More about validation and cross-validation

Previously, you learned how to split a dataset into training and testing data. Then, you fit a model to the training data and evaluated its performance on the test data. This basic process of building models with training data and evaluating them with held-out data is a fundamental part of machine learning and something that you should become very familiar with as a data professional. In this reading, you'll learn about validation and cross-validation, which are more rigorous ways of training and selecting a model.

Model validation

Fitting a model to training data and evaluating it on test data might be an adequate way of evaluating how well a single model generalizes to new data, but it's not a recommended way to compare multiple models to determine which one is best. That's because, by selecting the model that performs best on the test data, you never get a truly objective measure of future performance. The measure would be optimistic.

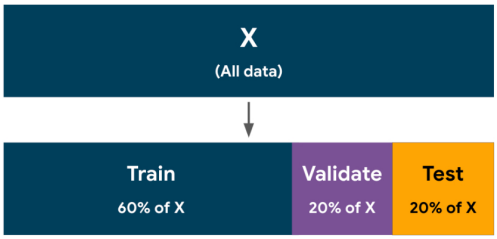
This may seem difficult to grasp or counterintuitive. You may be asking yourself: How is it not objective? After all, I'm not using the test data to tune my models. Well, if you're comparing how all of the models score on this data and then selecting the model with the best score as champion, in a way, you're "tuning" another hyperparameter—the model itself! The selection of the final model would itself behave as a tuning process, because you'd be using the test data to go back and make an upstream decision about your model. Put simply, if you want to use the test data to get a true measure of future performance, then you must never use it to make a modeling choice. Only use the test data to evaluate your final model. As a data professional, you will likely encounter scenarios where the test data is used to select a final model. It's not best practice, but it's unlikely that it will break your model. However, there are better, more rigorous ways of evaluating models and selecting a champion.

One such way is through a process called **validation**. Model validation is the whole process of evaluating different models, selecting one, and then continuing to analyze the performance of the selected model to better understand its strengths and limitations. This reading will focus on evaluating different models and selecting a champion. Post-model-selection validation and analysis is a discipline unto itself and beyond the scope of this certification.

Validation sets

The simplest way to maintain the objectivity of the test data is to create another partition in the data—a validation set—and save the test data for after you select the final model. The validation set is then used, instead of the test set, to compare different models.

Here is one common way of splitting data, but note that these proportions are not required. You can split to whichever ratios make the most sense for your use case.



This method—using a separate validation set to compare models—is most commonly used when you have a very large dataset. The reason for this is that the more data you use for validation, the less you have for training and testing. However, if you don't have enough validation data, then your models' scores cannot be expected to give a reliable measure that you can use to select a model, because there's a greater chance that the distributions in the validation data are not representative of those in the entire dataset.

When building a model using a separate validation set, once the final model is selected, best practice is to go back and fit the selected model to all the non-test data (i.e., the training data + validation data) before scoring this final model on the test data.

Cross validation

There is another approach to model validation that avoids having to split the data into three partitions (train / validate / test) in advance. **Cross-validation** makes more efficient use of the training data by splitting the training data into k number of "folds" (partitions), training a model on k – 1 folds, and using the fold that was held out to get a validation score. The training process occurs k times, each time using a different fold as the validation set. At the end, the final validation score is the average of all k scores. This process is also commonly referred to as k-fold cross validation.

Cross-validation (5-fold):

1	2	3	4	5
Validation fold	Train	Train	Train	Train
Train	Validation fold	Train	Train	Train
Train	Train	Validation fold	Train	Train
Train	Train	Train	Validation fold	Train
Train	Train	Train	Train	Validation fold

After a model is selected using cross-validation, that selected model is then refit to the entire training set (i.e., it's

retrained on all *k* folds combined).

The cross-validation process maximizes the usefulness of your data with the goal of getting a more accurate measure of model performance. It does so by averaging out the randomness imparted when splitting into training and validation folds. In other words, any time a dataset is split, the specific samples that go into each partition are usually random, which makes it possible for the distributions in each partition to diverge from those found in the full dataset.

Cross-validation reduces the likelihood of significant divergence of the distributions in the validation data from those in the full dataset. For this reason, it's often the preferred technique when working with smaller datasets, which are more susceptible to randomness. The more folds you use, the more thorough the validation. However, adding folds increases the time needed to train, and may not be useful beyond a certain point.

Model selection

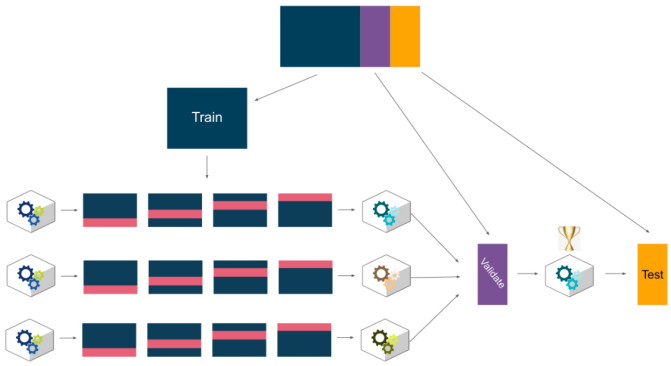
Once you've trained and validated your candidate models, it's time to select a champion. Of course, your models' validation scores factor heavily into this decision, but score is seldom the only criterion. Often you'll need to consider other factors too. How explainable is your model? How complex is it? How resilient is it against fluctuations in input values? How well does it perform on data not found in the training data? How much computational cost does it have to make predictions? Does it add much latency to any production system? It's not uncommon for a model with a slightly lower validation score to be selected over the highest-scoring model due to it being simpler, less computationally expensive, or more stable.

Once you have selected a champion model, it's time to evaluate it using the test data. The test data is used only for this final model. Your model's score on this data is how you can expect the model to perform on completely new data. Any changes you make to the model at this point that are based on its performance on the test data contaminate the objectivity of the score. Note that this does not mean that you can't make changes to the model. For instance, you might want to retrain the champion model on the entire dataset (train + validate + test) so it makes use of all available data prior to deployment. This is acceptable, but understand that at this point you have no way of meaningfully evaluating the model unless you acquire new data that the model hasn't encountered.

A review of the model development process

There is no single way to develop a model. Project-specific conditions will dictate the best approach. Over the course of your development as a data science professional, you'll likely encounter different variations of the train-validate-test process, some of which are more rigorous than others.

A rigorous approach to model development might use both cross-validation *and* validation. The cross-validation can be used to tune hyperparameters, while the separate validation set lets you compare the scores of different algorithms (e.g., logistic regression vs. Naive Bayes vs. decision tree) to select a champion model. Finally, the test set gives you a benchmark score for performance on new data. This process is illustrated in the diagram below.



For variations of this process, refer to the appendix.

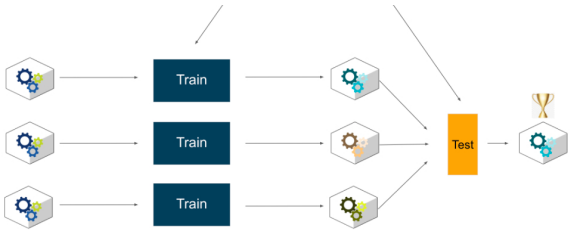
Key takeaways

- Model validation is the whole process of evaluating different models, selecting one, and then continuing to analyze the performance of the selected model to better understand its strengths and limitations. (Note that each unique combination of hyperparameters is a different model).
- Validation can be performed using a separate partition of the data, or it can be accomplished with cross-validation of the training data, or both.
- Cross-validation splits the training data into *k* number of folds, trains a model on *k* – 1 folds, and uses the fold that was held out to get a validation score. This process repeats *k* times, each time using a different fold as the validation set.
- Cross-validation is more rigorous, and makes more efficient use of the data. It's particularly useful for smaller datasets.
- Validation with a separate dataset is less computationally expensive, and works best with very large datasets.
- For a truly objective assessment of model performance on future data, the test data should not be used to select a final model.

Appendix

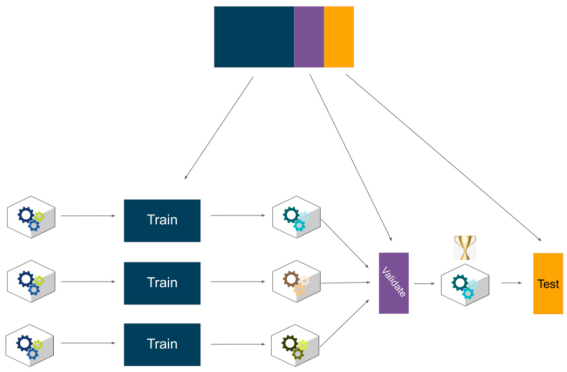
The following diagrams depict some variations of the model development process. Consider the implications of each choice.

A



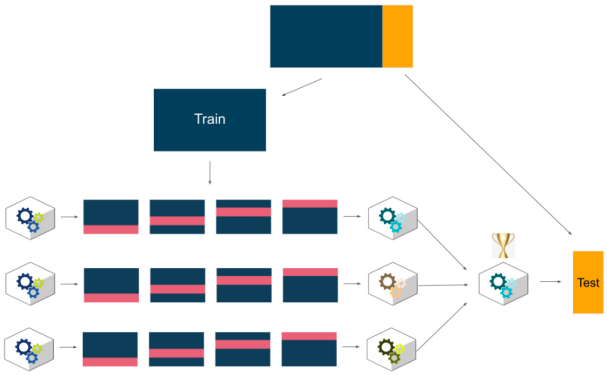
In this simple development scheme, data is split into train and test sets. Models are trained on the train data and all tested on the test data. The model with the best score on the test data is selected as champion. This approach does not iteratively tune hyperparameters or test the champion model on new data. The champion model's score on the test data would be an optimistic indicator of future performance.

B



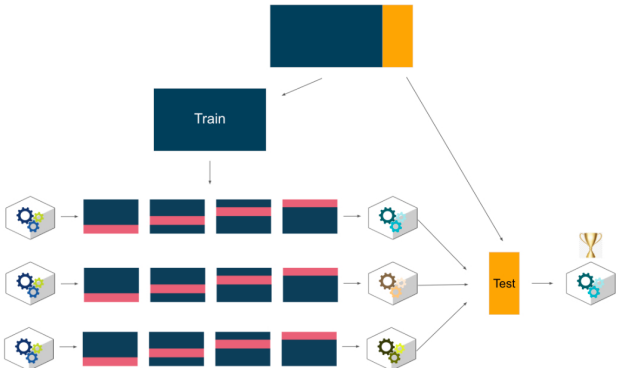
In this approach, the data is split into training, validation, and testing sets. Models are fit to the training data, and the champion model is the one that performs best on the validation set. This model alone is then scored on the test data. This is the same approach as example A, but with the added step of testing the champion model by itself to get a more reliable indicator of future performance.

C



This method splits the data into train and test sets. Models are trained and cross-validated using the training data. The model with the best cross-validation score is selected as the champion model, and this model alone is scored on the test data. The cross-validation makes the model more robust, and using the test data to evaluate only the champion model allows for a good understanding of future performance. However, selection of the champion model based on the cross-validation results alone increases the risk of overfitting the model to the training data.

D



In this variant, the data is split into training and testing sets. Models are trained and cross-validated using the training data, then all are scored on the test data. The model with the best performance on the test data is the champion. This is a very common approach, but note that it does not score the champion model on completely new data, so expected future performance may be optimistic. However, compared to approach C above, this method mitigates the risk of overfitting the model to the training data.

Mark as completed