Introduction to ML

Supervised vs Unsupervised

Unsupervised learning: No prediction involved, just finding patterns.

Supervised learning: Prediction

Sometimes, can be used together (semi-supervised learning), or can get some interpretation of relationships from ML models (interpretable ML).

What can you study with ML?

Recall: Focus is on prediction

- We are not making conclusions about associations between variables

Considerations:

- Ethical issues with how these applied.
- Bias, fairness very important but can be hard to detect or fully mitigate.

Ethical Implications

Considerations:

- What is the intervention? Is it punitive? Is it beneficial?
- Who does the ML model affect?
- How is affecting different groups of people differentially?

Example: When predicting recidivism, what if the model was built primarily on data with white inmates and ended up only being beneficial for that population?

The Machine Learning Process

Goal: Make predictions that are as good as possible.

In other words, when making decisions about models, the answer on what to do is always, **“The one that improves prediction on the test set.”**

Example: How you determine what set of predictors to use when building a linear model? Under the ML framework, you **choose the one that has the best performance on the test set.**

Performance Measures

Accuracy: Proportion of correct predictions.

Precision: Proportion of positive predictions that are actually positive.

Recall: Proportion of all positives that were predicted positive.

Accuracy

Good when you **don’t have a preference for any particular category** that you are predicting.

Good when you have **more than two categories**.

Example: ML is used to automate categorization from text. Accuracy is a common measure in these cases.

Precision

Good when you want to make sure that **your prediction that it is a positive case is correct.** Many times, this might be because the intervention has a negative effect.

Example: Using ML to predict whether to run an expensive ad campaign. Since it is expensive, you really want to make sure that you are right that it will be successful.

Recall

Good when you want to make sure that **get as many positive cases as possible.** Many times, this is when the intervention is beneficial and the downside of being wrong isn't very high.

Example: Predicting a baby is at high risk of lead poisoning when they actually aren't is ok, since the intervention won't hurt them. But, we want to prevent as many cases of infant lead poisoning as possible.

Relationship Between Precision and Recall

Suppose we use **logistic regression to predict a binary outcome**, and we get **predicted probabilities** on the test set. We can predict above 50% as 1 and below as 0, or we can set a different **threshold**.

To get perfect precision: Only predict above 99% probability to be a 1.

To get perfect recall: Predict everything above 0% probability to be a 1.

(In reality, we want something in between these two extremes.)

Other Performance Measures

F1 score is a commonly cited one and is meant to have **a balance between precision and recall**.

Good when you have imbalance outcomes and no preference between precision and recall.

Many times, however, you will want to have an emphasis on one or the other.

Other Constraints

It is important to think about the real-world application.

Example: What is the best way to have 100% recall for preventing lead poisoning in babies?

Predict yes to all, provide intervention for all babies.

Why does this not work?

Budget won't allow us to do this. So, we need to make sure that we find highest recall for the number of babies we can intervene on.

A note on the code

We will eventually use the **caret** package to automate the cross-validation and overall machine learning process. However, the code in the tutorials and R material might have loops to “manually” do this.

This is for **demonstration purposes only.**

Computation

This is something that is changing constantly as computers get better and better.

- This is why neural networks are much more popular now.

In general, we won't notice real slowdowns in code until we get to Boosting.

- XGBoost is a powerful method that is designed to run faster and more efficiently.