Creating a scatter plot

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

```
data = pd.read_csv('data/home_dataset.csv')

# Extracting features and target variables
house_size = data ['HouseSize'].values
house_prices = data ['HousePrice'].values

data.head()
```

	HouseSize	HousePrice
0	793	1300000
1	2477	3700000
2	1263	1480000
3	1291	2380000
4	603	955000

Looking at house prices vs house sizes

```
# Visualizing the data

plt.scatter(house_size, house_prices, marker = 'x', color = 'red')
plt.title('House Price vs House Size')
plt.xlabel('Size of House (sq ft)')
plt.ylabel('Price of House (USD $)')
plt.show
```

Train test

Now that we extracted our data and plotted on a graph, we can now perform linear regression.

A train-test splits the generated data into training and testing sets using the train_test_split function from sklearn.

The train test split is a model validation procedure commonly used in predictive machine learning to simulate how a model wil lwork with new-unknown data. This is mostly used in larger datasets or when we need a quick estimate.

```
x_train, x_test, y_train, y_test = train_test_split(house_size, house_prices,
test_size = 0.2, random_state=42)
```

This code above splits the data 80/20. So, 80% of the data will be used to train the algorithm in x_train and y_train.

20% will be used for testing the alogorithm in x_test and y_test.

In our next step, we are creating and training the model to make predictions.

```
# Reshaping the data
x_train = x_train.reshape(-1, 1)
x_test = x_test.reshape(-1, 1)

# Creating and train the model
model = LinearRegression()
model.fit(x_train, y_train)
```

```
LinearRegression()
```

Since we prepared our data for Numpy with .reshape(), we can create the linear regression model with LinearRegression() from sklearn and train it using our x_train and y_train data.

Predicting

The model we just trained will now be able to make predictions on the test set (x-test and y-test). Our results will be stored in a predictions table.

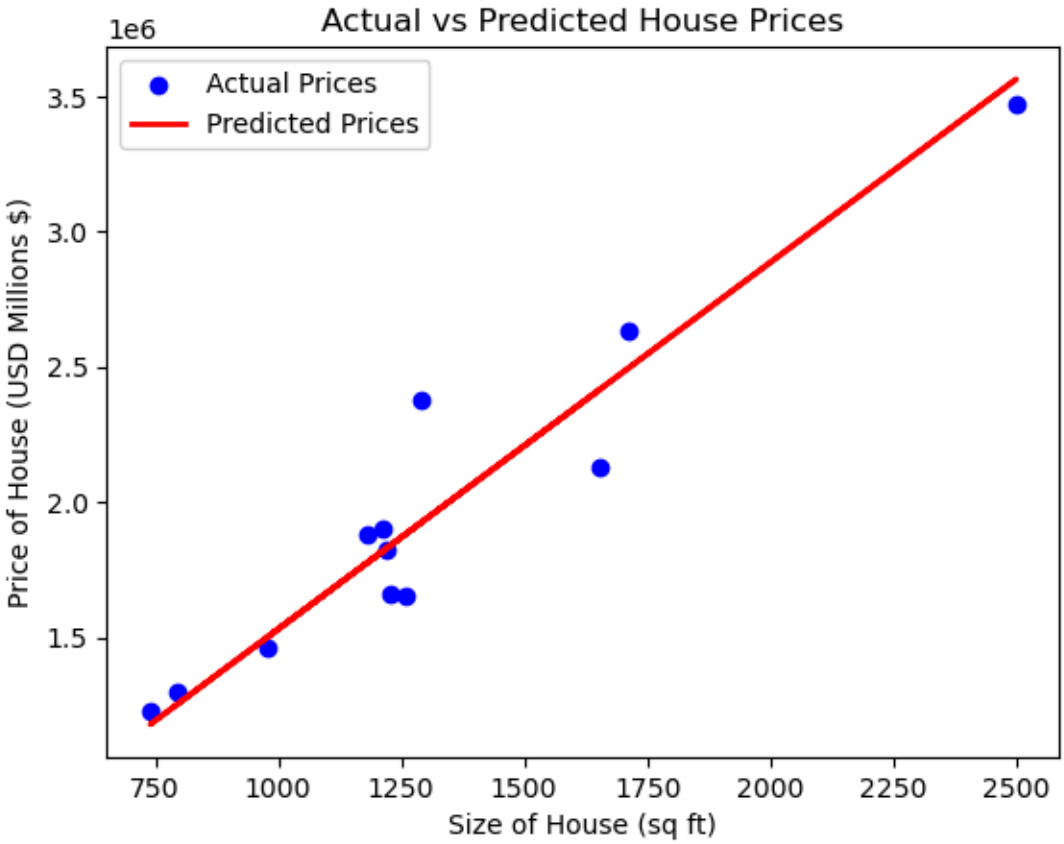
We will now visualize the results of our predictions.

```
# Predicting prices for the test set

predictions = model.predict(x_test)

# Visualizing the results

plt.scatter(x_test, y_test, marker = 'o', color = 'blue', label = 'Actual
Prices')
plt.plot(x_test, predictions, color = 'red', linewidth = 2, label = 'Predicted
Prices')
plt.title('Actual vs Predicted House Prices')
plt.xlabel('Size of House (sq ft)')
plt.ylabel('Price of House (USD Millions $)')
plt.legend()
plt.show()
```



This plot, titled "Actual vs Predicted House Prices," demonstrates a linear regression model's ability to predict house prices based on their size. The blue dots represent the actual prices of houses at various sizes, while the red line shows the predicted prices generated by the model. The model assumes a linear relationship, suggesting that as the size of a house increases, its price also increases proportionally. The plot allows for a visual assessment of the model's accuracy; the closer the blue dots are to the red line, the more accurate the predictions. While the model appears to be a reasonable fit for the data, some points are further from the line, indicating prediction errors or potential outliers.