How to split data into three sets (train, validation, and test) And why?

Sklearn train test split is not enough. We ... something better, and faster

Samarth Agrawal May 17 · 5 min read ★

Photo by Nathan Dumlao on Unsplash

INTRODUCTION

Why do you need to split data?

You don't want your model to over-learn from training data and perform poorly after being deployed in production. You need to have a mechanism to assess how well your model is generalizing. Hence, you need to separate your input data into training, validation, and testing subsets to prevent your model

from overfitting and to evaluate your model effectively.

In this post, we will cover the following things.

1. A brief definition of training, validation, and testing datasets
2. Ready to use code for creating these datasets (2 methods)
3. Understand the science behind dataset split ratio

. . .

Definition of Train-Valid-Test Split

Train-Valid-Test split is a technique to evaluate the performance of your machine learning model — classification or regression alike. You take a given dataset and divide it into three subsets. A brief description of the role of each of these datasets is below.

Train Dataset

- Set of data used for learning (by the model), that is, to fit the parameters to the machine learning model

Valid Dataset

- Set of data used to provide an unbiased evaluation of a model fitted on the training dataset while tuning model hyperparameters.
- Also play a role in other forms of model preparation, such as feature selection, threshold cut-off selection.

Test Dataset

- Set of data used to provide an unbiased evaluation of a final model fitted on the training dataset.

Read this article by Jason Brownlee if you want to know more about how experts in machine learning define train, test, and validation datasets. Link in the references sections below #1

. . .

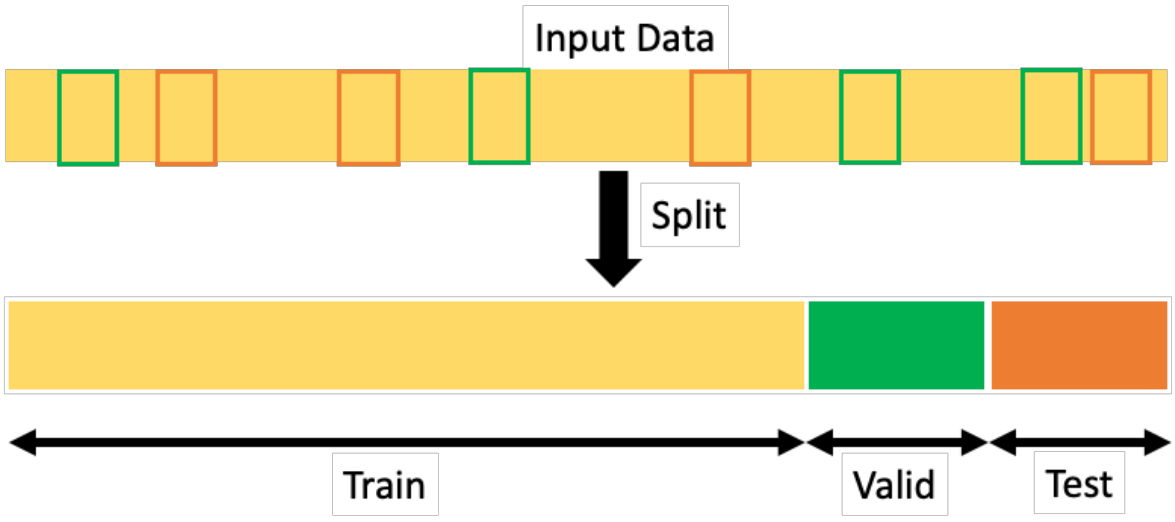
Ready to use code snippets

In this post we will see two ways of splitting the data into train, valid and test set —

1. Splitting Randomly
2. Splitting using the temporal component

1. Splitting Randomly

You can't evaluate the predictive performance of a model with the same data you used for training. It would be best if you evaluated the model with **new data** that hasn't been seen by the model before. Randomly splitting the data is the most commonly used method for that **unbiased evaluation.**



Randomly split the input data into train, valid, and test set. Image by Author

i. Using Sklearn → 'train_test_split'

In the code snippet below, you will learn how to use `train_test_split` twice to create the train | valid | test dataset of our desired proportions.

```
import pandas as pd

df = pd.read_csv('/kaggle/input/bluebook-for-bulldozers/TrainAndValid.c

from sklearn.model_selection import train_test_split

# Let's say we want to split the data in 80:10:10 for train:valid:test
train_size=0.8

X = df.drop(columns = ['SalePrice']).copy()
y = df['SalePrice']

# In the first step we will split the data in training and remaining da
X_train, X_rem, y_train, y_rem = train_test_split(X,y, train_size=0.8)

# Now since we want the valid and test size to be equal (10% each of ov
# we have to define valid_size=0.5 (that is 50% of remaining data)
test_size = 0.5
X_valid, X_test, y_valid, y_test = train_test_split(X_rem,y_rem, test_s

print(X_train.shape), print(y_train.shape)
```

ii. Using Fast_ml → 'train_valid_test_split'

In the code snippet below, you will learn how to use `train_valid_test_split` to create the train | valid | test dataset of our desired proportions in a single line of code.

```
import pandas as pd

df = pd.read_csv('/kaggle/input/bluebook-for-bulldozers/TrainAndValid.c

from fast_ml.model_development import train_valid_test_split

X_train, y_train, X_valid, y_valid, X_test, y_test = train_valid_test_s

print(X_train.shape), print(y_train.shape)
print(X_valid.shape), print(y_valid.shape)
```

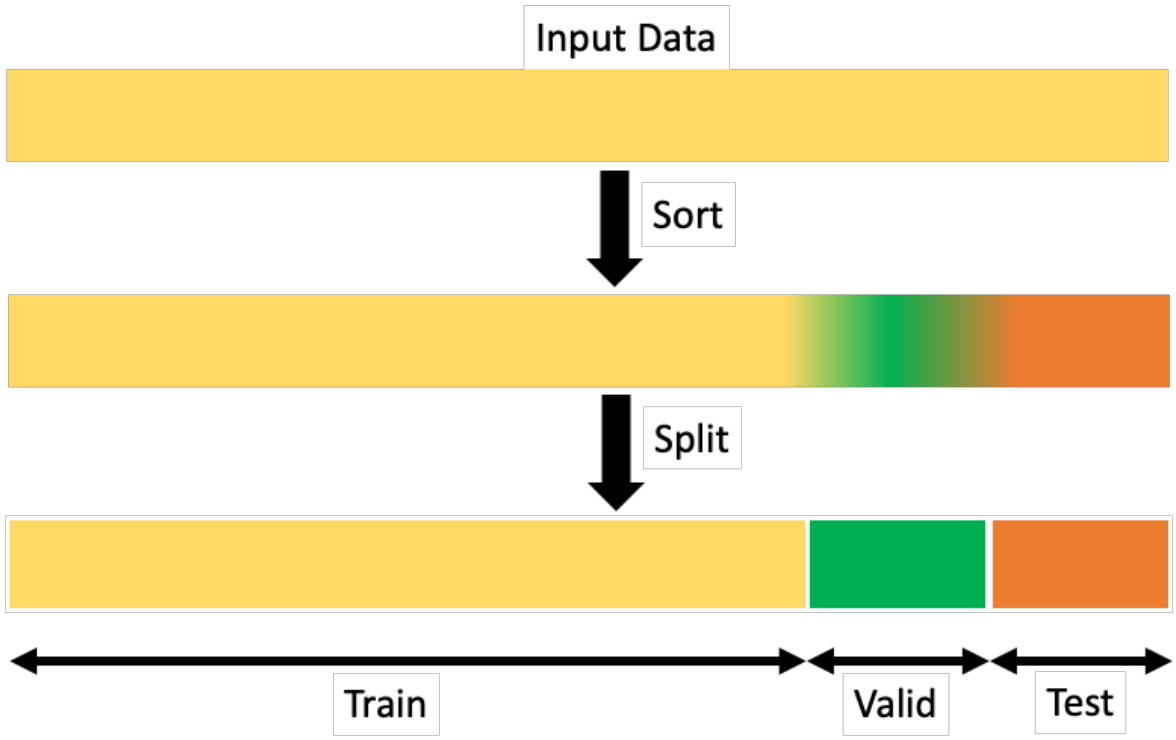
2) Splitting using the temporal component

You can listen to Jeremy Howard in his fast.ai lectures on Machine Learning: Introduction to Machine Learning for Coders. In Lesson 3, he talks about *"what makes a good validation set, and we use that discussion to pick a validation set for this new data." #2*

He uses an example, "Let's *say you are building a model to predict next month's sale. And if you have no way of knowing whether the model you have built is good at predicting sales a month ahead of time, then you have no way of knowing when you put a model in production whether it's going to be any good." #3*

Using that temporal variable is a more reliable way of splitting datasets whenever the dataset includes the date variable, and we want to predict something in the future. Hence we must use the latest samples for creating the validation and test dataset.

The main idea is always choosing a subset of samples representing the data faithfully in our model will receive afterward (whether we face a real-world problem or a Kaggle competition).



Train Valid Test Dataset after sorting the data. Image by Author

i. Custom code

In the code snippet below, you will learn how to write your custom code to create the train | valid | test dataset of our desired proportions after sorting the data. You can use this code directly after the slight modifications.

```
1  import pandas as pd
2
3  df = pd.read_csv('/kaggle/input/bluebook-for-bulldozers/TrainAndValid.c
```

```
# Let's say we want to split the data in 80:10:10 for train:valid:test
train_size = 0.8
valid_size=0.1

train_index = int(len(df)*train_size)

# First we need to sort the dataset by the desired column
df.sort_values(by = 'saledate', ascending=True, inplace=True)

df_train = df[0:train_index]
df_rem = df[train_index:]

valid_index = int(len(df)*valid_size)

df_valid = df[train_index:train_index+valid_index]
df_test = df[train_index+valid_index:]

X_train, y_train = df_train.drop(columns='SalePrice').copy(), df_train[
X_valid, y_valid = df_valid.drop(columns='SalePrice').copy(), df_valid[
X_test, y_test = df_test.drop(columns='SalePrice').copy(), df_test['Sal

```

ii. Using Fast_ml → 'train_valid_test_split'

In the code snippet below, you will learn how to use `train_valid_test_split` to create the train | valid | test dataset of our desired proportions after sorting the data. All of that in just a single line of code.

```
import pandas as pd

df = pd.read_csv('/kaggle/input/bluebook-for-bulldozers/TrainAndValid.c


from fast_ml.model_development import train_valid_test_split

```

```
X_train, y_train, X_valid, y_valid, X_test, y_test = train_valid_test_s


print(X_train.shape), print(y_train.shape)
print(X_valid.shape), print(y_valid.shape)
```

. . .

The science behind dataset split ratio

Often it is asked in what proportion to split your dataset into Train, Validation, and Test sets?

This decision mainly depends on two things. First, the total number of samples in your data, and second, on the actual model you are training.

- Some models need substantial data to train upon, so you would optimize for the more extensive training sets in this case.
- Models with very few hyper-parameters will be easy to validate and tune, so you can probably reduce the size of your validation set.
- But if your model has many hyper-parameters, you would want to have a significant validation set as well.
- If you happen to have a model with no hyper-parameters or ones that cannot be easily tuned, you probably don't need a validation set too.

References

#1 https://machinelearningmastery.com/difference-test-validation-datasets/ #2 https://www.fast.ai/2018/09/26/ml-launch/
#3 https://www.youtube.com/watch?v=YSFG_W8JxBo

Thanks for reading!!

- If you enjoyed this, follow me on medium for more.
- Interested in collaborating? Let's connect on Linkedin.
- Please feel free to write your thoughts/suggestions/feedback.
- Kaggle link
- Fast_ml link

Notebook is available at the following location with fully functional code:

train_valid_test_split instead of train_test_split

Explore and run machine learning code with Kaggle Notebooks | Using data from Blue Book for...

www.kaggle.com

Thanks to Anne Bonner.

Machine Learning | Data Science | Python | Analytics | Data Scientist