Vladimir Lyashenko | 10 min | 30th August, 2023

ML Model Development

In machine learning (ML), generalization usually refers to the ability of an algorithm to be effective across various inputs. It means that the **ML** model does not encounter performance degradation on the new inputs from the same distribution of the training data.

For human beings generalization is the most natural thing possible. We can classify on the fly. For example, we would definitely recognize a dog even if we didn't see this breed before. Nevertheless, it might be quite a challenge for an ML model. That's why checking the algorithm's ability to generalize is an important task that requires a lot of attention when building the model.

To do that, we use **Cross-Validation** (**CV**).

In this article we will cover:

- What is Cross-Validation: definition, purpose of use and techniques
- Different CV techniques: hold-out, k-folds, Leave-one-out, Leave-p-out, Stratified k-folds, Repeated k-folds, Nested k-folds, Time Series CV
- How to use these techniques: sklearn
- Cross-Validation in Machine Learning: sklearn, CatBoost
- Cross-Validation in Deep Learning: Keras, PyTorch, MxNet
- Best practices and tips: time series, medical and financial data, images

What is cross-validation?

Cross-validation is a technique for evaluating a machine learning model and testing its performance. CV is commonly used in applied ML tasks. It helps to compare and select an appropriate model for the specific predictive modeling problem.

CV is easy to understand, easy to implement, and it tends to have a lower bias than other methods used to count the model's efficiency scores. All this makes cross-validation a powerful tool for selecting the best model for the specific task.

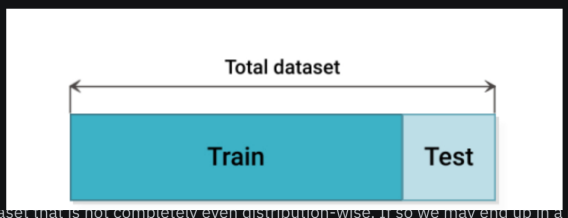
There are a lot of different techniques that may be used to **cross-validate** a model. Still, all of them have a similar algorithm:

Hold-out cross-validation is the simplest and most common technique. You might not know that it is a **hold-out** method but you certainly use it every day.

The algorithm of hold-out technique:

1. Divide the dataset into two parts: the training set and the test set. Usually, 80% of the dataset goes to the training set and 20% to the test set but you may choose any splitting that suits you better
2. Train the model on the training set
3. Validate on the test set
4. Save the result of the validation

2. Train the model on the training set
3. Validate on the test set
4. Save the result of the validation



For example, a dataset that is not completely even distribution-wise. If so we may end up in a rough spot after the split. For example, the training set will not represent the test set. Both training and test sets may differ a lot, one of them might be easier or harder.

Moreover, the fact that we test our model only once might be a bottleneck for this method. Due to the reasons mentioned before, the result obtained by the hold-out technique may be considered inaccurate.

k-Fold cross-validation

k-Fold cross-validation is a technique that minimizes the disadvantages of the hold-out method. k-Fold introduces a new way of splitting the dataset which helps to overcome the “test only once bottleneck”.

The algorithm of the k-Fold technique:

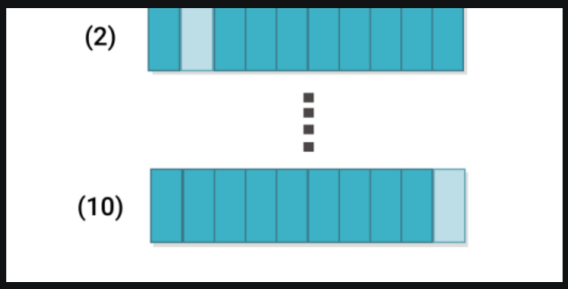
1. Pick a number of folds – k. Usually, k is 5 or 10 but you can choose any number which is less than the dataset’s length.
2. Split the dataset into k equal (if possible) parts (they are called folds)
3. Choose k – 1 folds as the training set. The remaining fold will be the test set
4. Train the model on the training set. On each iteration of cross-validation, you must train a new model

mentioned before, the result obtained by the hold-out technique may be considered inaccurate.

k-Fold cross-validation

k-Fold cross-validation is a technique that minimizes the disadvantages of the hold-out method. k-Fold introduces a new way of splitting the dataset which helps to overcome the “test only once bottleneck”.

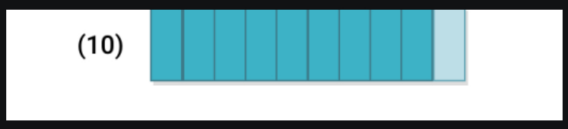
The algorithm of the k-Fold technique:



To perform k-Fold cross-validation you can use sklearn.model_selection.KFold.

```
import numpy as np
from sklearn.model_selection import KFold

X = np.array([[1, 2], [3, 4], [1, 2], [3, 4]])
y = np.array([1, 2, 3, 4])
```

```
y = np.array([1, 2, 3, 4])
kf = KFold(n_splits=2)
```



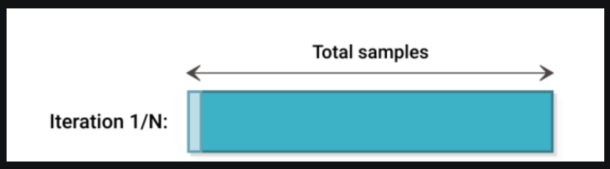
To perform k-Fold cross-validation you can use sklearn.model_selection.KFold.

```
import numpy as np
from sklearn.model_selection import KFold

X = np.array([[1, 2], [3, 4], [1, 2], [3, 4]])
y = np.array([1, 2, 3, 4])
kf = KFold(n_splits=2)
```

the number of samples in the dataset. Such **k-Fold** case is equivalent to **Leave-one-out** technique.

The algorithm of LOOCV technique:

1. Choose one sample from the dataset which will be the test set
2. The remaining n – 1 samples will be the training set
3. Train the model on the training set. On each iteration, a new model must be trained
4. Validate on the test set
5. Save the result of the validation
6. Repeat steps 1 – 5 n times as for n samples we have n different training and test sets
7. To get the final score average the results that you got on step 5.



3. Train the model on the training set. On each iteration, a new model must be trained
4. Validate on the test set
5. Save the result of the validation
6. Repeat steps 1 – 5 n times as for n samples we have n different training and test sets
7. To get the final score average the results that you got on step 5.


```
X = np.array([[1, 2], [3, 4]])
y = np.array([1, 2])
loo = LeaveOneOut()

for train_index, test_index in loo.split(X):
    print("TRAIN:", train_index, "TEST:", test_index)
    X_train, X_test = X[train_index], X[test_index]
    y_train, y_test = y[train_index], y[test_index]
```

The greatest advantage of Leave-one-out cross-validation is that it doesn't waste much data. We use only one sample from the whole dataset as a test set, whereas the rest is the training set. But when compared with k-Fold CV, LOOCV requires building n models instead of k models, when we know that n which stands for the number of samples in the dataset is much higher than k. It means LOOCV is more computationally expensive than k-Fold, it may take plenty of time to cross-validate the model using LOOCV.

Thus, the Data Science community has a general rule based on empirical evidence and different researches,

which suggests that 5- or 10-fold cross-validation should be preferred over LOOCV.

Leave-p-out cross-validation

```
    y_train, y_test = y[train_index], y[test_index]
```

The greatest advantage of Leave-one-out cross-validation is that it doesn't waste much data. We use only one sample from the whole dataset as a test set, whereas the rest is the training set. But when compared with k-Fold CV, LOOCV requires building n models instead of k models, when we know that n which stands for the number of samples in the dataset is much higher than k. It means LOOCV is more computationally expensive than k-Fold, it may take plenty of time to cross-validate the model using LOOCV.

Thus, the Data Science community has a general rule based on empirical evidence and different researches, which suggests that 5- or 10-fold cross-validation should be preferred over LOOCV.

Leave-p-out cross-validation

```
import numpy as np
from sklearn.model_selection import LeavePOut

X = np.array([[1, 2], [3, 4], [5, 6], [7, 8]])
y = np.array([1, 2, 3, 4])
lpo = LeavePOut(2)

for train_index, test_index in lpo.split(X):
    print("TRAIN:", train_index, "TEST:", test_index)
    X_train, X_test = X[train_index], X[test_index]
    y_train, y_test = y[train_index], y[test_index]
```

LpOC has all the disadvantages of the LOOCV, but, nevertheless, it's as robust as LOOCV.

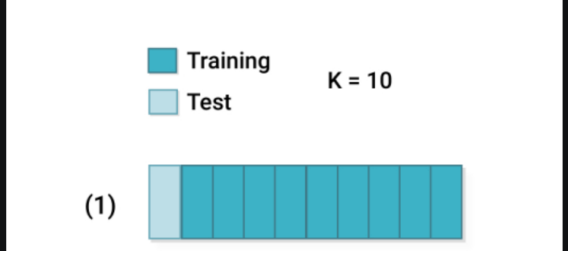
Stratified k-Fold cross-validation

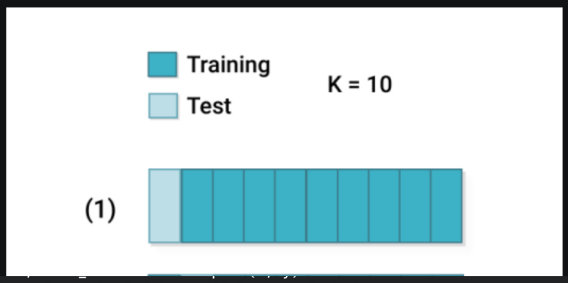
```
X = np.array([[1, 2], [3, 4], [5, 6], [7, 8]])
y = np.array([1, 2, 3, 4])
lpo = LeavePOut(2)

for train_index, test_index in lpo.split(X):
    print("TRAIN:", train_index, "TEST:", test_index)
    X_train, X_test = X[train_index], X[test_index]
    y_train, y_test = y[train_index], y[test_index]
```

LpOC has all the disadvantages of the LOOCV, but, nevertheless, it's as robust as LOOCV.

Stratified k-Fold cross-validation

6. Save the result of the validation.
7. Repeat steps 3 – 6 k times. Each time use the remaining fold as the test set. In the end, you should have validated the model on every fold that you have.
8. To get the final score average the results that you got on step 6.






```
print("TRAIN:", train_index, "TEST:", test_index)
X_train, X_test = X[train_index], X[test_index]
y_train, y_test = y[train_index], y[test_index]
```

All mentioned above about k-Fold CV is true for Stratified k-Fold technique. When choosing between different CV methods, make sure you are using the proper one. For example, you might think that your model performs badly simply because you are using k-Fold CV to validate the model which was trained on the dataset with a class imbalance. To avoid that you should always do a proper exploratory data analysis on your data.

Repeated k-Fold cross-validation

Repeated k-Fold cross-validation or **Repeated random sub-sampling CV** is probably the most robust of all **CV** techniques in this paper. It is a variation of **k-Fold** but in the case of **Repeated k-Folds k** is not the number of folds. It is the number of times we will train the model.

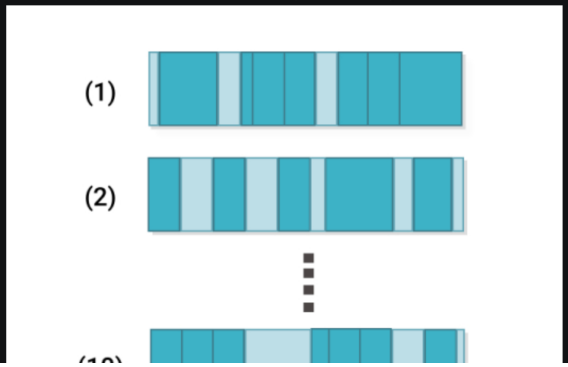
The general idea is that on every iteration we will randomly select samples all over the dataset as our test set. For example, if we decide that 20% of the dataset will be our test set, 20% of samples will be randomly selected

methods, make sure you are using the proper one. For example, you might think that your model performs badly simply because you are using k-Fold CV to validate the model which was trained on the dataset with a class imbalance. To avoid that you should always do a proper exploratory data analysis on your data.

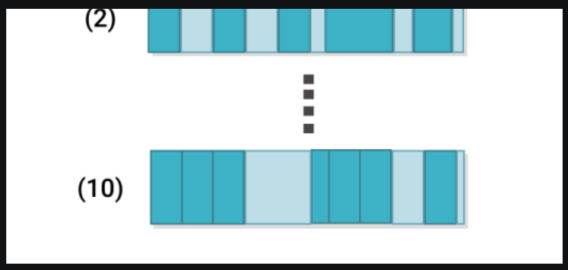
Repeated k-Fold cross-validation

Repeated k-Fold cross-validation or **Repeated random sub-sampling CV** is probably the most robust of all **CV** techniques in this paper. It is a variation of **k-Fold** but in the case of **Repeated k-Folds k** is not the number of folds. It is the number of times we will train the model.

The general idea is that on every iteration we will randomly select samples all over the dataset as our test set.



Repeated k-Fold has clear advantages over standard k-Fold CV. Firstly, the proportion of train/test split is not dependent on the number of iterations. Secondly, we can even set unique proportions for every iteration. Thirdly

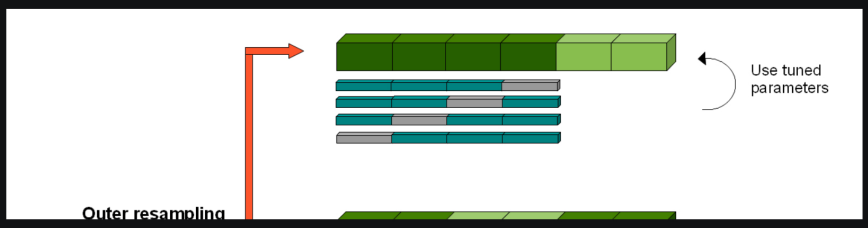


Repeated k-Fold has clear advantages over standard k-Fold CV. Firstly, the proportion of train/test split is not

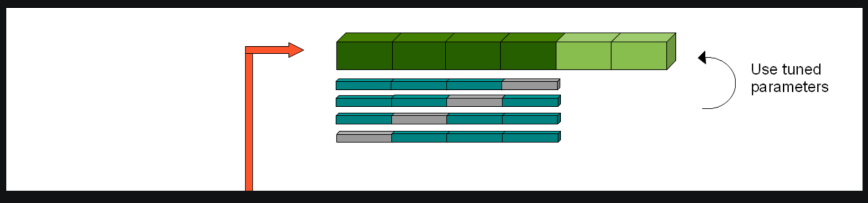
```
        x_train, x_test = x[train_index], x[test_index]
        y_train, y_test = y[train_index], y[test_index]
```

Nested k-Fold

Unlike the other CV techniques, which are designed to evaluate the quality of an algorithm, **Nested k-fold CV** is used to train a model in which hyperparameters also need to be optimized. It estimates the generalization error of the underlying model and its (hyper)parameter search.



Unlike the other CV techniques, which are designed to evaluate the quality of an algorithm, **Nested k-fold CV** is used to train a model in which hyperparameters also need to be optimized. It estimates the generalization error of the underlying model and its (hyper)parameter search.



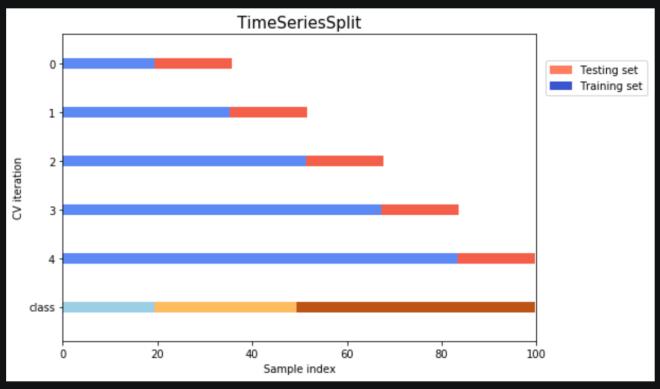
1. Define set of hyper-parameter combinations, C, for current model. If model has no hyper-parameters, C is the empty set.
2. Divide data into K folds with approximately equal distribution of cases and controls.
3. (outer loop) For fold k, in the K folds:
    - Set fold k, as the test set.
    - Perform automated feature selection on the remaining K-1 folds.
    - For parameter combination c in C:
        - (inner loop) For fold k, in the remaining K-1 folds:
            - Set fold k, as the validation set.
            - Train model on remaining K-2 folds.
            - Evaluate model performance on fold k

- Evaluate model performance on fold k.
- Calculate average performance over K-2 folds for parameter combination c.
- Train model on K-1 folds using hyper-parameter combination that yielded best average performance over all steps of the inner loop.
- Evaluate model performance on fold k.

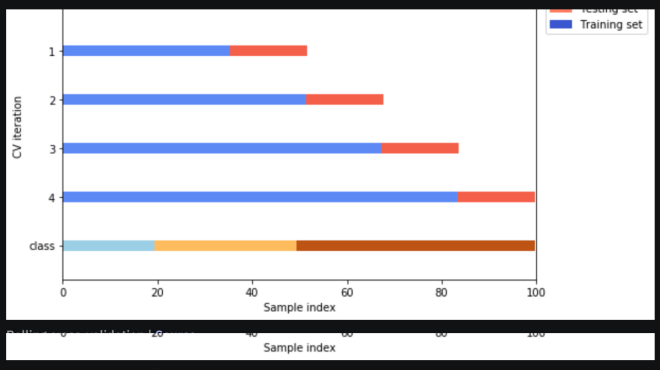
4. Calculate average performance over K folds.

- Perform automated feature selection on the remaining K-1 folds.
- For parameter combination c in C:
  - (inner loop) For fold k, in the remaining K-1 folds:
    - Set fold k, as the validation set.
    - Train model on remaining K-2 folds.
    - Evaluate model performance on fold k.
  - Calculate average performance over K-2 folds for parameter combination c.
- Train model on K-1 folds using hyper-parameter combination that yielded best average performance over all steps of the inner loop.
- Evaluate model performance on fold k.

Cross-validation is done on a rolling basis i.e. starting with a small subset of data for training purposes, predicting the future values, and then checking the accuracy on the forecasted data points. The following image can help you get the intuition behind this approach.



Rolling cross-validation | Source



Blocked cross-validation | Source

It works by adding margins at two positions. The first is between the training and validation folds in order to prevent the model from observing lag values which are used twice, once as a regressor and another as a response. The second is between the folds used at each iteration in order to prevent the model from memorizing patterns from one iteration to the next.

Cross-validation in Machine Learning

When is cross-validation the right choice?

Although doing cross-validation of your trained model can never be termed as a bad choice, **there are certain scenarios in which cross-validation becomes an absolute necessity:**

1. **Limited dataset**

Let's say we have 100 data points and we are dealing with a multi-class classification problem with 10 classes.

response. The second is between the folds used at each iteration in order to prevent the model from memorizing patterns from one iteration to the next.

Cross-validation in Machine Learning

When is cross-validation the right choice?

Although doing cross-validation of your trained model can never be termed as a bad choice, **there are certain scenarios in which cross-validation becomes an absolute necessity:**

1. **Limited dataset**

In the absence of cross-validation, we only get a single value of accuracy or precision or recall which could be an outcome of chance. When we train multiple models, we eliminate such possibilities and get a metric per model which results in robust insights.

4. **Hyperparameter tuning**

Although there are many methods to tune the hyperparameters of your model such as grid search, Bayesian optimization, etc., this exercise can't be done on training or test set, and a need for a validation set arises. Thus, we fall back to the same splitting problem that we have discussed above and cross-validation can help us out of this.

> **May be useful**
>
> How to Manage, Track, and Visualize Hyperparameters of Machine Learning Models?

Cross-validation in Deep Learning

Although there are many methods to tune the hyperparameters of your model such as grid search, Bayesian optimization, etc., this exercise can't be done on training or test set, and a need for a validation set arises. Thus, we fall back to the same splitting problem that we have discussed above and cross-validation can help us out of this.

> **May be useful**
>
> How to Manage, Track, and Visualize Hyperparameters of Machine Learning Models?

Cross-validation in Deep Learning

3. Testing – a part of the dataset for final validation of the model

Still, you can use cross-validation in DL tasks if the dataset is tiny (contains hundreds of samples). In this case, learning a complex model might be an irrelevant task so make sure that you don't complicate the task further.

Best practices and tips

It's worth mentioning that sometimes performing cross-validation might be a little tricky.

For example, it's quite easy to make a logical mistake when splitting the dataset which may lead to an untrustworthy CV result.

You may find some tips that you need to keep in mind when cross-validating a model below:

1. Be logical when splitting the data (does the splitting method make sense)
2. Use the proper CV method (is this method viable for my use-case)
3. When working with time series don't validate on the past (see the first tip)
4. When working with medical or financial data remember to split by person. Avoid having data for one person

learning a complex model might be an irrelevant task so make sure that you don't complicate the task further.

Best practices and tips

It's worth mentioning that sometimes performing cross-validation might be a little tricky.

For example, it's quite easy to make a logical mistake when splitting the dataset which may lead to an untrustworthy CV result.

You may find some tips that you need to keep in mind when cross-validating a model below:

Hopefully, with this information, you will have no problems setting up the CV for your next machine learning project!

Resources

1. https://www.geeksforgeeks.org/cross-validation-machine-learning/
2. https://machinelearningmastery.com/k-fold-cross-validation/
3. https://towardsdatascience.com/cross-validation-in-machine-learning-72924a69872f
4. https://towardsdatascience.com/why-and-how-to-do-cross-validation-for-machine-learning-d5bd7e60c189
5. https://scikit-learn.org/stable/modules/cross_validation.html

Resources

1. https://www.geeksforgeeks.org/cross-validation-machine-learning/
2. https://machinelearningmastery.com/k-fold-cross-validation/
3. https://towardsdatascience.com/cross-validation-in-machine-learning-72924a69872f
4. https://towardsdatascience.com/why-and-how-to-do-cross-validation-for-machine-learning-d5bd7e60c189
5. https://scikit-learn.org/stable/modules/cross_validation.html

Was the article useful?

Yes No

Manage your model metadata in a single place

Join 30,000+ ML Engineers & Data Scientists using Neptune to easily log, compare, register, and share ML metadata.

Try Neptune for free | Check out the Docs