Introduction to Machine Learning in Python

Presented by Nicholas Miklaucic

2024-07-22

Outline

Outline

Machine Learning (ML)

Machine learning (ML) is when a computer does something through some kind of experience, not just a pre-programmed algorithm.. More formally:

> A computer program is said to learn from experience E with respect to some class of tasks T, and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.
>
> — Tom Mitchell, **Machine Learning**

Artificial Intelligence (AI)

Artificial intelligence (AI) has a broader definition: computers that behave intelligently in some way. This is broader than machine learning and includes algorithms that don’t learn directly from experience.

We won’t be talking about non-ML AI right now, so let’s get back to ML.

Supervised Learning

- Prediction of some *target*, often y, from labeled training data, often X.

There are two broad kinds of supervised learning:

- *Classification* predicts one of a set of discrete *classes*, like diagnosing a disease.
- *Regression* predicts a numerical output, like predicting the price a house will sell at.

Figure 1: Zillow thinks this house is worth $4 million.

Figure 2: A ML model could classify this image as a picture of a dog.

Supervised Learning

Consider this an unofficial exercise 0: what are other examples of classification and regression? What examples might appear in materials science?

Outline

Evaluating ML Models

If we have a model making predictions, how do we assess how good our prediction is?

Let's call what our model predicts $\hat{y}$ ("y-hat"), and we have the actual y to compare it with.

Classification

We can plot $\hat{y}$ vs. y in a *confusion matrix*. This is basically all of the important information we need to evaluate the model. Sometimes we use the raw counts, sometimes we normalize by row like here.

The most common metric is *accuracy*: the percent of correct predictions. (This is the fraction of values on the main diagonal.)

There are many other metrics that can be better when classes are imbalanced. For now, we'll stick with accuracy, but looking at the full confusion matrix gives a lot more information.

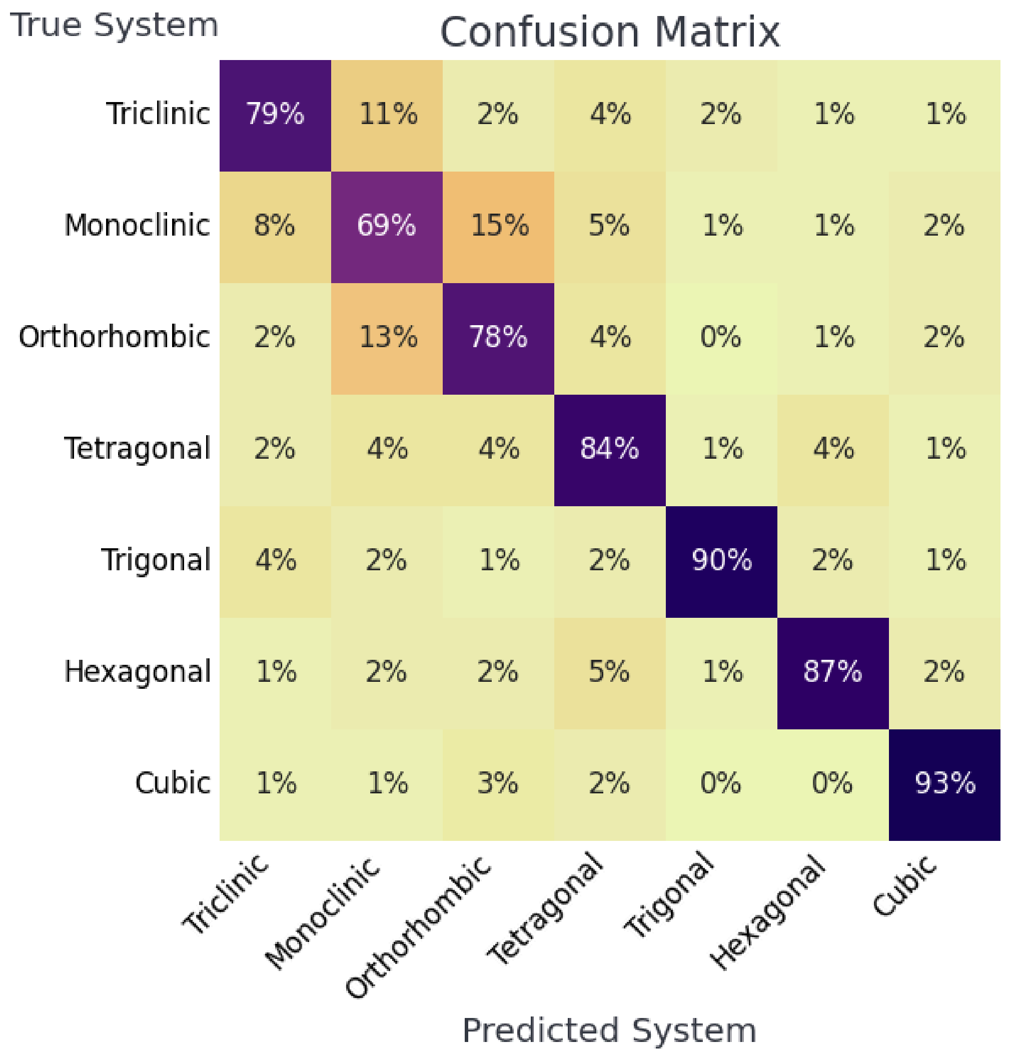


Figure 3: Reproduced from Li, Dong, Yang, & Hu 2021

Regression

We can also plot the predicted vs. actual values, as shown here.

There are a couple common metrics you will see:

Mean Squared Error (MSE): $\text{mean}\left((y - \hat{y})^2\right)$

Root Mean Squared Error (RMSE): $\sqrt{\text{mean}\left((y - \hat{y})^2\right)}$

Mean Absolute Error (MAE): $\text{mean}(|y - \hat{y}|)$

R^2: $1 - \dfrac{\text{Var}(y - \hat{y})}{\text{Var}(y)}$

MSE is in units of the target squared. RMSE and MAE are in the same units as the target: they're different versions of "average" error. R^2 is unitless: it measures the percent of the variance in the target we can explain with the inputs, so 0 is the same as no information and 1 is perfect.

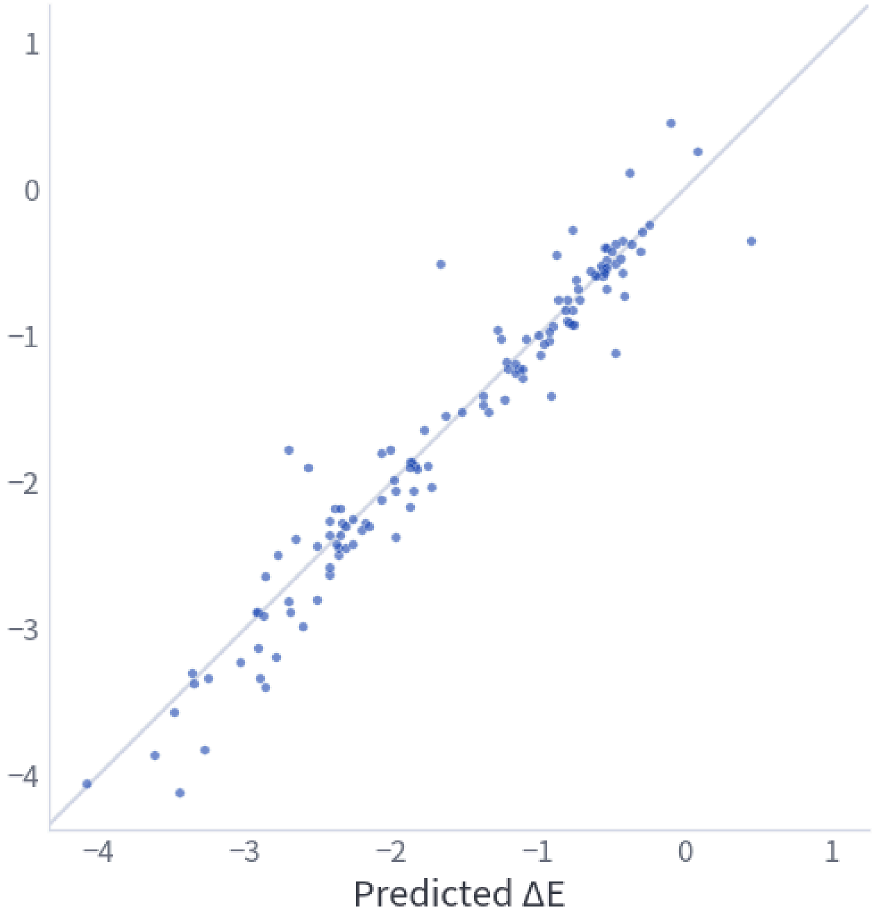


Figure 4: Formation energy prediction

Validation

We want to know how well our model will do on *new* inputs. If we can compute accuracy or MAE, then we already know those answers! Scientists don't want AI to tell them what they already know.

Moreover, performance on training data isn't predictive of future performance. Models can *overfit*, finding patterns that don't hold up.

We want to see how our model will do on data it hasn't seen before. How can we do that?

Hold-out Set

We can split our data into sets. Let's say we have 100 *samples*.

If we train our model on 80 of those samples, and then test it on the remaining 20, we have a better estimate of our model's future performance.

If we have a ton of data, this works well. If we don't have enough data, we face a tricky dilemma:

- If we use too much data for validation, our model won't have enough training data, and our performance will be much lower than it would be had we trained on the whole set.
- If we use too little data for validation, the estimates will be unreliable, and randomness in how our validation set is chosen will end up dominating the results.

Cross-Validation

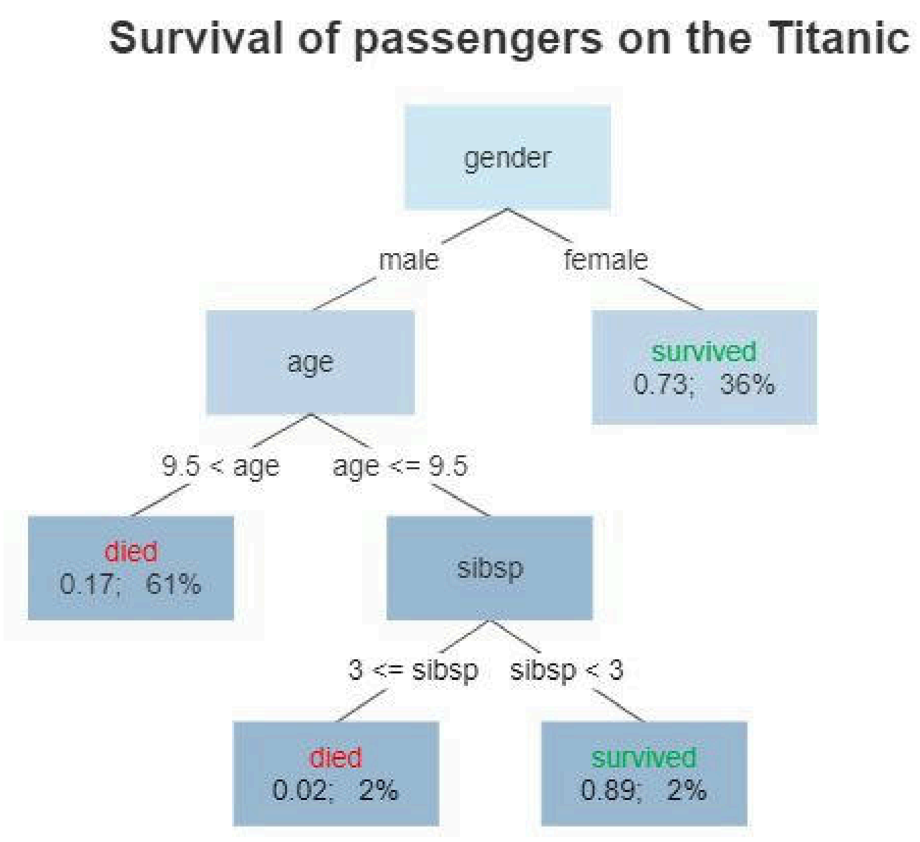
We can get the best of both worlds by training multiple models.

If we split our 100 samples into groups of 20, called *folds*, we can train 5 models. Each model is trained on the other 80 samples and then predicts for a single fold. At the end, we have predictions for each sample, none of which were tainted by the model being able to train on the correct answer.

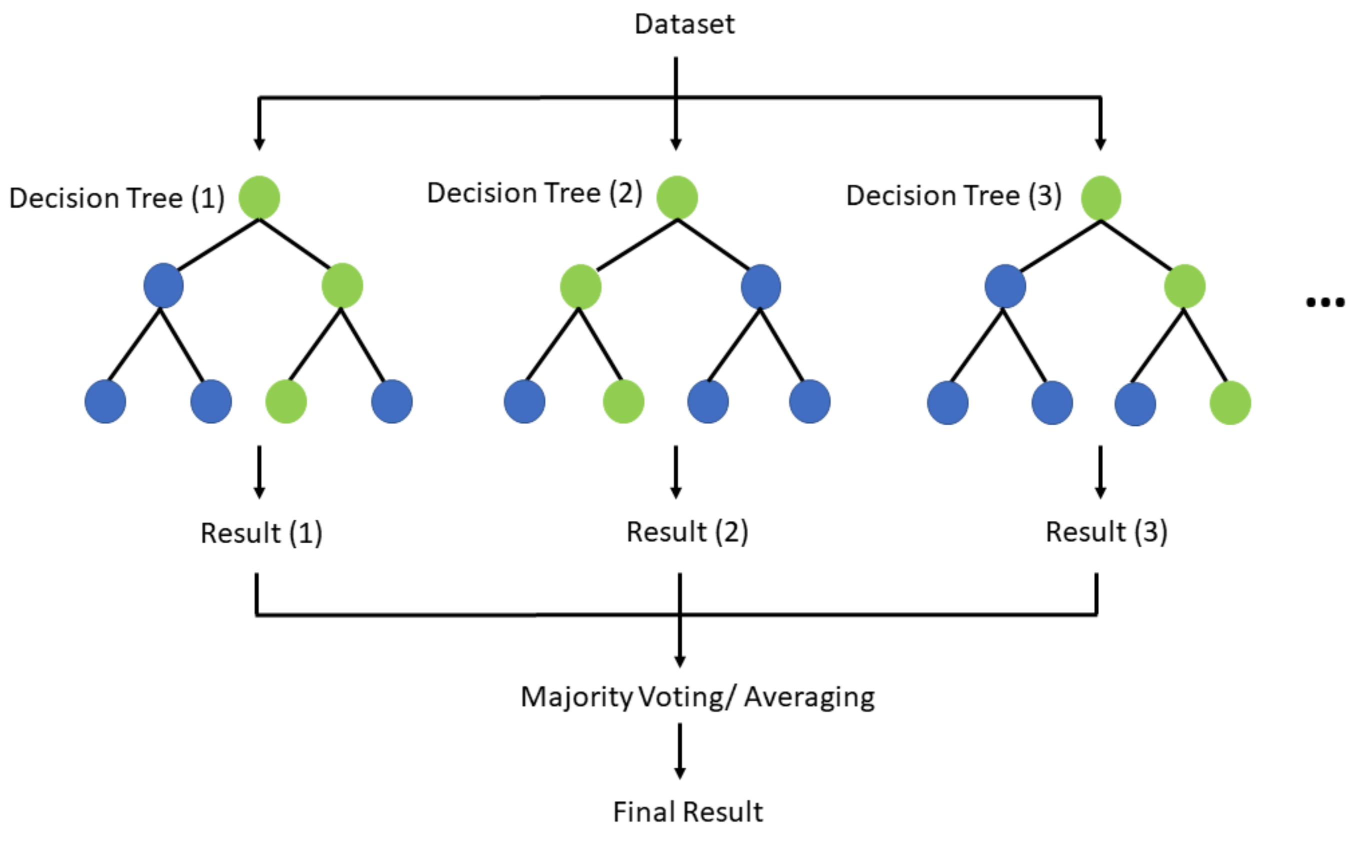
Now our tradeoff is purely computing power. The ideal is *leave-one-out cross-validation* (LOO-CV), where we train a model for every data point. LOO-CV is the best estimate of future performance, but it requires orders of magnitude more computing power. Normally, we use CV with 5 or 10 folds as a more reasonable baseline.

Outline

Decision Tree



Random Forest



Random Forest

Outline