K-Fold Cross Validation

What is K-Fold Cross Validation?

K-Fold Cross Validation is a technique used to **evaluate machine learning models** more reliably by splitting the dataset into **K parts (folds)** and running the training and testing process **K times.**

How does it work?

- **Split the dataset into K equal-sized folds.**
- **For each fold:**
    - **Use that fold as the test set.**
    - **Use the remaining K-1 folds as the training set.**
- **Train and test the model K times, each time with a different test fold.**
- **Finally, average the results to get a better estimate of performance.**

Why use it?

- **Help to Reduces overfitting: Since every data point is used for both training and testing.**
- **Useful when the dataset is small.(500 row<x)**
- **Provides a more accurate evaluation of model performance.**

✅ **If k-fold cross validation reduce overfitting**

K-fold CV itself doesn't directly "reduce" overfitting like a regularizer or dropout.

But it enables you to choose hyperparameters and models that generalize better, thereby indirectly reducing overfitting.

It also encourages using all data for training after evaluation, which helps improve generalization

1)Better model selection and hyperparameter tuning:

- **By evaluating the model's performance on multiple folds, you get a more stable and honest estimate of how your model behaves on unseen data.**
- **This prevents you from over-tuning your model to perform well on just one specific validation set.**
- **You can choose simpler models or hyperparameters that perform consistently well across folds, which usually means less overfitting.**

2) Use all data for training in the final model:

- **After finding the best model via k-fold CV, you can retrain it on the entire dataset (all folds combined).**
- **Since the model is validated on all parts of the data, you have more confidence it generalizes well.**
- **Training on more data usually reduces overfitting compared to training on a smaller training split.**

3) Reducing model complexity with validation feedback:

- **If your model overfits in CV (big difference between training and validation scores), you can reduce model complexity (e.g., fewer features, smaller networks).**
- **CV helps you find this sweet spot so you don’t overfit unnecessarily**

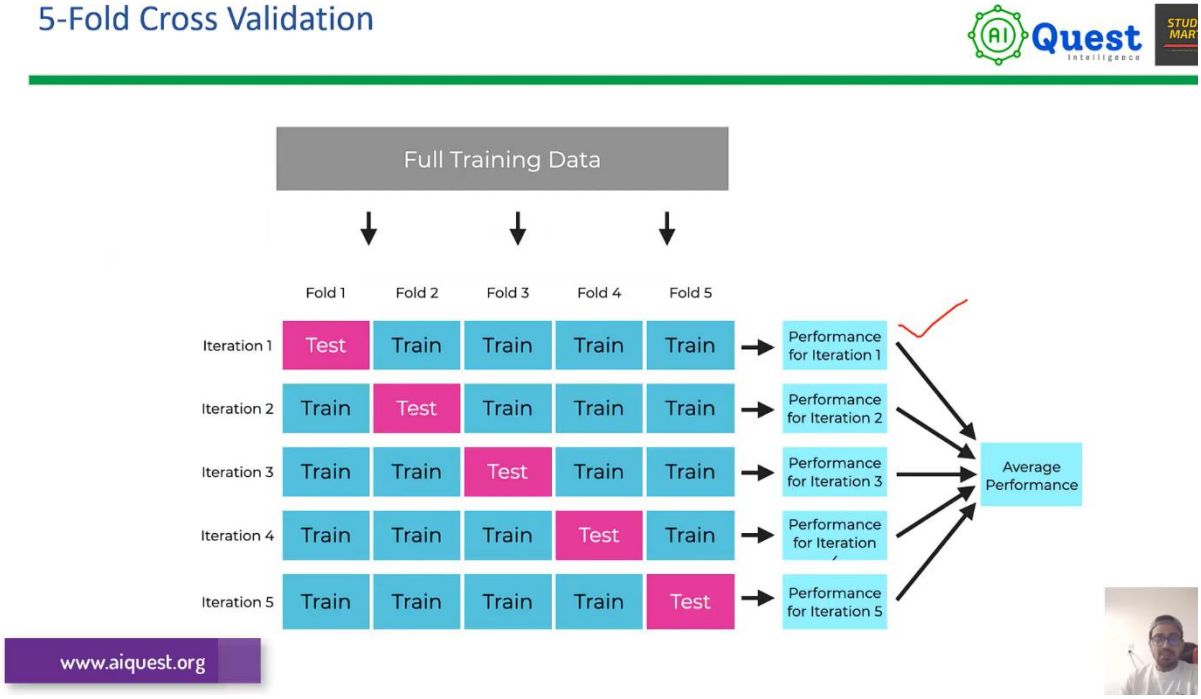
Why is there no leakage even after looking at all the data?

Let's say K=5K = 5K=5. Dataset = A, B, C, D, E (5 fold).

The 5 iterations of K-Fold will be as follows:

Iteration	Training Set	Validation Set
1	B, C, D, E	A
2	A, C, D, E	B
3	A, B, D, E	C
4	A, B, C, E	D
5	A, B, C, D	AND

- **In each iteration, the model is trained only with the training set .**
- **The validation fold's data model *is not then visible* .**
- **In the next iteration, the model is retrained — the model from the previous iteration is not reused.**
- **So even though the entire dataset is eventually used for validation at some point, no single model instance ever sees all the data .**



How we get validation data in k-fold cross validation?

1. For each fold i from 1 to k:

   Use the i-th fold as **validation data**.

   Use the other k-1 folds combined as **training data**.

2. Train your model on the training data and evaluate on the validation fold.

   After all k iterations, average the validation scores for a final performance estimate

What is StratifiedKFold?

StratifiedKFold ensures that **each fold** of the dataset **preserves the percentage of samples** for **each class**, just like the original dataset

Useful when:

- You have **imbalanced data** (e.g., 80% class A, 20% class B).
- You want **fair evaluation** in each fold with **similar class proportions**.

Real-life Example:

Let's say you have 10 samples like this:

Copy code

y = [0, 0, 0, 0, 0, 1, 1, 1, 1, 1] # 5 zeros and 5 ones

With **normal KFold**, one fold might have 4 zeros and 1 one, another 1 zero and 4 ones, causing **class imbalance** during training/testing.

But with **StratifiedKFold**, each fold will try to **keep class ratios similar** to original.

Summary

- K-Fold is a **cross-validation technique**.
- Avoids overfitting and underfitting by testing on all data.
- Can be combined with any model (e.g., Logistic Regression, SVM, etc.).