Cross-Validation

Model Development is a critical stage in the life cycle of a Data Science project. We attempt to train our data set using various forms of Machine Learning models, either supervised or unsupervised, depending on the Business Problem.

Given many models available for solving business problems, we need to ensure that the model we choose performs effectively on unknown data after this phase.

Cross-Validations are a resampling strategy that ensures our model's efficiency and correctness when applied to previously unknown data. It is a technique for assessing Machine Learning models that involves training numerous more Machine Learning models on subsets of the available input data set and evaluating them on the subgroup.

Why cross validation is needed?

So what is wrong with testing the model on the training dataset?

If we do so, we assume that the training data represents all the possible scenarios of real-world and this will surely never be the case. Our main objective is that the model should be able to work well on the real-world data, although the training dataset is also real-world data, it represents a small set of all the possible data points(examples) out there.

So to know the real score of the model, it should be tested on the data that it has never seen before and this set of data is usually called testing set. But if we split our data into training data and testing data, aren't we going to lose some important information that the test dataset may hold?

Whenever we do train test split, we use random state variable. When random state value changes accuracy also will change.

So we use Cross-Validation to make different splits of our data to train and test our model and we average the accuracy overall this iteration to see the overall performance of our model.

This means that instead of splitting our dataset into two parts, one to train on and another to test on, we split our dataset into multiple portions, train on some of these and use the rest to test on. We then use a different portion to train and test our model on. This ensures that our model is training and testing on new data at every new step.

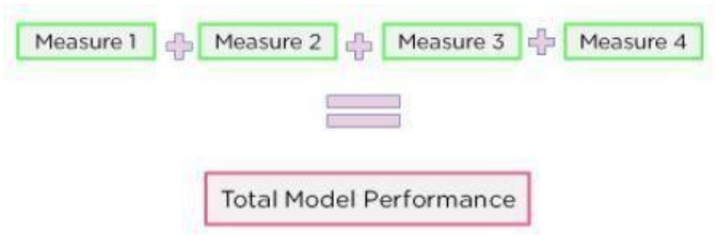
This also exposes our model to minority classes which may be present in the data. If we split our data into two and train only on one part, there is a chance that the test data contains a minority class that was not present in the testing data. In this case, our model will still perform well as the class constitutes only a small portion of the dataset but it will be desensitized to that data.

The goal of cross validation is to:

- Avoid sensitivity to test set selection
- Train on as much data as possible.
- Helps in preventing overfitting

The basic steps of cross-validations are:

- Reserve a subset of the dataset as a validation set.
- Provide the training to the model using the training dataset.
- Now, evaluate model performance using the validation set. To get the actualperformance metric the average of all measures is taken



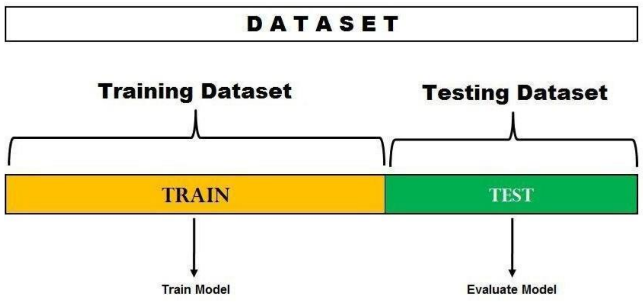
If the model performs well with the validation set, perform the further step, else checkfor the issues.

Types of cross-validation:

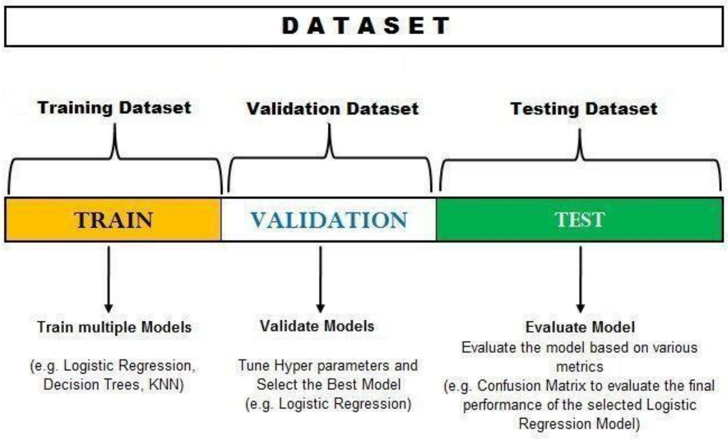
1. K-fold cross-validation
2. Hold-out cross-validation
3. Stratified k-fold cross-validation
4. Leave-p-out cross-validation
5. Leave-one-out cross-validation

Holdout cross-validation:

Also called a train-test split, holdout cross-validation has the entire dataset partitioned randomly into a training set and a validation set. A rule of thumb to partition data is that nearly 70% of the whole dataset will be used as a training set and the remaining 30% will be used as a validation set. Since the dataset is split into only two sets, the model is built just one time on the training set and executed faster.



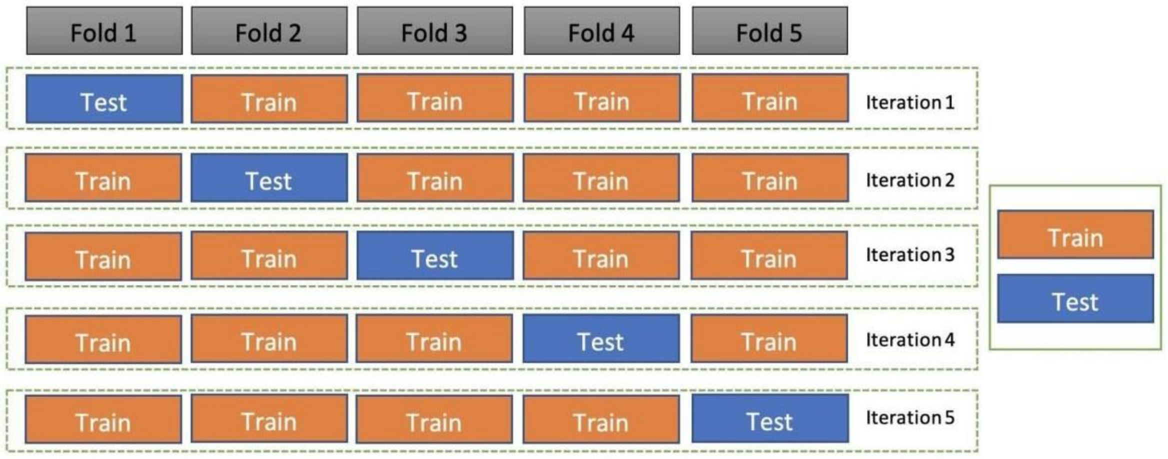
In the image above, the dataset is split into a training set and a test set. You can train the model on the training set and test it on the testing dataset. However, if you want to hyper-tune your parameters or want to select the best model, you can make a validation set like the one below.


DATASET
Training Dataset
Validation Dataset
Testing Dataset
TRAIN
VALIDATION
TEST
Train multiple Models
(e.g. Logistic Regression, Decision Trees, KNN)
Validate Models
Tune Hyper parameters and Select the Best Model (e.g. Logistic Regression)
Evaluate Model
Evaluate the model based on various metrics
(e.g. Confusion Matrix to evaluate the final performance of the selected Logistic Regression Model)

K-fold cross-validation:

In this technique, the whole dataset is partitioned in k parts of equal size and each partition is called a fold. It's known as k-fold since there are k parts where k can be any integer - 3,4,5, etc.

One-fold is used for validation and other K-1 folds are used for training the model. To use every fold as a validation set and other left-outs as a training set, this technique is repeated k times until each fold is used once.

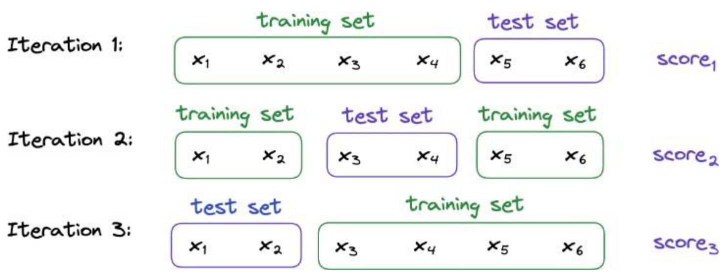

Fold 1
Fold 2
Fold 3
Fold 4
Fold 5
Test Train Train Train Train Iteration 1
Train Test Train Train Train Iteration 2
Train Train Test Train Train Iteration 3
Train Train Train Test Train Iteration 4
Train Train Train Train Test Iteration 5
Train
Test

The image above shows 5 folds and hence, 5 iterations. In each iteration, one fold is the test set/validation set and the other k-1 sets (4 sets) are the train set. To get the final accuracy, you need to take the accuracy of the k-models validation data.

For example, let’s suppose that we have a dataset S = {x1, x2, x3, x4, x5, x6} containing 6 samples and that we want to perform a 3-fold cross-validation.

First, we divide S into 3 subsets randomly. For instance:

S1 = {x1, x2} S2 = {x3, x4} S3 = {x5, x6}
Then, we train and evaluate our machine-learning model 3 times. Each time, two subsets form the training set, while the remaining one acts as the test set.

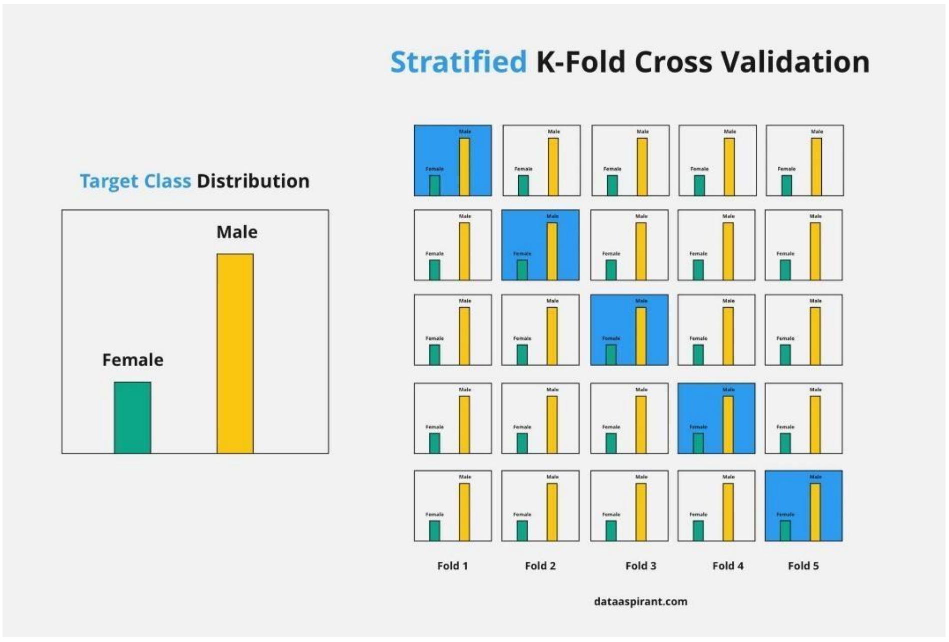


$$\text{overall score} = \frac{score_1 + score_2 + score_3}{3}$$

This validation technique is not considered suitable for imbalanced datasets as the model will not get trained properly owing to the proper ratio of each class's data.

Stratified k-fold cross-validation:

k-fold validation can’t be used for imbalanced datasets because data is split into k-folds with a uniform probability distribution. Stratified k-fold, which is an enhanced version of the k- fold cross-validation technique. Stratification is the process of rearranging the data so as to ensure that each fold is a good representative of the whole. Although it too splits the dataset into k equal folds, each fold has the same ratio of instances of target variables that are in the complete dataset. This enables it to work perfectly for imbalanced datasets, but not for time- series data.

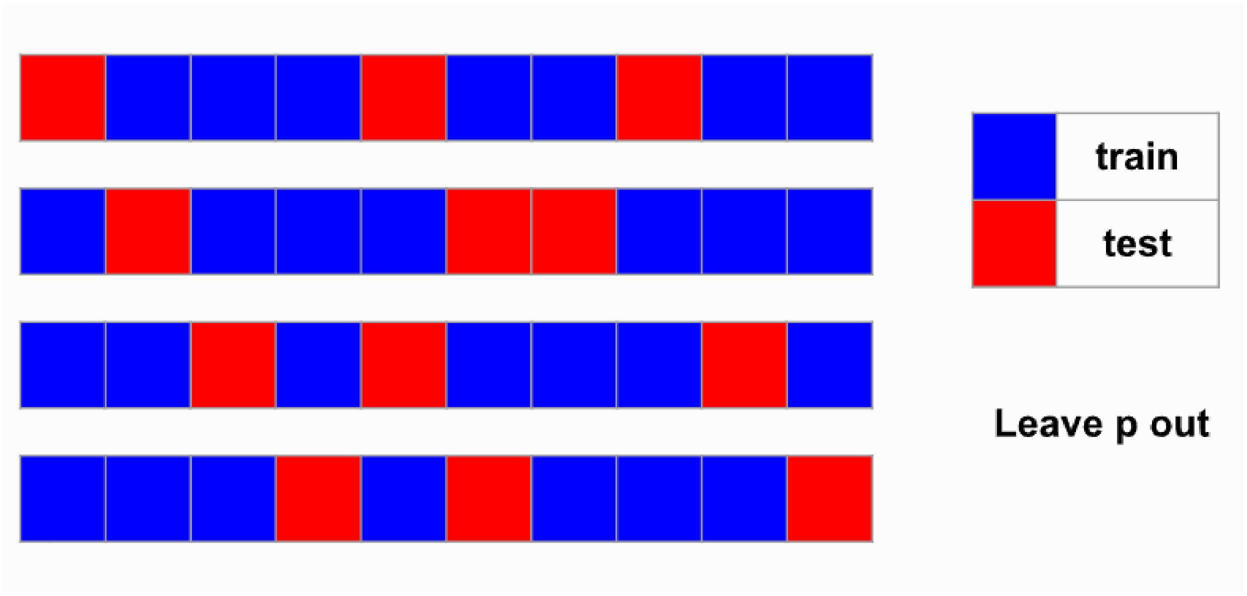


In the example above, the original dataset contains females that are a lot less than males, so this target variable distribution is imbalanced. In the stratified k-fold cross-validation technique, this ratio of instances of the target variable is maintained in all the folds.

Leave-p-out cross-validation:

An exhaustive cross-validation technique, p samples are used as the validation set and n-p samples are used as the training set if a dataset has n samples. The process is repeated until the entire dataset containing n samples gets divided on the validation set of p samples and the training set of n-p samples. This continues till all samples are used as a validation set.

The technique, which has a high computation time, produces good results. However, it's not considered ideal for an imbalanced dataset and is deemed to be a computationally unfeasible method. This is because if the training set has all samples of one class, the model will not be able to properly generalize and will become biased to either of the classes.

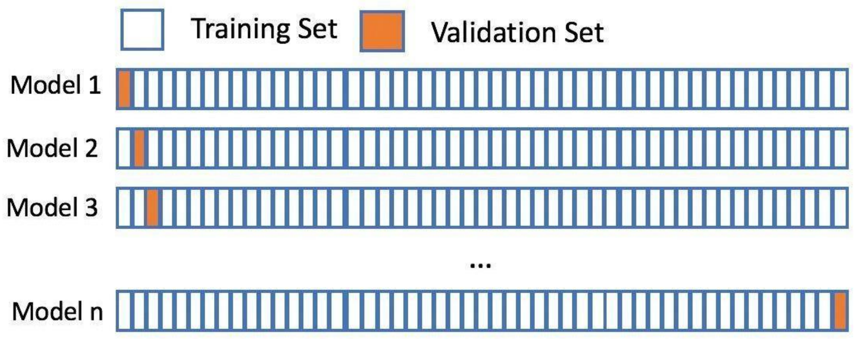


Leave-one-out cross-validation:

In this technique, only 1 sample point is used as a validation set and the remaining n-1 samples are used in the training set. Think of it as a more specific case of the leave-p-out cross-validation technique with P=1.

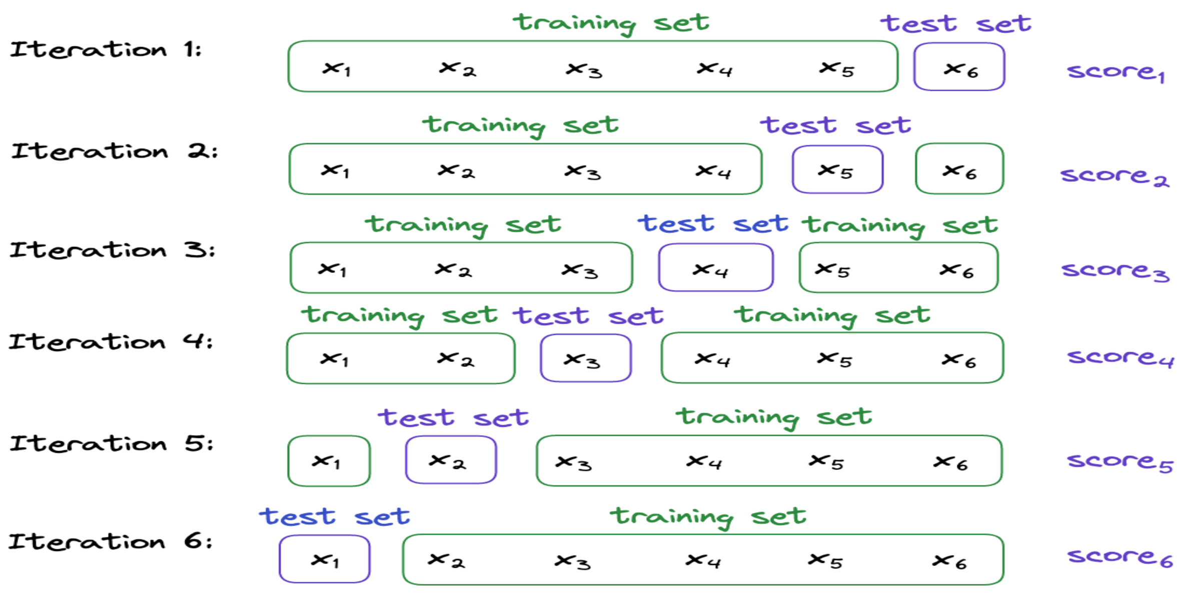
To understand this better, consider this example:

There are 1000 instances in your dataset. In each iteration, 1 instance will be used for the validation set and the remaining 999 instances will be used as the training set. The process repeats itself until every instance from the dataset is used as a validation sample.



In the leave-one-out (LOO) cross-validation, we train our machine-learning model n times where n is to our dataset's size. Each time, only one sample is used as a test set while the rest are used to train our model.
We'll show that LOO is an extreme case of k-fold where {k=n}. If we apply LOO to the previous example, we'll have 6 test subsets:
S1 = {x1}
S2 = {x2}
S3 = {x3}
S4 = {x4}
S5 = {x5}
S6 = {x6}

Iterating over them, we use S \ Si as the training data in iteration i=1,2… 6, and evaluate the model on Si:



The final performance estimate is the average of the six individual scores:

$$\text{overall score} = \frac{score_1 + score_2 + score_3 + score_4 + score_5 + score_6}{6}$$

The leave-one-out cross-validation method is computationally expensive to perform and shouldn't be used with very large datasets. The good news is that the technique is very simple and requires no configuration to specify. It also provides a reliable and unbiased estimate for your model performance.

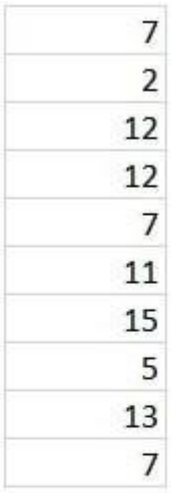
Bootstrapping:

● Sampling: With respect to statistics, sampling is the process of selecting a subset of items from a vast collection of items (population) to estimate a certain characteristic of the entire population

●Sampling with replacement: It means a data point in a drawn sample can reappear in future drawn samples as well

●Parameter estimation: It is a method of estimating parameters for the population using samples. A parameter is a measurable characteristic associated with a population. For example, the average height of residents in a city, the count of red blood cells, etc.

For example, if you were interested in a confidence interval of your population mean, the Bootstrap would give you a great estimation.
Let's say we have this population with 70 values in it, and we want to estimate the mean. Hint: The mean of the population is 7.264
Now imagine that we only had a sample of 10 random digits.



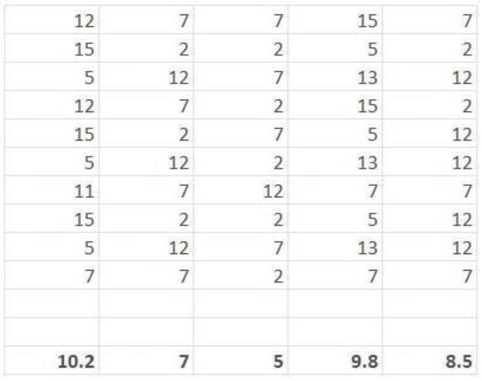
7
2
12
12
7
11
15
5
13
7

If we take the mean of just this set, we get 9.1

We can quickly see that this isn't a great representation of our population mean. That is where the idea around the Bootstrap comes in!

Now, let's take 5 Random Samples of our current set with a length of N (where N is the original length of our data) with replacement (the same value can be seen multiple times)

Those numbers can be seen here, with their respective means underneath.

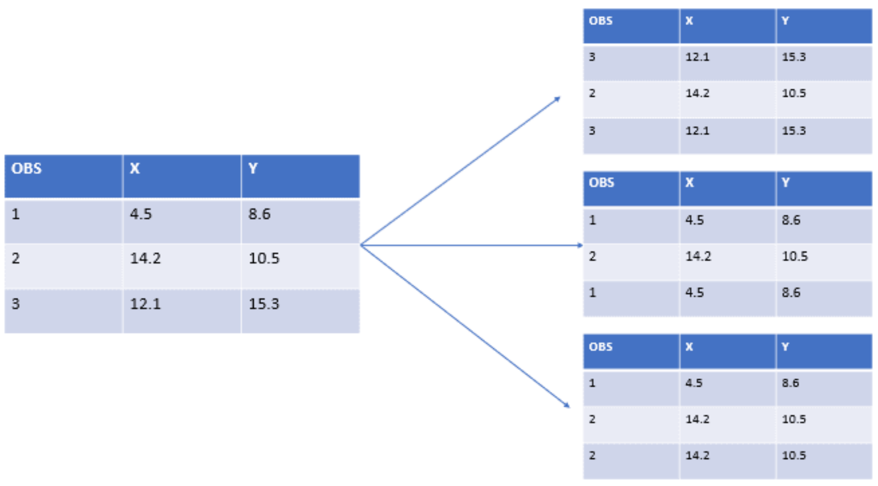


12	7	7	15	7
15	2	2	5	2
5	12	7	13	12
12	7	2	15	2
15	2	7	5	12
5	12	2	13	12
11	7	12	7	7
15	2	2	5	12
5	12	7	13	12
7	7	2	7	7
10.2	**7**	**5**	**9.8**	**8.5**

Now, if we take the mean of our 5 bootstrap samples, we have 8.1.

With only 5 bootstrapped samples, we've gotten much closer to correctly estimating our population mean.

Bootstrap sampling is a type of resampling where we create N datasets from our population (your dataset) with replacement. Each bootstrap data set is the same size as our original dataset. As a result, some observations may appear more than once in each bootstrap data set – and some not at all.
Here's an image that illustrates this idea, with n=3 observations



OBS	X	Y
1	4.5	8.6
2	14.2	10.5
3	12.1	15.3

OBS	X	Y
3	12.1	15.3
2	14.2	10.5
3	12.1	15.3

OBS	X	Y
1	4.5	8.6
2	14.2	10.5
1	4.5	8.6

OBS	X	Y
1	4.5	8.6
2	14.2	10.5
2	14.2	10.5

Advantages of bootstrapping in machine learning:

- Improve Model Real-World Accuracy

Since we will create a lot more data, bootstrapping will allow our model to generalize to the underlying population. We can take a dataset with extremely high variance due to our low number of data points and perform our sampling. This will increase our dataset's bias and training size, leading to a model that can converge and provide accurate insights.
Bootstrapping reduces the variance of your estimates, leading to more accurate predictions and models with higher production success.

- Increase Training Data Size

Sometimes, we're given datasets with insufficient data. This is one of the main advantages of bootstrap sampling.

Disadvantages of bootstrapping in machine learning:

● Independent Population:
When sampling with replacement, there is an underlying assumption that your data points are independent. Some data, like time series data, violates this, and the traditional bootstrap sampling method would not work.

● Computational Limits:
Since bootstrap sampling will create N new datasets, it is sometimes impossible to fit them into RAM. If our original dataset were 5,000 rows of data, our bootstrap sample would create 5,000 new datasets. While this parameter can be lowered, there is a computational and memory cap on bootstrap sampling that many run into.