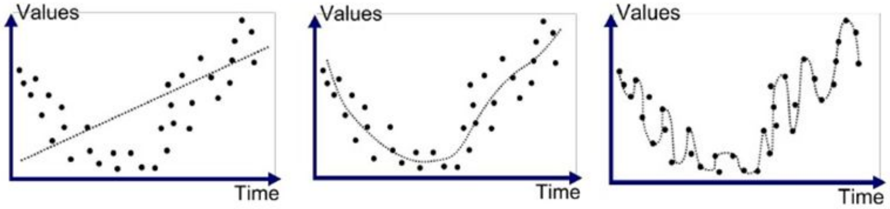
Underfitting & Overfitting

[4,8,2] [5,5
4.8

- The more model complexity you add the accuracy improves but this can lead to overfitting□ that means the model memorizes the data too much and not applying generalization.
- So, we should know about the trade-off between Bias and Variance.
- A model with a high bias error underfits data and makes very simplistic assumptions on it.
- A model with a high variance error overfits the data and learns too much from it.
- A good model is where both Bias and Variance errors are balanced



high bias
low variance

medium bias
medium variance

low bias
high variance

Generalization – Error Analysis - Guidelines

Train Error	Test Error	What to do?
Low Over-fitting	High	• Need a simpler model • Need more data (more data samples) حافظ مش فاهم

Generalization – Error Analysis - Guidelines

Train Error	Test Error	What to do?
Low Over-	High fitting	• Need a simpler model • Need more data (more data samples) حافظ مش فاهم
High Under -	High fitting	• Need more data • Difficult to learn f(x, z) with only x. Get also z • Get more data samples • Additional features (e.g. $\frac{1}{x^2}$) لا حافظ ولا فاهم • Need a more complex model

Generalization – Error Analysis - Guidelines

Train Error	Test Error	What to do?
Low Over-	High fitting	• Need a simpler model • Need more data (more data samples) حافظ مش فاهم
High Under -	High fitting	• Need more data • Difficult to learn f(x, z) with only x. Get also z • Get more data samples • Additional features (e.g. $\frac{1}{x^2}$) • Need a more complex model لا حافظ ولا فاهم
High	Low	Unusual: it could mean that the test data is too similar to the train data. Get more test data.

Generalization – Error Analysis - Guidelines

Train Error	Test Error	What to do?
Low Over-	High fitting	• • low bias, high variance حافظ مش فاهم
High Under -	High fitting	• Need more data Difficult to learn f(x, z) with only x. Get also z high bias, potentially high variance too لا حافظ مش فاهم x^2 • Need a more complex model
High	Low	Unusual: it could mean that the test data is too similar to the train data. Get more test data.
Low	Low	You' Trade-off between Bias and Variance

Underfitting & Overfitting

Techniques to reduce underfitting

- Increase model complexity.
- Increase number of features.
- Performing feature engineering.
- Remove noise from the data.

Underfitting & Overfitting

Techniques to reduce overfitting

- Increase training data.
- Remove correlated features.
- Use simpler model.
- **Regularization (add penalty bias).**

Train acc = 90% > error 10%
Test acc = 60% > error 40%

لما يشوف داتا جديدة (يعني المودل ما شافها قبل كده) بعرفش يكتشف النمط

حافظ مش فاهم overfitting

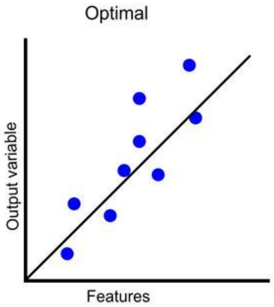
يعني حفظ بزيادة

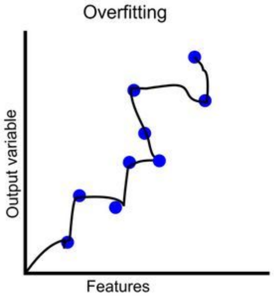
فانا بروح أعاقبه انو فهم بزياده ???????

ف بعمل Regularization يعني بضيف على المعادلة penalty

Regularization

- **Fine-tuning Machine Learning Models for Optimal Performance**



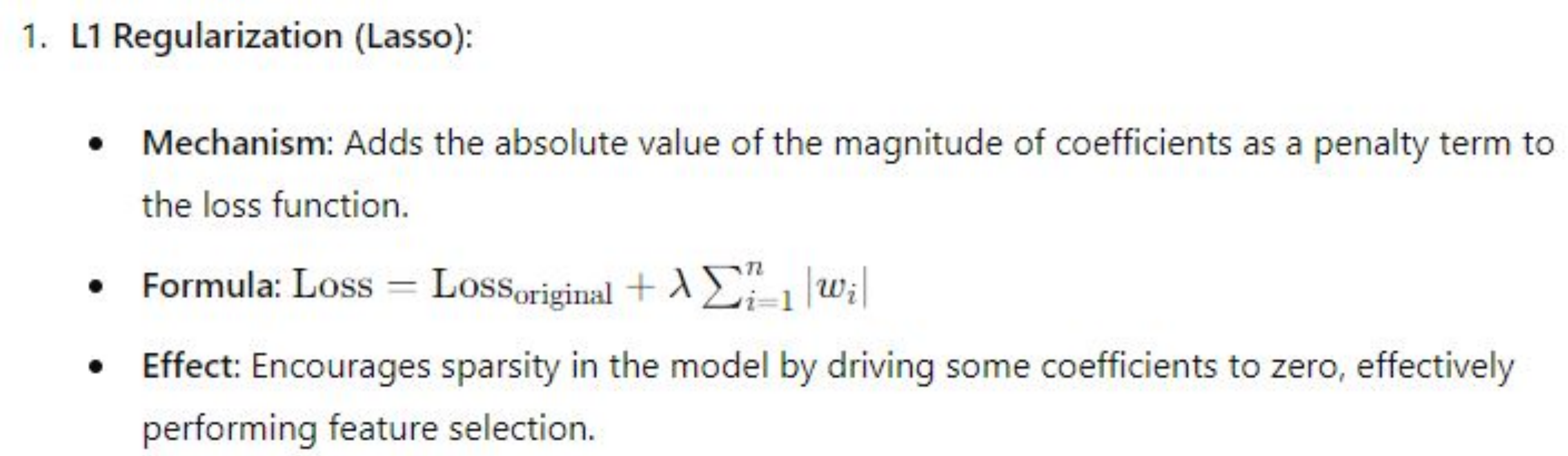


- **The Challenge:**
  - Overfitting: When a model learns the noise and random fluctuations in the training data, rather than the underlying pattern.
- **The Problem:**
  - Excellent performance on training data, but poor performance on new data.
  - Complex model that is difficult to interpret.
  - Inability to generalize well to real-world data.

What is Regularization?

- Techniques used to prevent or reduce overfitting.
- Works by adding a "penalty" to the model if it becomes too complex.
- Helps to simplify the model and improve its ability to generalize.

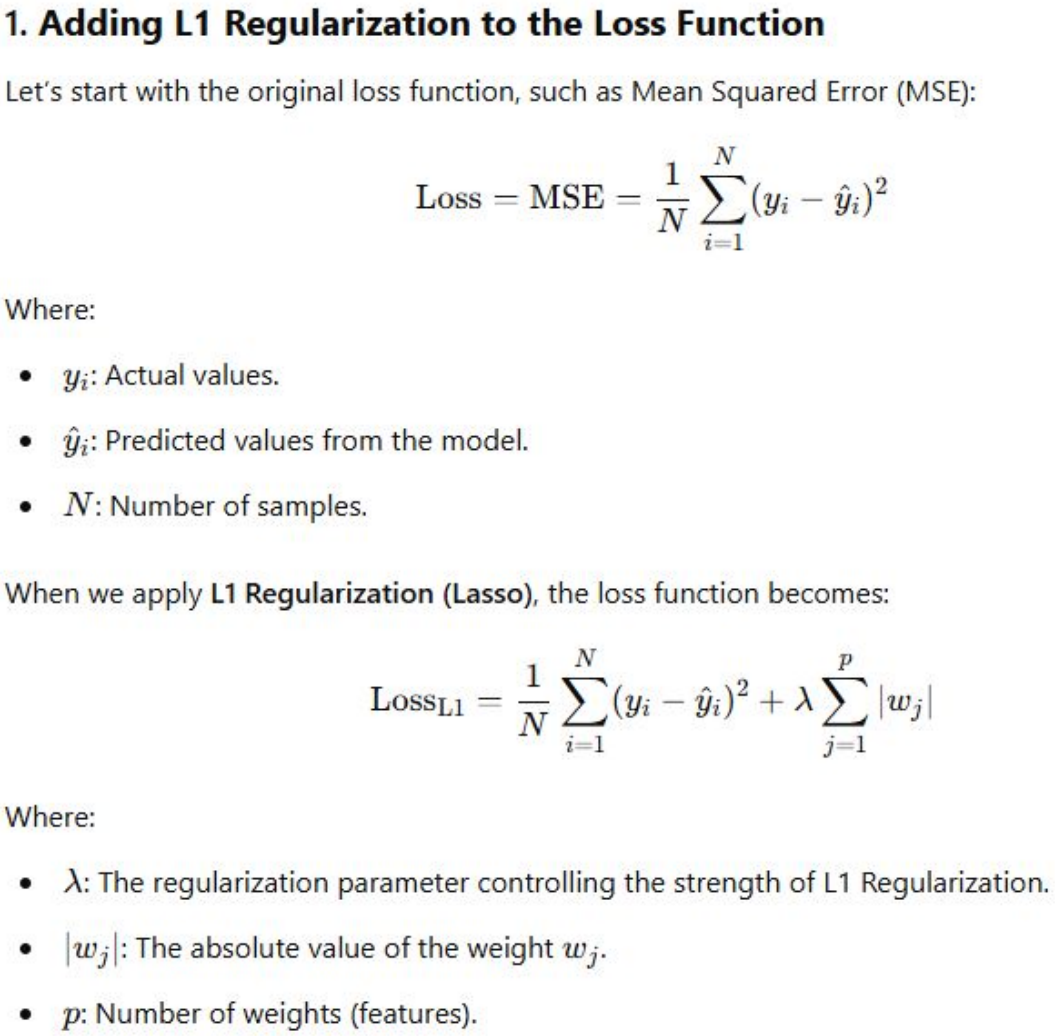
Types of Regularization:



1. **L1 Regularization (Lasso):**

- **Mechanism:** Adds the absolute value of the magnitude of coefficients as a penalty term to the loss function.
- **Formula:** $\text{Loss} = \text{Loss}_{\text{original}} + \lambda \sum_{i=1}^{n} |w_i|$
- **Effect:** Encourages sparsity in the model by driving some coefficients to zero, effectively performing feature selection.

Price = numRoom*a1 + houseSize * a2 + unv*a3
Price = numRoom*a1 + houseSize * a2 + unv*0.5
Price = numRoom*a1 + houseSize * a2 + unv*0



1. Adding L1 Regularization to the Loss Function

Let's start with the original loss function, such as Mean Squared Error (MSE):

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

Where:

- y_i: Actual values.
- $\hat{y}_i$: Predicted values from the model.
- N: Number of samples.

When we apply **L1 Regularization (Lasso)**, the loss function becomes:

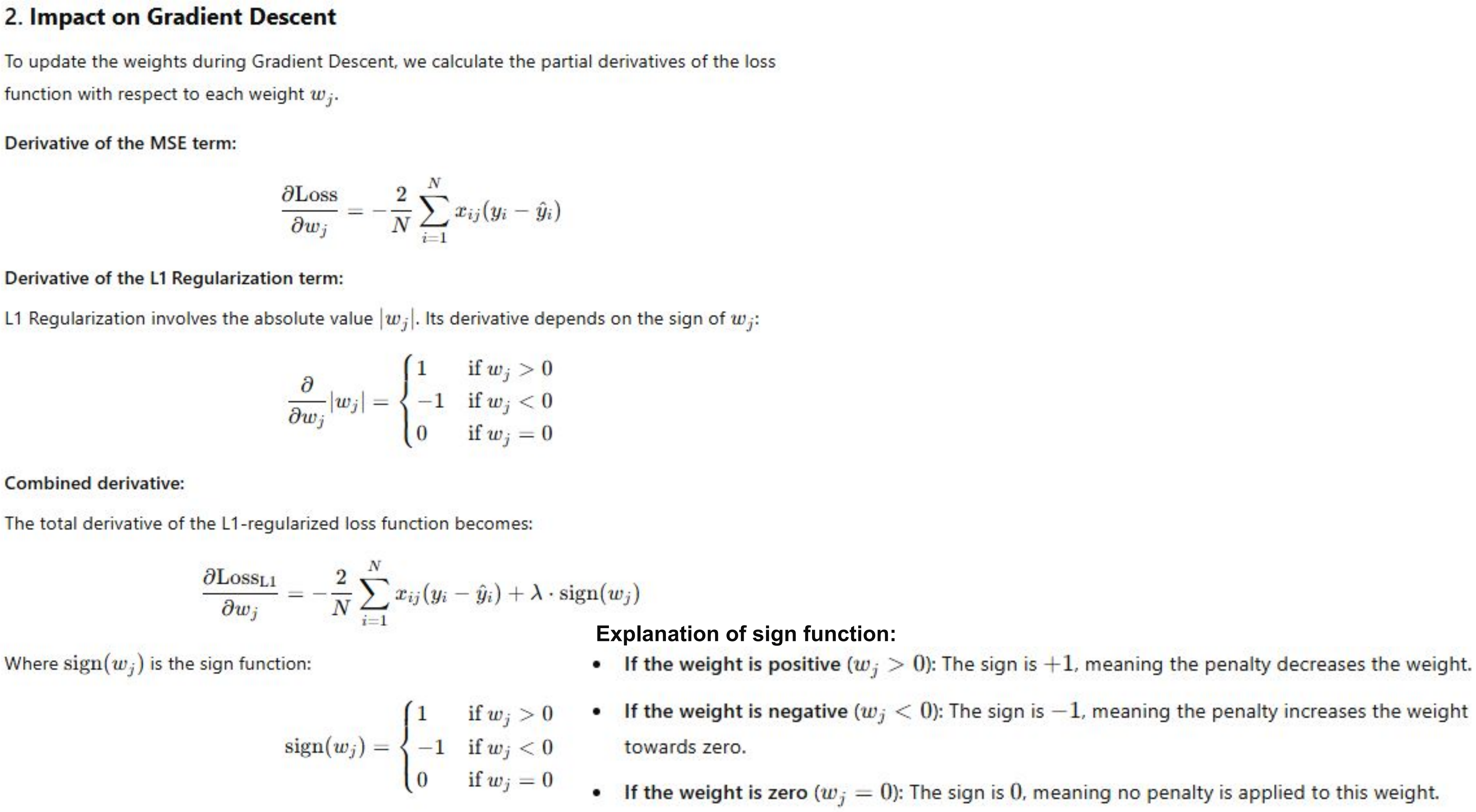
$$\text{Loss}_{\text{L1}} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 + \lambda\sum_{j=1}^{p}|w_j|$$

Where:

- λ: The regularization parameter controlling the strength of L1 Regularization.
- $|w_j|$: The absolute value of the weight w_j.
- p: Number of weights (features).

L1

$$\text{Loss} = \text{Loss}_{\text{original}} + \lambda\sum_{i=1}^{n}|w_i|$$



2. Impact on Gradient Descent

To update the weights during Gradient Descent, we calculate the partial derivatives of the loss function with respect to each weight w_j.

Derivative of the MSE term:

$$\frac{\partial \text{Loss}}{\partial w_j} = -\frac{2}{N}\sum_{i=1}^{N} x_{ij}(y_i - \hat{y}_i)$$

Derivative of the L1 Regularization term:

L1 Regularization involves the absolute value $|w_j|$. Its derivative depends on the sign of w_j:

$$\frac{\partial}{\partial w_j}|w_j| = \begin{cases} 1 & \text{if } w_j > 0 \\ -1 & \text{if } w_j < 0 \\ 0 & \text{if } w_j = 0 \end{cases}$$

Combined derivative:

The total derivative of the L1-regularized loss function becomes:

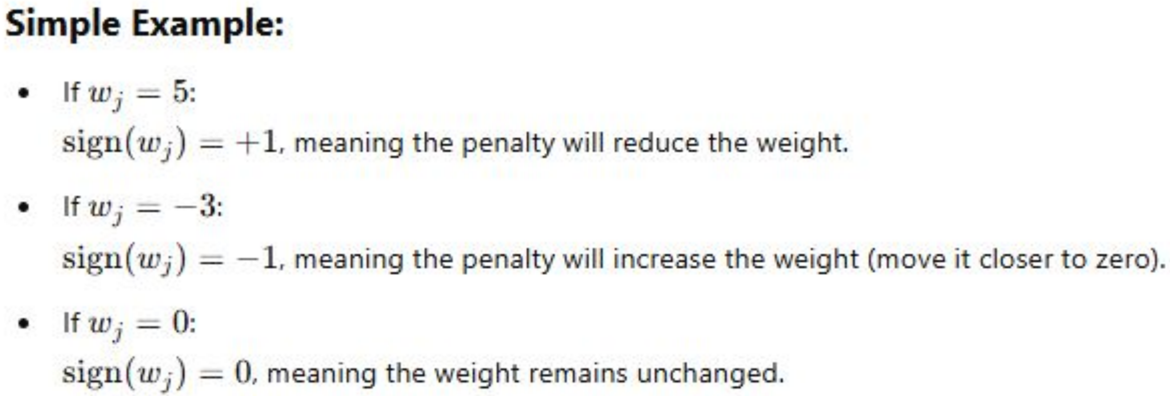
$$\frac{\partial \text{Loss}_{\text{L1}}}{\partial w_j} = -\frac{2}{N}\sum_{i=1}^{N} x_{ij}(y_i - \hat{y}_i) + \lambda \cdot \text{sign}(w_j)$$

Where $\text{sign}(w_j)$ is the sign function:

$$\text{sign}(w_j) = \begin{cases} 1 & \text{if } w_j > 0 \\ -1 & \text{if } w_j < 0 \\ 0 & \text{if } w_j = 0 \end{cases}$$

Explanation of sign function:

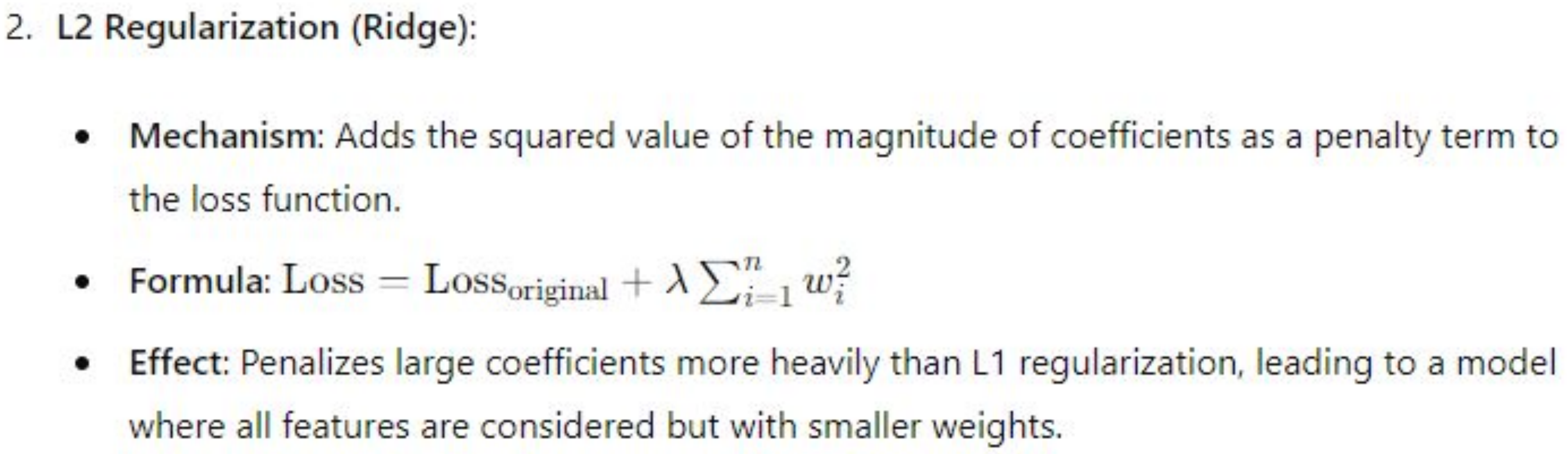
- **If the weight is positive** ($w_j > 0$): The sign is $+1$, meaning the penalty decreases the weight.
- **If the weight is negative** ($w_j < 0$): The sign is -1, meaning the penalty increases the weight towards zero.
- **If the weight is zero** ($w_j = 0$): The sign is 0, meaning no penalty is applied to this weight.



Simple Example:

- If $w_j = 5$:
  $\mathrm{sign}(w_j) = +1$, meaning the penalty will reduce the weight.
- If $w_j = -3$:
  $\mathrm{sign}(w_j) = -1$, meaning the penalty will increase the weight (move it closer to zero).
- If $w_j = 0$:
  $\mathrm{sign}(w_j) = 0$, meaning the weight remains unchanged.

Types of Regularization:



2. **L2 Regularization (Ridge):**

- **Mechanism:** Adds the squared value of the magnitude of coefficients as a penalty term to the loss function.
- **Formula:** $\text{Loss} = \text{Loss}_{\text{original}} + \lambda \sum_{i=1}^{n} w_i^2$
- **Effect:** Penalizes large coefficients more heavily than L1 regularization, leading to a model where all features are considered but with smaller weights.

Price = numRoom*a1 + houseSize * a2 + unv*a3

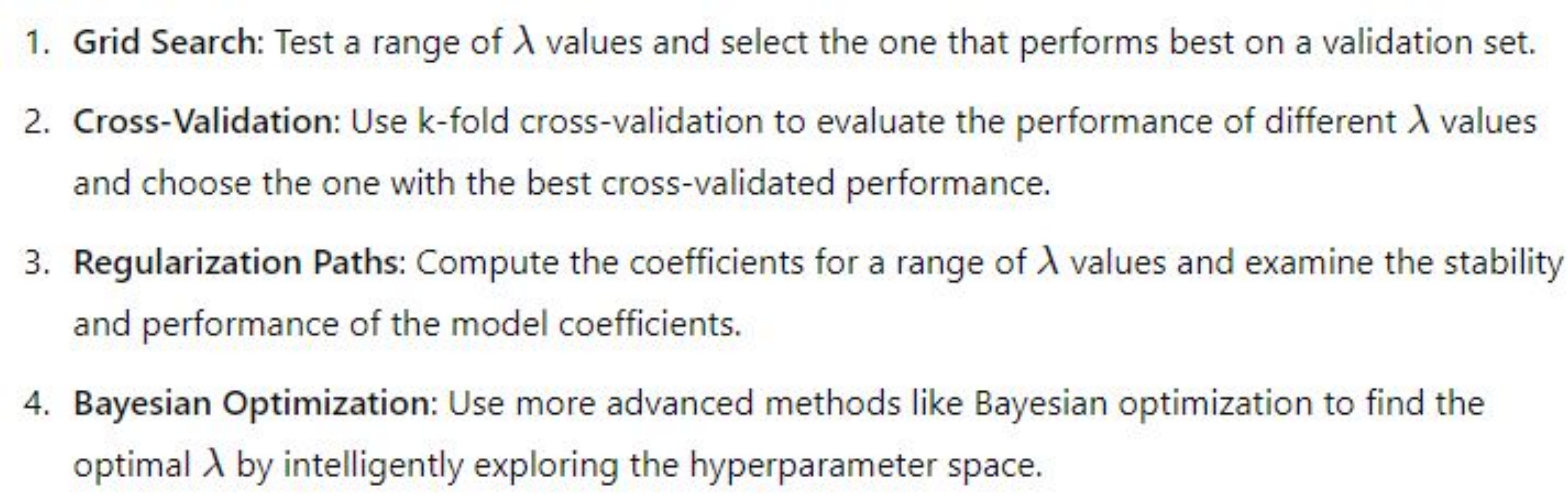
Price = numRoom*a1 + houseSize * a2 + unv*0.5

Price = numRoom*a1 + houseSize * a2 + unv*0.05

The regularization parameter λ

- λ **parameter:** controls the strength of the penalty applied to the coefficients.
- **A larger λ** increases the penalty, leading to more regularization (smaller coefficients),
  - leading to simpler models that **may underfit the data**
- **A smaller λ** reduces the penalty, resulting in less regularization.
  - Allowing the model to fit the training data more closely, which may result in **overfitting**.
- Selecting an appropriate λ is critical for balancing model complexity and generalization.

Methods to Select λ



1. **Grid Search:** Test a range of λ values and select the one that performs best on a validation set.
2. **Cross-Validation:** Use k-fold cross-validation to evaluate the performance of different λ values and choose the one with the best cross-validated performance.
3. **Regularization Paths:** Compute the coefficients for a range of λ values and examine the stability and performance of the model coefficients.
4. **Bayesian Optimization:** Use more advanced methods like Bayesian optimization to find the optimal λ by intelligently exploring the hyperparameter space.

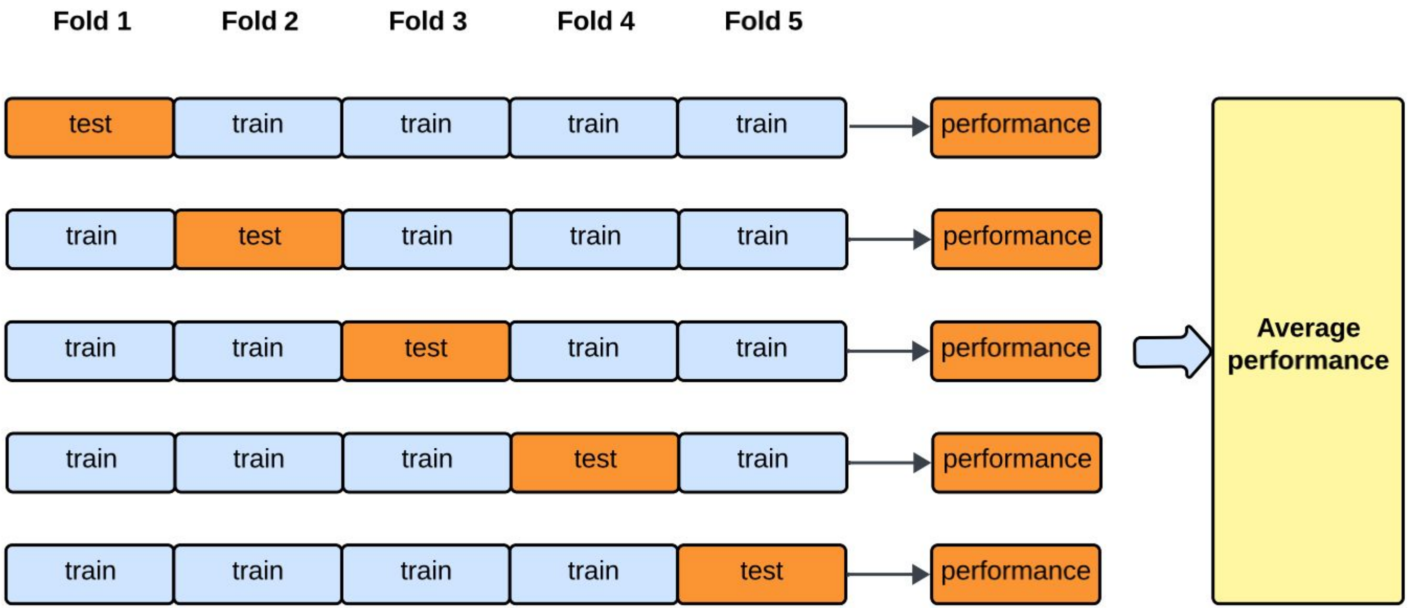
K-fold Cross Validation

cv=5 , lam = [0.1,0.01,0.02]

بعد 5 مرات اعطاني 90% avg على 0.1

بعد 5 مرات اعطاني 80% avg على 0.01

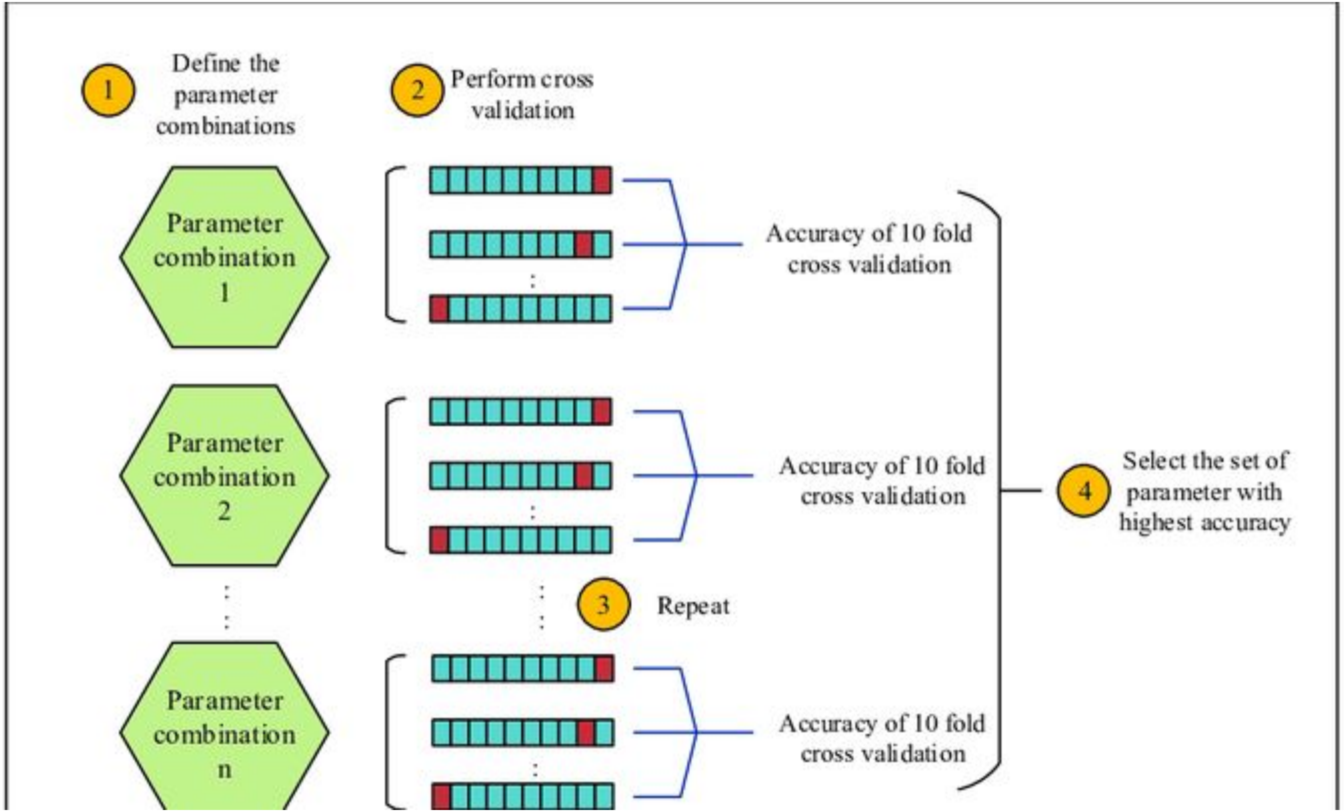
بعد 5 مرات اعطاني 85% avg على 0.02



Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	
test	train	train	train	train	performance
train	test	train	train	train	performance
train	train	test	train	train	performance
train	train	train	test	train	performance
train	train	train	train	test	performance

Average performance

Grid Search CV


1 Define the parameter combinations
2 Perform cross validation
Parameter combination 1
Accuracy of 10 fold cross validation
Parameter combination 2
Accuracy of 10 fold cross validation
4 Select the set of parameter with highest accuracy
3 Repeat
Parameter combination n
Accuracy of 10 fold cross validation

Grid Search

1. Definition

- A systematic method to tune hyperparameters.

- We define a **set of candidate values** (a “grid”) and test them one by one.

2. Process

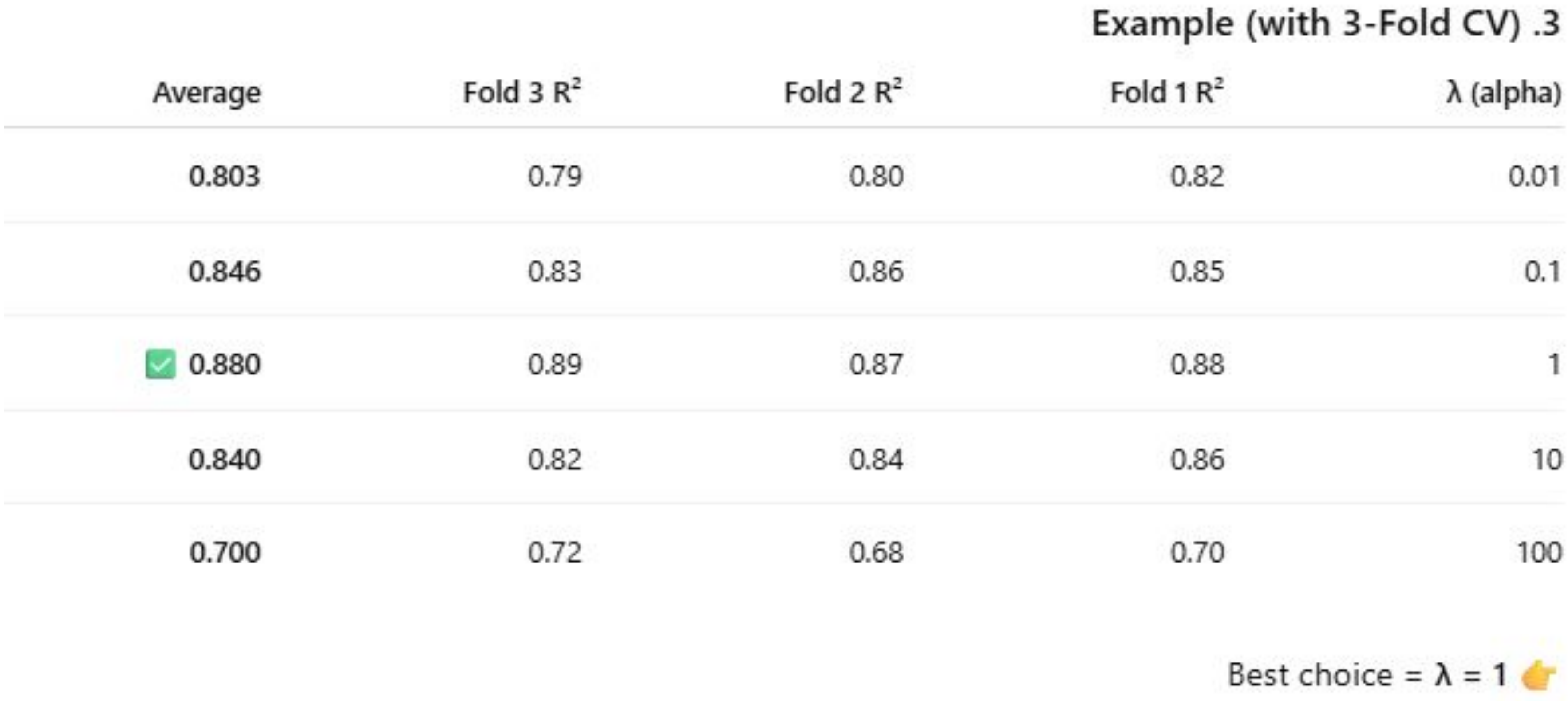
1. Choose a hyperparameter to optimize (e.g., **λ / alpha** in Ridge).

2. Define a grid of possible values, e.g.:
   {0.01, 0.1, 1, 10, 100}

3. For each value, train the model using **k-fold cross-validation**.

4. Compute the average performance (e.g., MSE, R^2, Accuracy).

5. Select the value with the **best score**.



Example (with 3-Fold CV) .3

Average	Fold 3 R^2	Fold 2 R^2	Fold 1 R^2	λ (alpha)
0.803	0.79	0.80	0.82	0.01
0.846	0.83	0.86	0.85	0.1
✅ **0.880**	0.89	0.87	0.88	1
0.840	0.82	0.84	0.86	10
0.700	0.72	0.68	0.70	100

Best choice = **λ = 1** 👉