DATA SPLITTING

Data splitting is the process of dividing a dataset into separate subsets for different stages of the machine learning workflow.

Importance of Data Splitting

It ensures that:

- The model learns patterns from one set (training set)
- Is tuned or validated using another (validation set)
- And is finally tested for generalization performance on unseen data (test set)

Types of Splits

Set	Purpose
Training Set	Used to train the model (learn parameters or patterns)
Validation Set	Used for hyperparameter tuning and early stopping
Testing Set	Used for final evaluation of the model's performance

comparison between Overfitting and Underfitting

Feature	Overfitting	Underfitting
Definition	Model learns too much, including noise and outliers	Model learns too little, missing important patterns
Model Complexity	High (too many parameters/features)	Low (too simple model)
Training Performance	Very good (low error)	Poor (high error)

Testing Performance	Poor generalization (high error)	Poor generalization (high error)
Bias	Low bias	High bias
Variance	High variance	Low variance
Cause	Too many features, small training set, or too many epochs	Model is too simple, insufficient training, high regularization
Examples	Polynomial regression with very high degree	Linear regression for non-linear data
Symptoms	Excellent accuracy on training, poor on test set	Poor accuracy on both training and test sets
Solution	Simplify model, more data, use regularization, pruning	Increase model complexity, reduce bias, improve features

```
Model Fit vs Complexity:

Underfitting ←──── Optimal Fit ────→ Overfitting

 High Bias                        Balanced                       High Variance
```

- Overfitting: Model is too smart for its own good — memorizes instead of generalizing.

- Underfitting: Model is too naive — doesn't learn enough to make good predictions.

"In Python, data splitting can be performed using the `train_test_split()` function from Scikit-learn."

```
train_test_split(
    *arrays,                 # X, y or other arrays
    test_size=0.25,          # Proportion or absolute number for test data
    train_size=None,         # Same as above; defaults to remaining
    random_state=None,       # Seed for reproducibility
    shuffle=True,            # Shuffle before splitting
    stratify=None            # Ensures proportional class distribution
)
```

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
iris = load_iris()
X, y = iris.data, iris.target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Q1. What is the purpose of splitting data into training, validation, and testing sets?
Answer:

- Training set: Train the model and learn patterns.
- Validation set: Tune hyperparameters and avoid overfitting.
- Testing set: Evaluate final performance on unseen data.

Q2. Why do we keep the test set completely unseen during model training?
Answer: To simulate real-world unseen data and prevent biased evaluation by the model memorizing the test data.

Q3. Differentiate between training set and validation set.
Answer:

- Training set: Used to fit model parameters.
- Validation set: Used for model tuning and selecting the best configuration without touching the test set.

Q4. Define underfitting.
Answer: Model is too simple to learn patterns, resulting in poor accuracy on both training and testing data.

Q5. Define overfitting.
Answer: Model learns the training data too well, including noise, leading to poor generalization on new data.

Q6. What is the main difference between high bias and high variance?
Answer:

- High bias: Model is too simple, underfits data.
- High variance: Model is too complex, overfits data.

Q7. Why is random_state used in train_test_split()?
Answer: To ensure reproducibility by generating the same split every time the code is run.

Q8. Explain the use of the stratify parameter in train_test_split().
Answer: Ensures the class distribution in training and testing sets matches the original dataset distribution.

1. Which of the following is a symptom of overfitting?
   a) High training accuracy, low testing accuracy
   b) Low training accuracy, high testing accuracy
   c) Low accuracy on both training and testing sets
   d) Model fits perfectly on test data
   Answer: a
2. Underfitting is most likely to occur when:
   a) Model is too complex
   b) Model is too simple
   c) Dataset has too many features
   d) There is no noise in data
   Answer: b
3. In train_test_split(X, y, test_size=0.2), what proportion of data will be in the training set?
   a) 20%
   b) 80%
   c) 50%
   d) Depends on random_state
   Answer: b
4. Which parameter ensures that the class distribution is similar in train and test sets?
   a) shuffle
   b) random_state
   c) stratify

d) train_size

Answer: c

Q1. You train a model and get 98% training accuracy but 72% testing accuracy. What problem is likely occurring? Suggest a solution.
Answer: Overfitting.
Solution examples:

- Simplify model (reduce complexity)
- Add regularization (L1/L2)
- Increase training data

Q2. Your model performs poorly on both training and testing datasets. What could be the issue? Suggest at least two fixes.
Answer: Underfitting.
Solutions:

- Increase model complexity
- Add more features
- Reduce regularization strength

Q4. In a classification problem, after splitting data without using stratify, you find that one class is missing in the test set. What caused this, and how would you fix it?
Answer: The random split created an imbalanced distribution.
Fix: Use stratify=y in train_test_split().

```
train_test_split(

   *arrays,                  # X, y or other arrays

   test_size=0.25,           # Proportion or absolute number for test data

   train_size=None,          # Same as above; defaults to remaining

   random_state=None,        # Seed for reproducibility
```

```
    shuffle=True,              # Shuffle before splitting
    stratify=None              # Ensures proportional class distribution
)
```