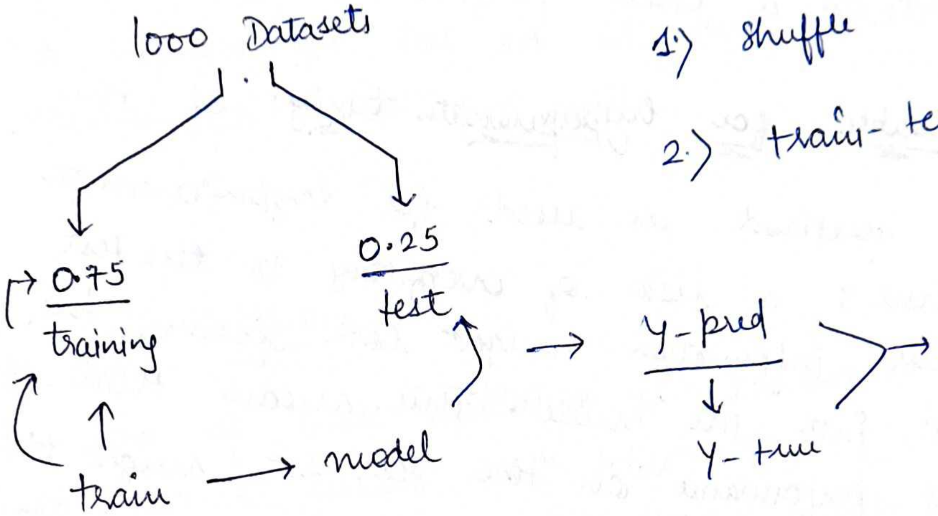
CROSS VALIDATION

The Hold-Out Approach → train-test-split

1.) shuffle

2.) train-test



Problem with Hold-out Approach

1. **Variability**: The performance of the model can be very sensitive to how the data is divided into training and testing sets. If the split is unfortunate, the training set may not be representative of the overall distribution of data, or the test set might contain unusually easy or difficult examples. This leads to higher variance in the estimation of the model's performance

2. **Data inefficiency**: The holdout method only uses a portion of the data for training and a different portion for testing. This means that the model doesn't get to learn from all available data, which can be particulary problematic if the dataset is small.

3. **Bias in performance estimation**: If some classes or patterns are over - or under- represented in the training set or test due to the random split, it can lead to a biased performance estimation.

4. **Less reliable for hyperparameter tuning**: If the holdout method is used for hyperparameter tuning, there's a risk of overfitting to the test set because information might leak from the test set into the model. This means that the model's performance on the test set might the overly optimistic and not representative of its performance on unseen data.

Why is hold-out approach used then?

1. **Simplicity**: The holdout method is straightforward and easy to understand. You simply divide your data into two sets: a training set and a test sets. This simplicity makes it appealing especially for initial exploratory analysis or simple projects.

2. **Computation Efficiency**: The holdout methods is computionally less intensive than methods like k-fold cross-validation. In k-fold cross-validation, you need to train and test your model k times, which can be computationally expensive, especially for large datasets or complex models

with the holdout method you only train the model once. ③

3. <u>large Datasets</u>: For very large datasets, even a small proportion of the data may be sufficient to form a representative test set. In these cases, the holdout method can work quite well.

Cross Validation

The idea of cross-validation is to divide the data into several subsets or "folds". The model is then trained on some of these subsets and tested on the remaining ones. This process is repeated multiple times, with different subsets used for training and validation each time. The results from each round are usually averaged to estimate the model's overall performance.

Sampling :- Population Data $\xrightarrow{pop}$ observation → estimate (eg:- Avg Salary of Emp. in India)

↑ Sample

Resampling → is the creation of new samples based on one observed sample.

Resampling → Cross val, Bootstrap

Iris dataset → Sample data → (100 dataset) Sample1, Sample2, Sample3

using Model → Sample1 M1, Sample2 M2, Sample3 M3

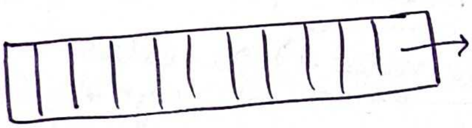
Resampling → use sample data Create another 3 sample data

Cross Validation

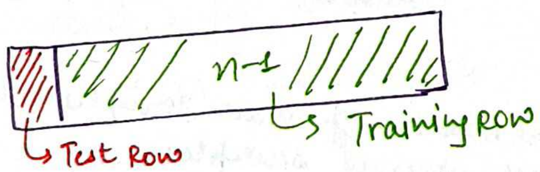
- Leave One Out CV (LOOCV)
  - Leave point CV
- k-fold CV
  - k-fold
  - stratified
  - nested k-fold
  - repeated k-fold
  - Time based → Time Series

Leave One Out Cross-Validation (LOOCV)

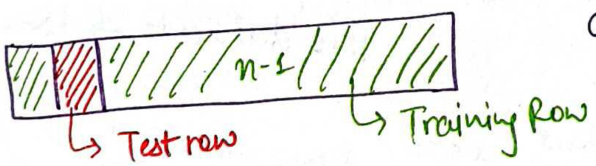
data → 1000 rows → model (1000) → model evaluation



→ 1000 row

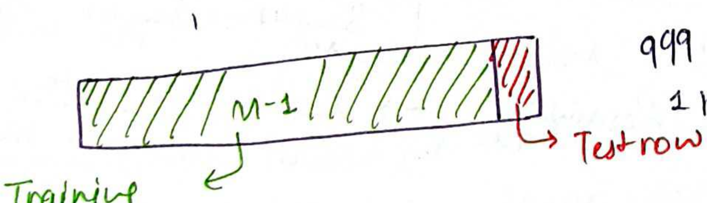


999 row → Training] model(1)
1 row → Test



999 row → Training] model (2)
1 row → Test

⋮



999 row → Training] model (1000)
1 row → Test

m1 + m2 + m3 - - - - m(100) → avg → final accuracy score

Advantages

1. **Use of data**: LOOCV uses almost all of the data for training, which can be beneficial in situation when the dataset is small and every data is valuable.

2. **Less Bias**: Since each iteration of validation is performed on just one data point. LOOCV is less biased than other method, such as k-fold cross validation. The validation process is less dependent on the random partitioning of data.

3. **No Randomness**: There's no randomness in the train/test split, so the evaluation is stable, without variation in the results due to different random split

Disadvantages

1. **Computational Expense**: LOOCV requires fitting the model N times, which can be computationally expensive and time-consuming for large datasets.

2. **High Variance**: LOOCV can lead to higher variance in the model performance since the training sets in all iterations are very to each other.

3. **Inappropriate Performance Metric**: Performance metrics like R^2 are not appropriate to be used with LOOCV as they are not defined when the validation set only has one sample.

4. **Not Ideal for Imbalance Data:** In classification problems. If you have imbalanced classes, LOOCV may not provide a reliable estimate of model performance because the single validation sample in each iteration may not be representitive of the overall class distribution.

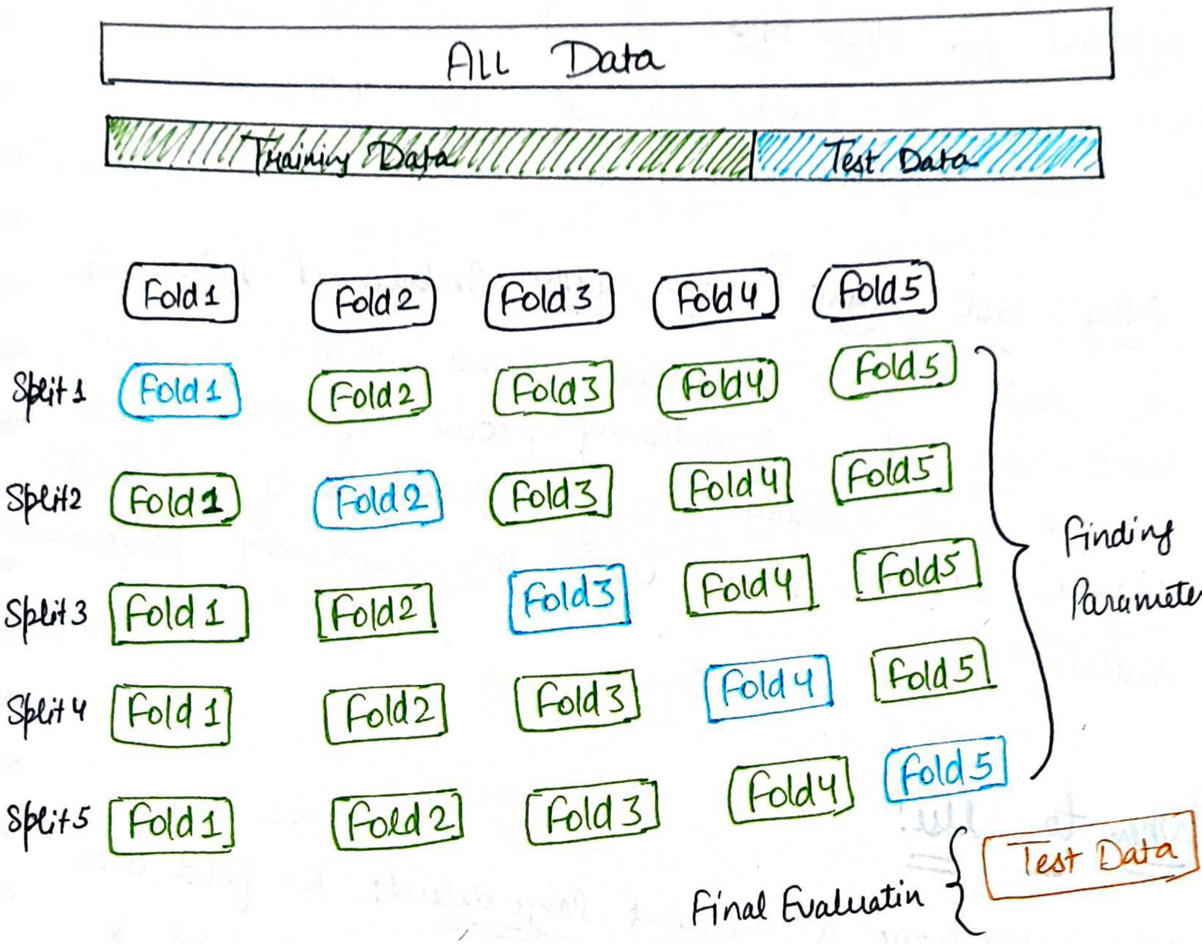
When to use:

1. **Small datasets:** LOOCV is most beneficial when you have a limited amount of data. With small datasets, you want to use as much data as possible for training to get a reliable model, which is exactly what LOOCV offers by using all but one data point for training.

2. **Balanced datasets:** LOOCV might not performance well on imbalanced datasets especially in classification problem, because the training set might end up missing some classes. Thus, it's more appropriate to use LOOCV when you have a balanced dataset.

3. **Need for less biased performance estimate:** Since LOOCV use nearly all the data for training, it gives a less biased estimate of model performance compared to other method like k-fold cross-validation

K-Fold Cross Validation



Advantage

1. **Reduction of variance**: By averaging over k different partitions, the variance of the performance estimating is reduced. This is beneficial because it means that the performance estimate is less sensitive to the particular random partitioning of the data

2. **Computationally Inexpensive**: Take less time and space in comparison to LOOCV.

Disadvantage

1. **Potential for High Bias:** If k is too small, there could be high bias if the test set is not represntive of the overall population.

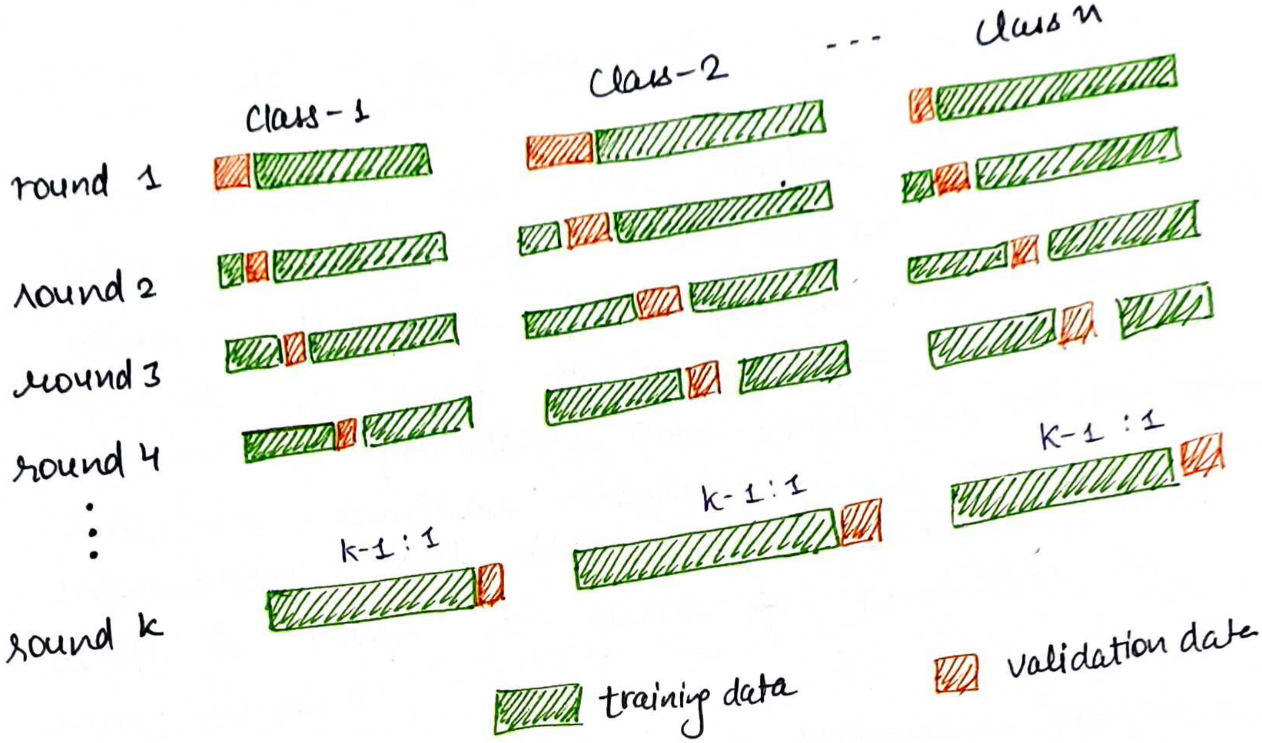
2. **May not work well with Imbalanced data:** If the data has imbalanced classes, there's a risk that in the partitioning, some of the folds might not contain any samples of minority classes, which can lead to misleading performance metrics.

When to Use:

1. **When you have a sufficient large dataset:** K-fold cross validation requires the model to be trained k times, so, it can be computationally intensive. However, if your dataset is large enough, this increased computational cost can be justified by the more reliable estimate of model performance.

2. **When your data is evenly distributed:** K-fold cross validation works best when the data is evenly distributed.

Stratified K-Fold CV



* Work on imbalanced data

Validation Set

Data

→ Data split into train, validation and test

Train | Validation | Test → unseen data

data leakage

Train → Model

Validation → 85% accuracy

Validation → Hyperparameter perform on validation

Test → 79% actual accuracy

So, we can test our model on unseen data and find actual accuracy