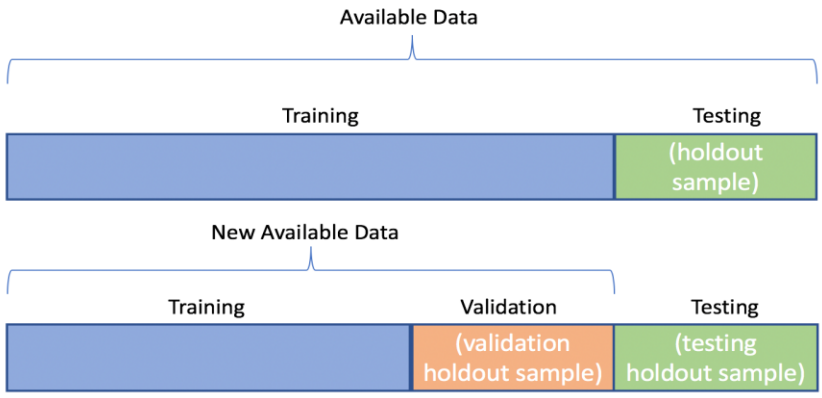
In **machine learning**, we couldn't fit the model on the training data and can't say that the model will work accurately for the real data. For this, we must assure that our model got the correct patterns from the data, and it is not getting up too much noise. For this purpose, we use the **cross-validation technique**. In this article, we'll delve into the process of cross-validation in machine learning.

What is 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. This process is repeated multiple times, each time using a different fold as the validation set. Finally, the results from each validation step are averaged to produce a more robust estimate of the model's performance. Cross validation is an important step in the machine learning process and helps to ensure that the model selected for deployment is robust and generalizes well to new data.



ChatGPT definition:

In cross-validation, the dataset is indeed split into multiple subsets, typically referred to as "folds." One common approach is k-fold cross-validation, where the dataset is divided into k equal-sized folds.

For each iteration, one of the folds is held out as the validation set, while the remaining folds are used as the training set. **The model is trained on the training set and evaluated on the validation set**. This process is repeated k times, each time with a different fold held out for validation.

The accuracy (or other evaluation metric) is then calculated for each iteration based on the predictions made on the validation set. After all iterations are completed, the overall performance metric is often calculated by averaging the results from each fold.

This technique helps in estimating the model's performance more accurately compared to a single train-test split, as it ensures that each data point is used for both training and validation, reducing the risk of overfitting or underfitting to a particular subset of the data.

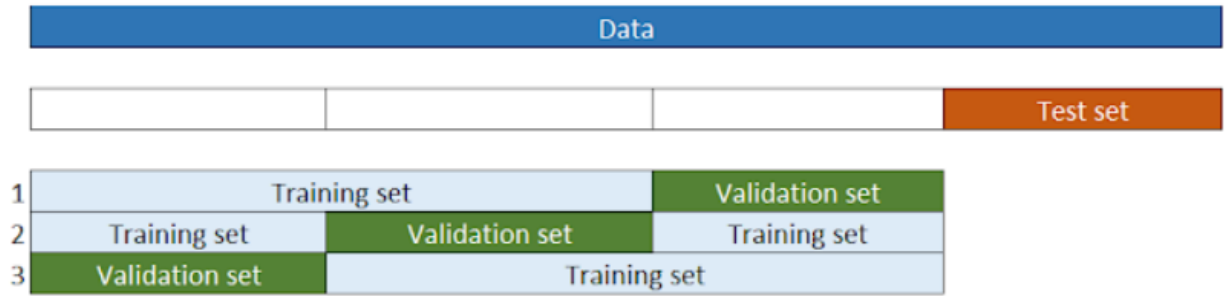
What is cross-validation used for?

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. By evaluating the model on multiple validation sets, cross validation provides a more realistic estimate of the model's generalization performance, i.e., its ability to perform well on new, unseen data.

NOTE: *Once we done with cross validation, it will pick the best hyperparameters needs to be initiated while training the model so that we will get best accuracy on the new unseen data.*

Explanation of Cross validation:

As we know our dataset is split into train and test dataset. Training dataset further divided into training and **validation data** set.



Validation dataset is used to tune the **hyper-parameters** (Hyper-parameter tuning)**

Hyperparameters means: The parameters which are initialized while training the model.

class sklearn.linear_model.LogisticRegression(*penalty='l2', *, dual=False, tol=0.0001, C=1.0, fit_intercept=True, intercept_scaling=1, class_weight=None, random_state=None, solver='lbfgs', max_iter=100, multi_class='auto', verbose=0, warm_start=False, n_jobs=None, l1_ratio=None*)

The below parameters are considered as the hyperparameters for above lasso regression model:

penalty='l2', **, dual=False, tol=0.0001****, C=1.0,*** *fit_intercept=True, intercept_scaling=1, class_weight=None, random_state=None, solver='lbfgs', max_iter=100, multi_class='auto', verbose=0, warm_start=False, n_jobs=None, l1_ratio=None*

***** Bold highlighted are discussed earlier***

A validation dataset is a sample of data held back from training your model that is used to give an estimate of model skill while tuning model's hyperparameters.

The validation dataset is different from the test dataset that is also held back from the training of the model, but is instead used to give an unbiased estimate of the skill of the final tuned model when comparing or selecting between final models.

For example, if you have 1000 training data points and you took cv=5 (cross validation=5 times) then, 1000 datapoints will divided into 1000/5 = 200 datapoints each.

As we considered **cv=5**,

at **cv=1** iteration, the first 200 points (0th index to 199th index) of training data will be considered to train the model (to tune hyper parameter) and the remaining 800 points are considered as training data and we will evaluate the model performance using the validation dataset and find the accuracy (Let's say it is accuracy1)

at **cv=2** iteration, another 200 points (200th index to 399th index) of training data will be considered to train the model (to tune hyper parameter) and the remaining 800 points are considered as training data and we will evaluate the model performance using the validation dataset and find the accuracy (Let's say it is accuracy2)

at **cv=3** iteration, another 200 points (400th index to 599th index) of training data will be considered to train the model (to tune hyper parameter) and the remaining 800 points are considered as training data and we will evaluate the model performance using the validation dataset and find the accuracy (Let's say it is accuracy3)

at **cv=4** iteration, another 200 points (600th index to 799th index) of training data will be considered to train the model (to tune hyper parameter) and the remaining 800 points are considered as training data and we will evaluate the model

performance using the validation dataset and find the accuracy (Let's say it is accuracy4)

at **cv=5** iteration, another 200 points (800th index to 999th index) of training data will be considered to train the model (to tune hyper parameter) and the remaining 800 points are considered as training data and we will evaluate the model performance using the validation dataset and find the accuracy (Let's say it is accuracy5)

In every cross validation, the training and validation datapoints will change and we will get a different accuracy at each cross validation. *Validation dataset is used to tune the hyper parameters and the final accuracy of the model will be the average of 5 accuracy's:*

Final accuracy= (accuracy1+ accuracy2+ accuracy3+accuracy4+accuracy5)/5

Important NOTE:

- ➔ During each iteration (each instance), the model will be trained on the training data and evaluated on validation data. The accuracy which we get after evaluated on validation dataset will be considered. **Once we done with all the iterations (instances) we will find average of all the accuracies.**

Final accuracy= (accuracy1+ accuracy2+ accuracy3+accuracy4+accuracy5)/5

Q) During cross validation, based on which dataset we find the accuracies? (Interview Question)

In each iteration of the cross-validation process, **the accuracy (or other evaluation metric) is calculated based on the predictions made on the validation data, not the training data.**

The purpose of using cross-validation is to obtain an unbiased estimate of how well the model will perform on unseen data. By evaluating the model on validation data that it hasn't been trained on, we can assess its generalization performance and identify any potential issues such as overfitting.

The training data is used solely for training the model during each iteration of cross-validation, while the validation data is used for evaluating the model's performance. This helps ensure that the evaluation is independent of the training process and provides a more reliable estimate of the model's true performance.

NOTE:

As we know during **train_test_split** we have a parameter called "**random_state**", if we initialized the different values for the parameter "**random_state**" Let's say 42, 102,234 etc., for each and every value we will get different values (data) in test and train datasets and also the accuracy will also change but it is very hard to choose the best random_state value which provides good accuracy, So, we use cross validation. In cross validation it will take the values randomly from train dataset and once we are done with cross validation process it will tune the hyperparameters. We can use those hyperparameters to predict the outcome of new unseen data.

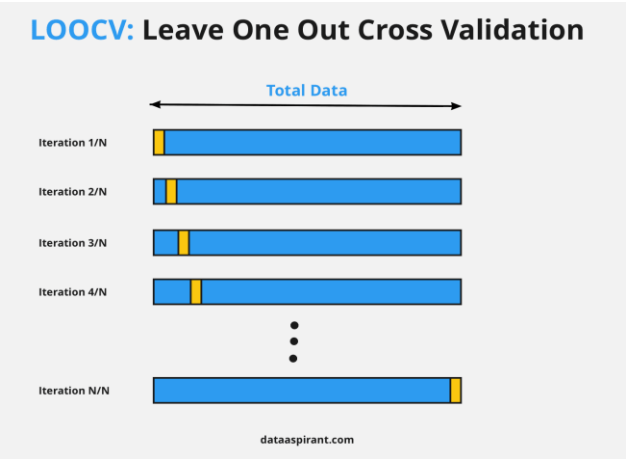
Types of Cross-Validation: (For every cross validation there is change in the way of picking the data for validation dataset)

There are several types of cross validation techniques, including **k-fold cross validation, leave-one-out cross validation, and Holdout validation, Stratified Cross-Validation.** The choice of technique depends on the size and nature of the data, as well as the specific requirements of the modeling problem.

leave-one-out cross validation (LOOCV):

In this method, we perform training on the whole dataset but leaves only one data-point of the available dataset and then iterates for each data-point. 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.

For example, if you have 500 training datapoints which ranging from 0^{th} index to 499^{th} index.



In 1st iteration:

- ➔ 0th index will be considered as “Validation data” and from 1st index to 499th index will be considered as training data.
- ➔ Train the model on the training set and evaluate it on the validation set.
- ➔ After evaluating on validation dataset, we will get $accuracy_i$ (“i” represents the iteration number)

In 2nd iteration:

- ➔ 1st index will be considered as “Validation data” and 0th index + 2nd index to 499th index will be considered as training data.
- ➔ Train the model on the training set and evaluate it on the validation set.
- ➔ After evaluating on validation dataset, we will get $accuracy_i$ (“i” represents the iteration number)

In 3rd iteration:

- ➔ 2st index will be considered as “Validation data” and 0th index, 1st index + 3rd index to 499th index will be considered as training data.
- ➔ Train the model on the training set and evaluate it on the validation set.
- ➔ After evaluating on validation dataset, we will get $accuracy_i$ (“i” represents the iteration number)

..... This process repeats for “n” times (where “n” is the length of training data) and we will get “n” number of accuracies

Final accuracies = Sum of “n” number of accuracy’s/ n → Average of all accuracies.

NOTE:

During this cross-validation process, internally it will find which **parameter** provides good accuracy and will finalize that for hyperparameter tuning.

Like below parameters:

penalty='l2', **, dual=False, tol=0.0001****, C=1.0,*** *fit_intercept=True, intercept_scaling=1, class_weight=None, random_state=None*

It has some advantages as well as disadvantages also.

An advantage of 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 (Overfit)** 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.

NOTE: Overfit will occur because the model well fitted on training data (as training data has n-1 data points every time) and we choose one data point at a time for validation dataset.

Training data accuracy will be high and test data accuracy will be low → **Overfitting**

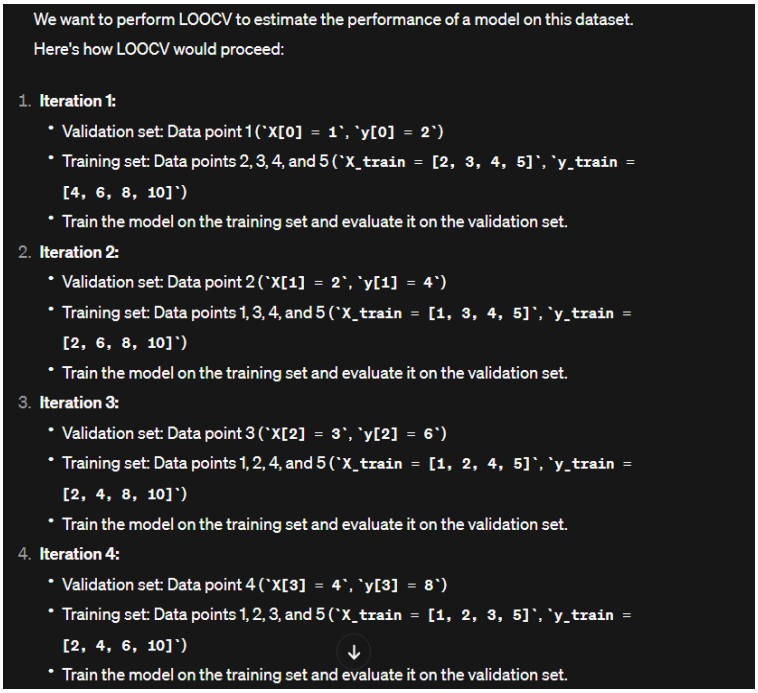
To avoid above draw back we have another cross-validation technique: **Leave-p-out cross validation: (LpOCV) (we will discuss later)**

Example to understand LOOCV:

Let's go through a simple example to illustrate how Leave-One-Out Cross-Validation (LOOCV) works:

Suppose we have a small dataset consisting of 5 data points:

X = [1, 2, 3, 4, 5] and y = [2, 4, 6, 8, 10]



5. **Iteration 5:**
   - Validation set: Data point 5 (`X[4] = 5`, `y[4] = 10`)
   - Training set: Data points 1, 2, 3, and 4 (`X_train = [1, 2, 3, 4]`, `y_train = [2, 4, 6, 8]`)
   - Train the model on the training set and evaluate it on the validation set.

After completing these iterations, we will have evaluated the model's performance on each individual data point. We can then calculate the average performance metric across all iterations to obtain an overall estimate of the model's performance through LOOCV.

Leave-p-out cross validation: (LpOCV)

Here p is the hyperparameter which needs to be initialized. Based on the p value the validation datapoints will be considered.

Leave-p-Out Cross-Validation (LpOCV) is a variation of cross-validation where, instead of leaving out just one data point in each iteration (as in LOOCV), you leave out p data points. This method involves generating all possible combinations of leaving out p data points and using the rest for training. It's a more generalized form of cross-validation that provides a compromise between the computational expense of LOOCV and the bias introduced by k-fold cross-validation.

Example to understand **LpOCV**: (In below example **p is considered as 2 so, L2OCV)**

Suppose we have a dataset with 6 data points:

```css
X = [1, 2, 3, 4, 5, 6]
y = [2, 4, 6, 8, 10, 12]
```

Now, let's say we want to perform Leave-2-Out Cross-Validation (L2OCV), where we leave out 2 data points in each iteration:

1. **Iteration 1:**
   - Validation set: Data points 1 and 2 (`X_val = [1, 2]`, `y_val = [2, 4]`)
   - Training set: Data points 3, 4, 5, and 6 (`X_train = [3, 4, 5, 6]`, `y_train = [6, 8, 10, 12]`)
   - Train the model on the training set and evaluate it on the validation set.
2. **Iteration 2:**
   - Validation set: Data points 1 and 3 (`X_val = [1, 3]`, `y_val = [2, 6]`)
   - Training set: Data points 2, 4, 5, and 6 (`X_train = [2, 4, 5, 6]`, `y_train = [4, 8, 10, 12]`)
   - Train the model on the training set and evaluate it on the validation set.

2. **Iteration 2:**
   - Validation set: Data points 1 and 3 (`X_val = [1, 3]`, `y_val = [2, 6]`)
   - Training set: Data points 2, 4, 5, and 6 (`X_train = [2, 4, 5, 6]`, `y_train = [4, 8, 10, 12]`)
   - Train the model on the training set and evaluate it on the validation set.
3. **Iteration 3:**
   - Validation set: Data points 1 and 4 (`X_val = [1, 4]`, `y_val = [2, 8]`)
   - Training set: Data points 2, 3, 5, and 6 (`X_train = [2, 3, 5, 6]`, `y_train = [4, 6, 10, 12]`)
   - Train the model on the training set and evaluate it on the validation set.

And so on, until all combinations of leaving out 2 data points have been evaluated.

LpOCV allows you to control the trade-off between computational cost and bias/variance of the estimate, with larger values of p leading to lower computational cost but potentially higher bias in the estimate of model performance.

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.

Note: *It is always suggested that the value of k should be 10 as the lower value of k is takes towards validation and higher value of k leads to LOOCV method.*

Example of k-fold cross-validation:

In K-fold cross validation, we need to initialize the “k” value, if you initialize k=5 and if you have 1000 data points then each validation dataset size will be 1000/5 = 200.

As we considered **k=5**,

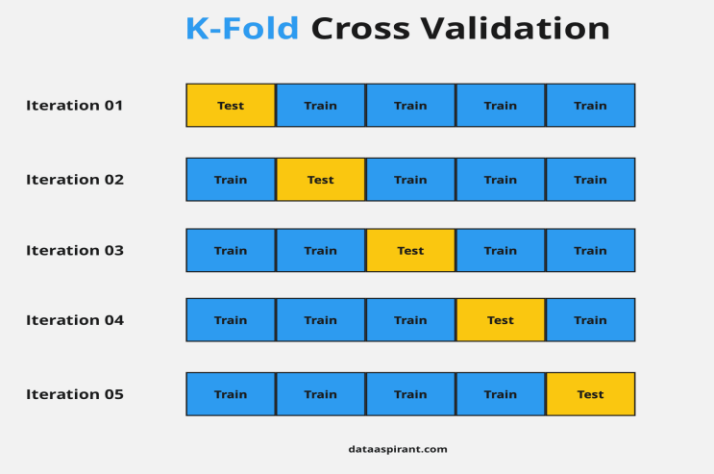
at **k=1** iteration, the first 200 points (0th index to 199th index) of training data will be considered to train the model (to tune hyper parameter) and the remaining 800 points are considered as training data and we will evaluate the model performance using the validation dataset and find the accuracy (Let's say it is accuracy1)

at **k=2** iteration, another 200 points (200th index to 399th index) of training data will be considered to train the model (to tune hyper parameter) and the remaining 800 points are considered as training data and we will evaluate the model performance using the validation dataset and find the accuracy (Let's say it is accuracy2)

at **k=3** iteration, another 200 points (400th index to 599th index) of training data will be considered to train the model (to tune hyper parameter) and the remaining 800 points are considered as training data and we will evaluate the model performance using the validation dataset and find the accuracy (Let's say it is accuracy3)

at **k=4** iteration, another 200 points (600th index to 799th index) of training data will be considered to train the model (to tune hyper parameter) and the remaining 800 points are considered as training data and we will evaluate the model performance using the validation dataset and find the accuracy (Let's say it is accuracy4)

at **k=5** iteration, another 200 points (800th index to 999th index) of training data will be considered to train the model (to tune hyper parameter) and the remaining 800 points are considered as training data and we will evaluate the model performance using the validation dataset and find the accuracy (Let's say it is accuracy5)



Another example:

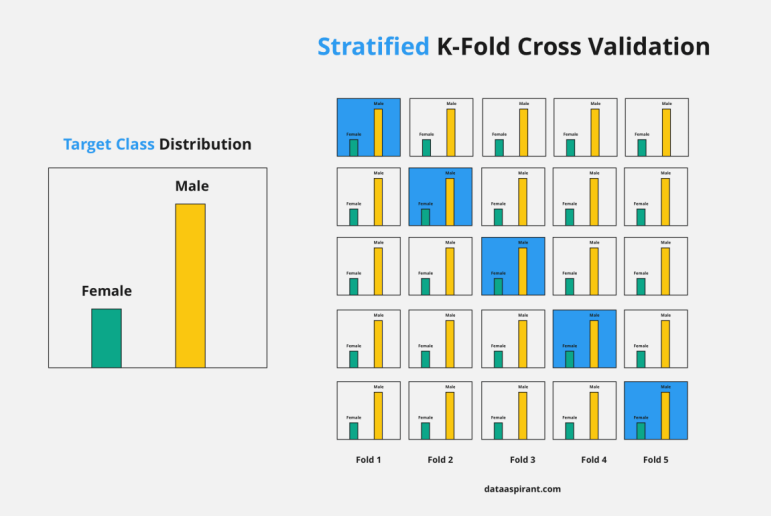
```
Total instances: 25
[0 1 2 3 4 5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24]
Value of k     : 5
No. Iteration                 Training set observations                      Testing set observations
 1      [ 5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24]   [0 1 2 3 4]
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

Stratified Cross-Validation or Stratified k-fold cross-Validation:

It is a technique used in machine learning to ensure that each fold of the cross-validation process maintains the same class distribution as the entire dataset. 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.

Stratified Cross-Validation is essential **when dealing with classification problems** where maintaining the balance of class distribution is crucial for the model to generalize well to unseen data.



PW Skills explanation for stratified K-fold CV: (It works as same as K-Fold)

- ➔ If you have imbalanced dataset having 500 datapoints. Class 0 has 150 datapoints and Class 1 has 350 data points
- ➔ If we take k=5 and we are having training dataset points=500, then validation dataset points range = 500/5 =100. That means we consider 100 datapoints as validation datapoints in every instance.
- ➔ Proportionality of Original training dataset: Class-0 is 150/500 ==30% and Class 1 is 350/500==70%
- ➔ In every Iteration, the validation data will be in the same proportion as like original training dataset (Please find the below image for more understanding)

In every iteration:

- ➔ The validation dataset will contain **Class-0:** 30% of 100 == 30 and **Class-1:** 70% of 100 == 70 datapoints and remaining rest 400 datapoints will consider as training data
- ➔ Train the model on the training set and evaluate it on the validation set.
- ➔ After evaluating on validation dataset, we will get accuracyi ("i" represents the iteration number)