In this module, we’ll dive into supervised learning with a focus on linear regression, a foundational technique in predictive modeling. We will cover the following topics: ¶

- 1- Introduction to Learning from Data
- 2- Simple and Multiple Linear Regression
- 3- Evaluating a Regression Model
- 4- Pros and Cons of Linear Regression

In [ ]:

```

```

1. Introduction to Learning from Data

- Definition:
  - Learning from data involves developing models that can identify patterns and make predictions based on input data.
  - In supervised learning, the model is trained on labeled data, where each input is associated with an output (label).
  - The goal is to learn a mapping from inputs to outputs that can be used to predict the labels for new, unseen data.
- Example: Consider a dataset containing information about houses (e.g., size, number of bedrooms) and their corresponding prices. The task is to predict the price of a new house based on its features.

In [ ]:

```

```

In [10]:

```python
import pandas as pd

# Sample data: House features and prices
data = {'Size (sq ft)': [1500, 1600, 1700, 1800, 1900],
        'Bedrooms': [3, 3, 3, 4, 4],
        'Price ($)': [300000, 320000, 340000, 360000, 380000]}

df = pd.DataFrame(data)
df
```

Out[10]:

	Size (sq ft)	Bedrooms	Price ($)
0	1500	3	300000
1	1600	3	320000
2	1700	3	340000
3	1800	4	360000
4	1900	4	380000

In [ ]:

```python

```

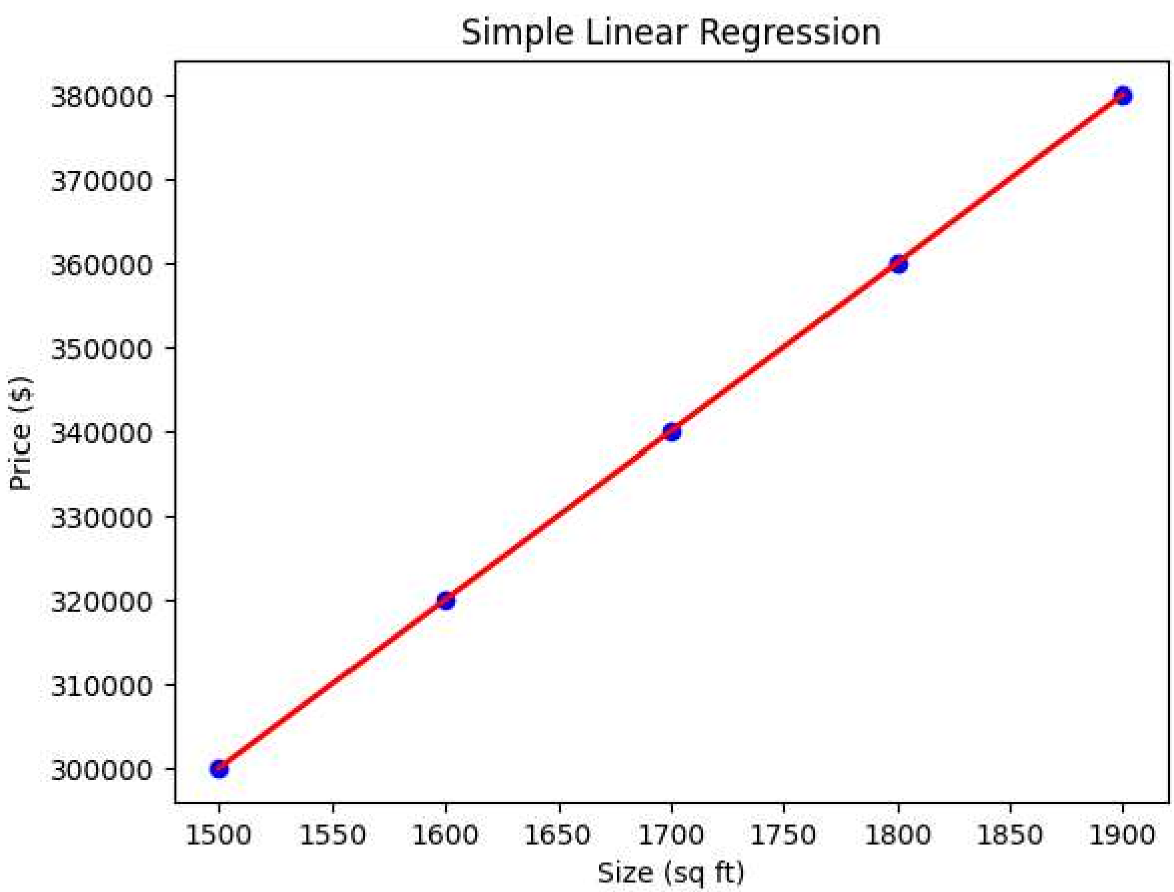
2. Simple and Multiple Linear Regression

2.1 Simple Linear Regression

- Definition:
  - Simple linear regression models the relationship between a single independent variable (feature) and a dependent variable (target) by fitting a straight line (linear relationship) to the data.
- Equation:
  - $y=\beta 0+\beta 1x+\epsilon$
  - Where:
    - y is the dependent variable (target)
    - x is the independent variable (feature)
    - β0 is the intercept
    - β1 is the slope
    - ϵ is the error term

In [3]:

```python
from sklearn.linear_model import LinearRegression
import numpy as np
import matplotlib.pyplot as plt

# Features (Size) and target (Price)
X = df['Size (sq ft)'].values.reshape(-1, 1)
y = df['Price ($)'].values

# Creating and training the model
model = LinearRegression()
model.fit(X, y)

# Predicting prices based on the model
predicted_prices = model.predict(X)

# Plotting the data and the regression line
plt.scatter(X, y, color='blue')
plt.plot(X, predicted_prices, color='red', linewidth=2)
plt.xlabel('Size (sq ft)')
plt.ylabel('Price ($)')
plt.title('Simple Linear Regression')
plt.show()

# Coefficients of the model
print(f"Intercept: {model.intercept_:.2f}")
print(f"Slope: {model.coef_[0]:.2f}")
```




```
Intercept: -0.00
Slope: 200.00
```

2.2 Multiple Linear Regression

- Definition:
  - Multiple linear regression extends simple linear regression by modeling the relationship between two or more independent variables and a dependent variable.
- Equation:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n + \epsilon$$

Where:

- y is the dependent variable
- $x_1, x_2, \ldots, x_n$ are the independent variables
- β_0 is the intercept
- $\beta_1, \beta_2, \ldots, \beta_n$ are the coefficients

In this case, the model learns the contribution of both the size of the house and the number of bedrooms to predict the price.

In [4]:

```python
# Features (Size and Bedrooms) and target (Price)
X = df[['Size (sq ft)', 'Bedrooms']].values

# Creating and training the model
model = LinearRegression()
model.fit(X, y)

# Predicting prices based on the model
predicted_prices = model.predict(X)

# Output model coefficients
print(f"Intercept: {model.intercept_:.2f}")
print(f"Coefficients: {model.coef_}")
```

```
Intercept: -0.00
Coefficients: [2.00000000e+02 2.00750019e-12]
```

Type *Markdown* and LaTeX: α^2

3. Evaluating a Regression Model

- To evaluate how well a regression model performs, we use several metrics:

3.1 Mean Absolute Error (MAE)

- Definition: The average of the absolute differences between the actual and predicted values.
- Equation: *

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

3.2 Mean Squared Error (MSE)

- Definition: The average of the squared differences between the actual and predicted values.
- Equation: *

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

3.3 Root Mean Squared Error (RMSE)

- Definition: The square root of the MSE, which gives an error measure in the same units as the target variable.
- Equation:
  - $$RMSE = \sqrt{MSE}$$

3.4 R-squared (R2)

- Definition: A measure of how well the independent variables explain the variability of the dependent variable.
- R2 ranges from 0 to 1, where 1 indicates a perfect fit.
- Equation:
  - $$R^2 = 1 - \frac{SS_{res}}{SS_{tot}}$$

In [5]:

```python
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

# Evaluating the model's predictions
mae = mean_absolute_error(y, predicted_prices)
mse = mean_squared_error(y, predicted_prices)
rmse = np.sqrt(mse)
r2 = r2_score(y, predicted_prices)

print(f"Mean Absolute Error (MAE): {mae:.2f}")
print(f"Mean Squared Error (MSE): {mse:.2f}")
print(f"Root Mean Squared Error (RMSE): {rmse:.2f}")
print(f"R-squared (R2): {r2:.2f}")
```

```
Mean Absolute Error (MAE): 0.00
Mean Squared Error (MSE): 0.00
Root Mean Squared Error (RMSE): 0.00
R-squared (R2): 1.00
```

4. Pros and Cons of Linear Regression

4.1 Pros:

- Simplicity: Linear regression is easy to understand and interpret.
- Efficiency: It's computationally efficient and works well with smaller datasets.
- Interpretability: The model coefficients provide clear insights into the relationship between features and the target variable.
- Baseline Model: It's often used as a baseline model to compare with more complex models.

4.2 Cons:

- Assumptions: Linear regression makes several assumptions (e.g., linearity, independence, homoscedasticity, normality) that may not hold in real-world data.
- Outliers: It's sensitive to outliers, which can significantly affect the model.
- Limited Flexibility: It can only model linear relationships. Non-linear relationships require more complex models.
- Overfitting: With too many features, the model can overfit the training data, leading to poor generalization.

Example: A discussion on the pros and cons in the context of predicting house prices:

- Pro: Linear regression provides a straightforward way to predict house prices based on features like size and the number of bedrooms.
- Con: If there are outliers in the data (e.g., a very expensive or very cheap house), the model's predictions could be skewed. Additionally, if the relationship between features and price is non-linear, linear regression may not capture it accurately.

****