Why (how) do we split train, valid, and test?

Photo by Pietro Jeng on Unsplash

During coding your machine learning project or learning machine learning, did you get confused about the data to split? I donâ€™t know about you, but I did. Not only train and test data set but also valid data set. Why do we split like that there and how to split? This is a good blog by Rachel Thomas from Fast.ai and you should read that blog too. His blog and mine will give you the same concepts and workflow, but with a different points of view.

Before talking about the train, validate and test. Letâ€™s talk about machine learning. As you know machine learning models/algorithms depend on the data. The more data you have, the better. Not just with more data, but it also should be the right data. Even though you have plenty of data with not the right features/values, your model cannot work well. So, this is where the train, valid, and test data set come in. See the short brief definition of these three:

Training set â€” to train your model on this data set Validation set â€” to test your model on this before testing the data set, to make hyperparameter tuning. Testing set â€” to test your model on this; after testing and tuning on the validation set, to know how well your model in real-time

Without the validation data set

In normal machine learning problems (without time series data), that is not a big deal without a validation data set. But anyway, you should split your data into train, valid, and test sets. Sometimes, without validation data set your model is working well on your local project but not getting a good deal on production.

The purpose of the testing data set is to test your model, right? So, you only should test on that, not making hyperparameter tuning. If youâ€™d make tuning on your testing data (some did on the training data), how can you test your model on the unseen data set/real-time data set? This is the key point of splitting your data set into train, valid, and test, and will help your model work well on production.

In machine learning problems with time series data (e.g. stock price prediction), do you think your data will be always constant? Not possible, your data will be changed continuously. So, in these problems, the role of the validation data set is more important. I will talk about this later in this blog.

How do split train, valid, and test?

All of you already know to split your data into 70% for the training data set and 30% for the testing data set. Some split into 80% and 20%, but it is not important. We are not just talking about training and testing data set. When the

validation data set came in, you should split your data into 70% for training data, 15% for validation, and 15% for the testing data set. See this below:

Splitting into train, valid, and test

Note that: splitting 70%, 15% and 15% is for general use. It depends on your data size and only mostly works on the problem without time series data. Letâ€™s see with the code:

The below codes do not contain the hole complete code for the machine learning project, just for splitting part

Scikit-Learn doesnâ€™t provide built-in train_valid_test_split function, so we will use built-in train_test_split . There are many ways to split into train, valid, and test. You can use it anyway as you prefer.

Splitting the normal data

We will import libraries, setup random seed, and read the data

```
import pandas as pd

import numpy as np

from sklearn.model_selection import train_test_split

# Setup random seed

np.random.seed(47)

# import data

df = pd.read_csv("data/heart-disease.csv")
```

And split into train, valid, and test sets as below

```
# Create X and y

X = df.drop("target", axis=1)

y = df["target"]

# first, split into X_train, X_valid_test, y_train, y_valid_test

# `test_size=0.3` split into 70% and 30%

X_train, X_valid_test, y_train, y_valid_test = train_test_split(X, y, test_size=0.3)

# second, split into X_valid, X_test, y_valid, y_test

# `test_size=0.5` split into 50% and 50%. The original data set is 30%,

# so, it will split into 15% equally.

X_valid, X_test, y_valid, y_test = train_test_split(X_valid_test, y_valid_test, test_size=0.5)
```

First of all, split into the training set and the combination of validation and testing sets. In this case, test_size=0.3 will split your data into 70% and 30% separately.

Second, we will split the combination of validation and testing set into validation and testing data set using test_size=0.5 . The original `valid_test` contains 30% of the original data set and so validation and testing data set will contains 15% of data equally.

Letâ€™s see with notation, and try not to confuse:

Splitting normal data

If you are confused above figure, you should check Scikit-Learnâ€™s train_test_split again.

Splitting the time series data

Working with normal data is not a big deal. But it can be a little bit tricky to work with the time series data/continuous data. There is a key one to note:

You never should use Scikit-Learnâ€™s train_test_split function. Because it splits your data randomly.

As you know, your data will always pass and create new data every time. Letâ€™s say you have data from 2010 to 2023 (now, we are here). Assume that it is a time series data and you want to predict data for tomorrow or for maybe next week. You use built-in train_test_split for that data and what if, if most of the data in the validation and testing data set are from 2010 to 2015?

Who knows the things will be the same? Time passed. Everything changed. So, your validation and testing data should be between 2021 and 2022, it is even better if your passed data is from around 5 months or 6 months ago. It cannot be a good deal if your data is far from 1 year.

Before coding for train, valid, and test for time series data, letâ€™s see the notation first:

Splitting time series data

As you saw in the above example, there is data for 11 months from the present, we are at the end of the year (December) and I split data backward from the present. Two months ago from present for the test, two months ago from the test set is for valid and the others are for the training data set. And maybe in the real world data set, there is more complicated than this. So, please note:

There is no 70% and 15% for the data in the above example and there is no specific rule. The more closed your valid and test set from the present, the better. Because time is passing.

210.00155555556

I will show you how you can easily split your data into that three parts

Note that: this is my point of view and this cannot be exactly true

Letâ€™s see in the code,

Import libraries, set up random seed, and create the data frame,

```
import pandas as pd
import numpy as np
# setup random seed
np.random.seed(47)
# read the data
# `parse_dates` is for easy to use date time column
df = pd.read_csv("data/netflix-stock-price-prediction.csv", parse_dates=True)
```

Sort the data frame and create X and y ,

```
# sort the dataframe by `Date`
# `Date` is the column name in the csv file
df = df.sort_values(by="Date", ascending=True)
# Create X and y
```

210.00155555556

```
X = df.drop("Volume", axis=1)

y = df["Volume"]

Split into training, validation, and testing sets

"""

The original data set has 1009 values and

We sorted the dataframe from top to bottom with `ascending=True`

In this case, real time data set cannot be same like this,

this is just an example

"""

# Indexing for training set

X_train, y_train = X[:800], y[:800]

# Indexing for validation set

X_valid, y_valid = X[801:900], y[801:900]

# Indexing for testing set

X_test, y_valid = X[901:], y[901:]

Full codes and CSV files are on GitHub
```

210.00155555556

The codes and get the CSV files I used in this example are uploaded to my GitHub account,

If you want to see the above figures in live-action, see here.

Conclusion

I think you are clear about what is validation set is, why it is important and why we need it. And make sure you read this good blog by Rachel Thomas. That can move to you the next step.

Reference

Educating Deaf Students: From Research to Practice

Negotiating the Complexities of Qualitative Research in Higher Education: Fundamental Elements and Issues