Cross validation:

Cross validation is a technique used in machine learning to evaluate the performance of a model on unseen data. It involves dividing the available data into multiple folds or subsets, using one of these folds as a validation set, and training the model on the remaining folds.

What is cross-validation used for?

The main purpose of cross validation is to prevent overfitting, which occurs when a model is trained too well on the training data and performs poorly on new, unseen data.

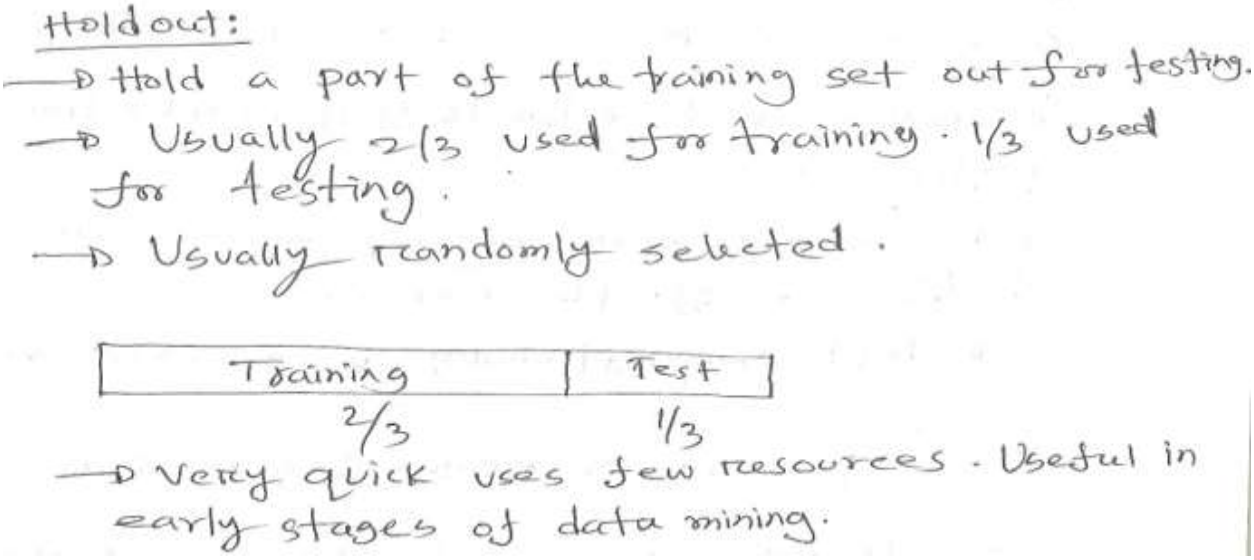
Types of Cross-Validation

There are several types of cross validation techniques, including

1. Holdout validation
2. Leave-one-out cross validation
3. K-fold cross validation
4. Stratified cross-validation

1. **Holdout Validation**

In Holdout Validation, we perform training on the 50% of the given dataset and rest 50% is used for the testing purpose. It's a simple and quick way to evaluate a model.



Drawback: This method is that we perform training on the 50% of the dataset, it may possible that the remaining 50% of the data **contains some important information** which we are leaving while training our model i.e. higher bias.

2. LOOCV (Leave One Out Cross Validation)

In LOOCV, the model is trained on (n−1) samples and tested on the one omitted sample, repeating this process for each data point in the dataset.

Advantage: Using this method is that we make use of **all data points** and hence it is **low bias.**

The major **drawback** of this method is that it **leads to higher variation** in the testing model as we are testing against one data point. If the data point is an **outlier** it can lead to higher variation. Another drawback is it takes a **lot of execution time** as it iterates over 'the number of data points' times.

- ➔ If the data point is an **outlier** it can lead to higher variation
- ➔ It takes a **lot of execution time** as it iterates over 'the number of data points' times.

3. K-Fold Cross Validation

In K-Fold Cross Validation, we split the dataset into k number of subsets (known as folds) then we perform training on the all the subsets but leave one **(k-**1) subset for the evaluation of the trained model.

In this method, we iterate k times with a different subset reserved for testing purpose each time.

Example of K Fold Cross Validation

Here, we have total 25 instances. In first iteration we use the first 20 percent of data for evaluation, and the remaining 80 percent for training ([1-5] testing and [5-25] training) while in the second iteration we use the second subset of 20 percent for evaluation, and the remaining three subsets of the data for training ([5-10] testing and [1-5 and 10-25] training), and so on.

```
Total instances: 25
Value of k     : 5
No. Iteration                    Training set observations                    Testing set observations
 1     [ 5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24]   [0 1 2 3 4]
 2     [ 0  1  2  3  4 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24]   [5 6 7 8 9]
 3     [ 0  1  2  3  4  5  6  7  8  9 15 16 17 18 19 20 21 22 23 24]   [10 11 12 13 14]
 4     [ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 20 21 22 23 24]   [15 16 17 18 19]
 5     [ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19]   [20 21 22 23 24]
```

4. Stratified Cross-Validation:

This is particularly important when dealing with **imbalanced datasets**, where certain classes may be underrepresented. In this method,

1. The dataset is divided into k folds while maintaining the proportion of classes in each fold.
2. During each iteration, one-fold is used for testing, and the remaining folds are used for training.

3. The process is repeated k times, with each fold serving as the **test set exactly once**.

Example Dataset:

Let's assume we have a small dataset with 10 samples and a binary class label (0 or 1). The class distribution is imbalanced:

Sample	Feature 1	Feature 2	Class
1	2.5	3.0	0
2	3.5	1.5	0
3	4.0	2.5	1
4	3.0	3.5	0
5	2.0	2.0	1
6	1.5	1.0	0
7	2.8	3.8	1
8	4.2	3.2	1
9	3.5	2.7	0
10	3.8	1.3	1

- **Class 0**: 5 samples
- **Class 1**: 5 samples

Goal:

We want to perform **Stratified 2-fold Cross-Validation** to ensure that each fold contains approximately the same ratio of Class 0 and Class 1 samples.

Step 1: Split the Dataset into 2 Folds

We'll split the dataset into **2 folds**, and because of the **stratified** nature, each fold must have an approximately equal proportion of Class 0 and Class 1 samples.

- The original dataset has **5 Class 0** and 5 **Class 1**.
- In each fold, there should be roughly 3 Class 0 samples and 3 Class 1 samples.

Step 2: Constructing the Folds

Fold 1 (Training set for first iteration)

Sample	Feature 1	Feature 2	Class
1	2.5	3.0	0
2	3.5	1.5	0
4	3.0	3.5	0
5	2.0	2.0	1
7	2.8	3.8	1
8	4.2	3.2	1

Fold 2 (Test set for first iteration)

Sample	Feature 1	Feature 2	Class
3	4.0	2.5	1
6	1.5	1.0	0
9	3.5	2.7	0
10	3.8	1.3	1

Fold 1 (Training set for second iteration)

Sample	Feature 1	Feature 2	Class
3	4.0	2.5	1
6	1.5	1.0	0
9	3.5	2.7	0
10	3.8	1.3	1

Fold 2 (Test set for second iteration)

Sample	Feature 1	Feature 2	Class
1	2.5	3.0	0
2	3.5	1.5	0
4	3.0	3.5	0
5	2.0	2.0	1
7	2.8	3.8	1
8	4.2	3.2	1