Advanced Machine Learning and Deep Learning

Dr. Mohammad Pourhomayoun
Assistant Professor
Computer Science Department
California State University, Los Angeles

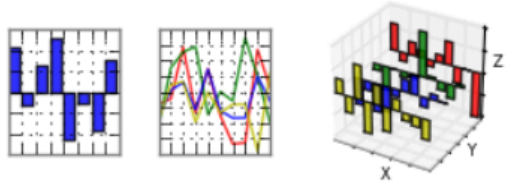
Data Science with Python

IP[y]: IPython
Interactive Computing

Keras

pandas

$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$



scikits learn
machine learning in Python

NumPy

SciPy.org Sponsored By ENTHOUGHT

matplotlib

Pandas

- **Pandas** is a powerful library to read, manipulate, and process large-scale data.
- **Pandas** introduces two new data structures to Python, **Series** and **DataFrame**, both of which are built on top of NumPy n-dimensional array.
- A **Series** is a one-dimensional vector similar to an array, list, or column in a table. It will assign an index to each item in the Series (starting from 0).
- A **DataFrame** is a structured data table (or Matrix) contains of rows and columns.
- Each column of a DataFrame can be considered as a Series.
- We covered the details of Pandas before. Please review it!

Scikit-Learn
Machine Learning Library

Scikit-Learn (sklearn)

- Scikit-learn is the Python Machine Learning Library.
- It includes optimal implementation of various classification, regression and clustering algorithms.
- It also includes hundreds of commands and functions for data preprocessing and processing along with a number of default datasets to work with.
- It is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.
- Scikit-learn has an exceptional documentation.

Important Note and Clarification

- In practice, we usually have a **Training Dataset** that is used to train your predictive model. Then, you will use your "trained model" to make prediction on **future data**.

- However, before using a model for real predictions, you will need to **evaluate** it using a known dataset!

- They are many different ways that we've learned so far. The **simplest** approach is just to split your existing dataset into two (or three) parts, called **Training Dataset, Testing Dataset (and sometimes validation dataset).** Then train your model on Training Dataset and test it on the Testing Dataset!

- Of course, they are better ways like **cross-validation** approach to evaluate a model. For now, let's start with the simplest method.

Evaluating the Accuracy of a Predictive Model Using Training-Testing Split

Evaluating The Accuracy Of Our Predictive Model

Here is the simplest way to evaluate the accuracy of our predictive model:

1- Let's split the dataset **RANDOMLY** into two new datasets: **Training Set** (e.g. 70% of the data samples) and **Testing Set** (e.g. 30% of the data samples).

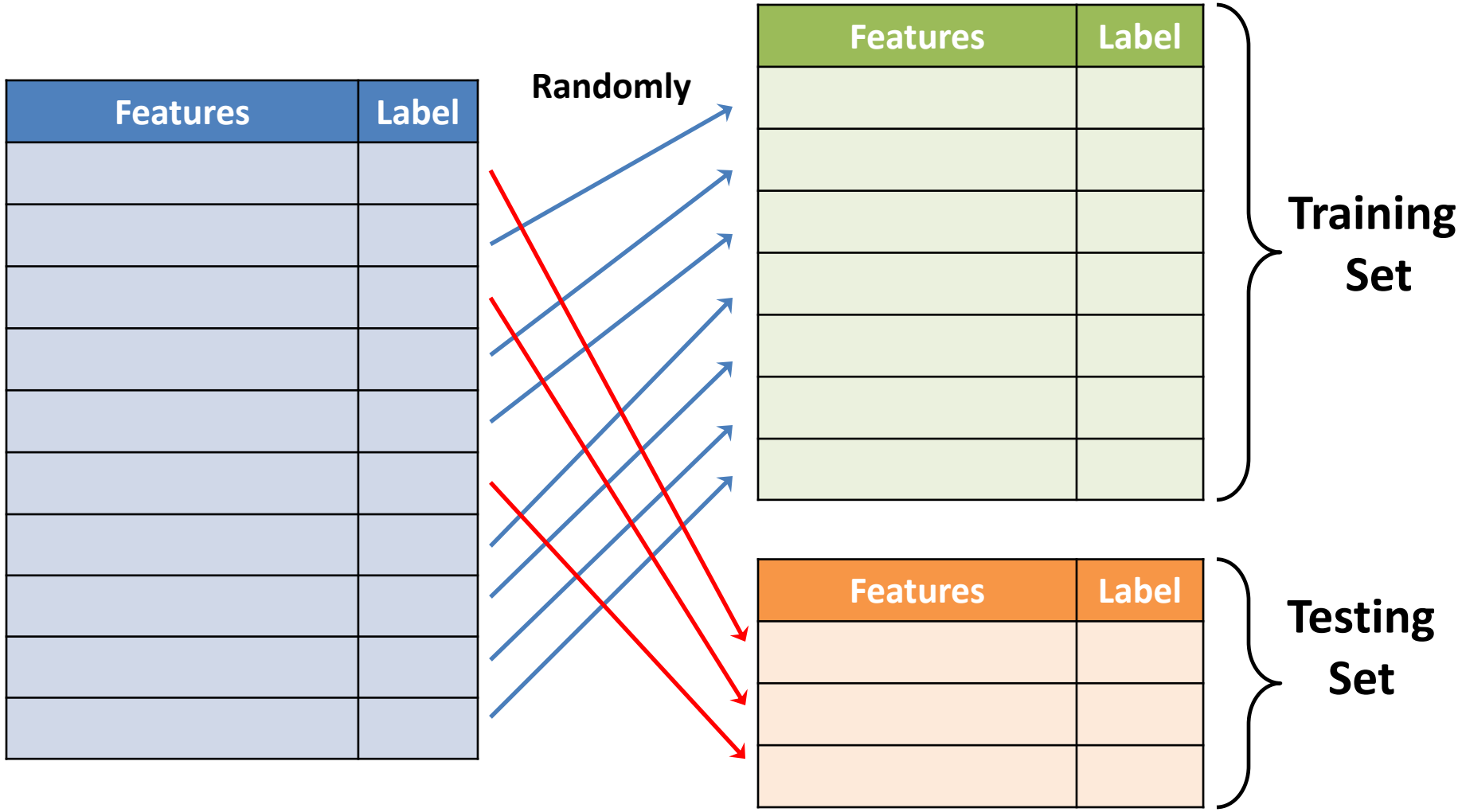
2- Let's **pretend** that we do **NOT** know the label of the Testing Set!

3- Let's Train the model **ONLY on Training Set**, and then Predict on the Testing Set!

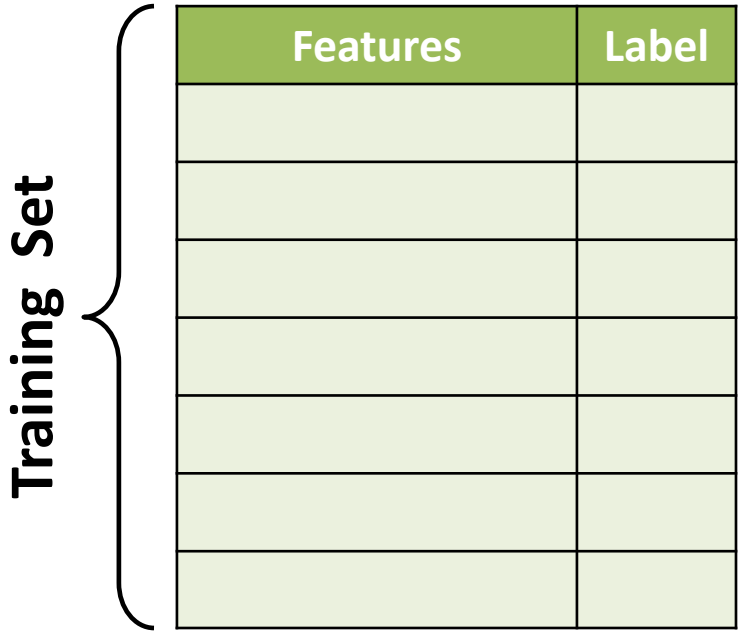
4- After prediction, we can compare the **predicted labels** for the Testing Set with the **actual labels** of it to evaluate the accuracy of our prediction!

Training and Testing Sets

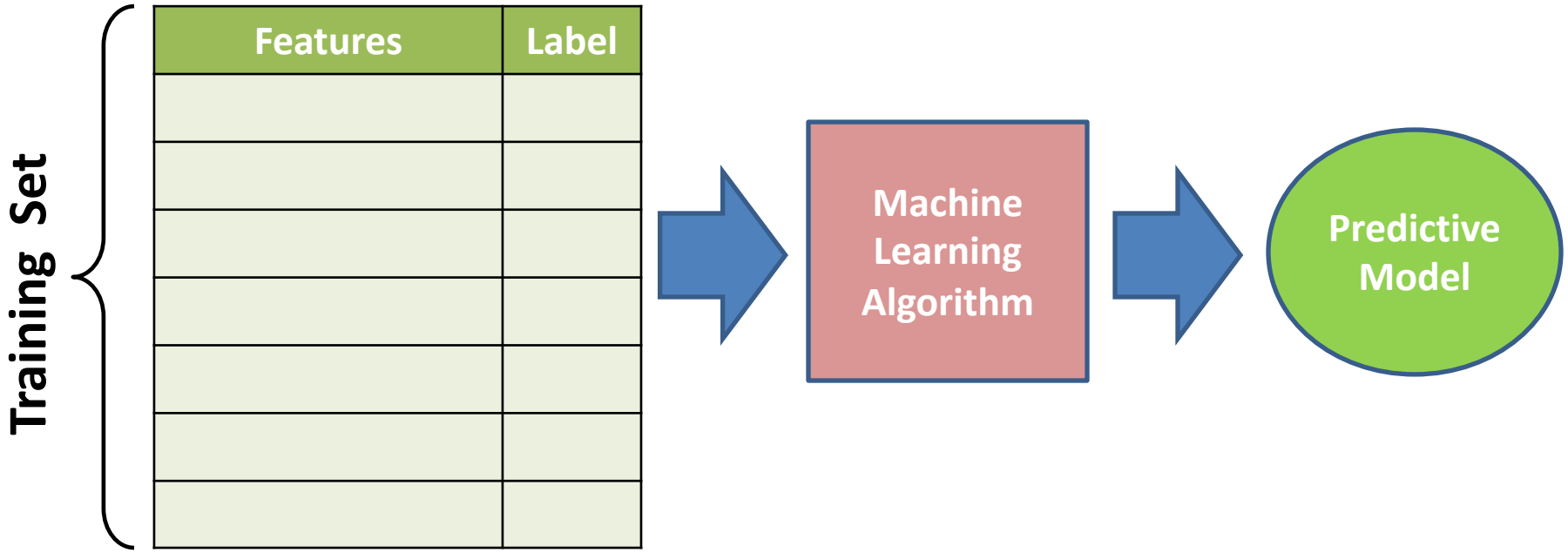
Training and Testing Sets



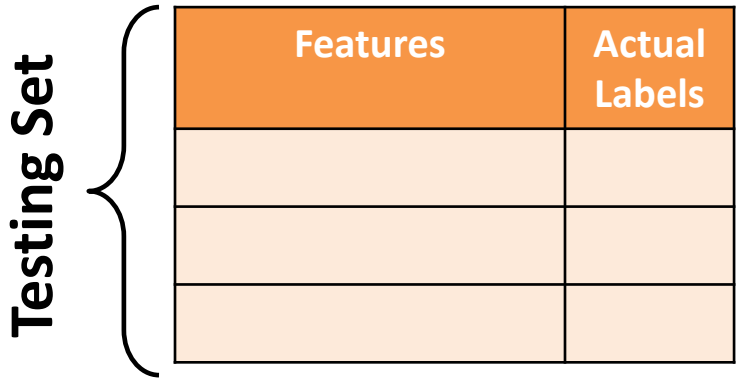
Training Stage



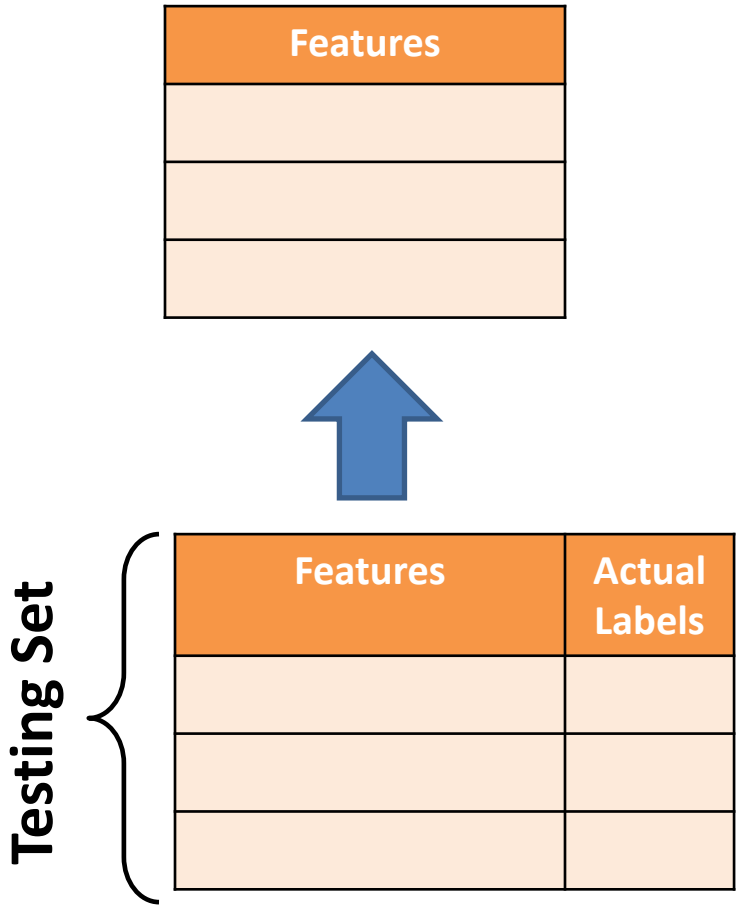
Training Stage



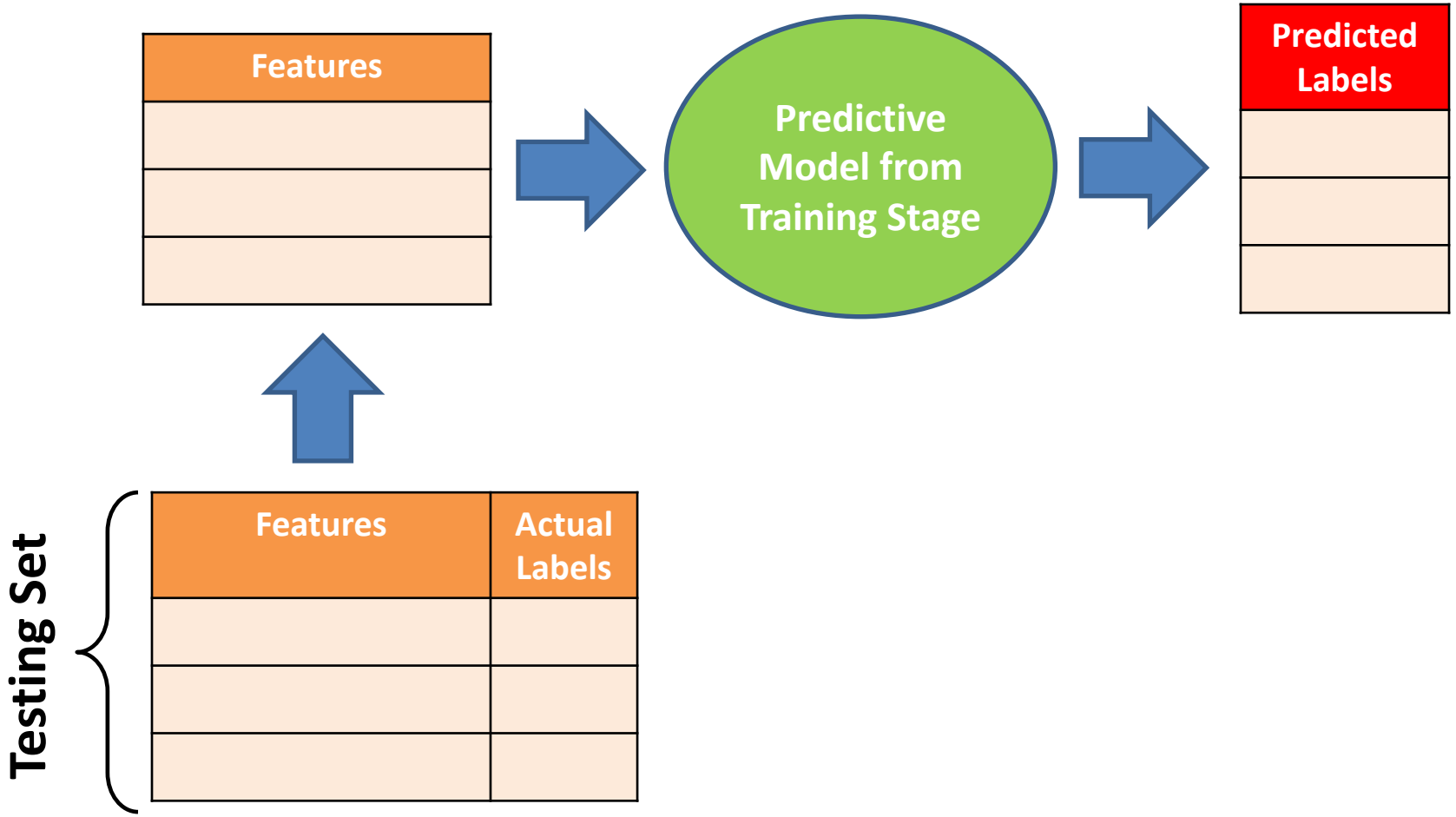
Testing Stage



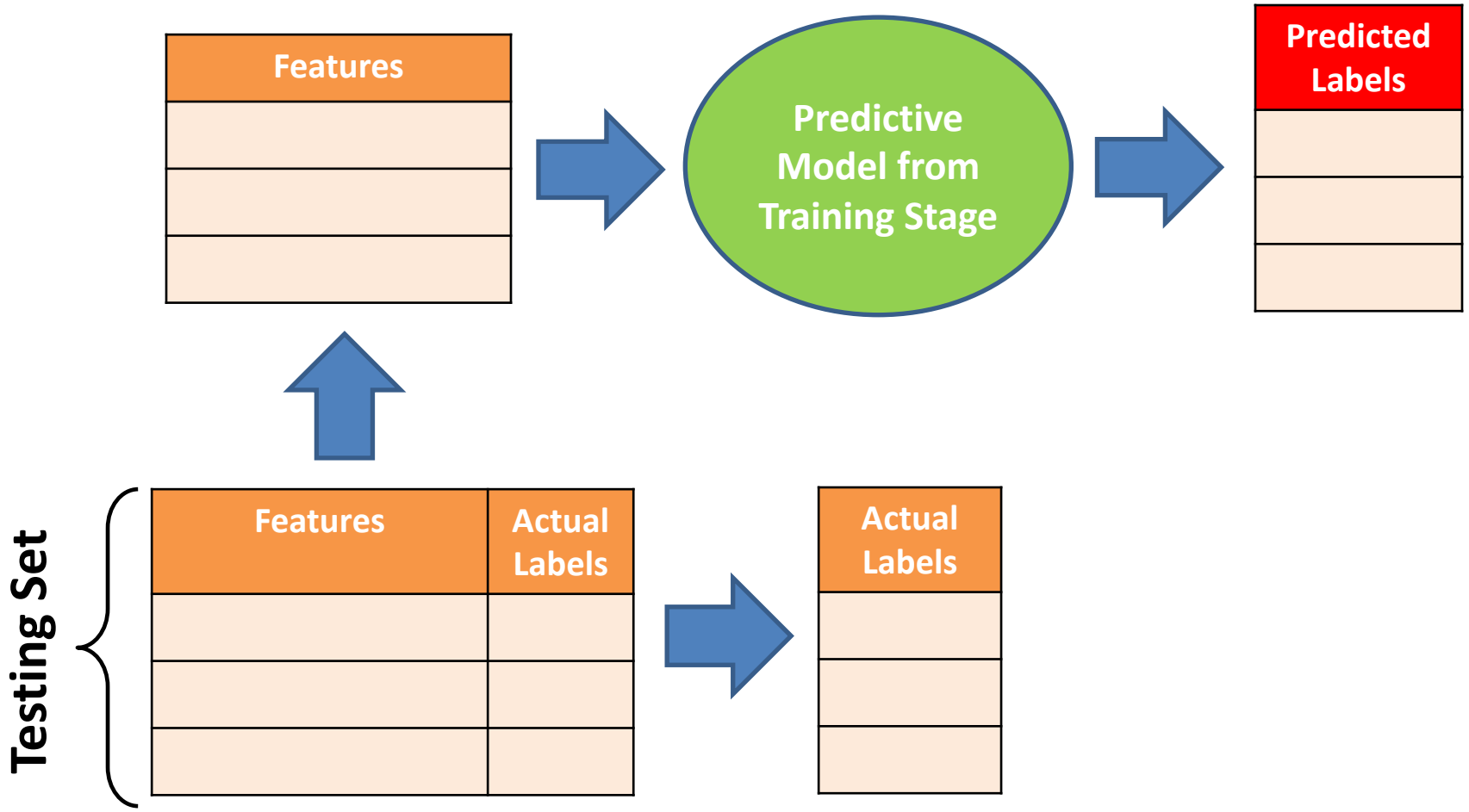
Testing Stage



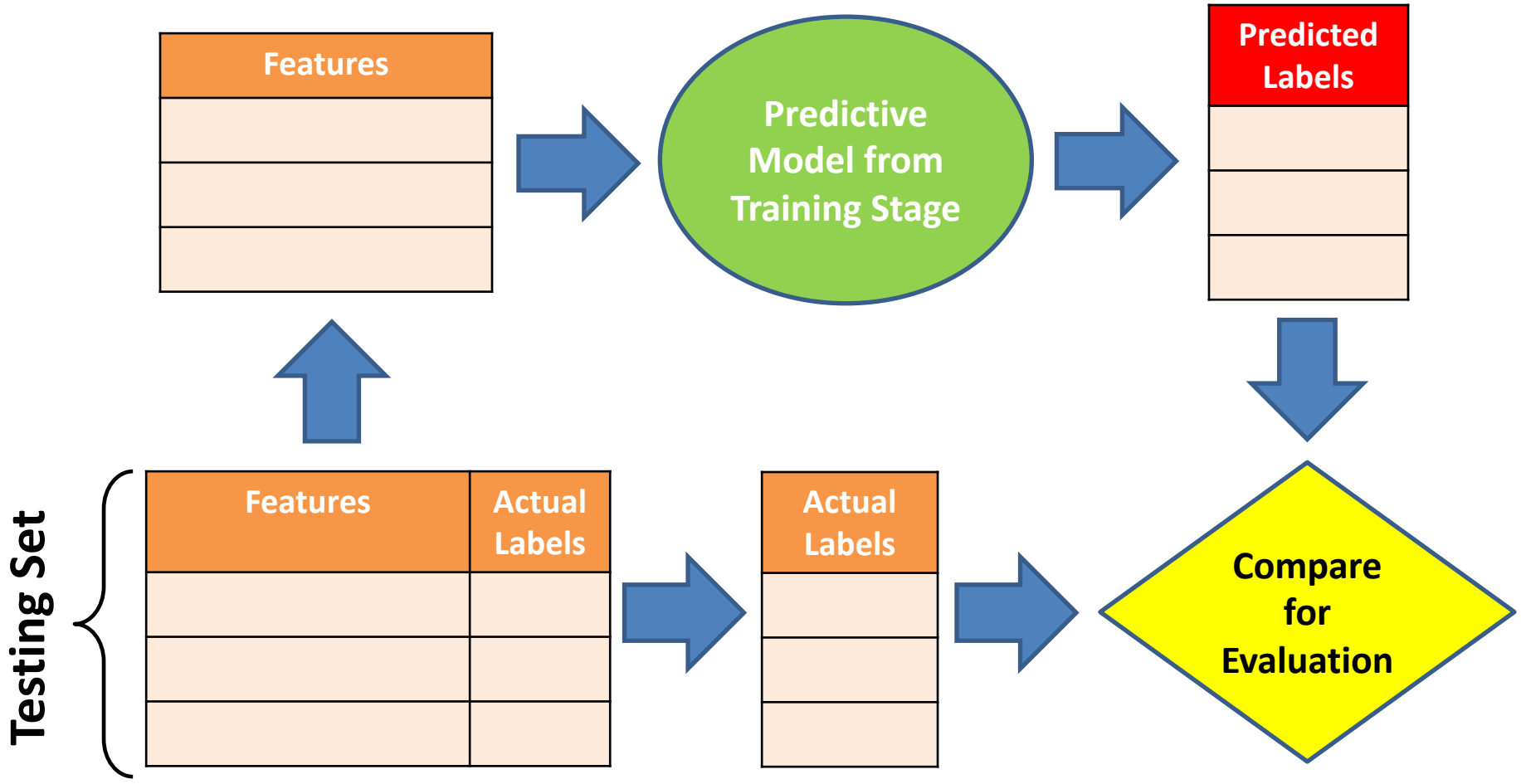
Testing Stage



Testing Stage



Testing Stage



Evaluating The Accuracy Of Our Predictive Model

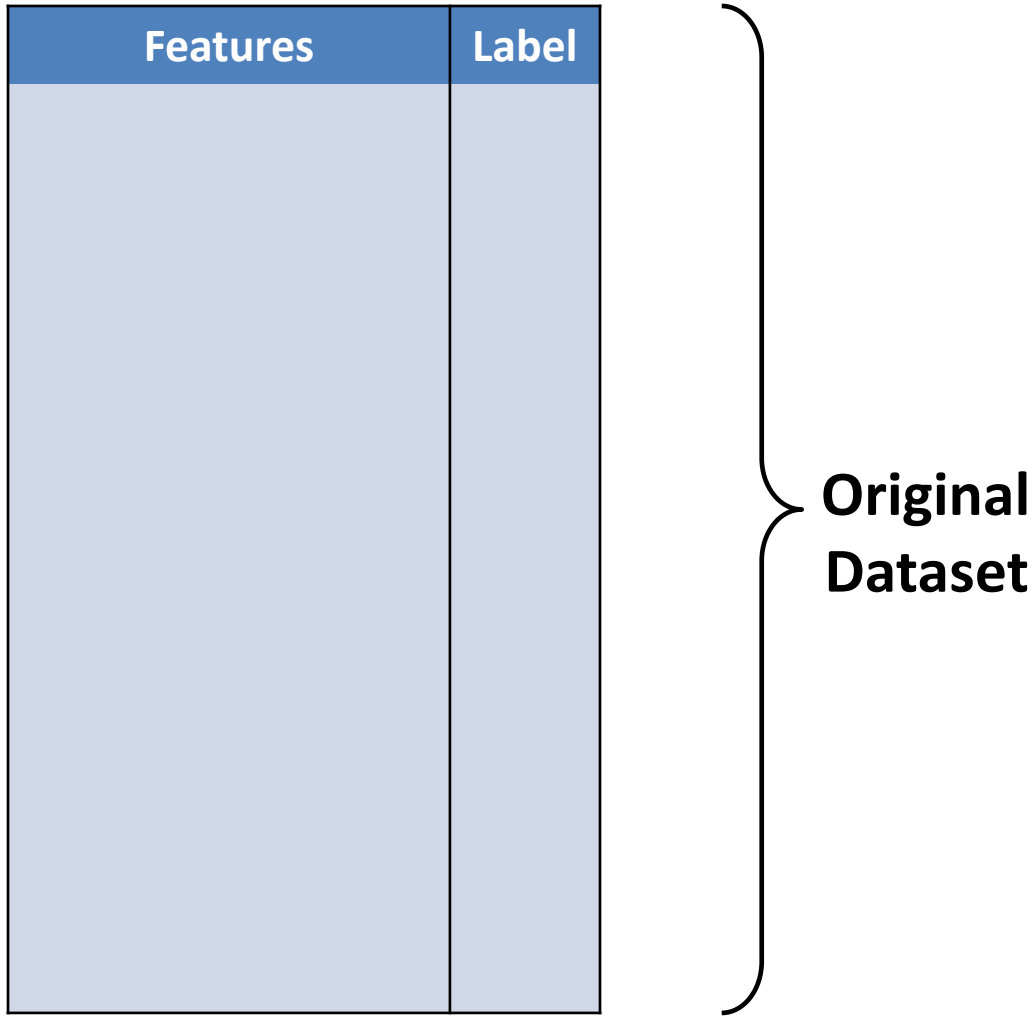
VERY IMPORTANT: There must be NO OVERLAP between Training Set and Testing Set!

Note: we can split the original dataset into 3 sets: **Training Set, Testing Set**, and **Validation Set**. In this case, We can use Validation set for adjusting the parameters, and then use Testing Set for final evaluation.

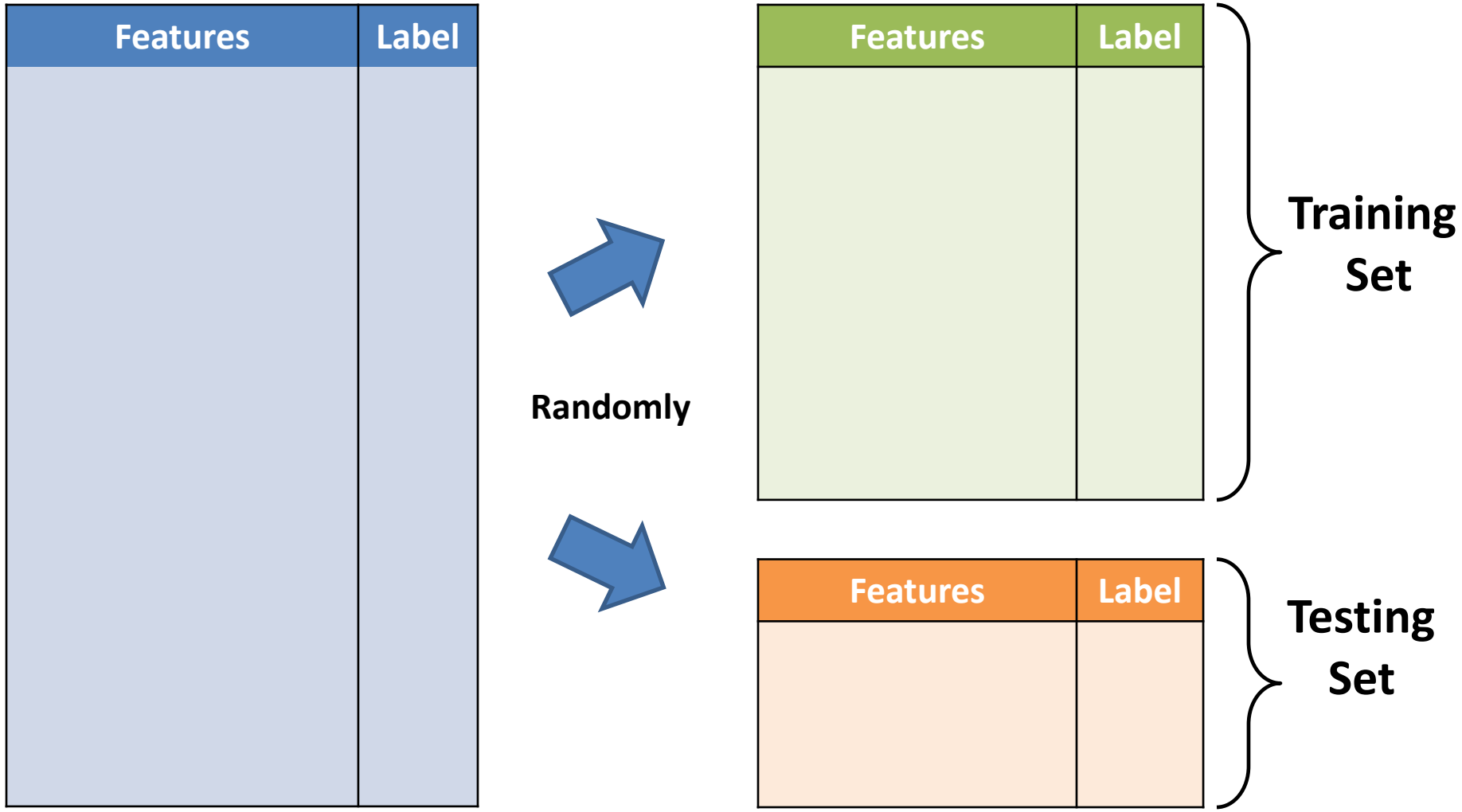
Evaluating the Accuracy of a Predictive Model Using

Cross Validation

Training and Testing Sets



Training and Testing Sets



Cross Validation

- We saw how to split the dataset into Training and Testing sets, Fit the model on "training set", and then predict on "testing set" to evaluate the accuracy.
- The problem with this method is that **the results may depend on the choice of split**. For example, if you are lucky, some easily predictable samples may happen to be located in the testing set (or vice versa!).
- In order to get fair results, we can repeat the splitting process several times, compute the prediction accuracy for each split, and then average the results.
- **Cross Validation tries to repeat the splitting procedure K times in a smart way such that all data samples will be used in "testing set" one time and in "Training Set" (K-1) times!**

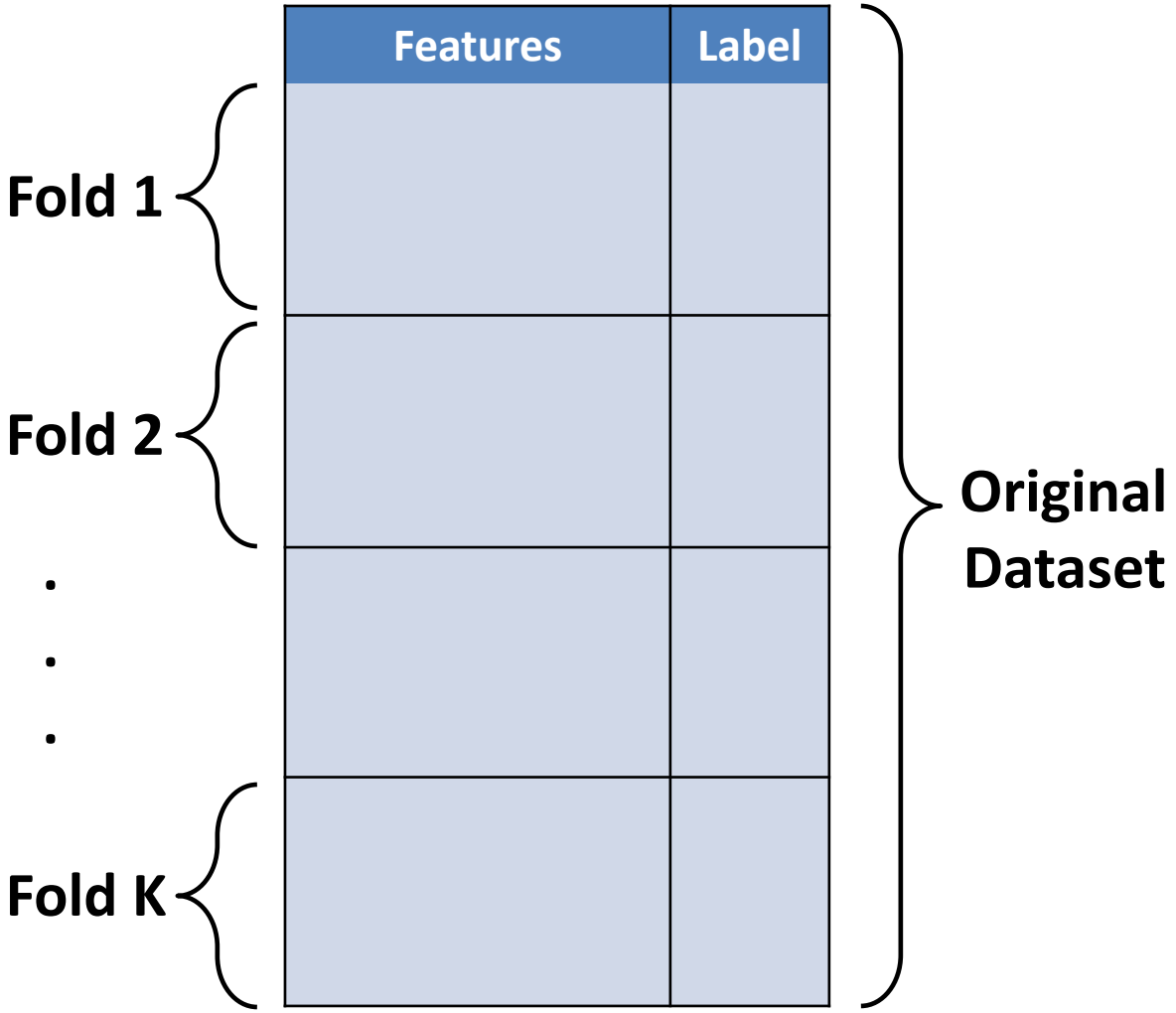
Cross Validation

Three main steps for K-fold cross-validation:

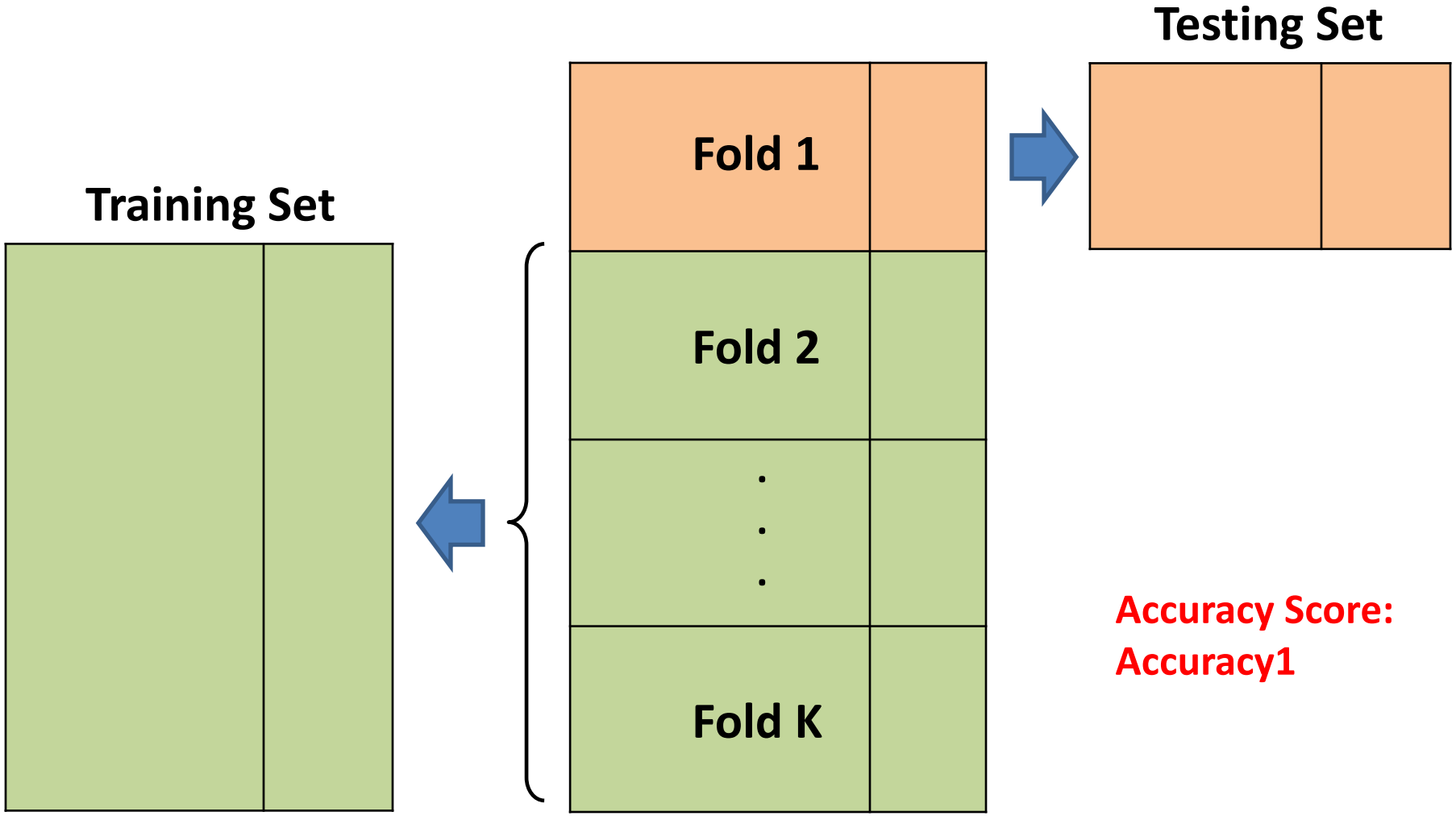
1. Partition the dataset Randomly into K equal, non-overlapping sections (called Fold).
2. Use one of the sections as **testing set** at a time and the union of the other (K-1) sections as the **training set**. Perform training stage, testing stage, and compute the accuracy based on the split each time. Repeat this procedure K times, so that each one of the K sections is used as **testing set** one time, and as a part of **training set** (K-1) times.
3. Calculate the average of the accuracies as final result.

Note: K is arbitrary, but Using K=10 (10-fold cross-validation) is very common and recommended in machine learning.

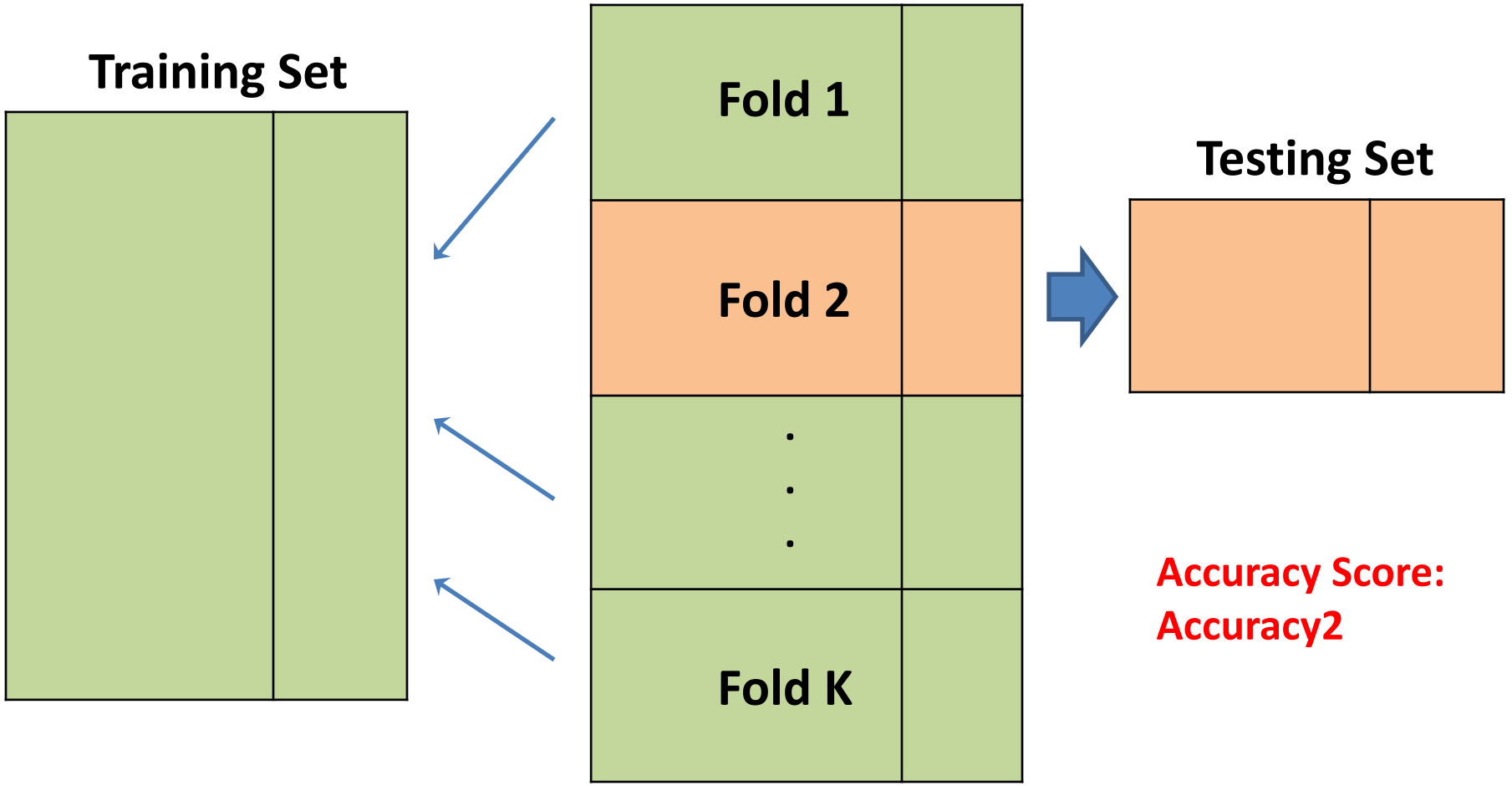
Cross Validation



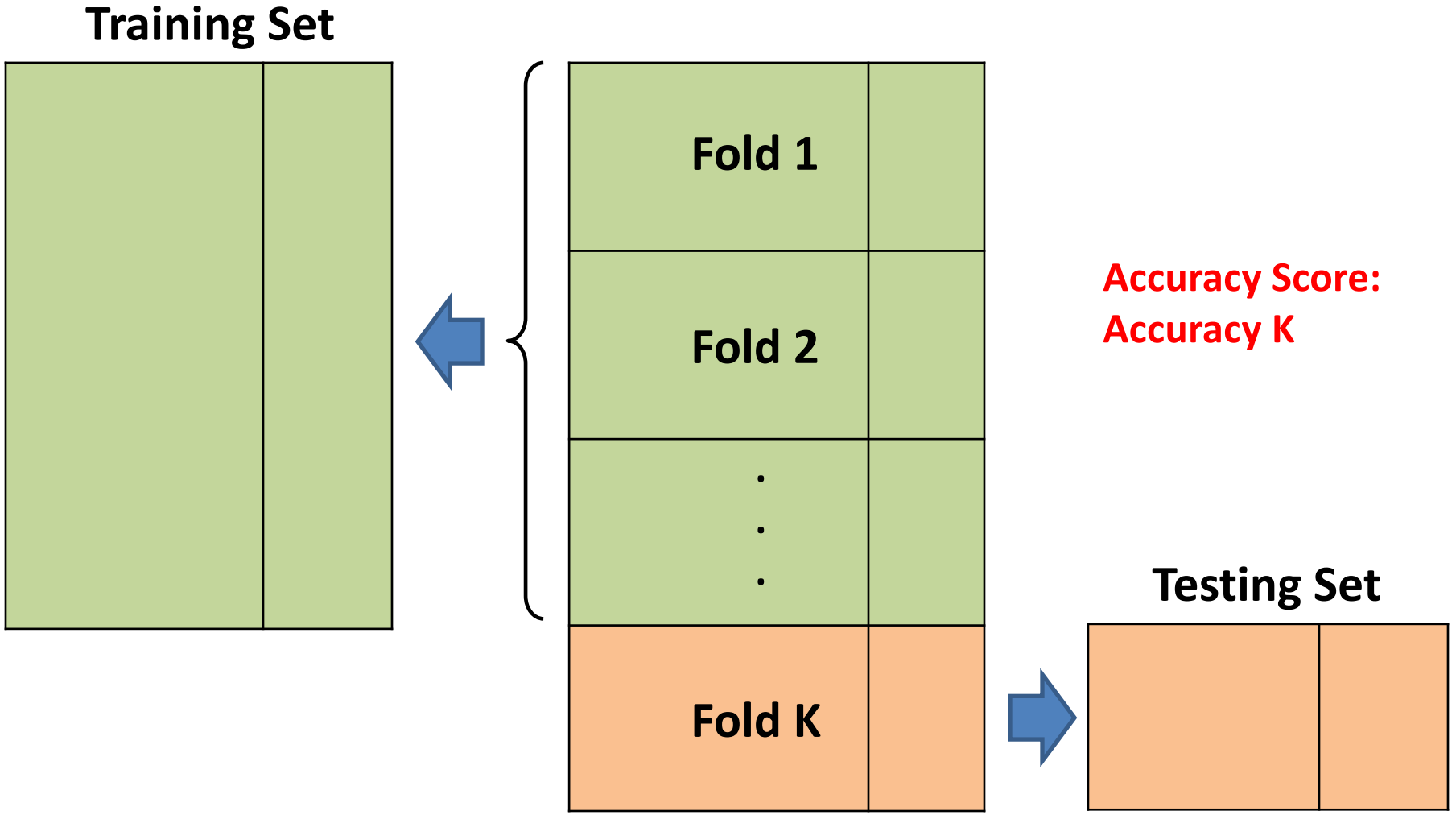
Cross Validation – Round 1



Cross Validation – Round 2



Cross Validation – Round K



Cross Validation

- **Accuracy_Score_Total =**

  (Accuracy 1 + Accuracy 2 + Accuracy 3 + … + Accuracy K) / K

Data Science Practical Tutorial

- Let's open the ***.ipynb*** tutorial file
  in Jupyter notebook to continue ….

Thank You!

Questions?