x)

  # R2, MSE and RMSE*
  r_score = r2_score(y, y_pred)
  MSE = mean_squared_error(y, y_pred)
  RMSE = np.sqrt(mean_squared_error(y, y_pred))

  print(
        'The linear regression equation is y = {} + {}*x'.format(a_0, a_1[0]))
  print('R2 is {} '.format(r_score))
  print('The mean squared error (MSE) is {}'.format(MSE))
  print('The root mean squared error (RMSE) is {}'.format(RMSE))
  print('\n')
```

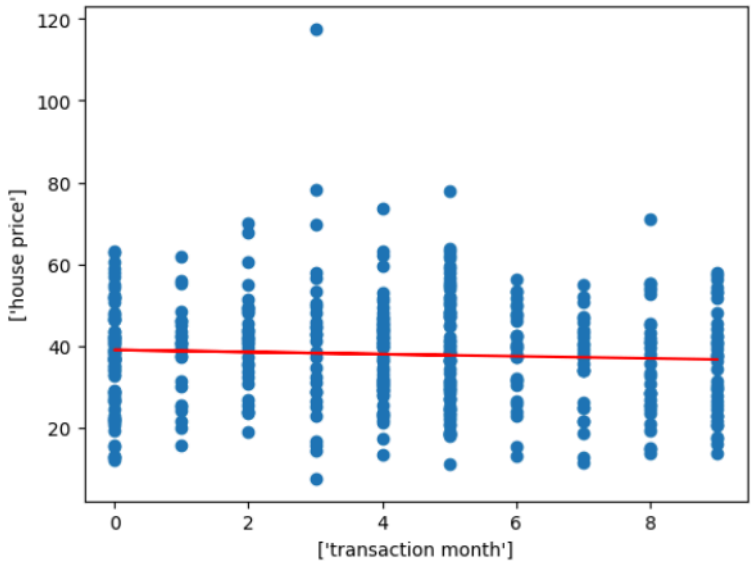
```

# Graph
plt.scatter(x, y)
plt.xlabel(x_label)
plt.ylabel(y_label)
plt.plot(x, y_pred, c='r')
plt.show()
```

```
for xdata in ['transaction year', 'transaction month', 'house age','distance to MRT', 'no of cov stores', 'latitude', 'longitude']:
  print('{}:'.format(xdata))
  x, x_label = df[[xdata]].values, list(df[[xdata]].columns)
  y, y_label = df[['house price']].values, list(df[['house price']].columns)
  linear_regression( x , y , x_label , y_label)

```

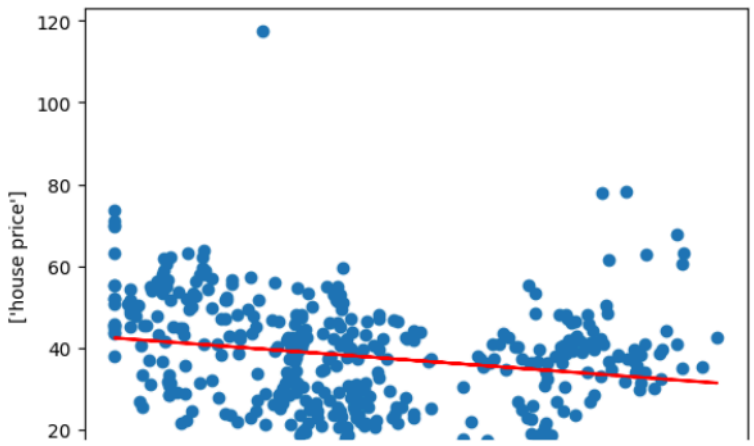
```
transaction year:
The linear regression equation is y = [-4809.46150794] + [2.40843254]*x
R2 is 0.006649508505266799
The mean squared error (MSE) is 183.4612246472663
The root mean squared error (RMSE) is 13.544785884142515
```

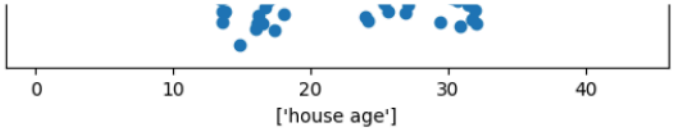



```
transaction month:
The linear regression equation is y = [39.05310539] + [-0.25869868]*x
R2 is 0.0030074678088517492
The mean squared error (MSE) is 184.133870659022
The root mean squared error (RMSE) is 13.569593606995825
```

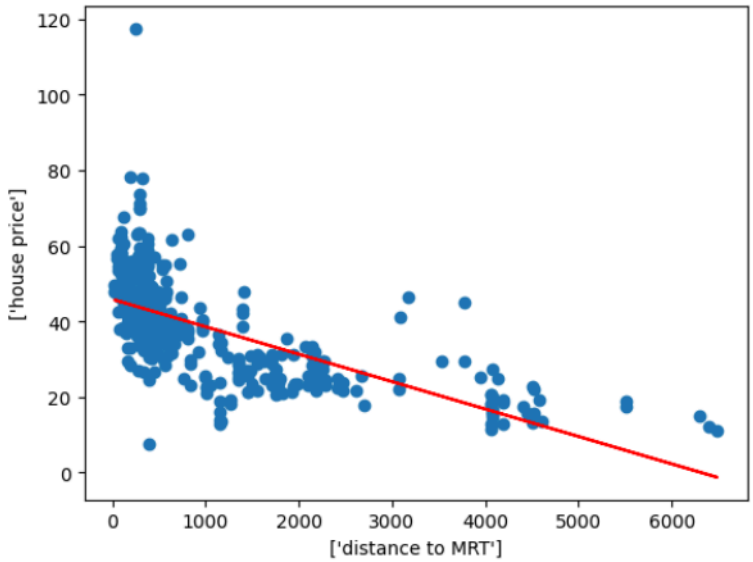



```
house age:
The linear regression equation is y = [42.43469705] + [-0.25148842]*x
R2 is 0.04433848097791171
The mean squared error (MSE) is 176.50047403131393
The root mean squared error (RMSE) is 13.285348095978287
```

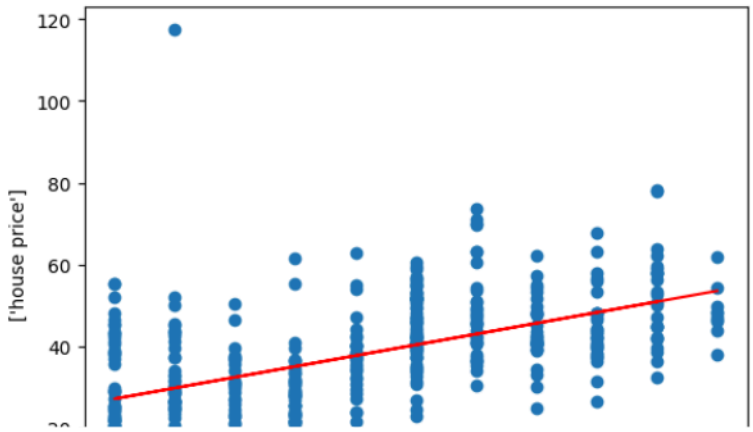





```
distance to MRT:
The linear regression equation is y = [45.85142706] + [-0.00726205]*x
R2 is 0.45375427891826703
The mean squared error (MSE) is 100.88574959799587
The root mean squared error (RMSE) is 10.044189842789505
```




```
no of cov stores:
The linear regression equation is y = [27.18110478] + [2.63765346]*x
R2 is 0.32604660851305056
The mean squared error (MSE) is 124.47199212769486
The root mean squared error (RMSE) is 11.156701668848857
```



Multilinear

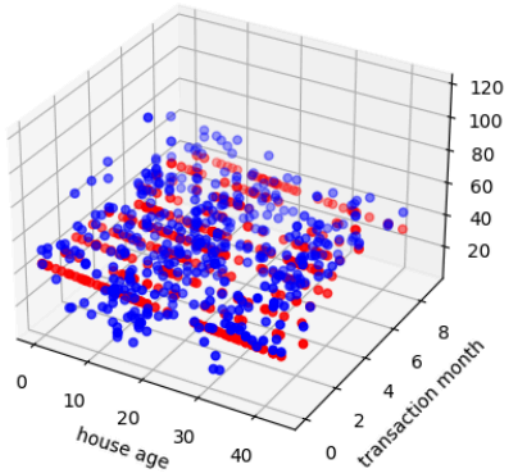

```
from mpl_toolkits.mplot3d import Axes3D

def multi_lin(x, y):
  model = linear_model.LinearRegression().fit(x,y)
  y_pred = model.predict(x)

  a_0 = model.intercept_[0]
  a_1 = model.coef_[0][0]
  a_2 = model.coef_[0][1]

  print('equation is  ,  {} + ({})*x1 + ({})*x2'.format(a_0 , a_1 , a_2))

  # R2, MSE and RMSE*
  r_score = r2_score(y, y_pred)
  MSE = mean_squared_error(y, y_pred)
  RMSE = np.sqrt(mean_squared_error(y, y_pred))

  print('\n')
  print('r_score is ' , r_score)
  print('MSE is     ' , MSE)
```

```
  print('RMSE is    ' , RMSE)
  print('\n')



  # Create a 3D scatter plot
  fig = plt.figure()
  ax = fig.add_subplot(111, projection='3d')
  ax.scatter(x.iloc[:,0] , x.iloc[:,1] , y, color='b') # Actual values
  ax.scatter(x.iloc[:,0] , x.iloc[:,1]  , y_pred , c ='r')  # Predicted values
  # ax.plot_surface(x[: , 0] , x[: ,1] , y_pred, cmap = plt.cm.Accent)
  # ax.plot_wireframe(x[: , 0] , x[: ,1] , y_pred)


  print(('--------------------'))
  # print(df[x])
  plt.xlabel(x.columns[0])
  plt.ylabel(x.columns[1])
  plt.show()



l = [['house age', 'transaction month'], ['house age','distance to MRT'], ['no of cov stores', 'latitude'], ['longitude', 'latitude']]
y = df[['house price']].values
for x in l :
  x= df[x]
  multi_lin(x,y)
```

```
equation is  ,  43.91559738967889 + (-0.2574902914166262)*x1 + (-0.3314391452352582)
```

```
r_score is  0.049249739289331296
MSE is      175.59341708405628
RMSE is     13.25116663105767
```

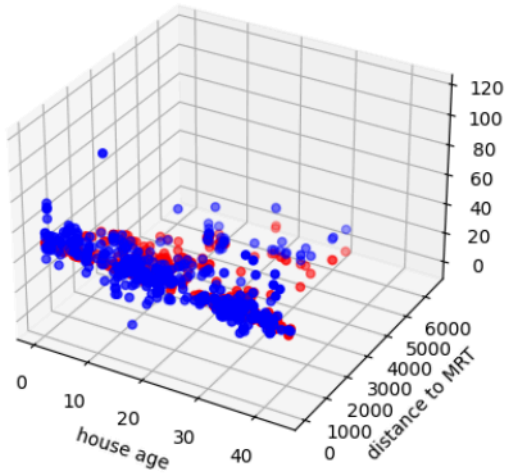
```
--------------------
```




```
equation is  ,  49.885585756906636 + (-0.2310265834572479)*x1 + (-0.0072086201430152
```

```
r_score is  0.49114669575911474
MSE is      93.97976963938086
RMSE is     9.69431635750458
```

```
--------------------
```




```
equation is  ,  -9951.734432278048 + (1.8895725187360568)*x1 + (399.77437262330074)*
```

```
r_score is  0.4327721306929613
MSE is      104.76092824048358
RMSE is     10.235278610789429
```

```
--------------------
```

1