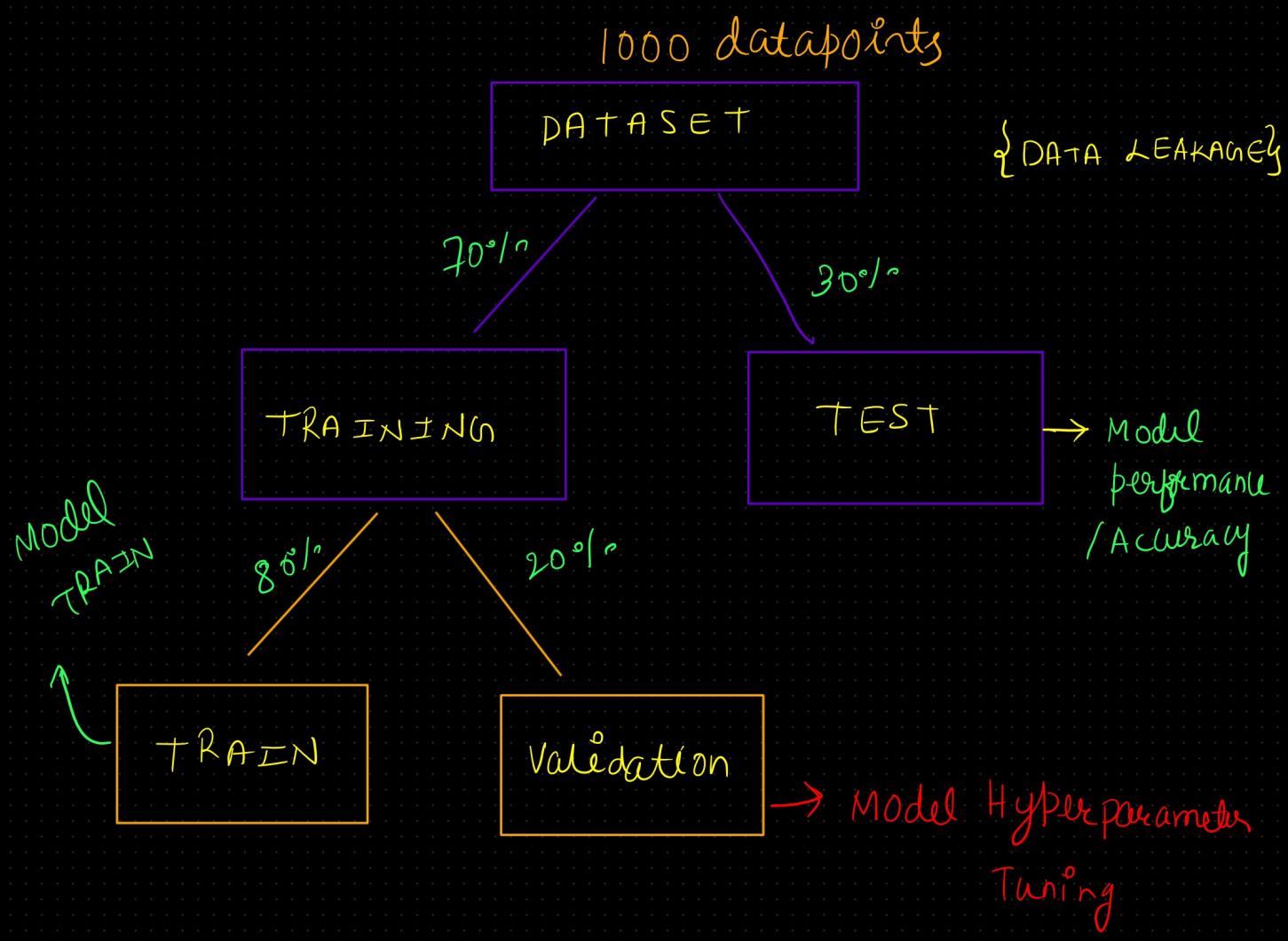
Cross Validation

Cross Validation (CV)

→ Cross-validation is a statistical technique used to evaluate the performance of a predictive model by partitioning the dataset into subsets, training the model on some subsets, and testing it on the remaining subset. This helps to assess how well the model generalizes to an independent dataset and avoids overfitting.

→ Cross-validation can be particularly useful for evaluating the model's performance and tuning its hyperparameters.



random-state = 42 → Acc → 87%

random-state = 100 → Acc → 80%

random-state = 35 → Acc → 90%

→ Acc → 95%

[80 – 95]

CROSS VALIDATION

CV = 5

Exp 1	TRAIN	VALIDATE ACC1
Exp 2	TRAIN	VALIDATE ACC2
Exp 3	TRAIN	VALIDATE ACC3
Exp 4	–	– ACC4
Exp 5	–	– ACC5

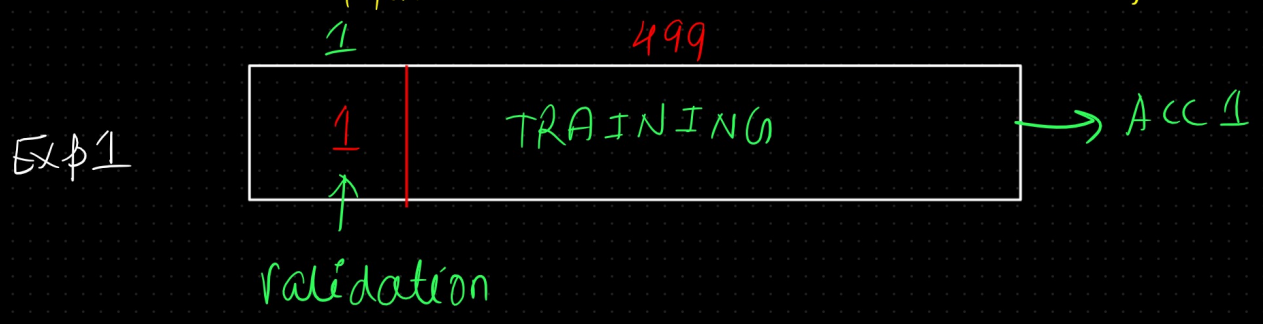
[ACC1, ACC2, ACC3, ACC4, ACC5]

$\boxed{\text{Final Accuracy = Mean of this Acc.}}$

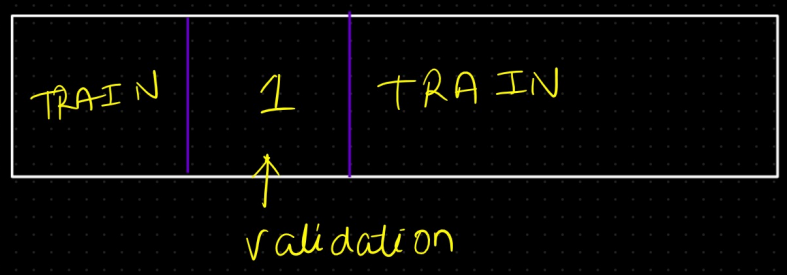
Types of CROSS VALIDATION

1. Leave One out cross validation

TRAINING DATASET → 500 datapoints



Exp 2



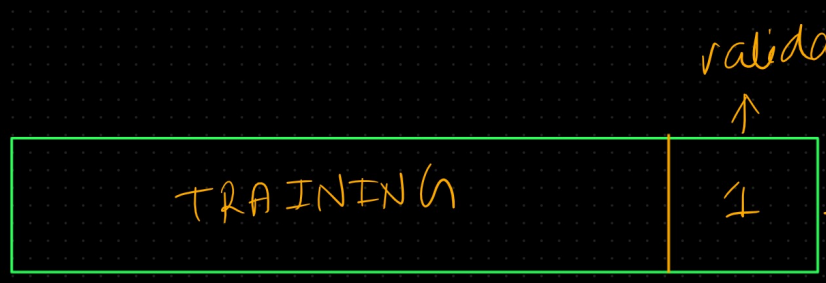
→ ACC 2

Exp 3

→ ACC 3

⋮

Exp 500



→ ACC 500

$\underbrace{\text{ACC 1, ACC 2, ACC 3, ---- ACC 500}}$

Final Accuracy = Mean

Disadvantages

① Time complexity is more for large TRAINING DATASET

② Model Overfit → TRAINING ↑↑

NEW DATA → validation → ACC ↓↓

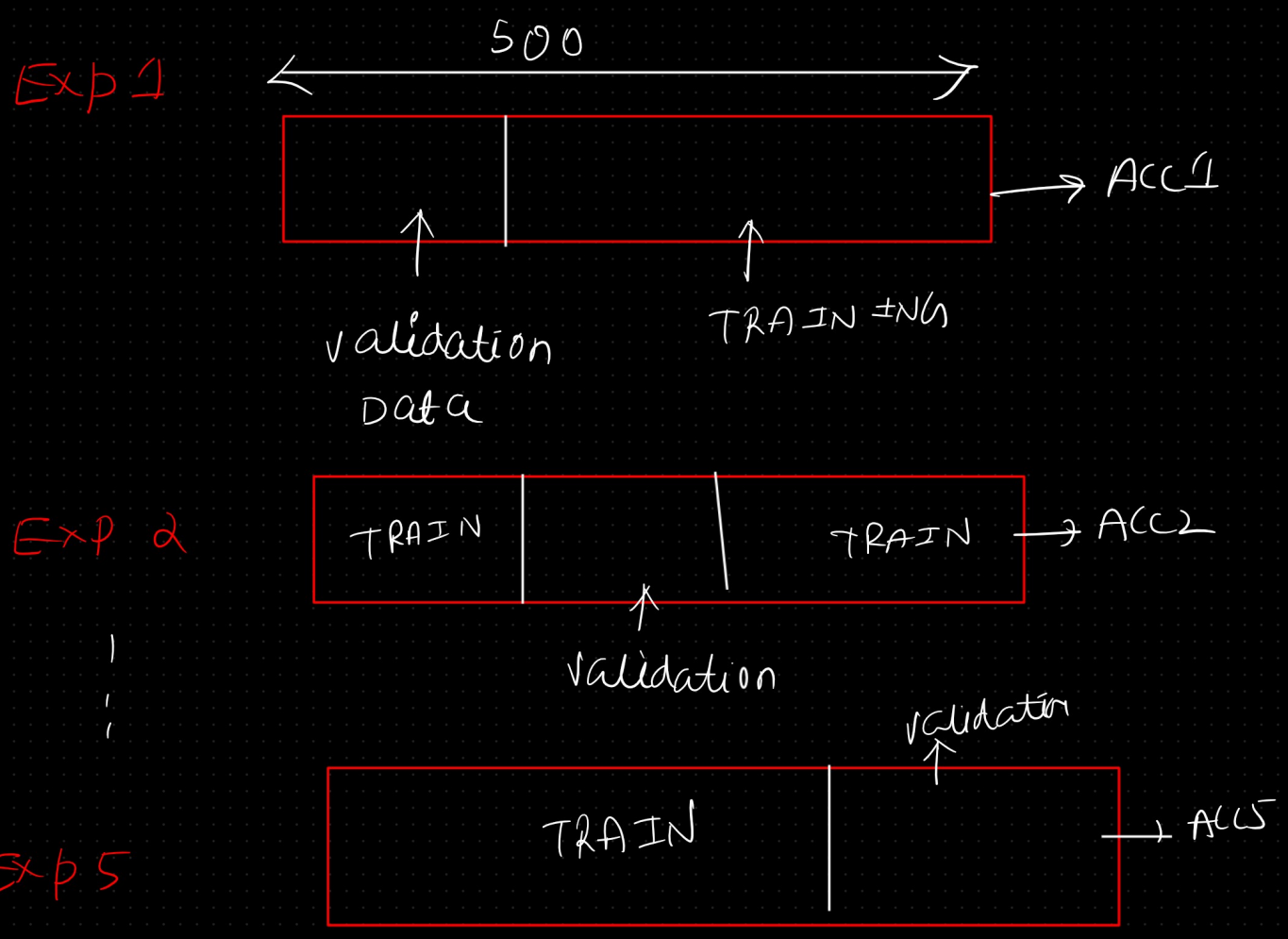
In LOOCV, each observation in the dataset is used as a test set, one at a time, with the remaining observations forming the training set.
This means if you have n observations, the model is trained on n-1 observations and tested on the one left out.

② Leave p out CV

$p = 10, 20, 30, 40 \rightarrow$ Hyperparameter

③ K - Fold CROSS VALIDATION

CV = 5



In k-fold cross-validation, the dataset is divided into k subsets (or folds) of approximately equal size.
The model is trained on k-1 folds and tested on the remaining fold. This process is repeated k times, each time using a different fold as the test set.
The final performance metric is calculated by averaging the performance of the model across all k folds.

For example, if you choose 5-fold cross-validation, the dataset will be divided into 5 subsets. The model will be trained on 4 subsets and tested on the remaining subset. This process is repeated 5 times with a different subset used for testing each time

4. Stratified K-Fold CV

{ Imbalanced DATASET } → 900 – 1, 100 – 0

Validation size $= \frac{500}{5} = 100$

EXP 1

Validation → ACC 1

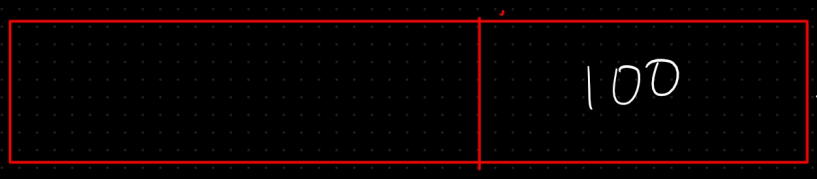
400 → 1
100 → 0 } Imbalanced Dataset

⋮

EXP 5

0 → 40, 60 → 100 f1

50, 50 — 100 f2



→ ACC 5

y: 0, 0, 0, 1, 0, 1, 1, ⋮

Stratified k-fold cross-validation is used when the dataset is imbalanced, meaning one class may be significantly more prevalent than the other(s).
It ensures that each fold maintains the same class distribution as the original dataset, thus providing more reliable performance estimates, especially for imbalanced datasets.
This is particularly important in logistic regression when dealing with binary classification problems with imbalanced classes.

Example

Example:
Suppose you have a dataset of emails labeled as spam (1) or not spam (0). You want to build a logistic regression model to classify emails. You decide to use 5-fold cross-validation to evaluate the model.

CV = 5

You divide the dataset into 5 folds.
For each fold:
You train the logistic regression model on 4 folds.
You test the model on the remaining fold.
You calculate the performance metric (e.g., accuracy, precision, recall) for this fold.
Repeat this process for all 5 folds and calculate the average performance metric across all folds to assess the model's overall performance.
This process helps you estimate how well your logistic regression model performs on unseen data and allows you to fine-tune its parameters to improve its generalization capability.

Hyper Parameter Tuning

↳ Finding the best parameters while training the model

1. Grid Search CV
2. Randomized Search CV

Hyperparameter tuning is the process of selecting the best set of hyperparameters for a machine learning model. Hyperparameters are parameters that are not directly learned from the data during the training process but are set prior to training. Tuning these hyperparameters is crucial for improving the performance of the model.

GRID SEARCH + CV

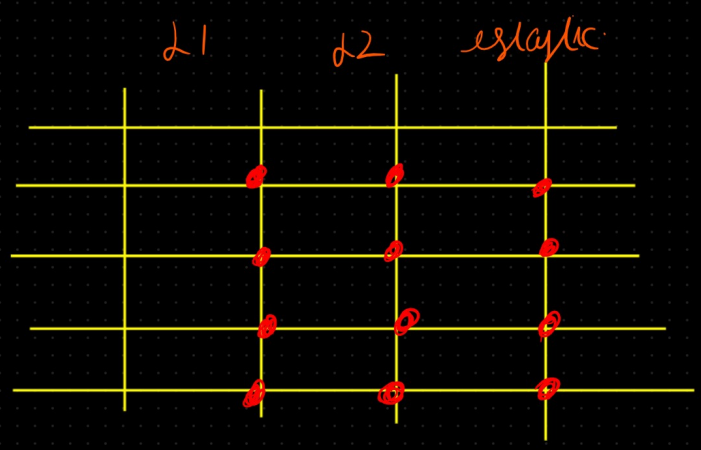
penalty{'l1', 'l2', 'elasticnet', None}

solver{'lbfgs', 'liblinear', 'newton-cg', 'newton-cholesky', 'sag', 'saga'}, default='lbfgs'

model = Logistic Regression

⇓

{elasticnet, newton_cg}



→ To Increase the Model performance or Accuracy

Disadvantages

→ Time complexity ↑↑ for training the model

In grid search, you specify a list of values for each hyperparameter that you want to tune.
The algorithm then exhaustively tries all possible combinations of hyperparameters from the specified lists.
It evaluates the model performance using cross-validation and selects the combination of hyperparameters that yields the best performance.
While grid search is simple and straightforward, it can be computationally expensive, especially for a large number of hyperparameters or when the ranges of values are broad

Randomized Search CV

n_iter = 10 + CV = 5

10 different combination + CV = 5

Random search selects random combinations of hyperparameters from predefined ranges.
Unlike grid search, which exhaustively searches through all combinations, random search samples hyperparameter values randomly.
Random search is more efficient than grid search in many cases, especially when only a few hyperparameters have a significant impact on the model's performance.
It is particularly useful when the search space is large and not all hyperparameters are equally important.