Linear Regression From Scratch

What is Linear Regression?

- **Linear regression is a statistical method used to establish a relationship between a dependent variable and one or more independent variables.**
- **It involves fitting a straight line to the data points to make predictions.**

Where it can used?

- **In healthcare, linear regression can be used to model the relationship between patient characteristics and health outcomes.**
- **In marketing, it can be used to determine the impact of advertising or promotional campaigns on sales.**
- **In sports, it can be used to analyze the relationship between a team's performance and various factors such as player statistics, weather conditions, or game strategies.**
- **In environmental science, it can be used to study the relationship between pollutants and their effects on the ecosystem.**

Formula

$$y = mx + c$$

```
y = Output / Dependent Variable.
```

```
m = Slope.
X = Input / Independent Variable.
C = Constant / y-intercept.
```

- **The `intercept` is the value where the regression line crosses the y-axis. It represents the predicted value of the dependent variable when all independent variables are equal to zero.**
- **The `intercept is called an intercept because` it represents the point where the regression line intercepts the vertical y-axis on a graph. It is also sometimes referred to as the `"y-intercept"` because it is the point where the regression line intersects the y-axis, which is the vertical axis in a two-dimensional coordinate system.**
- **The `slope` represents the change in the dependent variable (y) for a one-unit change in the independent variable (x). It is the rate at which the dependent variable changes with respect to the independent variable. The slope is also sometimes referred to as the `"regression coefficient"` or `"beta coefficient."`**

Example 01

Let's say you want to `predict a student's final exam score based on the number of hours they study`. In this case, the number of hours studied is the independent variable, also known as the predictor variable, and the final exam score is the dependent variable, also known as the response variable.

Here's a sample dataset of 6 students with their corresponding hours studied and final exam scores:

In [1]:

```
# Hours Studied     Final Exam Score
#       2                 60
#       3                 70
#       4                 80
#       5                 85
#       6                 90
#       7                 95
```

- To perform a linear regression, we can use the least squares method to find the line of best fit that minimizes the sum of the squared errors between the predicted values and the actual values.
- In a linear regression analysis, the goal is to find a line that best fits the observed data. The line is characterized by an intercept and a slope. The least squares method involves finding the values of the intercept and slope that minimize the sum of the squared differences between the predicted values of the dependent variable (y) and the actual values of the dependent variable (y) for a given set of independent variable (x) values.

- The equation for a simple linear regression model is: ##

$$y = mx + c$$

- where y is the dependent variable (final exam score), x is the independent variable (hours studied), c is the y-intercept, and m is the slope of the line.
- Using the dataset above, we can calculate the `slope` and `intercept` of the line of best fit as follows:

Formula of Slope

$$m= cov(x,y) / var(x)$$

$$m= \sum [(x - xMean) * (y - yMean)] / (x - xMean)^2$$

- `cov: covariance` , `var: variance`
- `Covariance:` Covariance is a measure of how much two variables are linearly related to each other. In this case, it measures how much the values of x and y vary together. When the covariance is positive, it indicates that as x increases, y tends to increase as well. When the covariance is negative, it indicates that as x increases, y tends to decrease. If the covariance is zero, it indicates that there is no linear relationship between x and y.
- `Variance:` Variance is a statistical measure of how much the values in a dataset vary from the mean (average) value. It measures the spread of the data around the mean. A high variance indicates that the data points are widely spread out, while a low variance indicates that the data points are closely clustered around the mean.

Finding the Mean

- `xMean` = (2+3+4+5+6+7)/6 = 4.5
- `yMean` = (60+70+80+85+90+95)/6 = 80
- `cov(x,y)` = sum((x - xMean) * (y - yMean) for x, y in zip(hours_studied, final_exam_scores)) = 120
- `var(x)` = sum((x - xMean)**2 for x in hours_studied) = 17.5
- `m` = cov(x,y) / var(x)= 120 / 17.5 = 6.8
- `c` = yMean - m(xMean) = 80 - 6.8(4.5) = 49.14

Therefore, the equation of the line of best fit is:

$$final\_exam\_score = 49.14 + 6.8*hours\_studied$$

We can use this equation to predict a student's final exam score based on the number of hours they study. For example, if a student studies for 8 hours, their predicted final exam score would be:

final_exam_score = 49.14 + 6.8*8 = 103.54

R Square

`R- squared` : R-squared value is a statistical measure of how close the data are to the fitted the regression line. It is also known as coefficient of determination, or the coefficient of multiple determination. If value of R^2 =

1 its mean no error occur 100 % line is fitted. If value of R^2 = 0 its mean error occur 100 % line is not fitted.

Formula

R^2= ∑ (yPredicted - yMean)^2 / (y - yMean)^2

Implementing the above Example

01. Importing Libraries

In [2]:

```
import numpy as np
import matplotlib.pyplot as plt
```

02. Creating arrays for x (Hours Studied) and y (Final Exam Score)

In [3]:

```
x = np.array([2,3,4,5,6,7])
y = np.array([60,70,80,85,90,95])

print(f"Hours Studied: {x}\nFinal Exam Score: {y}")
```

```
Hours Studied: [2 3 4 5 6 7]
Final Exam Score: [60 70 80 85 90 95]
```

03. Finding Mean of x and y

```
Mean = (Sum of all values)/(Total Values)
```

In [5]:

```
xMean = np.mean(x)
yMean = np.mean(y)
print(f"xMean: {xMean}\nyMean: {yMean}")
```

```
xMean: 4.5
yMean: 80.0
```

04. Finding Slope `""m""`

m= cov(x,y) / var(x)

m= ∑ [(x - xMean) * (y - yMean)] / (x - xMean)^2

In [8]:

```
covXY = 0
varX = 0

n = len(x) # we need this as required to iterate n times

for i in range(n):
    covXY += (x[i] - xMean) * (y[i] - yMean)
    varX += (x[i] - xMean)**2
```

```
m = covXY / varX
print(covXY)
print(varX)
print(f"Slope: {m}")
```

```
120.0
17.5
Slope: 6.857142857142857
```

05. Finding c or y-Intercept

In [10]:

```
c = yMean - (m * xMean)
print(f"c:{c}")
```

```
c:49.142857142857146
```

06. Predicting the Value of y (Final Exam Score)

In [14]:

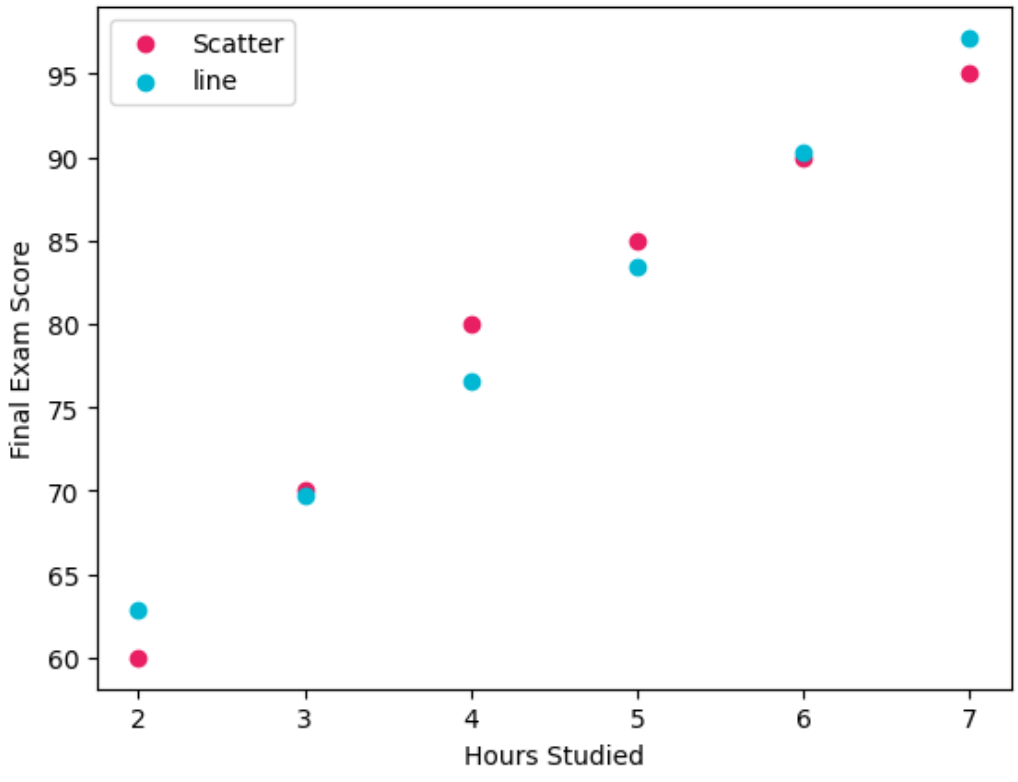
```
yPredict = m*x + c
print(yPredict)
```

```
[62.85714286 69.71428571 76.57142857 83.42857143 90.28571429 97.14285714]
```

07. Plotting the Original and Predicted Values

In [16]:

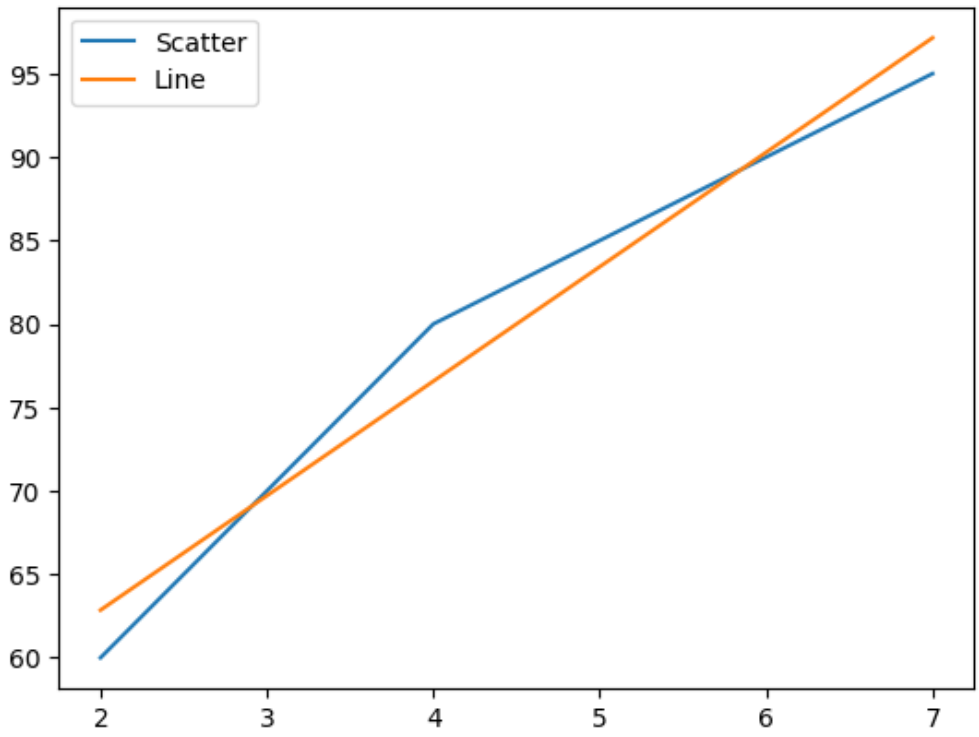
```
plt.scatter(x,y, color = "#e91e63", label = "Scatter")
plt.xlabel("Hours Studied")
plt.ylabel("Final Exam Score")
plt.scatter(x,yPredict, color = "#00b8d4", label = "line")
plt.legend()
plt.show()
```



In [21]:

```
plt.plot(x,y, label ="Scatter")
```

```
plt.plot(x,yPredict, label = "Line")
plt.legend()
plt.show()
```



08. Finding R Square

R^2= ∑ (yPredicted - yMean)^2 / (y - yMean)^2

In [27]:

```
rNum = 0
rDen = 0

for i in range(n):
    rNum += (yPredict[i] - yMean)**2
    rDen += (y[i] - yMean)**2

r = rNum / rDen

print(f"R Square: {r}")
print(f"Accuracy is: {round(r*100)}%")
print(f"Error: {round(100-r*100)}%")
```

```
R Square: 0.9680672268907555
Accuracy is: 97%
Error: 3%
```

Example 02 (multiple independent variable)

Predicting the house price based on size, bedrooms and age

In [28]:

```
# House Size (sq ft) Bedrooms Age (years)      Price (USD)
#   1     1500             3         10            200,000
#   2     2000             4         5            300,000
#   3     1200             2         20            150,000
#   4     1800             3         15            250,000
```

```
#    5     1600              2          12             220,000
#    6     2400              4          8             350,000
#    7     1300              3          18             180,000
#    8     1700              2          10             210,000
#    9     1900              3          7             280,000
#    10     2200               4             4             320,000
```

We can use multiple linear regression to model the relationship between the price of the house and its features:

Price = β0 + β1 *Size* + *β2* Bedrooms + β3 * Age

where β0 is the intercept, β1, β2, and β3 are the regression coefficients.

B1 = Σ(xi - x̄)(yi - ȳ) / Σ(xi - x̄)^2

B2 = Σ(xi - x̄)(zi - ẑ) / Σ(xi - x̄)^2

B3 = Σ(xi - x̄)(wi - ŵ) / Σ(xi - x̄)^2

B0 = ȳ - B1x̄1 - B2x̄2 - B3x̄3

1. Import Data

In [38]:

```
# Data
size = [1500, 2000, 1200, 1800, 1600, 2400, 1300, 1700, 1900, 2200]
bedrooms = [3, 4, 2, 3, 2, 4, 3, 2, 3, 4]
age = [10, 5, 20, 15, 12, 8, 18, 10, 7, 4]
price = [200000, 300000, 150000, 250000, 220000, 350000, 180000, 210000, 280000, 320000]
```

2. Specify the Independent and Dependent variable

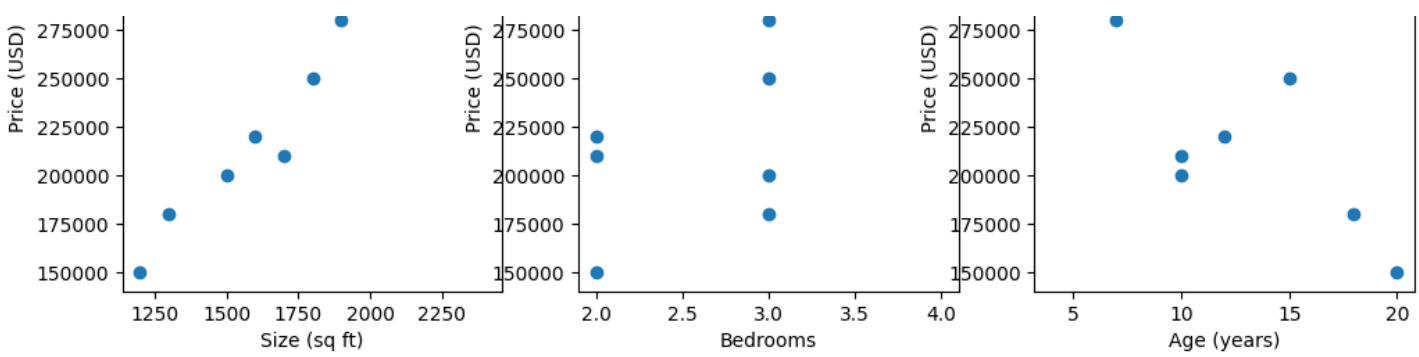
In [39]:

```
## Independent variables
x1 = size
x2 = bedrooms
x3 = age

# Dependent variable
y = price
```

3. Plot the Price against Each Feature

In [40]:

```
# Plotting
fig, axs = plt.subplots(1, 3, figsize=(12, 4))
axs[0].scatter(size, price)
axs[0].set_xlabel('Size (sq ft)')
axs[0].set_ylabel('Price (USD)')
axs[1].scatter(bedrooms, price)
axs[1].set_xlabel('Bedrooms')
axs[1].set_ylabel('Price (USD)')
axs[2].scatter(age, price)
axs[2].set_xlabel('Age (years)')
axs[2].set_ylabel('Price (USD)')

plt.show()
```



4. Find the Mean

In [44]:

```
x1Mean = np.mean(x1)
x2Mean = np.mean(x2)
x3Mean = np.mean(x3)

yMean = np.mean(y)
```

5. Finding Slope

```
y = m1*x1 + m2*x2 + m3*x3 + c
```

In [58]:

```
def findSlope(x,y):
    covXY = 0
    varX = 0

    xMean = np.mean(x)
    yMean = np.mean(y)

    n = len(x) # we need this as required to iterate n times

    for i in range(n):
        covXY += (x[i] - xMean) * (y[i] - yMean)
        varX += (x[i] - xMean)**2

    m = covXY / varX
    # print(covXY)
    # print(varX)
    # print(f"Slope: {m}")
    m = int(m)
    return m
```

`m1`

In [59]:

```
m1 = findSlope(x1,y)
print(f"Slope m1: {m1}")
```

```
Slope m1: 168
```

`m2`

In [60]:

```
m2 = findSlope(x2,y)
print(f"Slope m2: {m2}")
```

```
Slope m2: 65000
```

`m3`

In [61]:

```
m3 = findSlope(x3,y)
print(f"Slope m3: {m3}")
```

```
Slope m3: -9826
```

6. Finding c or y-Intercept

```
c = yMean - (m1 * x1Mean) - (m2 * x2Mean) - (m3 * x3Mean)
```

In [73]:

```
c = yMean - (m1 * x1Mean) - (m2 * x2Mean) - (m3 * x3Mean)
c
```

Out[73]:

```
-137576.59999999998
```

7. Predicting the Value of y

In [74]:

```
yPredict = m1*x1 + m2*x2 + m3*x3 + c
yPredict
```

Out[74]:

```
array([-136076.6, -135576.6, -136376.6, ..., -137574.6, -137573.6,
       -137572.6])
```

In [75]:

```
rNum = 0
rDen = 0

for i in range(n):
    rNum += (yPredict[i] - yMean)**2
    rDen += (y[i] - yMean)**2

r = rNum / rDen

print(f"R Square: {r}")
print(f"Accuracy is: {round(r*100)}%")
print(f"Error: {round(100-r*100)}%")
```

```
R Square: 38.32388181797056
Accuracy is: 3832%
Error: -3732%
```

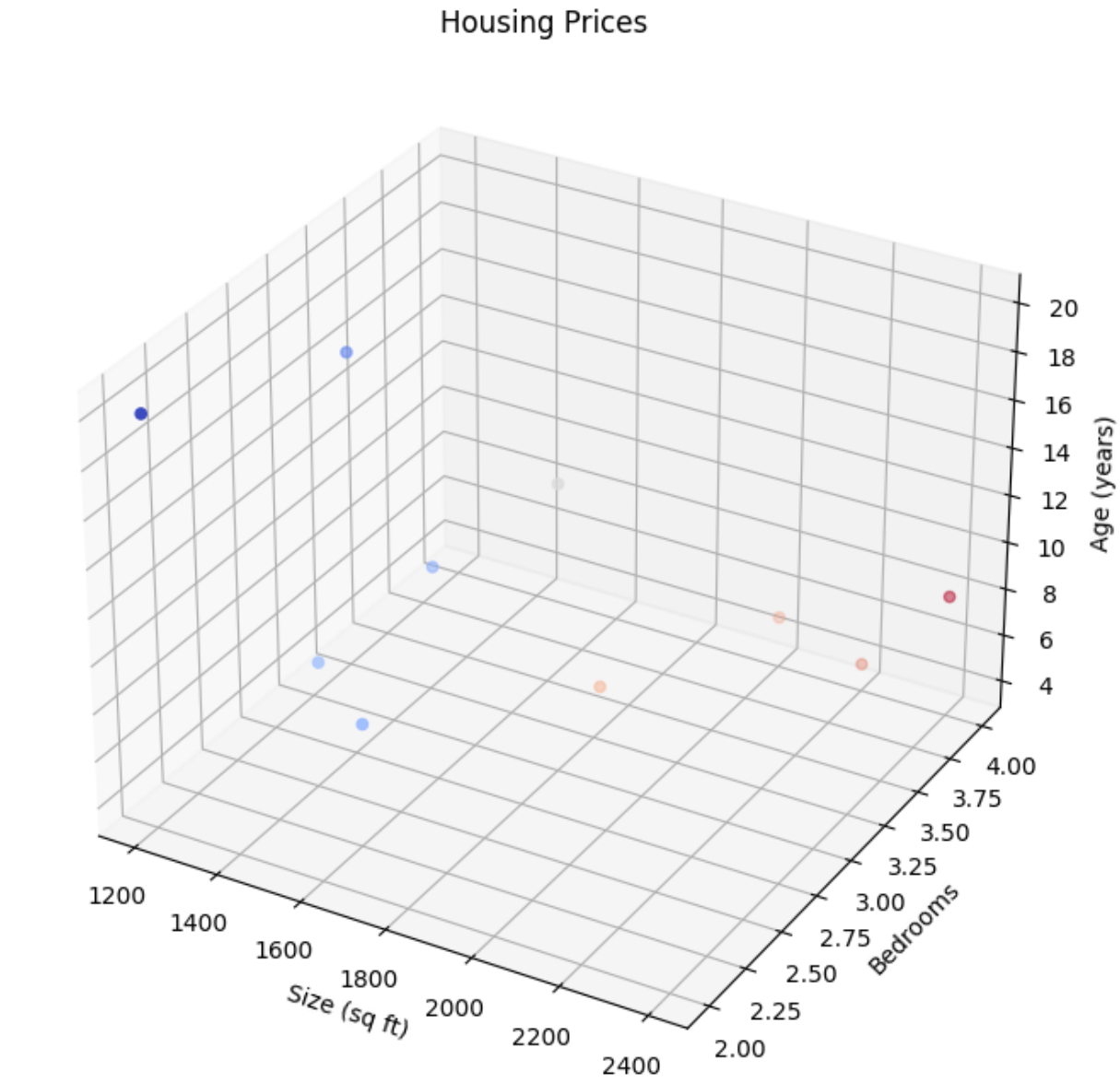
In [30]:

```
from mpl_toolkits.mplot3d import Axes3D
```

In [34]:

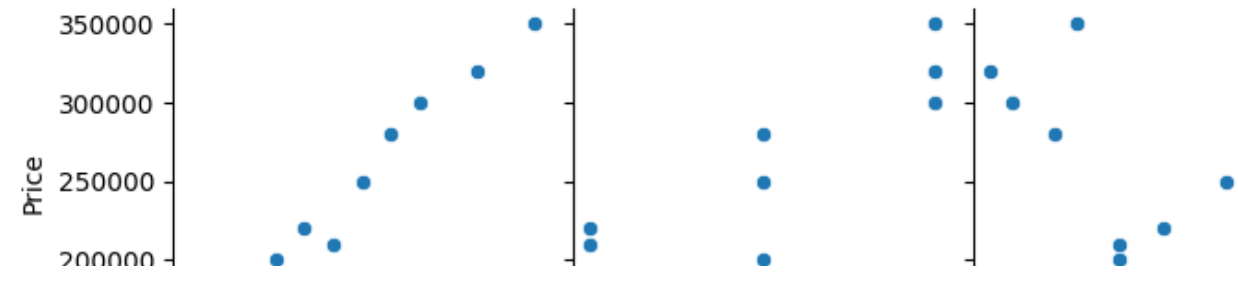
```
fig = plt.figure(figsize=(8, 8))
ax = fig.add_subplot(111, projection='3d')
ax.scatter(size, bedrooms, age, c=price, marker='o', cmap='coolwarm')
ax.set_xlabel('Size (sq ft)')
```

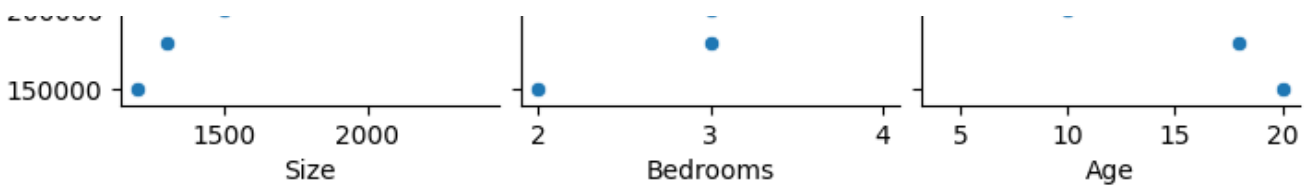
```
ax.set_ylabel('Bedrooms')
ax.set_zlabel('Age (years)')
ax.set_title('Housing Prices')
plt.show()
```



In [36]:

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

# Data
data = {'Size': [1500, 2000, 1200, 1800, 1600, 2400, 1300, 1700, 1900, 2200],
        'Bedrooms': [3, 4, 2, 3, 2, 4, 3, 2, 3, 4],
        'Age': [10, 5, 20, 15, 12, 8, 18, 10, 7, 4],
        'Price': [200000, 300000, 150000, 250000, 220000, 350000, 180000, 210000, 280000
, 320000]}
df = pd.DataFrame(data)

# Plotting
sns.pairplot(df, x_vars=['Size', 'Bedrooms', 'Age'], y_vars=['Price'], kind='scatter')
plt.show()
```





In [ ]: