Vinoothna Bavireddy

Train and Test Split for Machine Learning Models

What is Train-Test Split?

The train-test split is an evaluation technique to measure the performance of a machine learning model.[1] The dataset is divided into two subsets one called the training dataset and the other is called the test dataset. The **training dataset** is used for **building** the model and the **test dataset** is used for **evaluating** the model. The goal for performing splitting is to evaluate the performance of the algorithm on data that has not been used to train the model.

This technique of splitting the dataset into two (train-test) is only appropriate for large datasets. It is not a good technique to perform this technique on small datasets as it may lead to overfitting or underfitting. To overcome this we may use the k-fold cross-validation procedure.

What is the best % for Train-Test split?

The answer to this question is that there is no best split percentage for splitting a data into training and test datasets. Splitting the dataset depends on your project goals:

- Computational cost in training the model.
- Computational cost in evaluating the model.
- Training set representativeness.
- Test set representativeness.

Some common split percentages include:

- Train: 80%, Test: 20%
- Train: 67%, Test: 33%
- Train: 50%, Test: 50%

Parameters for splitting the data:

1. Test_size: We give the percentage of data that is to be split into test dataset.
2. Train_size: We give the percentage of data that is to be split into training dataset. You can specify either the test or training dataset values.
3. Random_state: We use the numpy library to set a random seed to split the datasets.

How to split the data?

To make the splitting of our data as random as possible python provides a library called **SciKit** which has a tool known as **Model Selection library**. The Model Selection library has a class named **'train_test_split'** which can easily split the dataset into the training and the testing datasets in various proportions.
For example: Splitting data in 80%-20% proportions

```
from sklearn.model_selection import train_test_split
xTrain, xTest, yTrain, yTest = train_test_split(x,y,train_size=0.8, random_state =0 )
```

Vinoothna Bavireddy

What is overfitting / under-fitting?

Overfitting

If we split small datasets where there are only a few combinations within it. The algorithm tends to memorize all the possible combinations from the training dataset and tries to fit the data from the test dataset leading to a condition called over-fitting. This is a common issue in models that are too complex or where there are a lot of variables.

Underfitting

The reverse of the above condition that is when the machine learning algorithm does not recognize the trends or patterns in the data to fit the training dataset then it is known as underfitting.

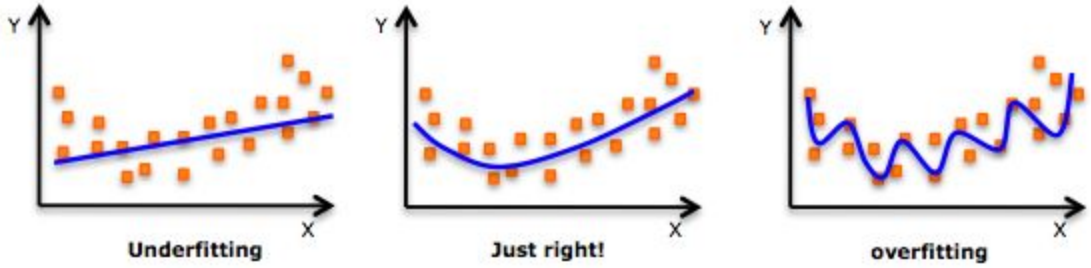


Image2:(Bronshtein,2017)

As stated above to prevent the issues of overfitting or underfitting we can use the **Cross Validation Technique.**

There are several types of cross-validation. The most common one is the K-fold

Other techniques can be found in the Scikit Library

K-Fold Cross Validation:

This technique is similar to splitting the data to training and test datasets but in several (K) folds. This means the entire dataset is divided into **k subsets** and we train our model on **k-1 subsets.** The last dataset (the one not used for training) serves as the **test data.** Each round of validation produces accuracy in percentage which is averaged to determine the overall accuracy for the training dataset. We compare this value against our test data accuracy to measure the performance of the model.

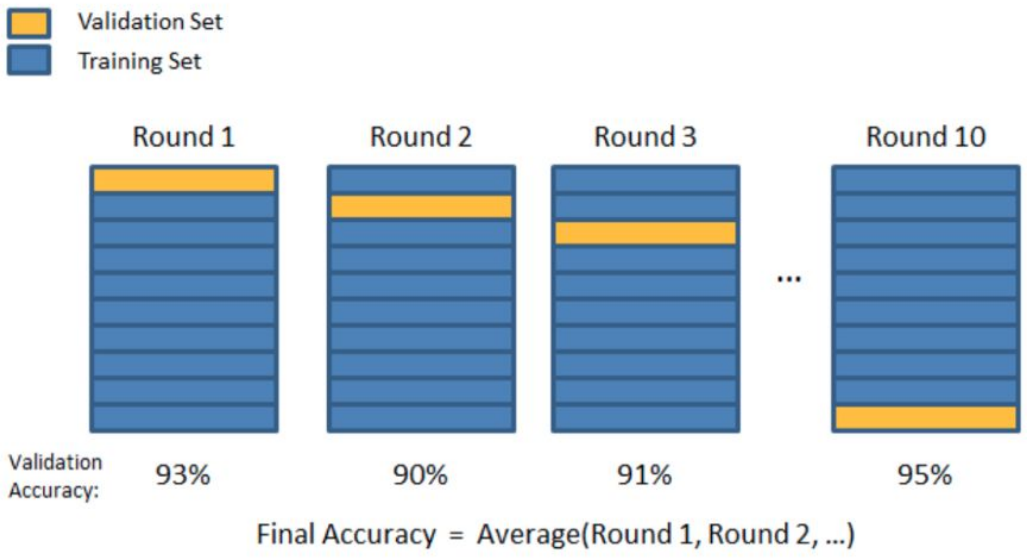
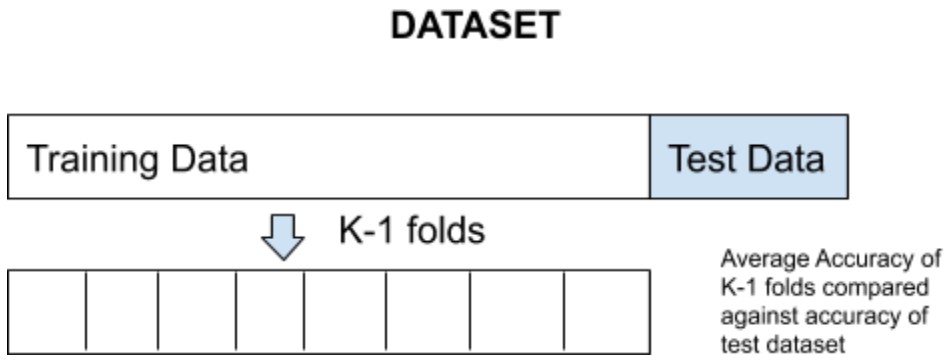


Image3: **K-Folds** (Bronshtein,2017)

What K to select?

The choice of **number of folds / k** is based on the size of each validation partition. It must be such that it allows for a partition size that is large enough to provide a fair estimate of the model's performance. At the same time K shouldn't be too small, say 2, such that we don't have enough trained models to evaluate. Usually, each validation set must contain around 15%–20% data in it.



Pros and Cons of Train-test split and K-fold Cross validation:

Pros of test/train split:

- Runs k-times faster than k-fold hence less time consuming
- Easy to implement and analyze the testing errors

Cons of test/train split:

- Cannot be applied on small datasets .
- Chance of missing randomness leading to inaccurate results.
- Issues of overfitting or underfitting.

Vinoothna Bavireddy

Pros of cross-validation:

- It is a better estimator of out-of-sample accuracy.
- More efficient use of data since each data is used for train and testing
- Less inaccuracies since the validation is performed on multiple folds
- Can be applied on small datasets without the fear of overfitting.

Cons of cross-validation:

- More time consuming than test-train split.
- Does not work well with sequential data such as data with time series.

The choice of train-test split / cross validation technique depends on the size of the dataset and also the kind of data (data type) you are predicting. Even though there is no one best technique to achieve the best model, we can make use of open source libraries such as Scikit to choose the best option for splitting the dataset or use the k-fold technique for evaluating the model.

References:

1. Jason B. (2020, July 24). Train-Test Split for Evaluating Machine Learning Algorithms. Python Machine Learning. https://machinelearningmastery.com/train-test-split-for-evaluating-machine-learning-algorithms
2. Srinidhi, S. (2020, January 9). How to split your dataset to train and test datasets using SciKit learn. Medium. https://medium.com/@contactsunny/how-to-split-your-dataset-to-train-and-test-datasets-using-scikit-learn-e7cf6eb5e0d
3. Poudyal, R. (2018, June 15). Train / Test split and cross-validation in Python. Medium. https://medium.com/@rabinpoudyal1995/train-test-split-and-cross-validation-in-python-434ecba10909
4. Anya. (2018, December 28). How to split train and test data. Medium.
5. https://medium.com/@karyaozmen/how-to-split-train-and-test-data-c1381d240fc4
6. Bronshtein, A. (2020, March 24). Train/Test split and cross validation in Python. Medium. https://towardsdatascience.com/train-test-split-and-cross-validation-in-python-80b61beca4b6
7. Bhattacharyya, M. (2020, July 15). 3 things you need to know before you train-test split. Medium. https://towardsdatascience.com/3-things-you-need-to-know-before-you-train-test-split-869dfabb7e50
8. Train-test split – NaadiSpeaks. (2017, Oct 12). NaadiSpeaks. https://naadispeaks.wordpress.com/tag/train-test-split/