```
#NAME:Shivani Gadkari
#ROLL NO:13342

#1.ALGORITHM(Housing Dataset)
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

 df=pd.read_csv("Housing.csv")

df
```

```
                  area_type   availability
location  \
0      Super built-up  Area         19-Dec  Electronic City Phase II

1                Plot  Area  Ready To Move          Chikka Tirupathi

2            Built-up  Area  Ready To Move               Uttarahalli

3      Super built-up  Area  Ready To Move        Lingadheeranahalli

4      Super built-up  Area  Ready To Move                  Kothanur

...                      ...           ...                       ...

13315        Built-up  Area  Ready To Move                Whitefield

13316  Super built-up  Area  Ready To Move             Richards Town

13317        Built-up  Area  Ready To Move     Raja Rajeshwari Nagar

13318  Super built-up  Area         18-Jun           Padmanabhanagar

13319  Super built-up  Area  Ready To Move              Doddathoguru


            size  society total_sqft  bath  balcony   price
0          2 BHK  Coomee        1056   2.0      1.0   39.07
1      4 Bedroom  Theanmp       2600   5.0      3.0  120.00
2          3 BHK      NaN       1440   2.0      3.0   62.00
3          3 BHK  Soiewre       1521   3.0      1.0   95.00
4          2 BHK      NaN       1200   2.0      1.0   51.00
...          ...      ...        ...   ...      ...     ...
13315  5 Bedroom  ArsiaEx       3453   4.0      0.0  231.00
13316      4 BHK      NaN       3600   5.0      NaN  400.00
13317      2 BHK  Mahla T       1141   2.0      1.0   60.00
13318      4 BHK  SollyCl       4689   4.0      1.0  488.00
13319      1 BHK      NaN        550   1.0      1.0   17.00
```

```
[13320 rows x 9 columns]
```

```
import pandas as pd
# Example: Loading from CSV
    df = pd.read_csv("Housing.csv")
# Example: Loading from a Python object
class Housing:
    df = pd.read_csv("Housing.csv")
df = pd.DataFrame(Housing.df)
```

```
  Cell In[8], line 3
    df = pd.read_csv("Housing.csv")
    ^
IndentationError: unexpected indent
```

```
import pandas as pd

# Example: Loading from CSV
df = pd.read_csv("Housing.csv")

# Example: Loading from a Python object
class Housing:
    # Indented correctly within the class
    df = pd.read_csv("Housing.csv")

# Now you can access the `df` attribute from the Housing class
df = pd.DataFrame(Housing.df)
```

```
df.columns
```

```
Index(['area_type', 'availability', 'location', 'size', 'society',
       'total_sqft', 'bath', 'balcony', 'price'],
      dtype='object')
```

```
df.isnull().sum()
```

```
area_type          0
availability       0
location           1
size              16
society         5502
total_sqft         0
bath              73
balcony          609
price              0
dtype: int64
```

```
x = df.drop(['price'], axis=1)
y = df['price']
```

```
from sklearn.model_selection import train_test_split
xtrain, xtest, ytrain, ytest =train_test_split(x, y, test_size
=0.2,random_state = 0)

from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
df['area_type'] = le.fit_transform(df['area_type'])
newdf=df

from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
df['availability'] = le.fit_transform(df['availability'])
newdf=df

from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
df['location'] = le.fit_transform(df['location'])
newdf=df

from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
df['size'] = le.fit_transform(df['size'])
newdf=df

from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
df['society'] = le.fit_transform(df['society'])
newdf=df

df
```

```
       area_type  availability  location  size  society total_sqft
bath  \
0              3            40       419    13      464       1056
2.0
1              2            80       317    19     2439       2600
5.0
2              0            80      1179    16     2688       1440
2.0
3              3            80       757    16     2186       1521
3.0
4              3            80       716    13     2688       1200
2.0
...          ...           ...       ...   ...      ...        ...   .
..
13315          0            80      1252    22      209       3453
4.0
13316          3            80      1004    18     2688       3600
5.0
13317          0            80       972    13     1216       1141
2.0
```

```
13318          3            32       907    18     2205        4689
4.0
13319          3            80       396     0     2688         550
1.0
```

```
       balcony   price  are_type
0          1.0   39.07         3
1          3.0  120.00         2
2          3.0   62.00         0
3          1.0   95.00         3
4          1.0   51.00         3
...        ...     ...       ...
13315      0.0  231.00         0
13316      NaN  400.00         3
13317      1.0   60.00         0
13318      1.0  488.00         3
13319      1.0   17.00         3

[13320 rows x 10 columns]
```

```
from sklearn.model_selection import train_test_split
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(x, y,
test_size=0.4,random_state=10)
```

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

```
print(df.dtypes)
```

```
area_type          int64
availability       int32
location           int32
size               int32
society            int32
total_sqft        object
bath             float64
balcony          float64
price            float64
are_type           int32
dtype: object
```

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
df['total_sqft'] = le.fit_transform(df['total_sqft' ])
newdf=df
```

```
print(df.dtypes)
```

```
area_type          int64
availability       int32
```

```
location          int32
size              int32
society           int32
total_sqft        int32
bath            float64
balcony         float64
price           float64
are_type          int32
dtype: object
```

```
print(X_train.head()) # Check if any columns are still categorical
```

```
                  area_type   availability      location   size
society  \
2022   Super built-up  Area  Ready To Move  Doddanekundi  3 BHK
AkletBa
7773   Super built-up  Area  Ready To Move  Channasandra  2 BHK  Unm
2El
3104   Super built-up  Area  Ready To Move      Nagavara  3 BHK
NaN
12487        Built-up  Area  Ready To Move  Sanjay nagar  2 BHK
NaN
5897   Super built-up  Area         18-May        Hebbal  2 BHK  Arcia
S

      total_sqft  bath  balcony
2022        2045   4.0      3.0
7773        1050   2.0      1.0
3104        2430   4.0      3.0
12487       1150   2.0      0.0
5897        1088   2.0      2.0
```

```
X_train = pd.get_dummies(X_train, drop_first=True)
```

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

# Prepare data
X = df.drop('price', axis=1)
y = df['price']

# Split the data
X_train, X_test, Y_train, Y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Drop rows with missing values
X_train = X_train.dropna()
Y_train = Y_train[X_train.index]  # Ensure Y_train is aligned with
X_train
```

```
X_test = X_test.dropna()
Y_test = Y_test[X_test.index]  # Ensure Y_test is aligned with X_test

# One-hot encoding of categorical variables (if any)
X_train = pd.get_dummies(X_train, drop_first=True)
X_test = pd.get_dummies(X_test, drop_first=True)

# Create and train the linear regression model
lm = LinearRegression()
lm.fit(X_train, Y_train)
```

```
LinearRegression()
```

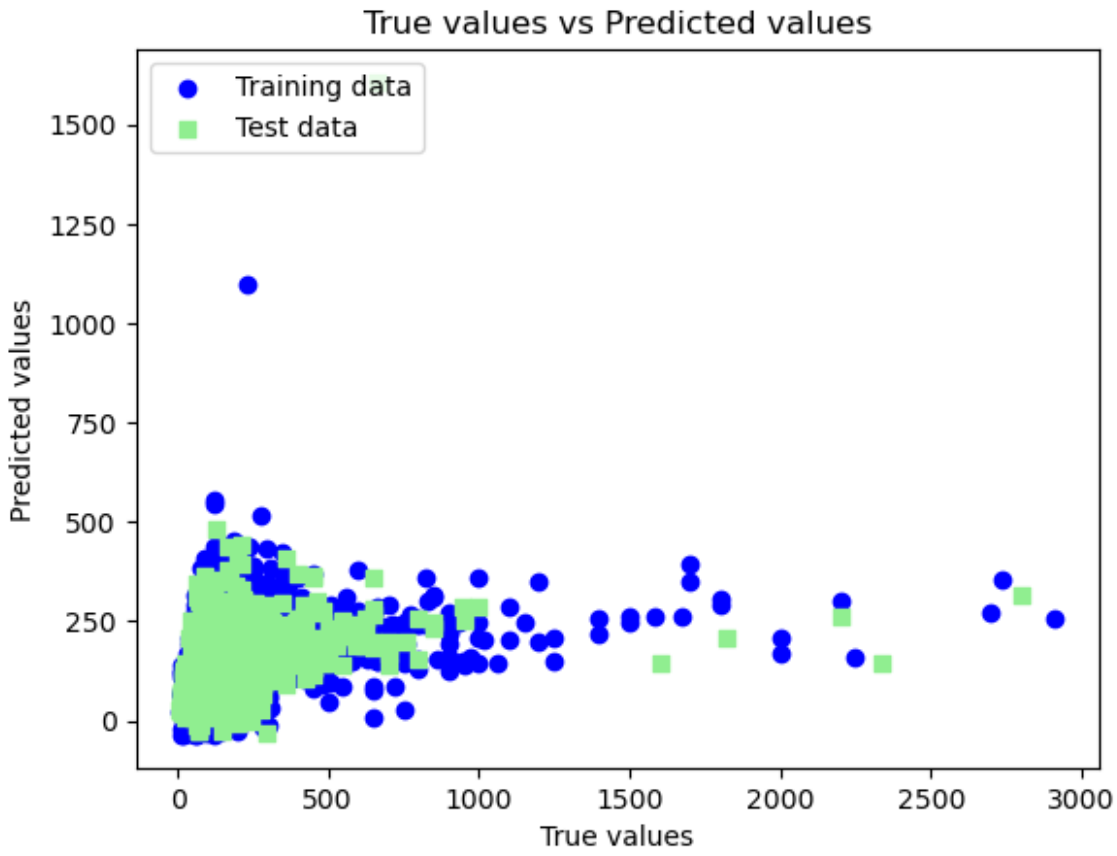
```
Y_pred = lm.predict(X_test)
print("Predictions:", Y_pred)
```

```
Predictions: [ 46.61943271 124.36567869  47.99075956 ... 132.29835756
63.36553523
 104.29057163]
```

```
import sklearn
from sklearn.linear_model import LinearRegression
lm = LinearRegression()
model=lm.fit(X_train, Y_train)

Ytrain_pred = lm.predict(X_train)
Ytest_pred = lm.predict(X_test)

df=pd.DataFrame(Ytrain_pred,Y_train)
df=pd.DataFrame(Ytest_pred,Y_test)

from sklearn.metrics import mean_squared_error, r2_score
mse = mean_squared_error(Y_test, Ytest_pred)
print(mse)
mse = mean_squared_error(Ytrain_pred,Y_train)
print (mse)
```

```
15042.627983565915
12708.433216734145
```

```
plt.scatter(Y_train, Ytrain_pred, c='blue', marker='o',
label='Training data')
plt.scatter(Y_test, Ytest_pred, c='lightgreen', marker='s',
label='Test data')
plt.xlabel('True values')
plt.ylabel('Predicted values')
plt.title("True values vs Predicted values")
plt.legend(loc='upper left')
plt.show()
```




```
#2.Algorithm (Synthesis Dataset):

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

x=np.array([95,85,80,70,60])
y=np.array([85,95,70,65,70])

model= np.polyfit(x, y, 1)

model

array([ 0.64383562, 26.78082192])

predict = np.poly1d(model)
predict(65)

68.63013698630137

y_pred= predict(x)
y_pred

array([87.94520548, 81.50684932, 78.28767123, 71.84931507,
65.4109589 ])
```

```
from sklearn.metrics import r2_score
r2_score(y, y_pred)
```

```
0.4803218090889326
```

```
y_line = model[1] + model[0]* x
plt.plot(x, y_line, c = 'r')
plt.scatter(x, y_pred)
plt.scatter(x,y,c='r')
```

```
<matplotlib.collections.PathCollection at 0x184dcb17dd0>
```