AMIT

AMIT

Machine Learning

Session 2

Agenda:

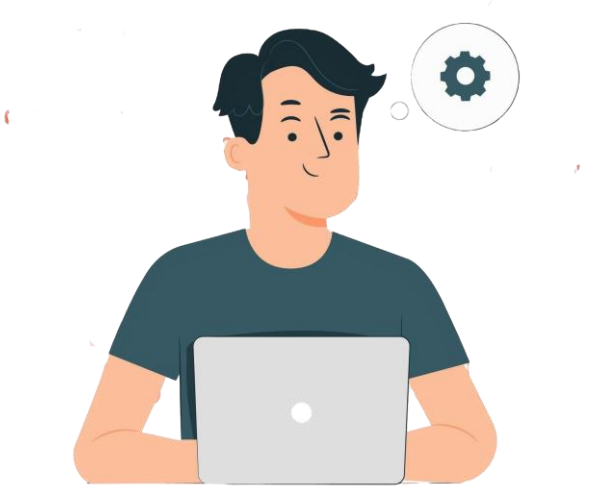
1	Encoding
2	Scaling
3	Overfitting and Underfitting

Encoding

Encoding

What is Encoding?

- **Encoding** refers to the process of converting categorical data into a numerical format that can be used by machine learning algorithms. Most algorithms cannot work directly with labels such as "**Male**" or "**Female**," so these labels must be converted into numbers.



Common Types of Encoding

1. Label Encoding

- Each category is **assigned** a **unique** integer value
- Example: “Tall" = 0, “Medium" = 1, “Short" = 2.
- Used when the categories have a meaningful order.

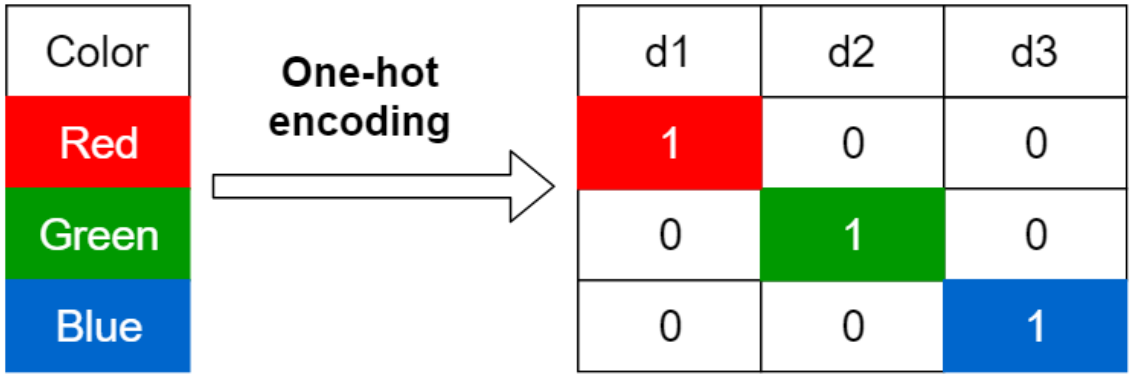
Height
Tall
Medium
Short

Height
0
1
2

Common Types of Encoding

2. One-Hot Encoding

- Creates a new binary column for each category
- Example: "Red" = [1, 0, 0], "Green" = [0, 1, 0], "Blue" = [0, 0, 1]
- Used when there is no ordinal relationship between categories.



Common Types of Encoding

When to Use Encoding ?

- **Label Encoding**: For **ordinal features**, such as education level (High School, Bachelor's, Master's, etc.), where the order matters.

- **One-Hot Encoding**: For **nominal features**, such as colors or product categories, where there is no inherent order.

Scaling

Scaling

What is Scaling?

- **Scaling** is the process of **transforming features** so that they are on the **same scale**. This is crucial because many machine learning algorithms (such as gradient descent, K-nearest neighbors, etc.) are sensitive to the scale of the features.

Scaling

Why Scaling is Important?

- Without scaling, features with larger values may dominate the model, leading to biased predictions.
- **Scaling** ensures that all features contribute equally to the model during training.

Common Methods of Scaling

Min-Max Scaling (Normalization)

- Transforms features to a fixed range, usually between 0 and 1

Formula $$X_{\text{scaled}} = \frac{X - X_{\text{min}}}{X_{\text{max}} - X_{\text{min}}}$$

Common Methods of Scaling

Standardization (Z-Score Scaling)

- Transforms features so that they have a **mean of 0** and **standard deviation of 1**.

Formula $$X_{\text{scaled}} = \frac{X - \mu}{\sigma}$$

  - where μ is the mean, and σ is the standard deviation.

When to Use Scaling

- **Normalization**: When you need your features to be in a bounded range (e.g., in neural networks).

- **Standardization**: When the algorithm expects normally distributed data (e.g., linear regression, logistic regression, etc.).

When to Use Scaling

Example Scenario

- We have a dataset with two features: **House Size (in square feet)** and **House Age (in years)**. The scales of these two features are very different, so we'll apply both **Min-Max Scaling** and **Standardization** to see how scaling changes the values.

House Size (sq ft)	House Age (years)
1500	10
1800	15
1200	20
2200	5
2500	12

When to Use Scaling

Original Dataset

```
Original Dataset:
   House Size  House Age
0        1500         10
1        1800         15
2        1200         20
3        2200          5
4        2500         12
```

When to Use Scaling

Min-Max Scaled Data (Values between 0 and 1)

```
Min-Max Scaled Data (0-1 range):
   House Size  House Age
0    0.230769   0.333333
1    0.461538   0.666667
2    0.000000   1.000000
3    0.769231   0.000000
4    1.000000   0.466667
```

- The **Min-Max Scaled** values range between 0 and 1. The size of the house and its age are now on a common scale, which is important for algorithms like **K-Nearest Neighbors** or **Neural Networks**.

When to Use Scaling

Standardized Data (Mean = 0, Standard Deviation = 1)

```
Standardized Data (mean=0, std=1):
   House Size  House Age
0   -0.727533  -0.479616
1   -0.085592   0.519584
2   -1.369474   1.518785
3    0.770329  -1.478817
4    1.412270  -0.079936
```

- The **Standardized Data** has a mean of 0 and standard deviation of 1. The values are no longer between 0 and 1, but they are still scaled relative to the dataset. This scaling is particularly useful for algorithms like **Logistic Regression** or **Support Vector Machines**.

When to Use Scaling

Why Scaling Matters

- **Before Scaling**: Features like **House Size** and **House Age** have very different ranges. If used in algorithms like **KNN** or **Gradient Descent**, the model might give more importance to features with larger ranges (like **House Size**), even though they might not be more important.

After Scaling

- **Min-Max Scaling**: All features are now within the same range (0-1), which ensures that no feature dominates over others based on magnitude.

- **Standardization**: Features are transformed to have a mean of 0 and a standard deviation of 1, which is important for algorithms that assume the data is normally distributed.

Overfitting and Underfitting

Overfitting and Underfitting

Bias

- Bias refers to the difference between the model's predicted values and the true values we are trying to predict. High bias means that the model has **oversimplified** the data and has missed key patterns.

  High Bias Model Characteristics

  - **Fails to capture patterns** in the data.
  - **Underfitting**: The model is too simple to accurately represent the data.
  - Produces **high error rates** on both training and test data.
  - The model might be **too generalized** and unable to handle the complexity of the data.

Overfitting and Underfitting

Variance

- Variance measures how much the model's predictions vary when applied to different training sets. A model with high variance will **overfit**, meaning it will perform well on training data but poorly on unseen data.

High Bias Model Characteristics

- **Overfitting**: The model captures noise or irrelevant patterns in the data.
- **Complex models** that adapt too closely to the training set.
- High sensitivity to **small changes** in the data.

Overfitting and Underfitting

Balancing Bias and Variance

- **Bias-Variance Tradeoff**: When building models, we aim to find the balance between bias and variance. Increasing the complexity of a model decreases bias but increases variance, and vice versa.

Ways to Reduce High Bias

- **Add more features**: This helps the model learn better from the data.
- **Decrease regularization**: Too much regularization forces the model to be simple, leading to underfitting.
- Use **more complex models**, like polynomial regression, to capture more patterns in the data.

Overfitting and Underfitting

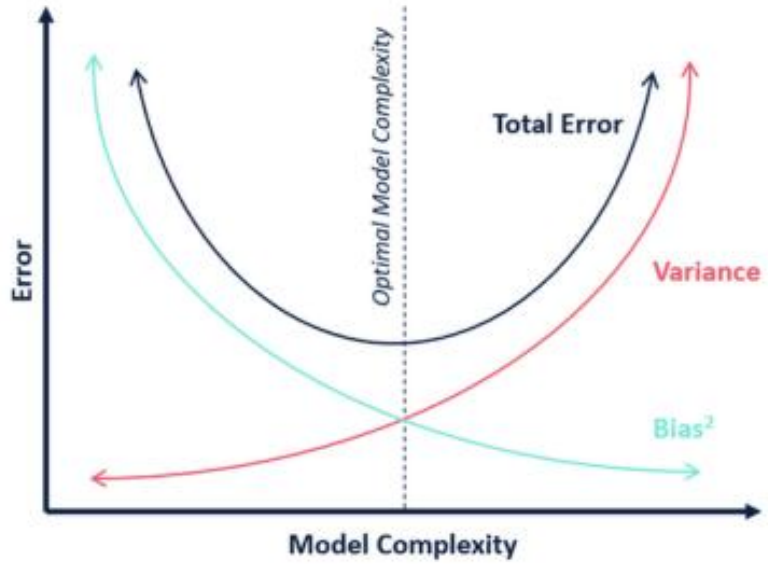
Balancing Bias and Variance

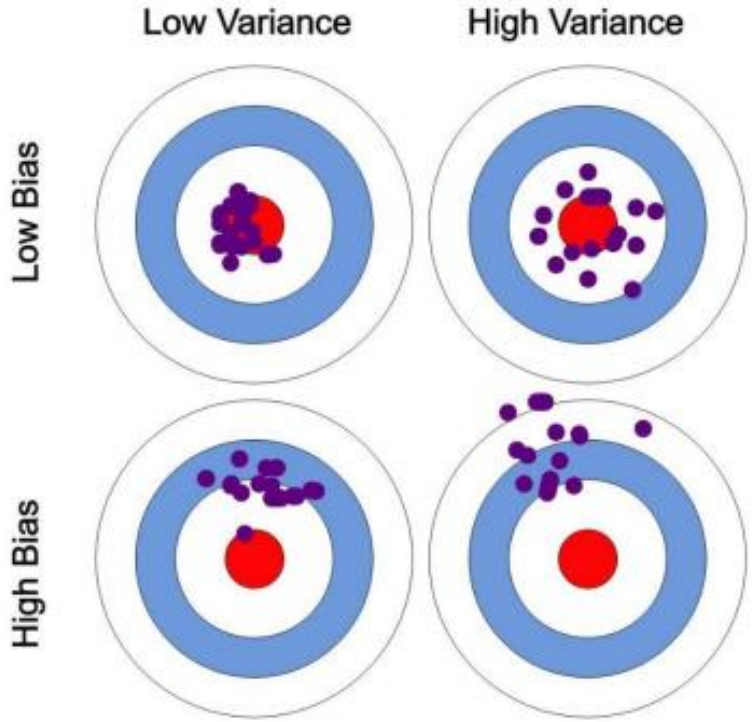
Ways to Reduce High Variance

- **Reduce the number of features** or **parameters**: Simplifying the model reduces overfitting.
- Avoid overly complex models.
- **Increase the training data size**: More data helps the model generalize better.
- **Increase regularization**: This helps prevent overfitting by keeping the model simpler.

Overfitting and Underfitting

Bias & Variance

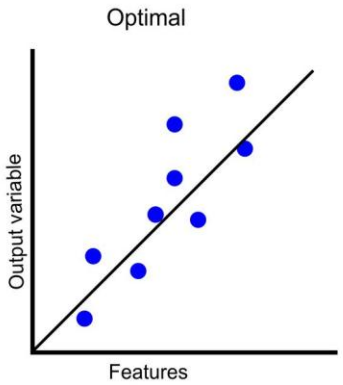


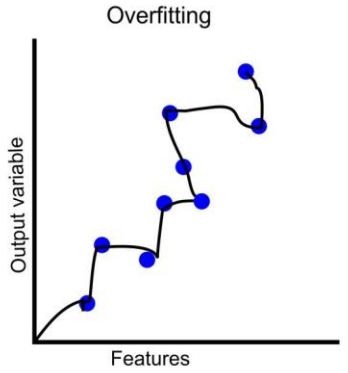


Overfitting and Underfitting

What is Overfitting?

- **Overfitting** occurs when a machine learning model performs extremely well on the training data but fails to generalize to new, unseen data. The model has essentially "memorized" the training data, capturing noise rather than learning the underlying patterns.





Overfitting and Underfitting

What is Underfitting?

- The model is too simple and does not capture the patterns in the data well. It performs poorly on both the training and test datasets, indicating that the model is not complex enough to understand the data.

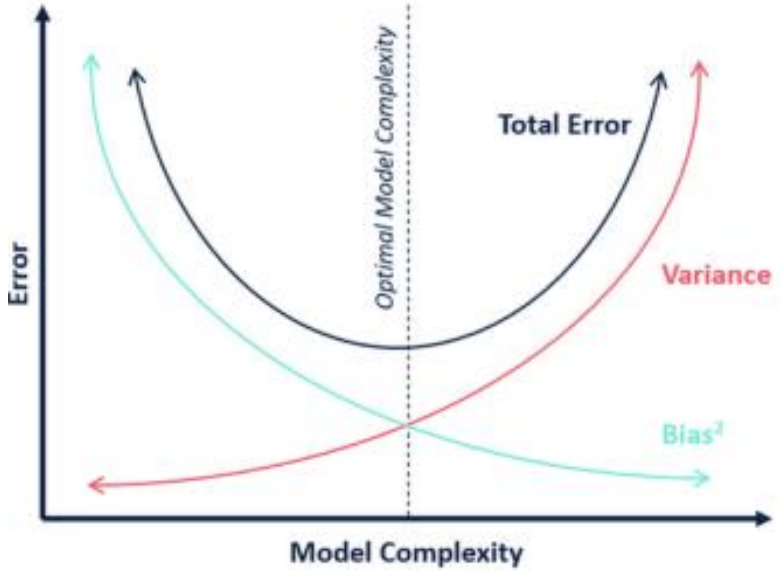
Overfitting and Underfitting

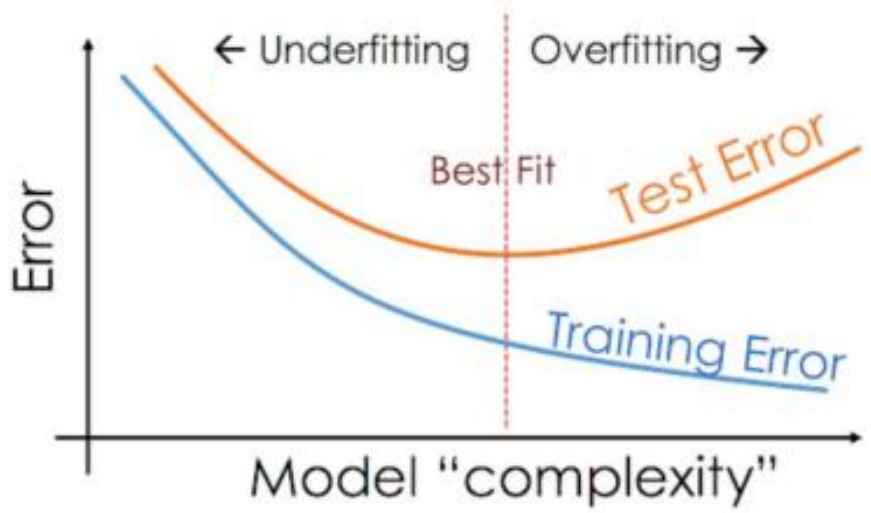
Understanding Overfitting and Underfitting

1. Underfitting
    - Occurs when the model is too simple.
    - It has **high bias** and cannot capture the patterns in the data.
2. Overfitting
    - Happens when the model is too complex.
    - It has **high variance** and fits the noise in the training data, making it perform poorly on unseen data.

Overfitting and Underfitting

Understanding Overfitting and Underfitting





Overfitting and Underfitting

Techniques to Reduce Underfitting

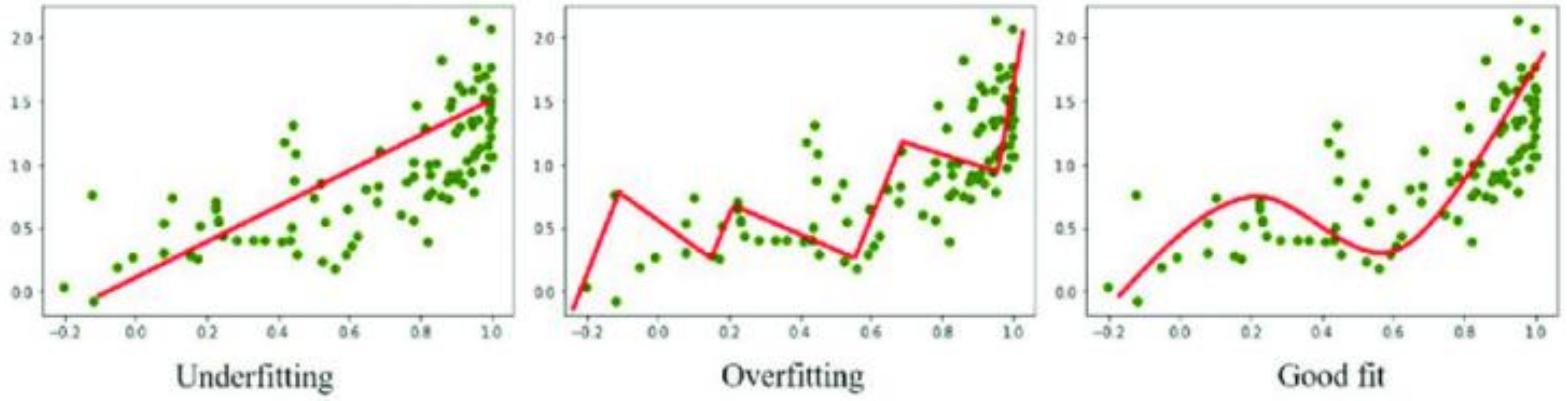
- **increase model complexity**: Add more features
- **Increase the number of features** or use **feature engineering** to capture more useful information from the data.
- **Remove noise**: Clean up the data to avoid irrelevant patterns.

Overfitting and Underfitting

Techniques to Reduce Overfitting

- **Increase training data**: More data helps the model generalize better
- **Reduce model complexity**: Simpler models have less variance and generalize better.
- **Regularization : L1 (Lasso)** and **L2 (Ridge)** regularization add penalties to the model to reduce complexity.

Overfitting and Underfitting



- **Underfitting**: The model fails to capture the trend and has high error.
- **Overfitting**: The model captures every detail (including noise) and fits the training data too well but fails to generalize.
- **Good Fit**: The model finds a balance between bias and variance, fitting the data well without being too complex or too simple.

Overfitting and Underfitting

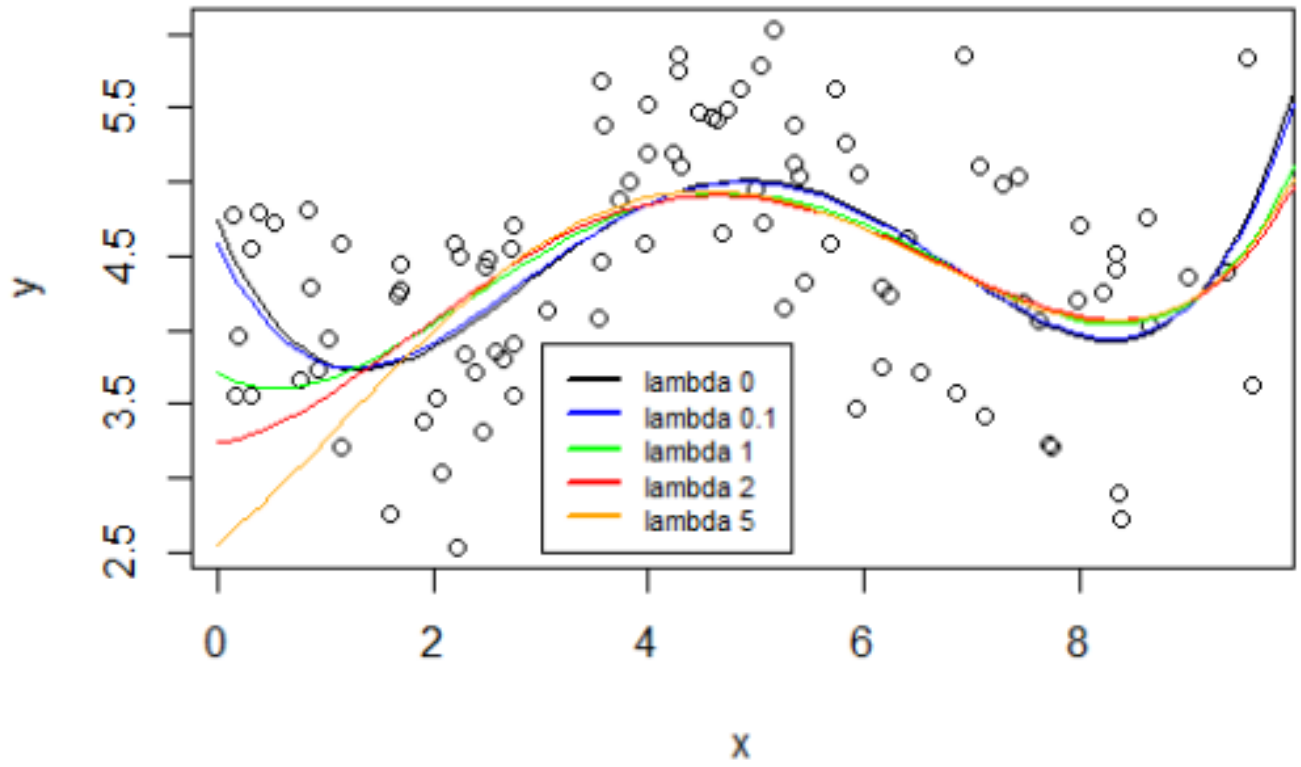
Regularization

- Regularization is a technique designed to **reduce overfitting** by adding a penalty term to the cost function of a model. Overfitting happens when a model is too complex and fits the noise in the training data, failing to generalize well on new, unseen data.

Regularized Regression

- Regularized regression constrains the coefficient estimates to prevent them from becoming too large or too flexible. This ensures that the model doesn’t overly adapt to the training data, reducing the likelihood of overfitting.

Overfitting and Underfitting



Overfitting and Underfitting

Regularization Parameter

Regularization Parameter (Lambda)

- Lambda (λ) is the **regularization parameter** that controls the strength of the regularization applied. A **higher lambda** increases the penalty on large coefficients, effectively reducing overfitting but potentially causing underfitting if λ is too large.

- If λ is **small**, the model will behave more like a traditional regression model (e.g., without regularization), potentially overfitting the data.

Effect of Increasing Lambda

- Increasing lambda results in **less overfitting** but at the cost of higher bias. This is often referred to as the **bias-variance tradeoff**.

$$\text{minimize}(\text{Loss}(\text{Data}|\text{Model}) + \lambda \, \text{complexity}(\text{Model}))$$

Overfitting and Underfitting

Types of Regularization (L1 & L2)

1. L2 Regularization (Ridge)

- In L2 regularization, a **squared penalty** is added to the sum of the squared coefficients. It helps in shrinking all coefficients, leading to a simpler model with smaller weights.
- This method **reduces the impact of irrelevant features** but doesn't force the coefficients of less important features to exactly zero.

Overfitting and Underfitting

Types of Regularization (L1 & L2)

2. L1 Regularization (Lasso)

- L1 regularization adds a **penalty equal to the absolute value** of the magnitude of the coefficients. This has the effect of shrinking some coefficients to **zero**, which can result in a sparse model (i.e., feature selection).

Overfitting and Underfitting

Types of Regularization (Mathematical Representation)

L1 Regularization (Lasso) Equation

- The cost function for L1 regularization adds a penalty term proportional to the absolute value of the coefficients (|W|). This results in some coefficients being zeroed out (feature selection).

$$\text{Cost} = \sum_{i=0}^{N} (y_i - \sum_{j=0}^{M} x_{ij} W_j)^2 + \lambda \sum_{j=0}^{M} |W_j|$$

Overfitting and Underfitting

Types of Regularization (Mathematical Representation)

L2 Regularization (Ridge) Equation

- The cost function for L2 regularization adds a penalty proportional to the **square of the coefficients** (W^2). This shrinks the coefficients without setting any to zero, which is effective when all features are useful, but we want to reduce their influence.

$$\text{Cost} = \underbrace{\sum_{i=0}^{N} (y_i - \sum_{j=0}^{M} x_{ij} W_j)^2}_{\text{Loss function}} + \underbrace{\lambda \sum_{j=0}^{M} W_j^2}_{\text{Regularization Term}}$$

Quiz

What are the common techniques used to prevent overfitting in machine learning models?

Quiz

Expected Answer

- **Cross-Validation:** A technique where the dataset is divided into multiple parts (folds), and the model is trained and validated multiple times using different combinations of these folds. This provides a more reliable estimate of model performance.

- **Regularization (L1 and L2):** Regularization techniques (Lasso and Ridge) add a penalty to the model's coefficients, discouraging it from fitting too closely to the training data.