Cross Validation in Machine learning

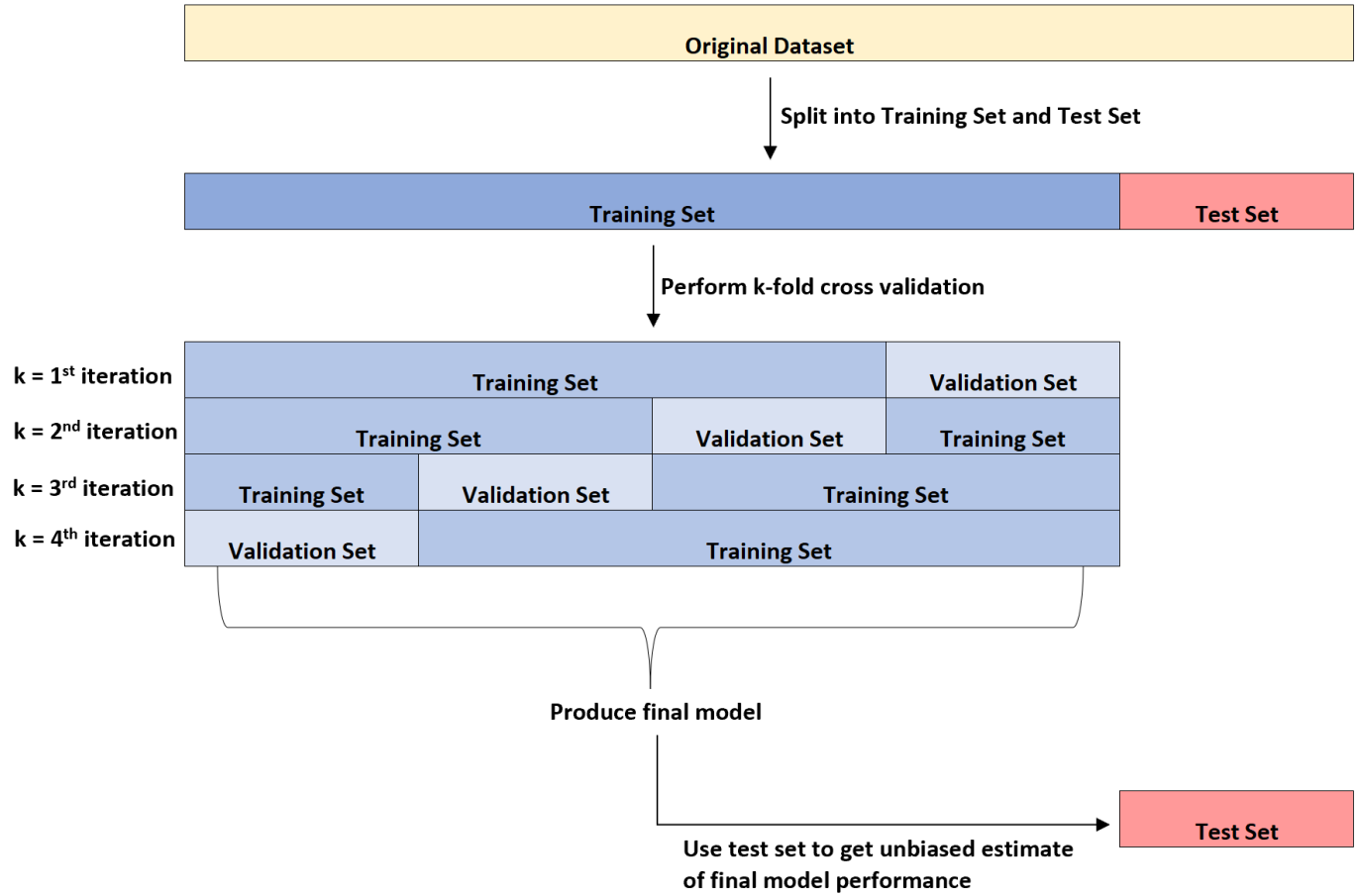
- Cross-validation is a statistical method used to estimate the performance (or accuracy) of machine learning models.
- It is used to protect against overfitting in a predictive model.
- In cross-validation, you make a fixed number of folds (or partitions) of the data, run the analysis on each fold, and then average the overall error estimate.

Cross Validation in Machine learning

- Whenever we fit a machine learning algorithm to a dataset, we typically split the dataset into three parts:

- Training Set: Used to train the model.
- Validation Set: Used to optimize model parameters.
- Test Set: Used to get an unbiased estimate of the final model performance.

Cross Validation in Machine learning



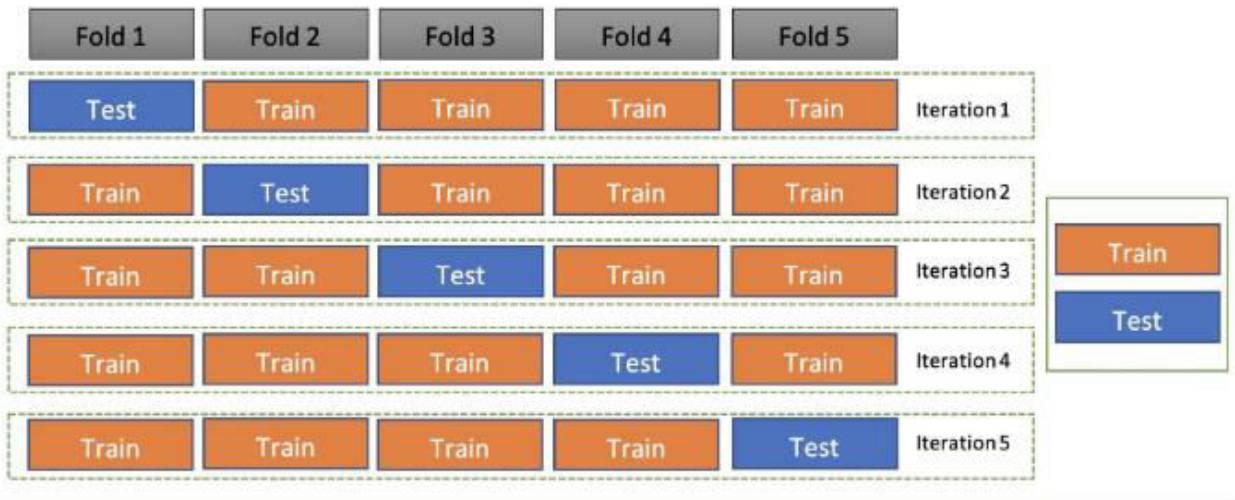
Types of Cross Validation

- K-fold cross-validation
- Stratified k-fold cross-validation
- Leave-p-out cross-validation
- Leave-one-out cross-validation

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.



K-fold cross-validation

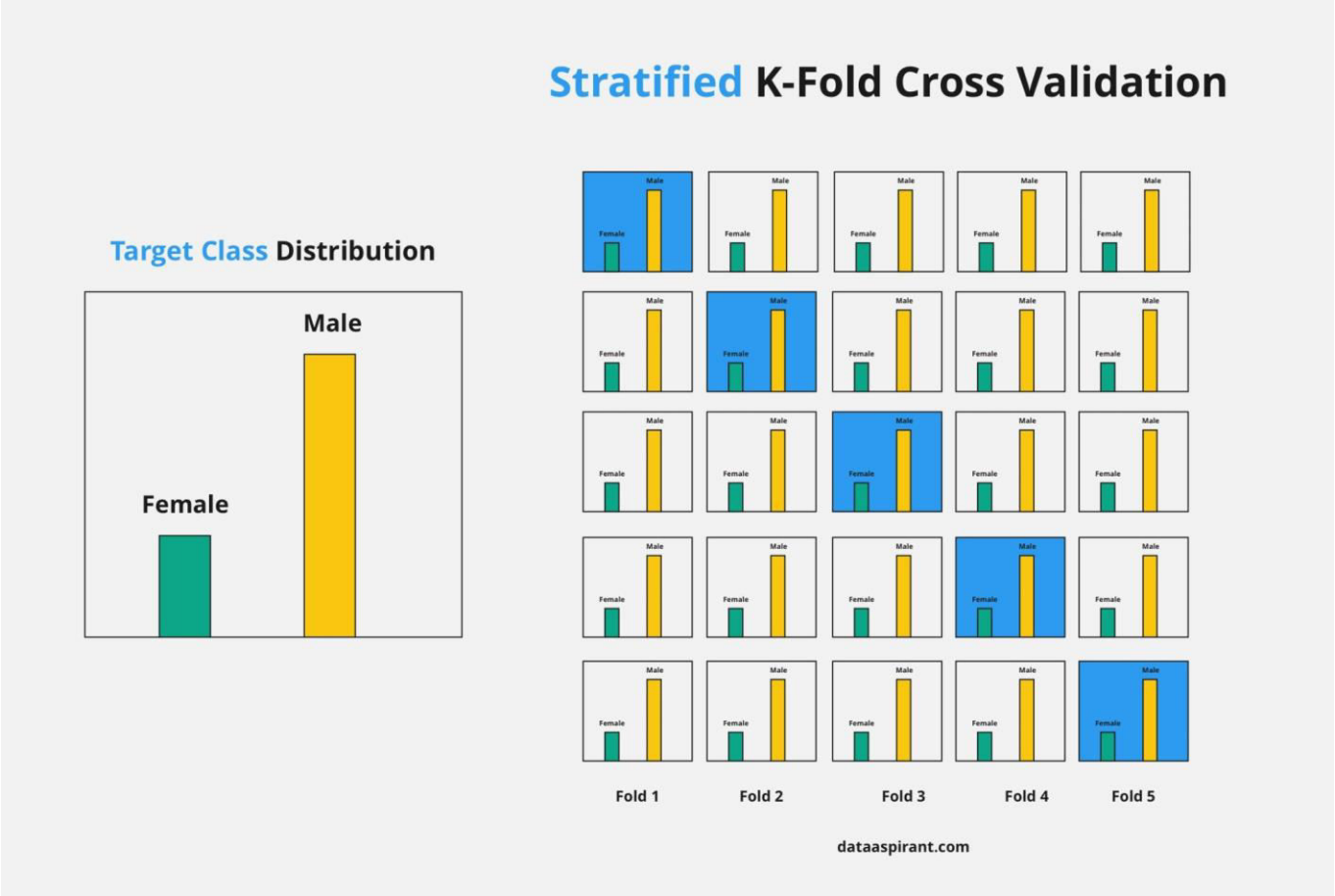
```
from sklearn.datasets import load_iris
from sklearn.model_selection import cross_val_score,KFold
from sklearn.linear_model import LogisticRegression
iris=load_iris()
X=iris.data
Y=iris.target
lr=LogisticRegression()
k_fold=KFold(n_splits=7)
score=cross_val_score(lr,X,Y,cv=k_fold)
print("Cross Validation Scores:{}".format(score))
print("Average Cross Validation score:{}".format(score.mean()))
```

```
Cross Validation Scores:[1.         1.         1.         0.80952381
0.95238095 0.85714286
 0.9047619 ]
Average Cross Validation score:0.9319727891156463
```

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.
- Not so with stratified k-fold, which is an enhanced version of the k-fold cross-validation technique.
- 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.

Stratified k-fold cross-validation



Stratified k-fold cross-validation

```
from sklearn.datasets import load_iris
from sklearn.model_selection import cross_val_score,StratifiedKFold
from sklearn.linear_model import LogisticRegression
iris=load_iris()
X=iris.data
Y=iris.target
lr=LogisticRegression()
st_kf=StratifiedKFold(n_splits=3)
score=cross_val_score(lr,X,Y,cv=st_kf)
print("Cross Validation Scores:{}".format(score))
print("Average Cross Validation score:{}".format(score.mean()))
```

```
Cross Validation Scores:[0.98 0.96 0.98]
Average Cross Validation score:0.9733333333333333
```

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.

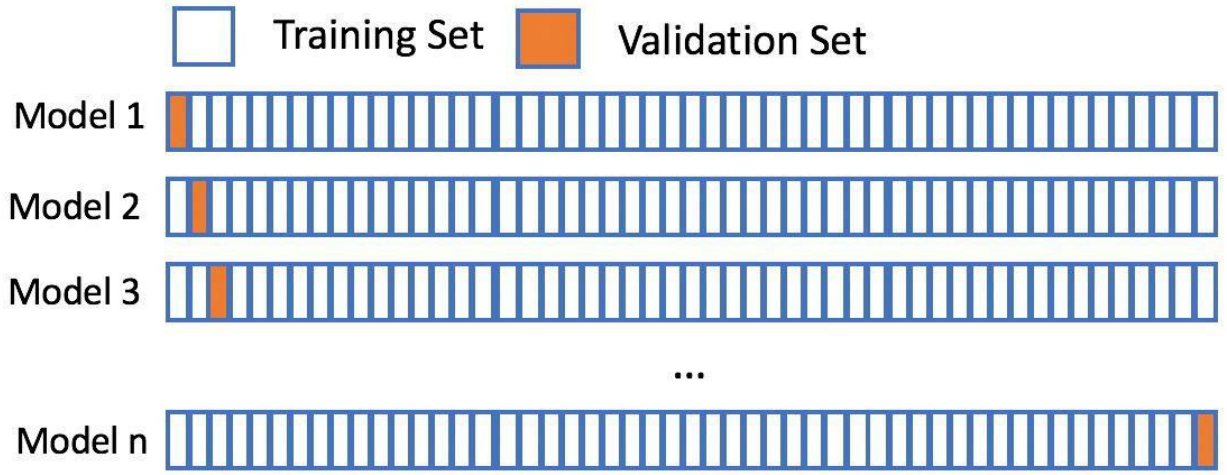
Leave-p-out cross-validation

```
from sklearn.model_selection import LeavePOut,cross_val_score
from sklearn.datasets import load_iris
from sklearn.ensemble import RandomForestClassifier
iris=load_iris()
X=iris.data
Y=iris.target
lpo=LeavePOut(p=1)
lpo.get_n_splits(X)
tree=RandomForestClassifier(n_estimators=5,max_depth=3,n_jobs=-1)
score=cross_val_score(tree,X,Y,cv=lpo)
print("Cross Validation Scores-{}".format(score))
print("Average Cross Validation score :{}".format(score.mean()))
```

```
Cross Validation Scores-[1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1.
 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
0.
 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1.
 1. 1. 1. 1. 1. 1.]
Average Cross Validation score :0.9533333333333334
```

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.



Leave-one-out cross-validation

```
from sklearn.datasets import load_iris
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import LeaveOneOut,cross_val_score
iris=load_iris()
X=iris.data
Y=iris.target
leave_one_out=LeaveOneOut()
rfc=RandomForestClassifier(n_estimators=7,max_depth=4,n_jobs=-1)
score=cross_val_score(rfc,X,Y,cv=leave_one_out)
print("Cross Validation Scores:{}".format(score))
print("Average Cross Validation score:{}".format(score.mean()))
```

```
Cross Validation Scores:[1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0.
1.
 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
0.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 0. 1. 1. 1. 0. 1. 1. 1. 1.
1.
 1. 1. 1. 1. 1. 1.]
Average Cross Validation score:0.9466666666666667
```