```
1 import pandas as pd
2 import numpy as np
```

```
1 data=pd.read_csv('/content/train.csv')
2
```

```
1 data.head()
2
```

	beds	baths	size	size_units	lot_size	lot_size_units	zip_code	price
0	3	2.5	2590.0	sqft	6000.00	sqft	98144	795000.0
1	4	2.0	2240.0	sqft	0.31	acre	98106	915000.0
2	4	3.0	2040.0	sqft	3783.00	sqft	98107	950000.0
3	4	3.0	3800.0	sqft	5175.00	sqft	98199	1950000.0
4	2	2.0	1042.0	sqft	NaN	NaN	98102	950000.0

Next steps: Generate code with `data` View recommended plots New interactive sheet

```
1 data.shape
2
```

```
(2016, 8)
```

```
1 for column in data.columns:
2     print(data[column].value_counts())
3     print("*"*20)
```

```
98144    113
98122    109
98118    100
98116     88
98107     83
98126     80
98106     78
98125     78
98105     73
98199     72
98119     70
98133     61
98109     61
98136     60
98102     60
98121     59
98112     57
98178     44
98168     44
98146     41
98108     33
98177     27
98101     23
98104     14
98164      1
Name: count, dtype: int64
**********
price
750000.0     27
700000.0     25
850000.0     23
950000.0     20
900000.0     19
             ..
205000.0      1
3400000.0     1
1278500.0     1
6250000.0     1
659000.0      1
Name: count, Length: 767, dtype: int64
**********
```

```
1 data.isna().sum()
2
```

	0
beds	0
baths	0
size	0
size_units	0
lot_size	347
lot_size_units	347
zip_code	0
price	0

dtype: int64

```
1 data.drop(columns=['lot_size','lot_size_units'],inplace=True)
2
```

```
1 data.describe()
2
```

	beds	baths	size	zip_code	price
count	2016.000000	2016.000000	2016.000000	2016.000000	2.016000e+03
mean	2.857639	2.159970	1735.740575	98123.638889	9.636252e+05
std	1.255092	1.002023	920.132591	22.650819	9.440954e+05
min	1.000000	0.500000	250.000000	98101.000000	1.590000e+05
25%	2.000000	1.500000	1068.750000	98108.000000	6.017500e+05
50%	3.000000	2.000000	1560.000000	98117.000000	8.000000e+05
75%	4.000000	2.500000	2222.500000	98126.000000	1.105250e+06
max	15.000000	9.000000	11010.000000	98199.000000	2.500000e+07

```
1 data['beds'].value_counts()
2
```

beds	count
3	645
2	560
4	398
1	256
5	123
6	22
9	5
7	3
8	2
15	1
14	1

dtype: int64

```
1 data.head()
2
```

	beds	baths	size	size_units	zip_code	price
0	3	2.5	2590.0	sqft	98144	795000.0
1	4	2.0	2240.0	sqft	98106	915000.0
2	4	3.0	2040.0	sqft	98107	950000.0
3	4	3.0	3800.0	sqft	98199	1950000.0
4	2	2.0	1042.0	sqft	98102	950000.0

Next steps: Generate code with `data` View recommended plots New interactive sheet

```
1 data['price_per_sqft'] = data['price'] * 100000 / data['size']
2
```

```
1 data['price_per_sqft']
2
```

	price_per_sqft
0	3.069498e+07
1	4.084821e+07
2	4.656863e+07
3	5.131579e+07
4	9.117083e+07
...	...
2011	6.642336e+07
2012	6.186727e+07
2013	5.373832e+07
2014	7.421384e+07
2015	3.853801e+07

2016 rows × 1 columns

dtype: float64

```
data.drop(columns=['price_per_sqft'],inplace=True)
```

```
data.to_csv("final_dataset.csv")
```

```

X=data.drop(columns=['price', 'size_units' ])
y=data['price']
```

```

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression,Lasso,Ridge
from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.compose import make_column_transformer
from sklearn.pipeline import make_pipeline
from sklearn.metrics import r2_score
```

```
X_train,X_test,y_train,y_test = train_test_split(X,y, test_size=0.2, random_state=0)
```

```
1 print(X_train.shape)
2 print(y_train.shape)
```

```
(1612, 4)
(1612,)
```

```
1 X_train
```

	beds	baths	size	zip_code
1354	3	3.5	1644.0	98112
942	2	1.0	2060.0	98106
1170	1	1.0	513.0	98121
651	2	2.0	1644.0	98101
360	3	3.5	1968.0	98199
...	...	...	...	...
835	3	2.0	1650.0	98126
1216	1	1.0	718.0	98119
1653	2	1.0	1760.0	98126
559	2	2.5	1460.0	98106
684	1	1.0	707.0	98133

1612 rows × 4 columns

Next steps: Generate code with X_train | View recommended plots | New interactive sheet

```
1 column_trans = make_column_transformer((OneHotEncoder(sparse=False), ['beds']), remainder
2
```

```
1 scaler = StandardScaler()
2
```

```
1 from sklearn.linear_model import LinearRegression
2
3 lr = LinearRegression()
```

```
1 # Identify columns with non-numeric data
2 non_numeric_columns = X.select_dtypes(include=['object']).columns
3 print(non_numeric_columns)
4
```

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

```
1 X
```

	beds	baths	size	zip_code
0	3	2.5	2590.0	98144
1	4	2.0	2240.0	98106
2	4	3.0	2040.0	98107
3	4	3.0	3800.0	98199
4	2	2.0	1042.0	98102
...	...	...	...	...
2011	3	2.0	1370.0	98112
2012	1	1.0	889.0	98121
2013	4	2.0	2140.0	98199
2014	2	2.0	795.0	98103
2015	3	2.0	1710.0	98133

2016 rows × 4 columns

Next steps: Generate code with X | View recommended plots | New interactive sheet

```
1 # Convert non-numeric columns to numeric using one-hot encoding
2 X_numeric = pd.get_dummies(X, drop_first=True)
3
```

```
1 X_numeric
```

	beds	baths	size	zip_code
0	3	2.5	2590.0	98144
1	4	2.0	2240.0	98106
2	4	3.0	2040.0	98107
3	4	3.0	3800.0	98199
4	2	2.0	1042.0	98102
...	...	...	...	...
2011	3	2.0	1370.0	98112
2012	1	1.0	889.0	98121
2013	4	2.0	2140.0	98199
2014	2	2.0	795.0	98103
2015	3	2.0	1710.0	98133

2016 rows × 4 columns

Next steps: Generate code with X_numeric | View recommended plots | New interactive sheet

```
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import StandardScaler

# Scaling the numeric data
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X_numeric)

# Fit the Linear Regression model
lr = LinearRegression()
lr.fit(X_scaled, y)

```

```
LinearRegression()
```

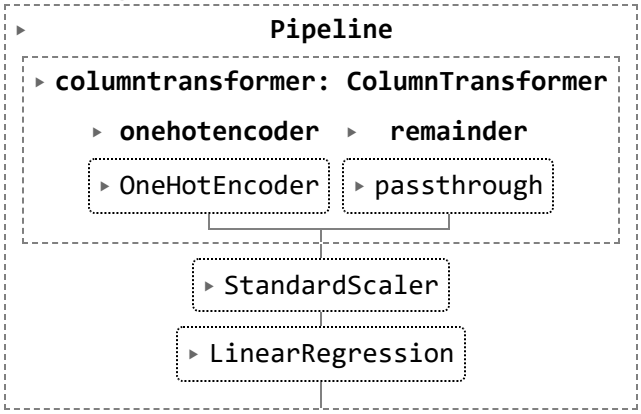
```
pipe = make_pipeline(column_trans,scaler, lr)

```

```
pipe.fit(X_train,y_train)

```

```
/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/_encoders.py:975: FutureWa
  warnings.warn(
```




```
1 y_pred_lr = pipe.predict(X_test)
2
```

```
1 r2_score(y_test,y_pred_lr)
2
```

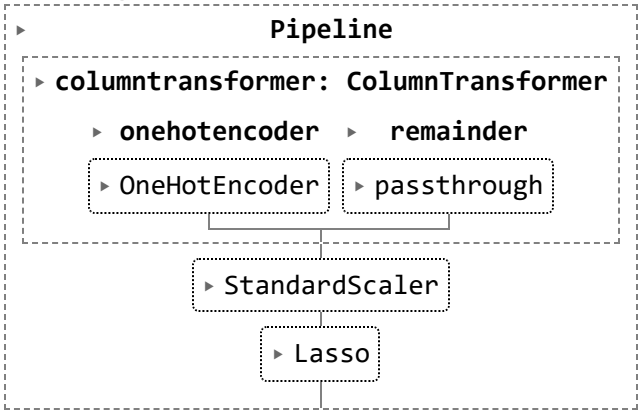
```
0.574030519852051
```

```
1 lasso = Lasso()
2
```

```
1 pipe = make_pipeline(column_trans,scaler, lasso)
2
```

```
1 pipe.fit(X_train,y_train)
2
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/_encoders.py:975: FutureWa
  warnings.warn(
```



```
1 y_pred_lasso = pipe.predict(X_test)
2 r2_score(y_test,y_pred_lasso)
```

```
0.5746817917322382
```

```
1 ridge = Ridge()
2
```

```
1 pipe.fit(X_train,y_train)
2
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/_encoders.py:975: FutureWa
  warnings.warn(
```