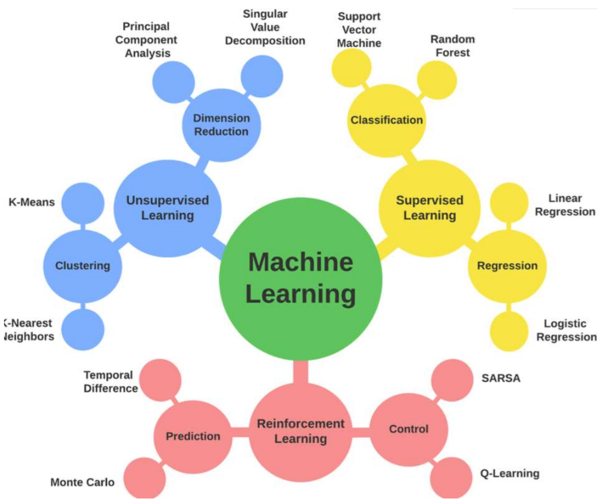
Các phương pháp học máy
Machine learning methods

4 TC: 2 LT – 2 TH

Giảng viên: **Tạ Hoàng Thắng**

tahoangthang@gmail.com

0975399307



Regression

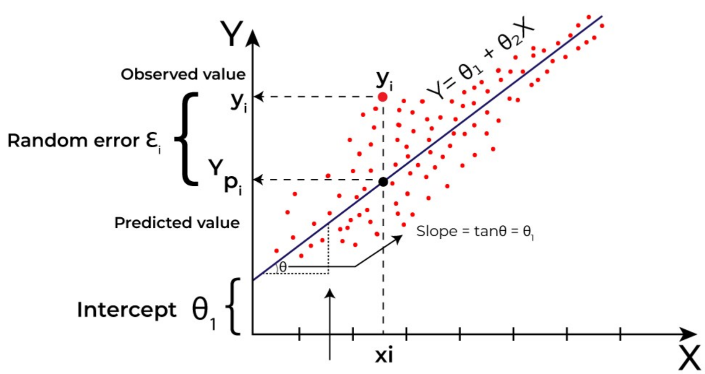
Definitions

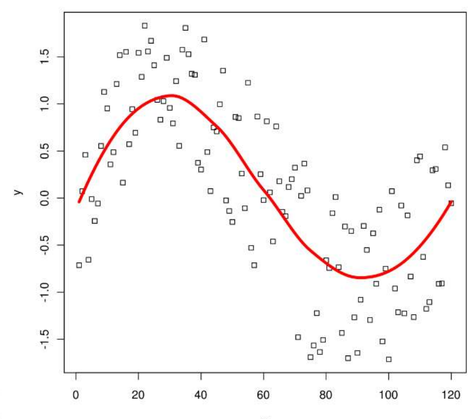
Regression is a **type of statistical and machine learning tech**nique used to model and analyze the relationship between

- **a dependent variable** (also called the target or response variable) and one or more
- **independent variables** (also called features or predictors).

The goal of regression is:

- to predict the value of the dependent variable based on the values of the independent variables.





Regression

Linear Regression

- Calculate slope, intercept of **ŷ = x*slop + intercept**:
  - x = [5, 7, 17, 25]
  - y = [20, 40, 70, 90]

Step 1: Calculate X*Y, X^2, Y^2, ΣX, ΣY, ΣX*Y, ΣX^2, and ΣY^2

	X	Y	X*Y	X^2	Y^2
	5	20	100	25	400
	7	40	280	49	1600
	17	70	1190	289	4900
	25	90	2250	625	8100
SUM	**54**	**220**	**3820**	**988**	**15000**

Regression

Linear Regression

- Calculate slope, intercept of **ŷ = x*slop + intercept**:
  - x = [5, 7, 17, 25]
  - y = [20, 40, 70, 90]
  - n = 4

	X	Y	X*Y	X^2	Y^2
	5	20	100	25	400
	7	40	280	49	1600
	17	70	1190	289	4900
	25	90	2250	625	8100
SUM	**54**	**220**	**3820**	**988**	**15000**

Step 2: Calculate intercept (b_0)

$[(\Sigma Y)(\Sigma X^2) - (\Sigma X)(\Sigma XY)] \ / \ [n(\Sigma X^2) - (\Sigma X)^2]$

= (220*988 – 54*3820) / (4*988 – 54*54)

= 10.6949…

Regression

Linear Regression

- Calculate slope, intercept of **ŷ = x*slop + intercept**:
  - x = [5, 7, 17, 25]
  - y = [20, 40, 70, 90]
  - n = 4

Step 3: Calculate slope (b_1)

[n(ΣXY) – (ΣX)(ΣY)] / [n(ΣX^2) – (ΣX)^2]

= (4*3820 – 54*220) / (4*988 – 54*54)

= 3.2818…

	X	Y	X*Y	X^2	Y^2
	5	20	100	25	400
	7	40	280	49	1600
	17	70	1190	289	4900
	25	90	2250	625	8100
SUM	**54**	**220**	**3820**	**988**	**15000**

=> ŷ = x*3.28… + 10.69…

Regression

Linear Regression

- **Bài tập nhóm:** Calculate slope, intercept of **ŷ = x*slop + intercept**:
  - x = [7, 5, 3, 1]
  - y = [2, 4, 6, 8]
  - n = 4

Regression

Linear Regression

Code:

```
# Import necessary libraries
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score

# Create a simple dataset
# Independent variable (X)
X = np.array([[1], [2], [3], [4], [5], [6], [7], [8], [9], [10]])

# Dependent variable (y)
y = np.array([3, 4, 2, 5, 7, 8, 8, 9, 10, 12])

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create a linear regression model
model = LinearRegression()
```

Regression

Linear Regression

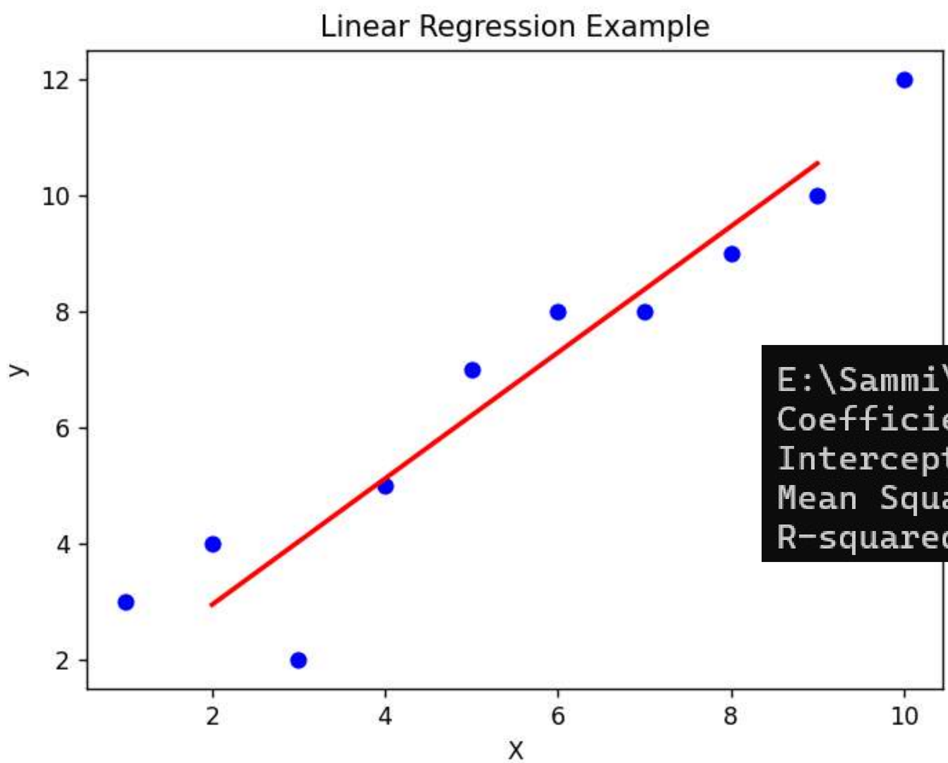
Code:

```

# Train the model
model.fit(X_train, y_train)

# Make predictions using the testing set
y_pred = model.predict(X_test)

# Print the coefficients
print("Coefficient:", model.coef_)
print("Intercept:", model.intercept_)

# Calculate performance metrics
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print("Mean Squared Error (MSE):", mse)
print("R-squared:", r2)

# Plot the results
plt.scatter(X, y, color='blue')  # Plot the original
plt.plot(X_test, y_pred, color='red', linewidth=2)  #
plt.xlabel('X')
plt.ylabel('y')
plt.title('Linear Regression Example')
plt.show()
```

Regression

Linear Regression

Code:




```
E:\Sammi\ML\Lab3>python test.py
Coefficient: [1.0862069]
Intercept: 0.7758620689655178
Mean Squared Error (MSE): 0.7052615933412598
R-squared: 0.92163760073986
```

Regression

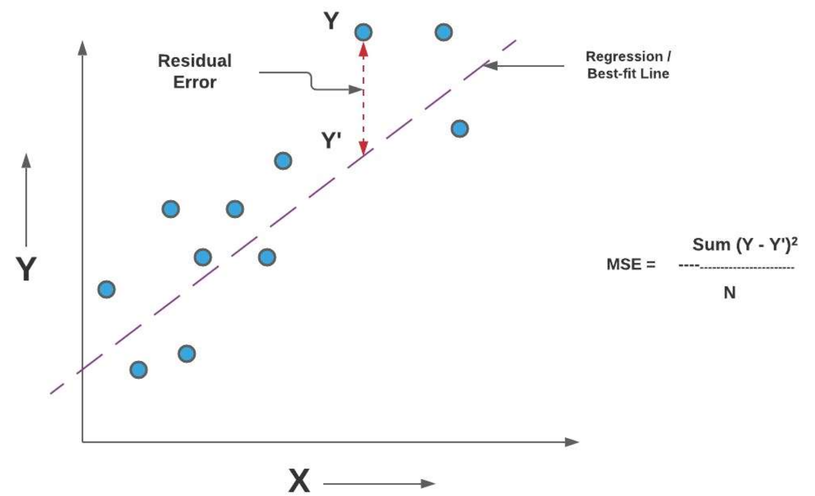
Mean Squared Error (MSE)

- is a metric used to evaluate the accuracy of a regression model.
  - measure **the average squared difference between the predicted values and the actual values**.
  - a **lower MSE indicates** better model performance

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

Where:

- n is the number of data points.
- y_i is the actual value.
- $\hat{y}_i$ is the predicted value.



Regression

Mean Squared Error (MSE)

- **Bài tập nhóm: Tính MSE theo yêu cầu sau:**

Suppose we have a dataset with 5 actual values y and the corresponding predicted values $\hat{y}$ from a regression model:

- Actual values (y): [3, 5, 2, 7, 1]
- Predicted values ($\hat{y}$): [2.5, 5.3, 2.1, 6.8, 1.2]

Regression

Root Mean Squared Error (RMSE)

- is a widely used metric to evaluate the accuracy of a regression model.

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

- **Bài tập nhóm: Tính MSE theo yêu cầu sau:**

Suppose we have a dataset with 5 actual values y and the corresponding predicted values $\hat{y}$ from a regression model:

- Actual values (y): [3, 5, 2, 7, 1]
- Predicted values ($\hat{y}$): [2.5, 5.3, 2.1, 6.8, 1.2]

Regression

R-squared

- the coefficient of determination, is a statistical metric used to evaluate the **goodness of fit of a regression model**.
  - Indicate how well the independent variables explain the variability of the dependent variable.
  - R-squared values range from 0 to 1
- Scale values:
  - $R^2 = 1$: perfectly explain any of the variability in the dependent variable
  - $R^2 = 0$: does not explain any of the variability in the dependent variable
  - $0 < R^2 < 1$: explain a proportion of the variability in the dependent variable, close to 1 better.

Regression

R-squared

- the coefficient of determination, is a statistical metric used to evaluate the goodness of fit of a regression model.

$$R^2 = 1 - \frac{SS_{RES}}{SS_{TOT}} = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \overline{y})^2}$$

- $\mathrm{SS}_{\mathrm{res}}$ is the **sum of squares of residuals** (also known as the sum of squared errors, or SSE):

$$\mathrm{SS}_{\mathrm{res}} = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

- $\mathrm{SS}_{\mathrm{tot}}$ is the **total sum of squares** (which measures the total variance in the dependent variable):

$$\mathrm{SS}_{\mathrm{tot}} = \sum_{i=1}^{n} (y_i - \bar{y})^2$$

Regression

R-squared

- **Bài tập nhóm: Tính R-squared theo yêu cầu sau:**

Suppose we have a dataset with 5 actual values y and the corresponding predicted values $\hat{y}$ from a regression model:

- Actual values (y): [3, 5, 2, 7, 1]
- Predicted values ($\hat{y}$): [2.5, 5.3, 2.1, 6.8, 1.2]

Regression

Multiple Linear Regression

- is an **extension of simple linear regression** that models the relationship **between a dependent variable and two or more independent variables**.
- to understand how changes in the independent variables influence the dependent variable and to predict the dependent variable's value based on those inputs.

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p + \epsilon$$

Regression

Multiple Linear Regression

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p + \epsilon$$

- y is the dependent variable (the outcome or target variable).
- $x_1, x_2, \ldots, x_p$ are the independent variables (predictors or features).
- β_0 is the intercept (the value of y when all independent variables are zero).
- $\beta_1, \beta_2, \ldots, \beta_p$ are the coefficients for the independent variables (indicating the change in y for a one-unit change in the corresponding x, holding all other variables constant).
- ϵ is the error term (representing the difference between the observed and predicted values).

Regression

Multiple Linear Regression

Example: Let's say we want to predict a house's price based on its size in square feet, the number of bedrooms, and the age of the house.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from sklearn.linear_model import LinearRegression

# Step 1: Create the Dataset
np.random.seed(42)
Size = np.random.randint(1500, 4000, 10)  # Random sizes between 1500 and 4000 sq ft
Bedrooms = np.random.randint(2, 6, 10)    # Random number of bedrooms between 2 and 6
Price = 50000 + 150 * Size + 30000 * Bedrooms + np.random.randn(10) * 10000  # Price calculation

df = pd.DataFrame({'Size': Size, 'Bedrooms': Bedrooms, 'Price': Price})
print(df)
```

Regression

Multiple Linear Regression

Example: Let's say we want to predict a house's price based on its size in square feet, the number of bedrooms, and the age of the house.

```python
# Step 2: Plot the Data
fig = plt.figure(figsize=(10, 7))
ax = fig.add_subplot(111, projection='3d')
ax.scatter(df['Size'], df['Bedrooms'], df['Price'],
           color='blue', label='Actual Data')
ax.set_xlabel('Size (sq ft)')
ax.set_ylabel('Bedrooms')
ax.set_zlabel('Price ($)')
plt.title('3D Scatter Plot of House Prices')
plt.legend()
plt.show()
```

Regression

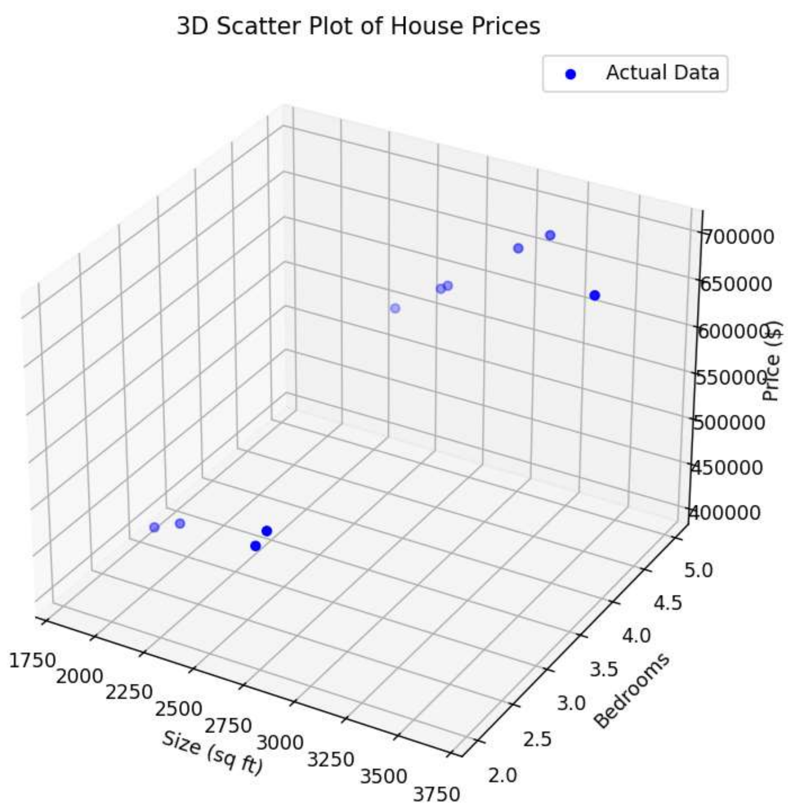
Multiple Linear Regression

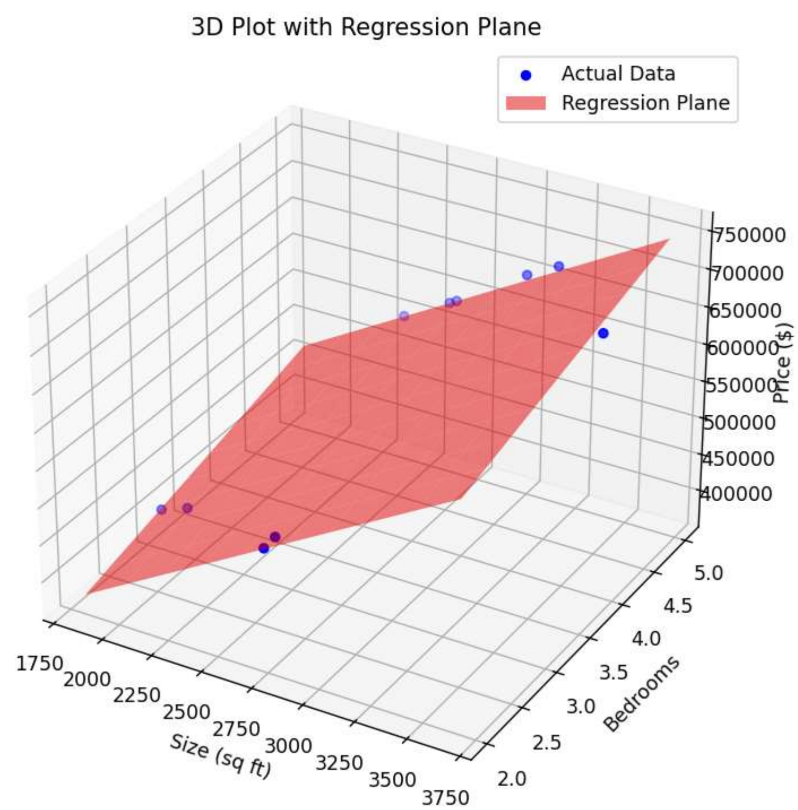
Example: Let's say we want to predict a house's price based on its size in square feet, the number of bedrooms, and the age of the house.

```
# Step 3: Perform Multiple Linear Regression
X = df[['Size', 'Bedrooms']]
y = df['Price']
model = LinearRegression()
model.fit(X, y)
print("Intercept:", model.intercept_)
print("Coefficients:", model.coef_)

# Step 4: Plot the Regression Plane
Size_grid, Bedrooms_grid = np.meshgrid(np.linspace(Size.min(), Size.max(), 10),
                                       np.linspace(Bedrooms.min(), Bedrooms.max(), 10))
Price_pred_grid = model.intercept_ + model.coef_[0] * Size_grid + model.coef_[1] * Bedrooms_grid

fig = plt.figure(figsize=(10, 7))
ax = fig.add_subplot(111, projection='3d')
ax.scatter(df['Size'], df['Bedrooms'], df['Price'], color='blue', label='Actual Data')
ax.plot_surface(Size_grid, Bedrooms_grid, Price_pred_grid, color='red', alpha=0.5, label='Regression Plane')
ax.set_xlabel('Size (sq ft)')
ax.set_ylabel('Bedrooms')
ax.set_zlabel('Price ($)')
plt.title('3D Plot with Regression Plane')
plt.legend()
plt.show()

```

Regression

Multiple Linear Regression





Regression

Multiple Linear Regression: Manual Calculation

Refer: https://www.statology.org/multiple-linear-regression-by-hand/

Suppose we have the following dataset with one response variable *y* and two predictor variables X_1 and X_2:

y	X_1	X_2
140	60	22
155	62	25
159	67	24
179	70	20
192	71	15
200	72	14
212	75	14
215	78	11

Regression

Multiple Linear Regression: Manual Calculation

Step 1: Calculate X_1^2, X_2^2, X_1y, X_2y and X_1X_2.

	y	X_1	X_2
	140	60	22
	155	62	25
	159	67	24
	179	70	20
	192	71	15
	200	72	14
	212	75	14
	215	78	11
Mean	181.5	69.375	18.125
Sum	1452	555	145

	X_1^2	X_2^2	X_1y	X_2y	X_1X_2
	3600	484	8400	3080	1320
	3844	625	9610	3875	1550
	4489	576	10653	3816	1608
	4900	400	12530	3580	1400
	5041	225	13632	2880	1065
	5184	196	14400	2800	1008
	5625	196	15900	2968	1050
	6084	121	16770	2365	858
Sum	38767	2823	101895	25364	9859

Regression

Multiple Linear Regression: Manual Calculation

Step 2: Calculate Regression Sums.

Next, make the following regression sum calculations:

- $\Sigma x_1^2 = \Sigma X_1^2 - (\Sigma X_1)^2 / n = 38{,}767 - (555)^2 / 8 = \mathbf{263.875}$
- $\Sigma x_2^2 = \Sigma X_2^2 - (\Sigma X_2)^2 / n = 2{,}823 - (145)^2 / 8 = \mathbf{194.875}$
- $\Sigma x_1 y = \Sigma X_1 y - (\Sigma X_1 \Sigma y) / n = 101{,}895 - (555*1{,}452) / 8 = \mathbf{1{,}162.5}$
- $\Sigma x_2 y = \Sigma X_2 y - (\Sigma X_2 \Sigma y) / n = 25{,}364 - (145*1{,}452) / 8 = \mathbf{-953.5}$
- $\Sigma x_1 x_2 = \Sigma X_1 X_2 - (\Sigma X_1 \Sigma X_2) / n = 9{,}859 - (555*145) / 8 = \mathbf{-200.375}$

Regression

Multiple Linear Regression: Manual Calculation

Step 2: Calculate Regression Sums.

	y	X_1	X_2
	140	60	22
	155	62	25
	159	67	24
	179	70	20
	192	71	15
	200	72	14
	212	75	14
	215	78	11
Mean	181.5	69.375	18.125
Sum	1452	555	145

	X_1^2	X_2^2	X_1y	X_2y	X_1X_2
	3600	484	8400	3080	1320
	3844	625	9610	3875	1550
	4489	576	10653	3816	1608
	4900	400	12530	3580	1400
	5041	225	13632	2880	1065
	5184	196	14400	2800	1008
	5625	196	15900	2968	1050
	6084	121	16770	2365	858
Sum	38767	2823	101895	25364	9859
Reg Sums	263.875	194.875	1162.5	-953.5	-200.375

Regression

Multiple Linear Regression: Manual Calculation

Step 3: Calculate b_0, b_1, and b_2.

The formula to calculate b_1 is: $[(\Sigma x_2^2)(\Sigma x_1 y) - (\Sigma x_1 x_2)(\Sigma x_2 y)] \ / \ [(\Sigma x_1^2)(\Sigma x_2^2) - (\Sigma x_1 x_2)^2]$

Thus, $\mathbf{b_1} = [(194.875)(1162.5) - (-200.375)(-953.5)] \ / \ [(263.875)(194.875) - (-200.375)^2] = \mathbf{3.148}$

Regression

Multiple Linear Regression: Manual Calculation

Step 3: Calculate b_0, b_1, and b_2.

The formula to calculate b_2 is: $[(\Sigma x_1^2)(\Sigma x_2 y) - (\Sigma x_1 x_2)(\Sigma x_1 y)] \;/\; [(\Sigma x_1^2)(\Sigma x_2^2) - (\Sigma x_1 x_2)^2]$

Thus, $\mathbf{b_2} = [(263.875)(-953.5) - (-200.375)(1152.5)] \;/\; [(263.875)(194.875) - (-200.375)^2] = \mathbf{-1.656}$

The formula to calculate b_0 is: $\overline{y} - b_1\overline{X}_1 - b_2\overline{X}_2$

Thus, $\mathbf{b_0} = 181.5 - 3.148(69.375) - (-1.656)(18.125) = \mathbf{-6.867}$

Regression

Multiple Linear Regression: Manual Calculation

Step 5: Place b_0, b_1, and b_2 in the estimated linear regression equation.

The estimated linear regression equation is: $\hat{y} = b_0 + b_1 * x_1 + b_2 * x_2$

In our example, it is $\mathbf{\hat{y} = -6.867 + 3.148x_1 - 1.656x_2}$

Regression

Multiple Linear Regression: Manual Calculation

Bài tập nhóm:

- Tính b0, b1, và b2 của MLR cho bảng dữ liệu:
  - Y = [1, 0, 1, 0]
  - X1 = [170, 155, 1650, 165]
  - X2 = [75, 45, 56, 49]

Regression

Polynomial Regression

- is a type of regression analysis where the relationship between the independent variable and the dependent variable is modeled as an n-th degree polynomial.
- Unlike simple linear regression, which fits a straight line to the data, polynomial regression fits **a curved line** to capture more complex relationships.

Regression

Polynomial Regression

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \cdots + \beta_n x^n + \epsilon$$

where:

- β_0 is the intercept,
- $\beta_1, \beta_2, \ldots, \beta_n$ are the coefficients of the polynomial terms,
- x is the independent variable,
- y is the dependent variable,
- ϵ is the error term.

Regression

Polynomial Regression

Code:

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
from sklearn.pipeline import make_pipeline

# Generate example data
np.random.seed(0)
X = np.sort(5 * np.random.rand(100, 1), axis=0)
y = np.sin(X).ravel() + np.random.normal(0, 0.1, X.shape[0])

# Fit polynomial regression model
degree = 3
poly = PolynomialFeatures(degree)
X_poly = poly.fit_transform(X)
model = LinearRegression()
model.fit(X_poly, y)

# Predict using the model
X_fit = np.linspace(0, 5, 100)[:, np.newaxis]
X_fit_poly = poly.transform(X_fit)
y_fit = model.predict(X_fit_poly)
```

Regression

Polynomial Regression

Code:

```
# Fit polynomial regression model
degree = 3
poly = PolynomialFeatures(degree)
X_poly = poly.fit_transform(X)
model = LinearRegression()
model.fit(X_poly, y)

# Predict using the model
X_fit = np.linspace(0, 5, 100)[:, np.newaxis]
X_fit_poly = poly.transform(X_fit)
y_fit = model.predict(X_fit_poly)

# Plot the results
plt.scatter(X, y, color='blue', label='Data')
plt.plot(X_fit, y_fit, color='red', label='Polynomial Regression')
plt.xlabel('X')
plt.ylabel('y')
plt.title('Polynomial Regression (Degree 3)')
plt.legend()
plt.show()
```

Regression

Polynomial Regression

Code: