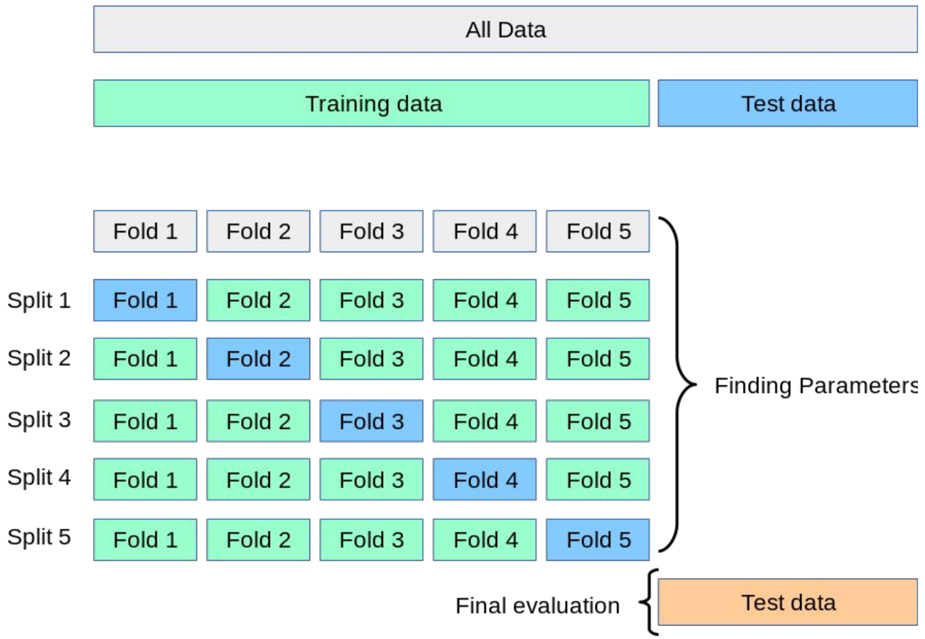
U:Pass Week 8

36106: Machine Learning Algorithms and Application
By Marisara Satrulee

Cross Validation (CV) Method

Divide the dataset into two parts: one for training, other for testing

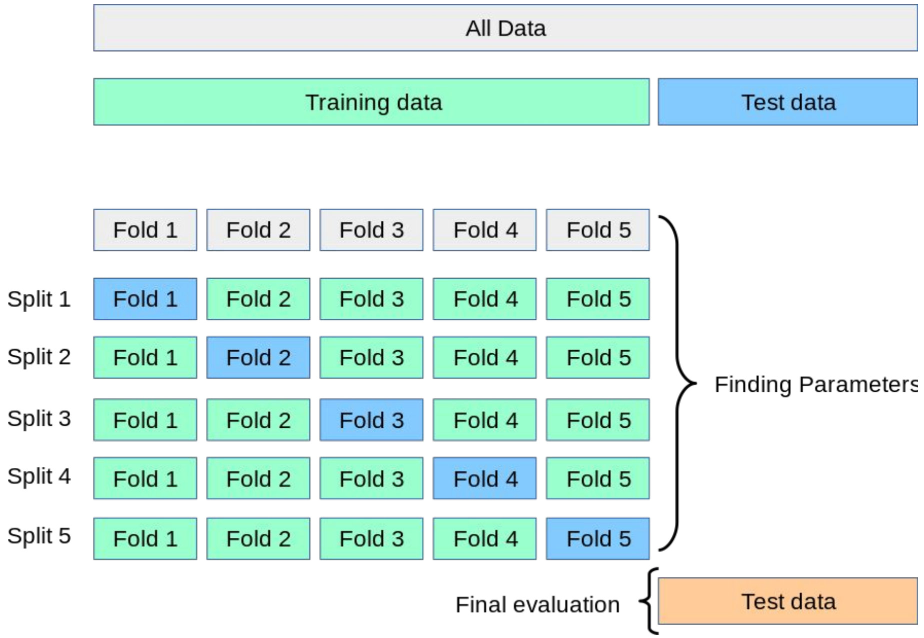
1. Train the model on the training set
2. Validate the model on the test set
3. Repeat 1-2 steps a couple of times. This number depends on the CV method that you are using



Cross Validation (CV) Method

cross-validation methods that will be covered in this article.

- Hold-out
- K-folds
- Leave-one-out
- Leave-p-out
- Stratified K-folds
- Repeated K-folds
- Nested K-folds

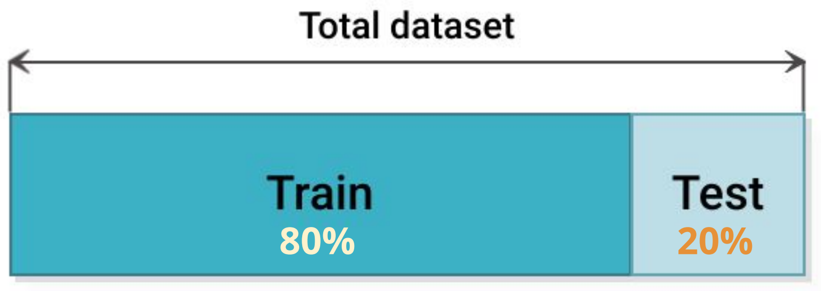


Reference: https://neptune.ai/blog/cross-validation-in-machine-learning-how-to-do-it-right

Hold-out

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

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



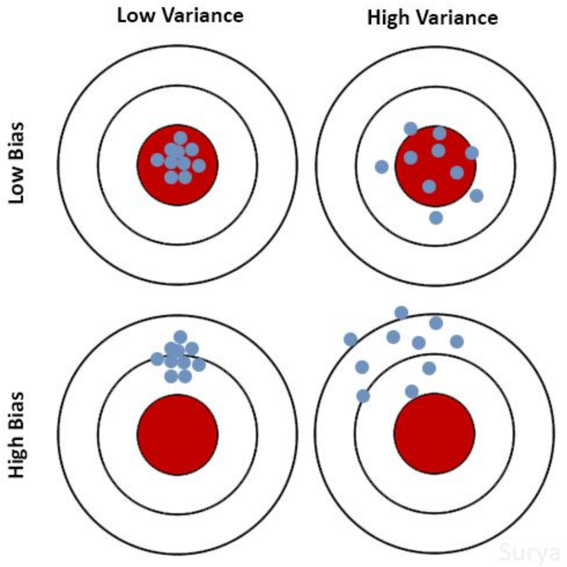
```
import numpy as np
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=111)
```

Hold-out

Disadvantage:

1. For a dataset that is not evenly distributed. The test error rates are highly variable (**high variance**) as it totally depends on which observations end up in the training set and test set
2. Only a part of the data is used to train the model (**high bias**) which is not a very good idea when data is not huge and this will lead to overestimation of test error.

High Bias - High Variance: Predictions are inconsistent and inaccurate on average.

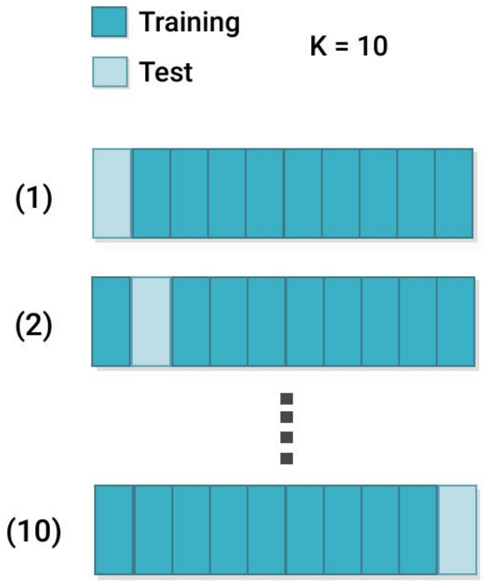


Reference: https://medium.com/analytics-vidhya/difference-between-bias-and-variance-in-machine-learning-fec71880c757

K-folds

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 independently of the model trained on the previous iteration
5. Validate on the test set
6. Save the result of the validation
7. Repeat steps 3 – 6, k times.
8. To get the final score average the results that you got on step 6.



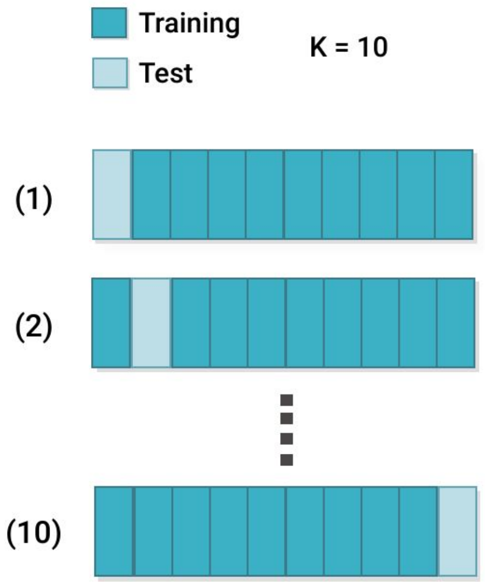
The **mean of errors** from all the iterations is calculated as the CV test error estimate.

$$\mathrm{CV}_{(k)} = \frac{1}{k}\sum_{i=1}^{k} \mathrm{MSE}_i .$$

Reference: https://neptune.ai/blog/cross-validation-in-machine-learning-how-to-do-it-right

K-folds

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

X = [1, 20, 30, 40, 50, 60, 70, 80, 90, 100, 1000]
kf = KFold(n_splits=2)

for train_index, test_index in kf.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]
```



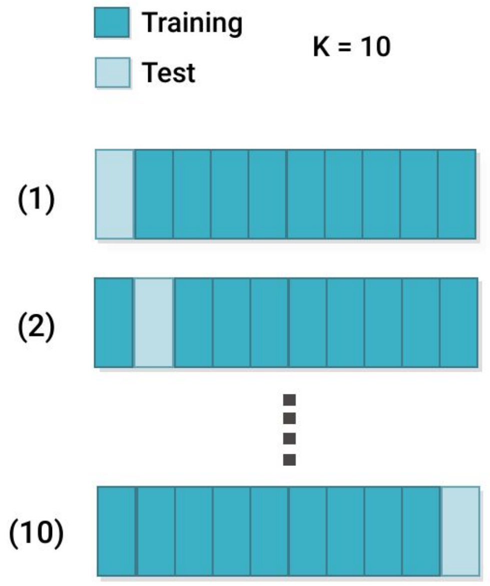
The mean of errors from all the iterations is calculated as the CV test error estimate.

$$\mathrm{CV}_{(k)} = \frac{1}{k}\sum_{i=1}^{k} \mathrm{MSE}_i .$$

Reference: https://neptune.ai/blog/cross-validation-in-machine-learning-how-to-do-it-right

K-folds

In general, it is always better to use k-Fold technique instead of hold-out. However, the the more iteration or **more k** could make the training process **expensive** and **time-consuming**.

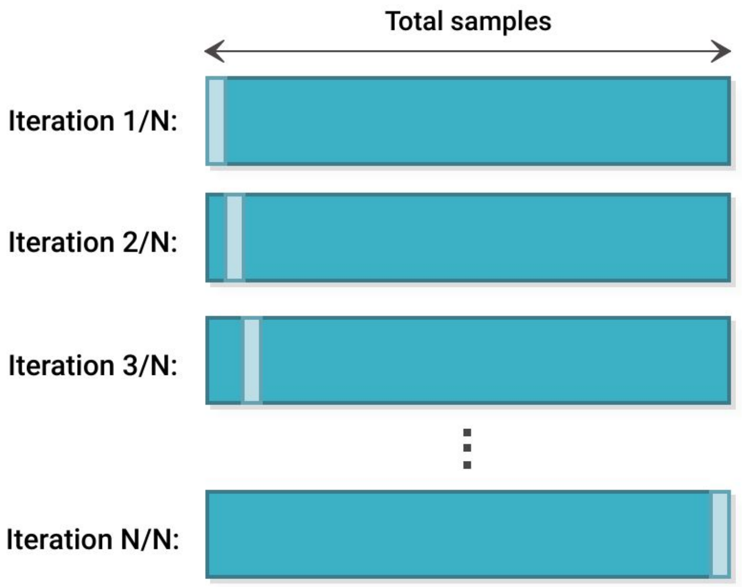


Reference: https://neptune.ai/blog/cross-validation-in-machine-learning-how-to-do-it-right

Leave-one-out (LOOCV)

LOOCV is an extreme case of k-Fold CV, as we select a **single** observation as **test data**, and everything else is training data.

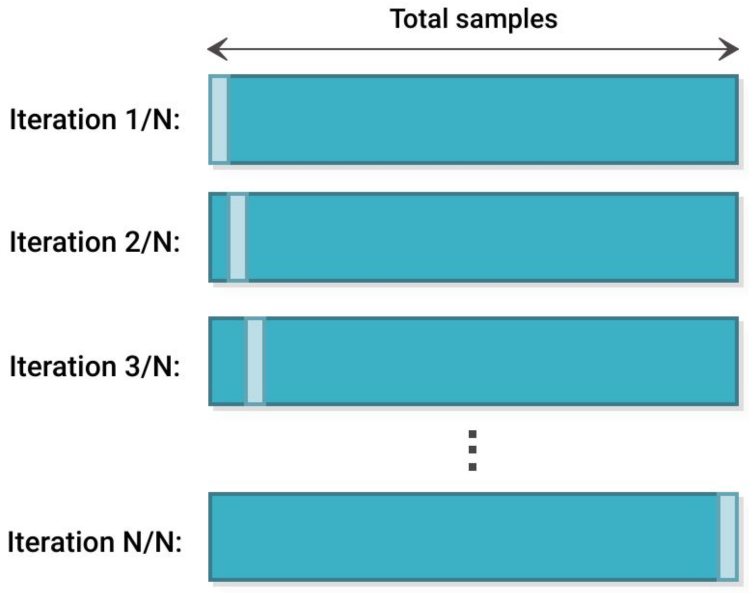
```
import numpy as np
from sklearn.model_selection import LeaveOneOut

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



Reference: https://neptune.ai/blog/cross-validation-in-machine-learning-how-to-do-it-right

Leave-one-out (LOOCV)

What is the disadvantage for this method?

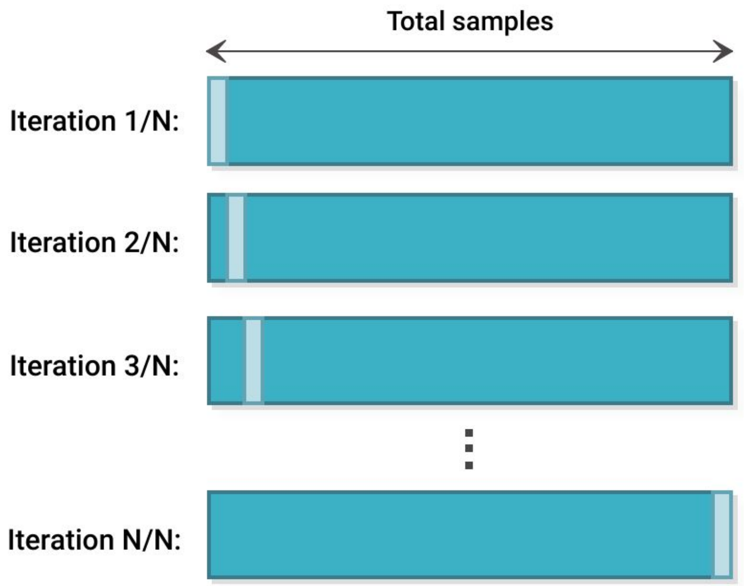


Reference: https://neptune.ai/blog/cross-validation-in-machine-learning-how-to-do-it-right

Leave-p-out (LpOCV)

LpOCV is is similar to Leave-one-out CV as it creates all the possible training and test sets by using p samples as the test set.

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

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



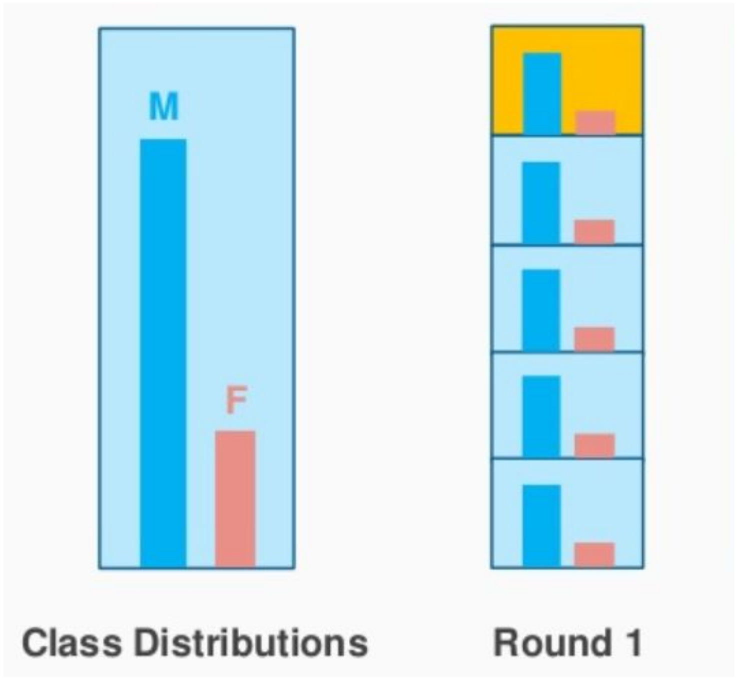
Reference: https://neptune.ai/blog/cross-validation-in-machine-learning-how-to-do-it-right

Stratified K-folds

Let’s say we have an imbalance dataset of 1000 customers, which **70%** is female and **30%** is male

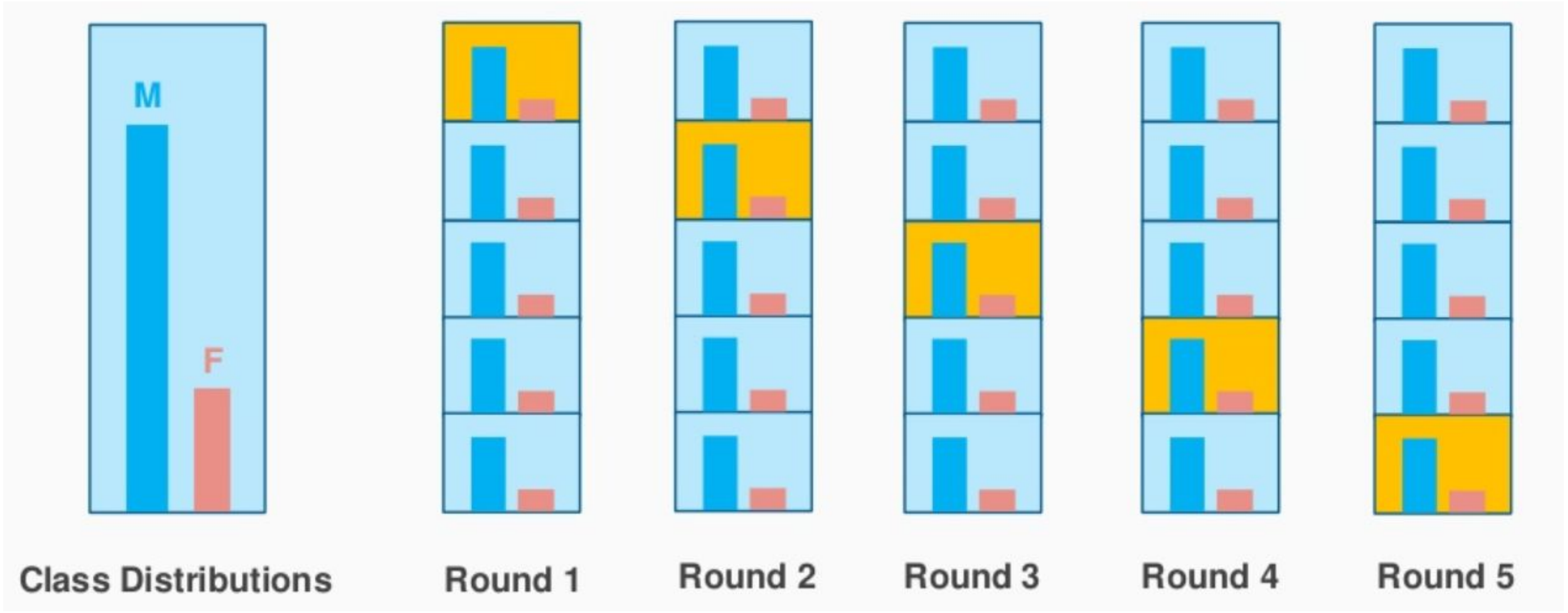
An (**70:30**) stratified split of this dataset will be **490** females and **210** males from the total **700** training data after splitting.

In a similar fashion, 20% of 1000 customers of stratified split will be **210** females and **90** males.



Reference: https://www.analyticsvidhya.com/blog/2021/05/4-ways-to-evaluate-your-machine-learning-model-cross-validation-techniques-with-python-code/

Stratified K-folds

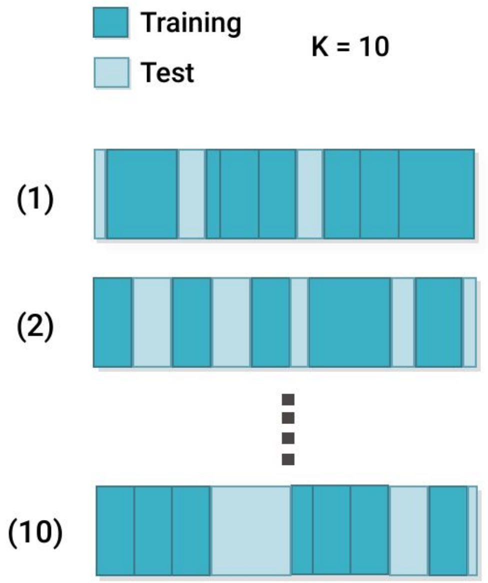


Reference: https://www.analyticsvidhya.com/blog/2021/05/4-ways-to-evaluate-your-machine-learning-model-cross-validation-techniques-with-python-code/

Repeated K-folds

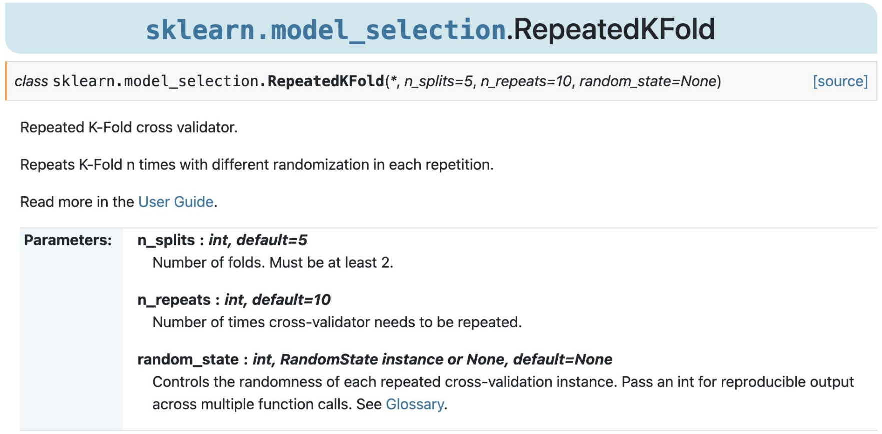
Repeated k-Fold cross-validation or Repeated random sub-sampling CV is probably the **most robust** of all CV

If we decide that 20% of the dataset will be our test set, 20% of samples will be randomly selected and the rest 80% will become the training set.



Reference: https://www.analyticsvidhya.com/blog/2021/05/4-ways-to-evaluate-your-machine-learning-model-cross-validation-techniques-with-python-code/

Repeated K-folds



sklearn.model_selection.RepeatedKFold

class `sklearn.model_selection.RepeatedKFold`(**, n_splits=5, n_repeats=10, random_state=None*) [source]

Repeated K-Fold cross validator.

Repeats K-Fold n times with different randomization in each repetition.

Read more in the User Guide.

Parameters:

n_splits : *int, default=5*
Number of folds. Must be at least 2.

n_repeats : *int, default=10*
Number of times cross-validator needs to be repeated.

random_state : *int, RandomState instance or None, default=None*
Controls the randomness of each repeated cross-validation instance. Pass an int for reproducible output across multiple function calls. See Glossary.

In sklearn, you must have (**n_splits**) and the number of times the split will be performed (**n_repeats**) for each iteration. It guarantees that you will have different folds on each iteration.

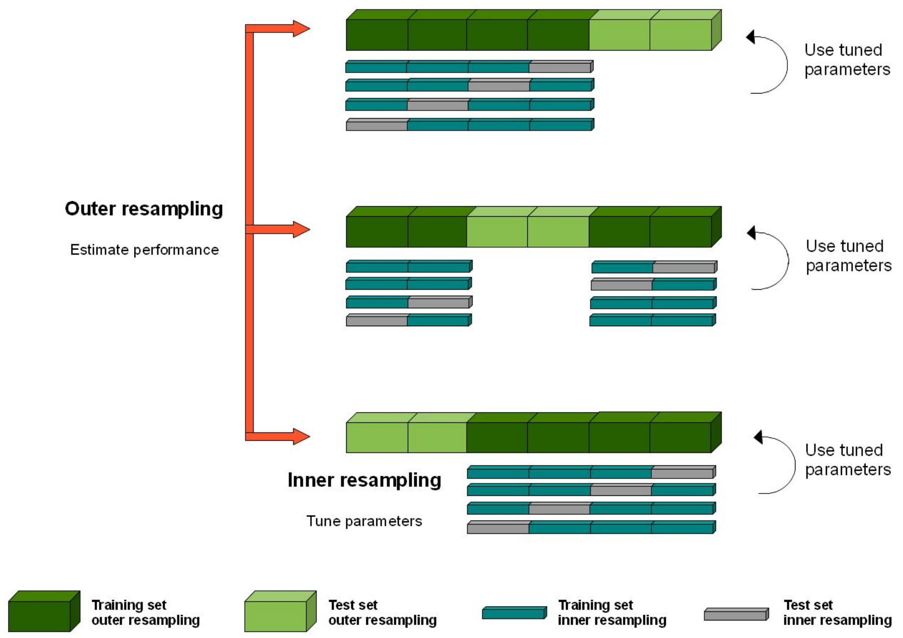
RepeatedStratifiedKFold guarantees different fold on each iteration. For example, 5 repeats (i.e., **n_repeats=5**) of 10-fold cross-validation (**n_splits=10**) were used for estimating the model's performance, it means that 50 different models would need to be fitted (trained) and evaluated

Reference: https://www.analyticsvidhya.com/blog/2021/05/4-ways-to-evaluate-your-machine-learning-model-cross-validation-techniques-with-python-code/

Nested K-folds

The algorithm of the k-Fold technique:

1. **Split Data**: Split the data into training-validation and test sets.
2. **Outer Loop** (Model Evaluation): Perform k-fold cross-validation on the training-validation set, evaluating model performance.
3. **Inner Loop** (Model Selection): Within each outer fold, perform k-fold cross-validation to select the best model or hyperparameters.
4. **Model Evaluation**: Average the performance metrics across all outer folds to estimate model performance.



Use **WHAT** with **WHAT**

Hold-out based cross-validation:

If we have a large amount data then we should use hold-out cross-validation. As it could be expensive for other CV methods.

K-fold cross-validation:

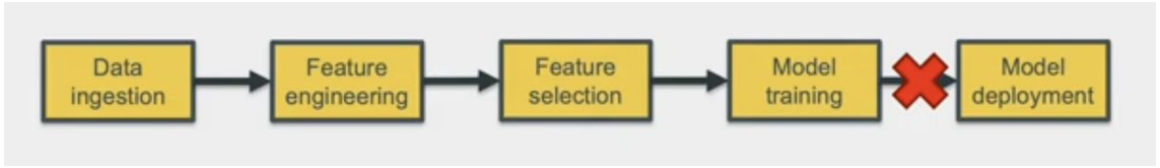
We can use this process with almost all kinds of dataset.

Nested k-fold cross-validation:

Nested CV estimates the generalization error of the underlying model and its (hyper)parameter search. Basically, it is used to test the model before putting it into production.

Stratified k-fold cross-validation:

A skewed dataset for binary classification with 90% positive samples and 10% negative samples. To use stratified k-fold for regression problem,we have to divide target into bins and then we use stratified k-fold for classification.



Reference: https://medium.com/the-rise-of-unbelievable/what-is-cross-validation-and-when-to-use-which-cross-validation-327d25bbb3f3

Mentimeter

https://www.menti.com/alnz1mswbbmh

Model Selection

Based on Type of Data:

- **Images and videos**

Convolutional Neural Network model, if your application mainly focuses on images and videos, for example, image recognition.

- **Text data or speech data**

Recurrent neural networks (RNN) are employed if your problem includes speech or text data.

- **Numerical data**

You may use Support Vector Machine (SVM), logistic **regression**, and decision trees if your data is numerical.

Based on Task:

- **Classification** Tasks

SVM, logistic regression, and decision trees.

- **Regression** Tasks

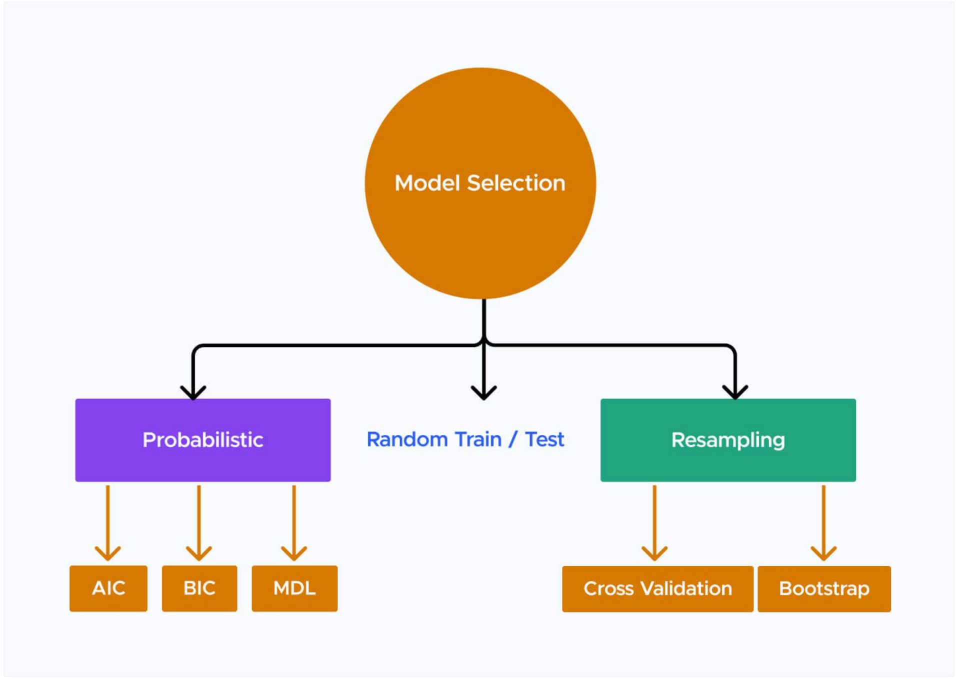
Linear regression, Random Forest, Polynomial regression, etc.

- **Clustering** Tasks

K means clustering, hierarchical clustering.

Reference: https://censius.ai/blogs/machine-learning-model-selection-techniques

Model Selection Technique



- Train-Test Split
- Cross-Validation
- Grid Search
- Random Search
- Bayesian optimization
- Model averaging
- Information Criteria
- Domain Expertise & Prior Knowledge
- Model Performance Comparison

Reference: https://censius.ai/blogs/machine-learning-model-selection-techniques