Understanding Cross Validation

29 September 2022 06:42

- Machine learning is an iterative process.
- You will face choices about predictive variables to use, what types of models to use, what arguments to supply those models, etc.
- We make these choices in a data-driven way by measuring model quality of various alternatives.
- You've already learned to use train_test_split to split the data, so you can measure model quality on the test data.
- Cross-validation extends this approach to model scoring (or "model validation.")
- Compared to train_test_split, cross-validation gives you a more reliable measure of your model's quality, though it takes longer to run.

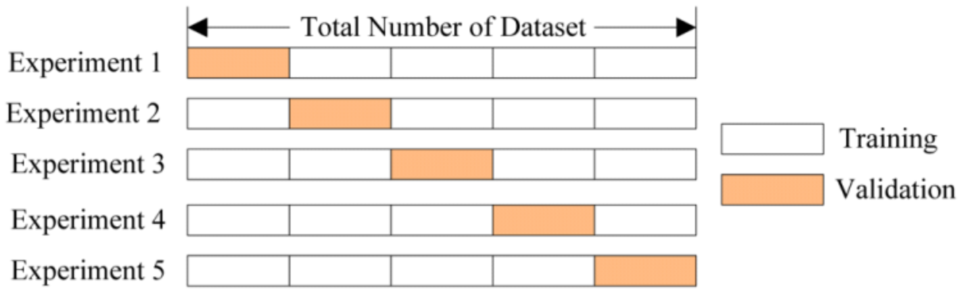
The Shortcoming of Train-Test Split

- Imagine you have a dataset with 5000 rows.
- The train_test_split function has an argument for test_size that you can use to decide how many rows go to the training set and how many go to the test set.
- The larger the test set, the more reliable your measures of model quality will be.
- At an extreme, you could imagine having only 1 row of data in the test set.
- If you compare alternative models, which one makes the best predictions on a single data point will be mostly a matter of luck.
- You will typically keep about 20% as a test dataset. But even with 1000 rows in the test set, there's some random chance in determining model scores.
- A model might do well on one set of 1000 rows, even if it would be inaccurate on a different 1000 rows.
- The larger the test set, the less randomness (aka "noise") there is in our measure of model quality.
- But we can only get a large test set by removing data from our training data, and smaller training datasets mean worse models.
- In fact, the ideal modelling decisions on a small dataset typically aren't the best modelling decisions on large datasets.

The Cross-Validation Procedure

07 October 2022 02:31

In cross-validation, we run our modeling process on different subsets of the data to get multiple measures of model quality. For example, we could have 5 folds or experiments. We divide the data into 5 pieces, each being 20% of the full dataset.



- We run an experiment called experiment 1 which uses the first fold as a holdout set, and everything else as training data.
- This gives us a measure of model quality based on a 20% holdout set, much as we got from using the simple train-test split.
- We then run a second experiment, where we hold out data from the second fold (using everything except the 2nd fold for training the model.)
- This gives us a second estimate of model quality.
- We repeat this process, using every fold once as the holdout.
- Putting this together, 100% of the data is used as a holdout at some point.

Returning to our example above from train-test split, if we have 5000 rows of data, we end up with a measure of model quality based on 5000 rows of holdout (even if we don't use all 5000 rows simultaneously.

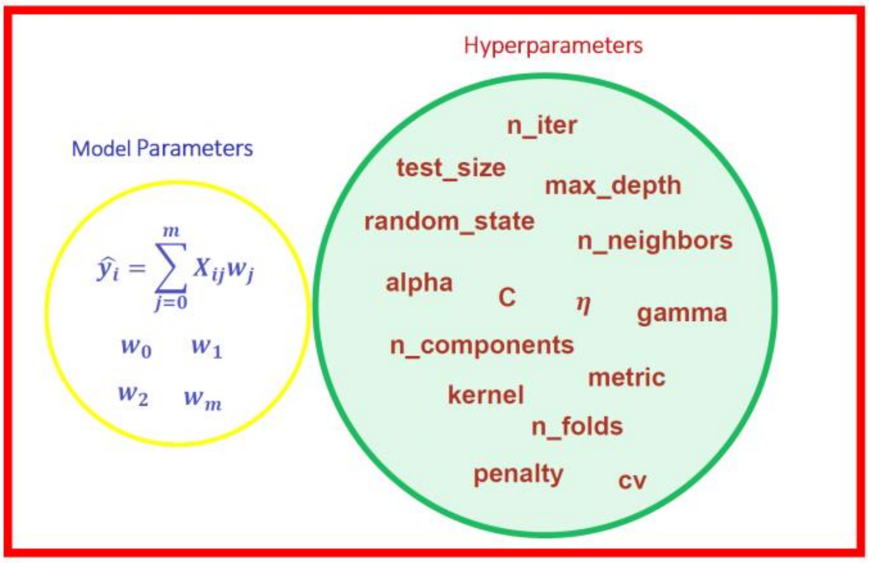
Trade-offs Between Cross-Validation and Train-Test Split

07 October 2022 02:33

- Cross-validation gives a more accurate measure of model quality, which is especially important if you are making a lot of modeling decisions.
- However, it can take more time to run, because it estimates models once for each fold. So it is doing more total work.
- Given these trade-offs, when should you use each approach?
- On small datasets, the extra computational burden of running cross-validation isn't a big deal.
- These are also the problems where model quality scores would be least reliable with train-test split.
- So, if your dataset is smaller, you should run cross-validation.
- For the same reasons, a simple train-test split is sufficient for larger datasets. It will run faster, and you may have enough data that there's little need to re-use some of it for holdout.
- There's no simple threshold for what constitutes a large vs small dataset.
- If your model takes a couple minute or less to run, it's probably worth switching to cross-validation. If your model takes much longer to run, cross-validation may slow down your workflow more than it's worth.
- Alternatively, you can run cross-validation and see if the scores for each experiment seem close. If each experiment gives the same results, train-test split is probably sufficient.

Understanding Parameters & Hyperparameters

07 October 2022 02:35



In a machine learning model, there are 2 types of parameters:

1. **Model Parameters:** These are the parameters in the model that must be determined using the training data set. These are the fitted parameters.
2. **Hyperparameters:** These are adjustable parameters that must be tuned in order to obtain a model with optimal performance.

For example, suppose you want to build a simple linear regression model using an m-dimensional training data set. Then your model can be written as:

$$\hat{y}_i = \sum_{j=0}^{m} X_{ij} w_j$$

where **X** is the predictor matrix, and **w** are the weights. Here w_0, w_1, w_2, ...,w_m are the **model parameters**. If the model uses the gradient descent algorithm to minimize the objective function in order to determine the weights w_0, w_1, w_2, ...,w_m, then we can have an optimizer such as GradientDescent(eta, n_iter). Here eta (learning rate) and n_iter (number of iterations) are the **hyperparameters** that would have to be adjusted in order to obtain the best values for the model parameters w_0, w_1, w_2, ...,w_m. For