CAPS Notes - Lecture 2

Supervised Learning

Leo Klenner, Henry Fung, Cory Combs

Last updated: 11/2/2019

Overview

In lecture 2, we begin our foray into supervised learning. To begin, we start with a crucial component of the infrastructure that allows us to use machine learning: data. From there, we explore model construction, applications, and the means to assess machine learning systems .

As we enter the technical portion of the course, we aim to provide the means to engage with engineers and to understand the power and limitations of these systems. There are books worth of material for each of these topics, and we emphasize that the coverage here is far from complete; rather, it reflects our assessment of what you *need* to know to get started in AI policy and strategy, and aims to provide solid launching points for continued engagement.

The session 2 notes cover:

- Data
- The branches of supervised learning
- The basics of supervised learning model construction
- Regression models
- Classification models
- Tree-based models for classification and regression
- Model evaluation
- Data labeling system case study

> *Information is the oil of the 21st century, and analytics is the combustion engine.*
>
> *- Peter Sondergaard*

1. Engaging Data

Modes of Engagement

In a course, say that an instructor gives you sample problems. You try to solve them, then the instructor informs you whether your answers are correct or incorrect. If anything is amiss, you try again, adjusting your responses. You are not taught new methods; you are only told that your answers were incorrect and you must try again using the given methods. In an ideal world, you continue to practice and iterate until your success rate is quite high and you can successfully answer new problems of the same form. This is a human analogy to **supervised learning**.

In contrast, imagine an pattern-finding exercise in which you have no training problems, no context, and no feedback. You are alone with your observations and your logical capabilities. What patterns will you identify? No doubt reasonable results may differ between people with slightly different processes - and all could, in principle, lead to different insights. This is a human analogy to **unsupervised learning**.

Moving from analogies to more precise machine learning specifications:

- **Supervised learning** allows us to train machines that *learn from labeled data*. This includes making inferences and predictions about new, unlabeled observations based on previous observations. This includes regression and classification.
- **Unsupervised learning** allows untrained machines to autonomously *learn about unlabeled data*. This includes pattern recognition and categorization. (The distinction between categorization and classification will become clear in the next session.)

Today, we focus on supervised learning, the most commonly deployed form of machine learning in current industry and a basis of much economic and financial analysis. Supervised learning is highly restricted by the available data, requiring not only accurately labeled data appropriate for each unique use case, but very large amounts of it. These requirements pose significant challenges to achieving "good" outcomes, for which we will focus on three criteria: *correct* inferences, *useful* results, and *ethical* implementation. All of these come down, at some point, to the data used to train the model.

Features and Observations

We can decompose all datasets into two components:

- **Features**: the variables in a dataset (the columns in a table).
- **Observations**: the number of unique records (the rows in a table).

As an example:

ID [Observation Number]	Province [Feature 1]	Security Perception [Feature 2]
3032	Badghis	3
3033	Samangan	2
3034	Wardak	5

Table 1: Example compilation from USAID MISTI Survey Dataset features. A security perception value of 1 indicates a response of "improved a lot"; 2 of "improved a little"; 3 of "stayed the same"; 4 of "worsened a little"; and 5 of "worsened a lot". Explore the MISTI dataset here: 1.

Note that the data may include both continuous (numerical) and categorical variables (e.g. province names). Machine learning models require numerical data for calculation purposes; to achieve this, categorical variables are transformed into numerical values for training and evaluation purposes. As Table 1 demonstrates, some datasets may already represent some categorical data in numerical form, while leaving other entries in their original form. Adapting variables for modeling is part of the data cleaning and manipulation process and is taught as part of data science.

Labels

Where do *labels* come into play?

- **Labels refer to target variables.** To say that a supervised learner "trains on labeled data" means that it learns to correlate a specified target variable with the selected features. When a supervised learner predicts a target variable, it is predicting a label.
- If no variable serves as a valid target variable, the problem is not suitable for supervised learning (unless or until a suitable target variable is added).
- Features selected to predict the target variable become the hypothesized independent variables.

To restate the supervised vs. unsupervised distinction:

- Supervised learners train to *predict* or *infer* a target variable based on a range of features, or independent variables, using examples for which the target variables are already known.
- Unsupervised learners do not train, and run various algorithms to find patterns among the features.

Given real-world datasets, this poses an important question: if you stumble upon data "in the wild", how do you know whether it is suitable for supervised learning? In principle, the solution has two components: the question you want to answer, and a judgment call about the features available. If you want a model to learn to *predict* a target variable, you will pursue supervised learning; however, you must then separately assess whether the target variable *can be reasonably well predicted* given the available features. This requires hypothesis testing of the form familiar from regression analysis, much of which is best served by domain expertise.

Many datasets are constructed for sake of answering a particular question. Most well-designed experimental studies produce results with explicit target variables and features selected through principled hypothesis formation. Such experimental data collection is expensive and usually not conducted without rigorous specifications. On the other hand, less targeted large-scale data collection is more and more accessible, particularly in the world of "big data"; here, data may be collected without clear prior expectations. Data science is in large part the science of parsing datasets for useful results. Another major component is cleaning and preparing the data for use. Selecting, understanding, and processing data are essential foundations for all machine learning.

> *In God we trust. All others must bring data.**
>
> *- *W. Edwards Deming*

"Better" data

One often hears that more data beats a better model. In general, this is on the right track. But this generalization requires a vital caveat: more data is *not automatically* better than less data. Rather, more *relevant* data is better than less applicable data. More bad data changes nothing. While this sounds quite obvious, when researching solutions to a problem it can be tempting to go after more readily available datasets - or the data you seek may not be available at all, making

a potential proxy seem all but necessary. Choosing relevant data is, ultimately, a matter of **feature selection**: what are you asking your model to consider, and why?

Machine learning follows the precept: Garbage In, Garbage Out.

Irrelevant features, or even partially relevant features, can cause a model to make inferences that do not hold beyond the training set. In a human analogy, it is taught an inaccurate view of how the world works. Given new data, its predictions and inferences will not hold.

Given relevant features, the model still needs sufficient examples of each feature to be able to discern signal from noise. Given the noisiness of the real world, "sufficient" can often means tens of thousands up to millions of data points, if not more.

A key take-away for assessing supervised learning models:

- **Appropriate feature selection** and **numerous observations for each feature** are **joint necessities** for successful models.

In the final section, we will explore formal metrics of model performance. However, while we have robust tools to assess whether a model performs well, identifying features to measure and test in the first place can be as much art as science. Domain expertise serves a vital role in feature selection.

Key Data Risks: Overfitting and Bias

We now turn to two of the most critical data-related risks in supervised learning: overfitting and bias. The former is primarily a function of how the algorithm learns from training data, and can be adjusted without changing the data. The latter is primarily a function of data selection, and generally requires changing what data is used to train the model.

Overfitting indicates that a model has learned to map *noise*, rather than real, meaningful *signals* in the training data. An overfitted model will reflect trends in the training data pool that *do not exist* in other data pools. In other words, an overfitted model will make inaccurate predictions for the data you want to analyze, for example showing 99% accuracy on training data but merely 50% on new data. (Underfitting is also possible, though a much less common issue in today's data-rich era.) Overfitting is something we can - and must - test for and guard against, using a variety of metrics depending on the context. We will discuss key metrics shortly.

Bias in the algorithmic context is, most simply, a systematic skewing of results. The term is fraught with other meanings, however, from the purely mathematical to the purely ethical, with different roles even across machine learning types (e.g. it is an essential ingredient of neural networks). As such, when people speak of "bias" in AI, they can mean many things. (This is enabled by a history of jargon adoption across fields - hence the importance of being specific in these discussions!) For now, we will consider the general, technical sense of bias as a skewing of results to a examine a key property of predictive supervised learning models: the bias-variance trade-off. Bias in the sense of AI ethics is discussed in a future session.

2. Data Collection Under Uncertainty

Data is the heart of all analysis; turning data into useful insight is our central goal. With the above content all in place, we turn to a case study on the challenge of collecting relevant data for open-ended problems.

3. Two Branches: Classification and Regression

Supervised learning has two principal branches:

- **Classification**, in which the model predicts a categorical target variable.
- **Regression**, in which the model predicts a continuous (numerical) target variable.

Consider two representative situations:

1a) *You are a loan officer responsible for determining whether to approve appplications based on whether each applicant is likely to default. The market you are responsible for has been historically stable and you have tens of thousands of records on hand to judge past experience of similar applicants.*

2a) *You are a USAID analyst evaluating the impact of renewable energy credits on household emissions outcomes. You have data from a well-designed randomized control trial collected from tens of thousands of households representing the majority of the population.*

What modes of analysis are most appropriate for these situations?

- Situation 1a) calls for **classification**: to predict whether the given applicant is likely to default on their loan.
- Situation 2a) calls for **regression**: to test statistical correlations between factors. The target is a continuous variable.

In every supervised learning case, we should begin with two key questions:

- Are we likely to have **enough observations** for robust results?
- Are we likely to have **relevant features** for robust results?

Each is a complex question and the subject of considerable study. In practice, this should be an opening discussion with the engineers on your team. For now, we will assume that tens of thousands of observations is sufficiently robust for both problems on the basis that we have a statistically representative spread of results in each case.

For the relevant features, the question is open: in an ideal world, **what features would *you* select**? What are **potential issues with each**?

Now consider variants of the above two situations, the first for classification and the second for regression:

1b) *You are a loan specialist developing a national micro-lending platform for an emerging market with historically limited small-scale lending. You have a small pool of data from low-n trial programs in three cities.*

2b) *You are a State Department analyst assessing the impact of a stabilization initiative on local security using survey data. You have just under one thousand responses collected by volunteers in their home regions.*

Revisit the two key questions regarding observations and features. We now have new, serious concerns about the data.

For situation1b):

- Is there enough historical data for statistically significant results?
- Is the market changing in such a way that the existing data will no longer be representative, i.e. the values for the same features would be different today?
- Is the market is changing such that different features are relevant?

For situation 2b):

- Are the survey results sufficiently complete for statistically significant results (considering NA values and missing entries)?
- Given the subjective nature of survey data, what can we, and can we not, infer from the dataset?
- How can and should we assess the quality and representativeness of the data?

Some questions, e.g. for the survey data, should be accounted for by research design. Non-experimental data, however, rely almost solely on separate research and domain expertise.

Based on the new contexts, what features would you select for each case? How might the different *environments* change your approach to feature selection? There are a range of reasonable responses. The goal for this discussion is simply to open the proverbial box.

4. Introduction to Supervised Learning Model Creation

Basics of Model Development

We turn now to the foundations of supervised learning model development. This knowledge empowers direct engagement with engineers during the development process and anchors our understanding of how keys policy-relevant risks arise from model implementation.

A highly simplified template of model development is as follows:

1. Data preparation
2. Model construction
3. Model training
4. Model testing
5. Model deployment

Data preparation - including cleaning and manipulation - is the cornerstone of all data science. Without it, no model development is feasible. Model training and testing are highly iterative; based on evaluation, a model will undergo multiple reconfigurations to maximize accuracy, mitigate bias, and otherwise optimize performance. Deployment indicates that a model has gone live and is being used as intended; at this stage, as the Knight Capital example demonstrated, changes must be made with great caution.

In the next sections, we explore model construction, training, and testing, as the most essential components for later policy considerations.

High-level model construction overview

Machine learning models involve sophisticated algorithms that implement a variety of statistical and mathematical tools. Happily, the machine learning field has developed and made public a variety of tools that wrap these algorithms into reusable, high-level processes. Following the data cleaning and manipulation stage, baseline model construction follows a tidy prescription:

1. **Split** the data into training and test datasets
2. **Build** the model
3. **Fit** the model to the training data
4. **Predict** values using the test data
5. **Evaluate** the model's performance

Let's explore the logic behind the procedure.

Train-test-split

Supervised learning trains on labeled data. However, in addition to training the model on labeled data, we need to *test* it on labeled data to properly assess its performance. Hence, rather than training on all available data, we want to set aside a subset of the data that is hidden from the model. Hence, we split our data into two subsets, traditionally called *train* and *test*. (For those in the know: we are setting aside validation datasets for the time being.)

As the names imply, the model will be trained on *train* and then tested on *test*; if the performance using the training set is much higher than on the test set, the model is likely overfitting to the training data and failing to generalize to the test set.

The proportion of the dataset assigned to each of the train and test datasets can vary. Assuming the features are relevant and well designed, more training data *generally* means better performance in the long run. However, more test data enables better insight into potential overfitting and underfitting, enabling designers to correct errors before the model is deployed. This direct trade-off is constant and unavoidable, and mitigated most simply by ensuring a large enough labeled dataset.

What about inconsistencies from different splits? If you are thinking that the *way* the data is split might affect the outcomes, you are absolutely correct. How can this be overcome? One simple technique is known as **cross-validation**. Cross-validation splits the data into a specified number of *different* training and test sets; trains and tests the model using each split; and compares the results of each case. The outcomes may be averaged or dealt with in any number of related ways, with the outcome of smoothing the potential for bias in any given random selection.

As a key take-away: if a potential problem is statistical in nature, ask the engineer on your team how it is addressed, or if it comes with a trade-off, how it is balanced. From the policy side, there may be outstanding problems posed by unique circumstances; statistical challenges, on the other hand, are frequently endemic to the modeling process, meaning that much research has gone into their management.

Instantiate, fit, predict, evaluate

With the reasoning behind the train-test-split step in place, the rest of the procedure is fairly predictable.

Instantiating the model is a formal Python way to say "tell your program which machine learning model to use". We will explore key options shortly. Most contemporary model construction is abstracted, meaning that, rather than writing out lines of linear algebra and calculus, we use existing code libraries and packages that have all of those details saved, letting us worry only about the data and specific parameters of interest. For example, we might specify "logistic regression" or "random forest" as our model.

Fitting the model means training the model on the training data. This is also where the terms "underfitting" and "overfitting" come in. Supervised learning models work to minimize error rates (also known as loss); in other words, to produce results close to the true values, i.e. fitting results to the truth. "Training" is a more general term: you can overfit a model, but it isn't "overtraining" the model - it is simply training it in a way that leads to suboptimal performance.

Predicting refers to running the trained model on the test data. In this phase, the model predicts the test data's labels.

Evaluation compares the model's predicted labels to the actual test data labels to see how well the model performed. Evaluation should occur after every round of training. The results of evaluation inform the designers how they should change the model or various parameters to improve performance. There are myriad metrics used to evaluate models, depending on the model type and specific application. There is no magic number for any metric; instead designers will look to maximize particular metrics of interest, or achieve a balance among measurable trade-offs, depending on the goals. Evaluation is one of the most essential topics in modeling.

5. Regression Models

Regression is the bread and butter of continuous variable prediction. At its core, regression models involve finding a line of best fit among a group of data based on established correlations between variables, enabling new data to be placed along the line based on one dimension (a selected feature) to estimate the likely value in the other dimension (the target variable).

Linear regression is the most common form of regression. More complex models are abundant, and often lead to higher accuracy among the training data, but are highly prone to overfitting and failing to generalize to new data. As such, these are generally only used in cases of a well-established nonlinear correlation.

Mathematically, linear regressions relate independent variables (features) x_n to the dependent variable (target variable) y in the following general function:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n + \epsilon$$

where β_i values indicate the weights associated with each independent variable and ϵ (epsilon) indicates an error term. When given actual values for the β_i terms, this becomes a **hypothesis** predicting how the variables are related.

The central task of linear regression is to calculate the coefficients, or weights, that optimally predict y based on the independent variables. That is, to find the best hypothesis, which will be applied to new data in the future.

"Optimal" prediction, as discussed in session 1, suggests "minimal error". Here, the goal is to achieve a minimal error term ϵ across the space of predictions, which indicates maximal fit to the true values. At this point, the earlier-discussed considerations of overfitting apply, and engineers must work to balance model fit with overfitting, primarily by comparing performance on training data and test data.

Recall that the independent variables used for prediction are initially selected during feature selection. Statistical testing will ultimately help identify the best selection of features among the available options. However, it cannot suggest new features for collection or testing; for that, domain expertise is an essential input.

Other forms of regression, such as polynomial regression, are variations on this theme, typically incorporating nonlinear relationships such as powers of x and interaction terms such as $x_1 x_2$; for example, polynomial regression is of the general form

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_{11} x_1^2 + \beta_{22} x_2^2 + \beta_{12} x_1 x_2 + \epsilon$$

where β_{ij} values indicate interaction term weights of the same sort as β_i values for non-interaction term weights.

Regression is a widespread tool, particularly in economics, and as such its role as a machine learning tool can be overlooked. However, it remains one of the most commonly deployed tools in data analysis and is a fundamental tool in supervised learning - including as the foundation of categorical variable classification, explored in the next section.

6. Classification

Foundations

Classification models predict categorical target variables. For instance: does a given image show a cat or a dog? Should the loan application be approved or not? These two examples are cases of **binary classification**, where "binary" simply indicates two possible outputs, frequently represented numerically as 0 and 1. Logistic regression is the leading means of binary classification.

Take another question: what type of animal does a given image show? The question as posed is not well formed for a supervised learning classification model. Instead, the system requires specified options for the target variable. For instance, you might specify a list of animals an image might possibly show. This could then become a well-formed question of the form: which of the specified animals (i.e. categories X, Y, Z, ...) does this image show? This problem is a candidate for **multiclass classification**. This can be achieved through multinomial logistic regression and tree-based models such as random forest.

Key Model: Logistic Regression

The archetypal model for **binary classification** is **logistic regression**. Note that, despite the name, logistic regression is a form of classification and not regression. To get started on this topic, let's look at a situation discussed earlier:

1a) *You are a loan officer responsible for determining whether to approve appplications based on whether each applicant is likely to default.*

The problem at hand calls for binary classification of applications: those to be approved and those to be rejected. We decide to base our decision on whether the applicant is deemed likely to default.

Looking at historical data, we see a feature called "default", with values 1, indicating that the person defaulted, and 0, where they did not default. We set "default" as our target variable. How do we go about classifying each application as 1 or 0 based on selected features?

Logistic regression employs a logistic function to transform an array of possible values into a value close to 1 or 0. The function produces a probabilistic output, and the developer can decide a probability cutoff, known as the **decision boundary**, to classify values as 1 or 0.

The **sigmoid function** is the basic logistic function for binary classification:

$$p = \frac{1}{1 + e^{-z}} = \frac{e^z}{e^z + 1}$$

where p is a predicted probability, which will end up close to 1 or 0 for the vast majority of inputs. (Both formulations are fairly common; they are listed together to avoid future confusion.) Before turning to explain the essential variable, z, let's see the general behavior of the sigmoid function:

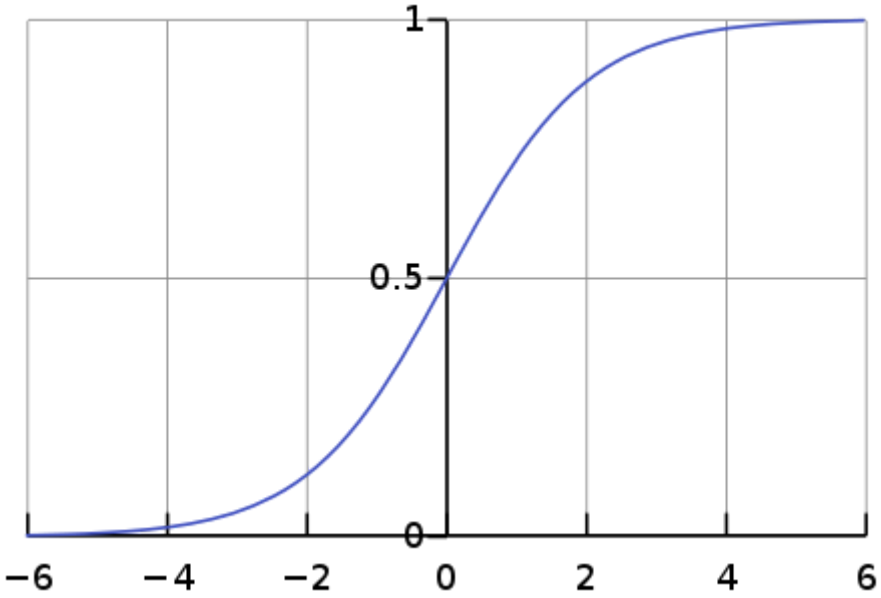

Figure 1: The sigmoid function. Credit: Wikimedia Commons.

Notice that the maximum y value of the sigmoid function is 1, and the minimum is 0. Outputs are essentially "pushed" toward one extreme or the other, allowing us to split the dataset.

What is z? As it turns out, z is simply a regression equation! That is, z is the **hypothesis** regarding the relationship between the target variable and independent variables. Again, the general form of the function, using a linear regression equation, will be:

$$z = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_{11} x_1^2 + \beta_{22} x_2^2 + \beta_{12} x_1 x_2 + \epsilon$$

Rather than predicting a value from a continuous range, we put this hypothesis into the sigmoid function (or any other logistic function) to predict either 1 or 0, corresponding to our labels of interest.

The same considerations of overfitting apply again. However, the particular metrics of interest can vary considerably depending on the dataset involved. We will explore this further in the metrics section.

Now we ask: how do you, as a loan officer, intend to approach the decisions? Following feature selection and model construction, two key questions remain.

Data imbalance

First, data are frequently **imbalanced**: one group may have far more data than another. For example, default tends to be rare. If you had 995 examples of "no default" and a five examples of "default" in your training data, how accurate could your model be? How much could it learn? In practice, very little: you require far more instances of "default" to learn effectively.

However, even with robust samples (say, 100,000 of "no default" and 2,000 of "default"), the imbalance leads to a **measurement problem**. If you simply scrapped the system and declared *all* applications to show (incorrectly) "no default", the *accuracy* of the results would be a healthy 98% - despite failing to acheive any intended goals. Something is clearly amiss. We will solve this problem shortly, when studying evaluation.

Goal specification

Second, a successful loan officer needs to know: what is the bank's or lender's **risk tolerance**? Mathematically, this question becomes: what is the appropriate **decision boundary**, i.e. the value above which we classify an application as 1 and below which we classify it as a 0? The value frequently is not a halfway point, but another value that is predicted to result in a desirable amount of lending (if that is a fixed end goal). This boundary-setting reflects the bank's goals and priorities. It may vary between banks, for a single bank at different points in time, for a single bank by each region of lending, etc. Modeling is rarely a one-and-done affair, but something that continues over time, both to improve on reaching a given goal and to refocus on new goals.

As a final note, **multiclass classification** adds complexity, as may be expected, but relies on similar mechanics. We will have the opportunity to explore this area later in the course.

7. Tree-based Models for Classification and Regression

The Decision Tree

Tree-based models provide an alternative means to approach both branches of supervised learning. Such models are often grouped under the heading of Classification and Regression Tree (CART) analysis. All of these models are based on the humble, human-interpretable decision tree:

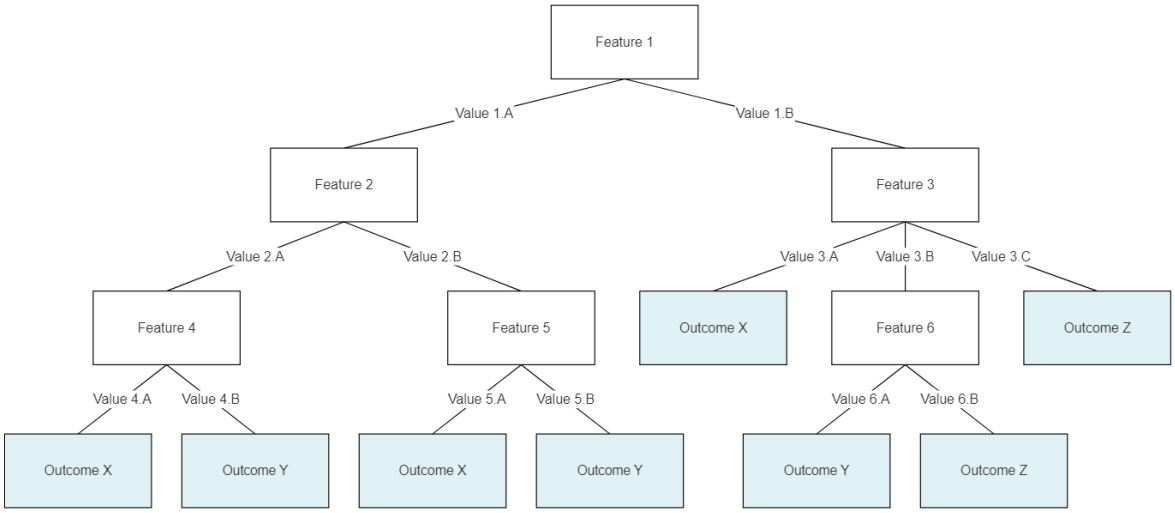

Figure 2: Graphical representation of an example decision tree for a classification problem with six features and possible target variable values X, Y, and Z. Note that decision trees may be represented in other ways, but the core concepts remain the same.

The basic intuition of a decision tree is similar to the game "twenty questions", in which a series of yes/no questions leads to a prediction of the hidden answer. The decision tree formalizes the process of questioning into a reproducible analysis of specific features, while allowing for a far broader range of questions.

In the figure above, the features assessed are represented as transparent boxes known as **nodes**, with possible values represented as lines and known as **branches**. The potential outcomes - which could be the results of either classification or regression - are represented as blue boxes and known as **leaves**. As the figure suggests, nodes may have varying numbers of branches, and leaves may be reached at any level. This flexibility allows considerable model adaptation.

In a regression problem, where the features are continuous variables, a node can assess whether a feature value is above or below a certain value. A sophisticated classification model may assess both categorical and numerical variables in different nodes.

Training a Tree

Numerous algorithms exist for decision tree learning; in general, however, the algorithms perform the same basic function: to identify the best splits for each node, working from the top down. The general procedure is:

1. Test which feature best splits the labeled data
2. Set the identified feature as the first node
3. Test which feature best splits each now-segregated set of labeled data
4. Establish the identified features as the next respective nodes

And so forth, until the measure used to test how well a feature splits the data falls below a specified threshold or reaches zero (indicating no further benefit to splitting). Where continuous variables are employed, the model also determines threshold values that best split the data.

Key Algorithms

Three important algorithms are discussed below, all of which identify splits (for further discussion, see 2). The following presentations assume binary splits, i.e. two branches per node, although many algorithms can be adapted for other cases.

Information Gain

- This algorithm determines which features yield the most *information*, using a concept called **entropy**, which can be expressed as:

$$Entropy = -\sum_{j=1}^{n} -p_j log_2 p_j$$

  where $E(S)$ is entropy, p is the probability of outcome 1 and q is the probability of outcome 2.

- Information gain is equal to $1 - E(S)$. The algorithms identifies the split that minimizes entropy, which indicates maximal information gained.

Gini Index

- The Gini index (also known as Gini impurity), measures the probability that a random classification will be incorrect. If *all* data belong to the same category (e.g. in the loan case, if the dataset contains zero defaults), then the Gini index returns a value of 0; this category is then known as "pure". A completely random distribution of data across classes results in a 1 - completely "impure".
- The Gini index is largely famous for its use in measuring income inequality.
- The most basic expression of the Gini index for decision tree-based classification is:

$$Gini = 1 - \sum_{i=1}^{n} p_i^2$$

  where n is the number of categories and p_i is the probability of a data point's classification in the given category.

Chi-Square

- Chi-square measures the statistical significance of a given split, allowing the model to choose the maximally significant split.
- The chi-square formula is, for a given observation O and a given expected value E:

$$\chi^2 = \sum_{i=1}^{n} \frac{(O_i - E_i)^2}{E_i}$$

Notes

Other algorithms include variance reduction, standard deviation reduction, and the like. As the names imply, they select splits that minimize variance, standard deviation, etc. between the parent and child nodes of each split.

Complete application of these techniques is far more involved that the base formulas above may imply; gladly, the data transformations involved are packaged into machine learning libraries for efficient development. The base formulas presented do, however, shed light on the central goal of each given algorithm.

Overfitting

There are, in general, two approaches to handle overfitting in a single tree-based model.

1. Allow the tree to grow to completion, allowing overfitting, then **prune** the tree by identifying and removing features that have a very low impact on the outcome.
2. Speciify a maximum tree depth, and **halt** tree growth before the point at which overfitting is expected.

While consciously allowing a model to overfit before correcting it may seem counter-intuitive, the alternative makes clear a difficult challenge: how deep *should* it be allowed to grow? While the decision is unlikely to be a complete guess, it is also far from an exact science. Pre-specified cutoffs risk both halting a tree *before* optimal depth is reached, and halting it *after* optimal depth is reached, in which case one returns to pruning the tree (albeit likely less so than if no cutoff is specified).

How does one prune a tree? To prune, developers will use the selected statistical measure to determine the contribution of each feature. While pruning, the model is tested repeatedly to measure the accuracy or other metric of interest, until overfitting is corrected and the desired performance is achieved.

How does one halt a tree before overfitting? There are multiple options. Most simply, an engineer can simply specify that a tree shall have no more than X levels or splits; the algorithm should cease if the specified depth is reached. As a more technical approach, they might also decide a threshold of minimum improvement for the algorithm to cease. As we discussed above, the algorithms can be allowed to proceed in making splits until the splits cease to yield improvements, where the improvements in question are dictated by the selected algorithm (e.g. when entropy is no longer decreased, for the case of information gain). An engineer can specify a higher threshold at which "improvement" is no longer sufficient to continue splitting, in principle ending the process before overfitting occurs.

Trial and Error

At the end of the day, models require iteration. While there are tools to hasten the process, there is no crystal ball to reach an ideal system from the first attempt. Instead, best first attempts are put in place, and then models are *evaluated*, reconfigured, and *evaluated again*, until satisfactory results are achieved. Or, if resources are running low, a suboptimal model might be let free as

"good enough"... hence the importance of ensuring project leaders understand what goes into optimization, and making sure the assessment of "good enough" truly is, in fact, good enough.

Beyond Single Trees

As mentioned at the start of the section, decision trees are the foundations of tree-based models. Building on these are **ensemble models**, such as the **random forest model**, which in effect averages results across multiple decision trees, thereby smoothing potential overfitting by each individual model. Such models lay beyond the scope of what can reasonably be covered here, but with the given foundations you will be prepared to research them moving forward.

Example Decision Tree Code

8. Model Evaluation

Model evaluation is tailored to both the model type and the specific goals of the model use case. We will discuss the foundational ideas, with the usual caveat that this short introduction cannot prove complete coverage. However, it does include the key foundations for working with the full range of metrics one might encounter in the future.

Regression

Regression models are evaluated by variations on a theme: how *close* are the predicted results to the values? Foundational techniques include R^2 and adjusted R^2. The most important, more robust methods, however, are **mean absolute error (MAE)** and **root mean square error (RMSE)**.

$$MAE = \frac{1}{n}\sum_{i}^{n}|y - \hat{y}|$$

where n is the number of data points, y is a true value, and $\hat{y}$ is a predicted value.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i}^{n}(y - \hat{y})^2}$$

where the variables are identical to the above. Note that the only different between MAE and RMSE is that RMSE first squares the difference between the true and predicted values, the takes the square root to return to the original units.

How do we choose between MAE and RMSE? We will get into further detail later in the course, but for now we can highlight key high-level trade-offs: **MAE tends to be easier to interpret**, aiding in human understanding and, if needed, correction. It is also generally robust to outliers. **RMSE tends to allow for more efficient calculations**, in large part because it is smoothly differentiable. It is, however, less robust to outliers. Which one is preferable in a given case is primarily an engineering question. Knowing the essential trade-offs, however, provide the foundations to engage in that conversation with your development team. (For accessible introductions to these topics, see 3 and 4.)

Classification

Competing classification metrics differ in surprisingly vital ways. All reflect goal specifications, frequently also reflecting implicit value judgments; this is because the metric for which a model is optimized suggests choosing one side of a trade-off over another. Classification metrics can bring policy and value considerations to the forefront.

The confusion matrix

The essential classification model metrics are anchored in a single table known as the **confusion matrix**. In the simple case of a binary classification, in which all data are labeled "1" or "0", also known as "positive" or "negative", the confusion matrix is as follows:

	Actual Values Positives: 1 Negatives: 0	
Predicted Values Positives: 1 Negatives: 0	**True Positives (TP)** *predicted: 1 actual: 1*	**False Positives (FP)** *predicted: 1 actual: 0*
	False Negatives (FN) *predicted: 0 actual: 1*	**True Negatives (TN)** *predicted: 0 actual: 0*

Table 2: The confusion matrix for a binary classification problem.

- **Unpacking the confusion matrix**: We first want to understand the differences between the four groups. **TP** and **TN** indicate *correct* predictions, while **FP** and **FN** indicate *incorrect* predictions. Why not have only two categories: correct and incorrect? In many cases, **in matters in which direction we are wrong**.

  Consider again the loan problem. Would we rather deny applications unlikely to default, or approve applications likely to default?

Essential Metrics

With the confusion matrix in place, we can outline the four fundamental metrics drawn from it. (For further detail, see Mohammed Sunasra's excellent introduction: X.)As an example to contextualize these metrics, consider a classification algorithm designed to detect cancer. Assume a relatively low incidence in the given population - say, 0.5%.

- **Accuracy**: measures percentage of total correct predictions: (TP * TN)/(TP + FP + FN + TN). This is a common metric for balanced datasets, but a very poor metric when a large majority of outcomes are in one class rather than another - for example, when one is low incidence, such as cancer in the assumed population.
- **Precision**: measures percentage of predicted positives that were correct: (TP)/(TP + FP). For example, of the people the model predicted to have cancer, how many actually had cancer?
- **Sensitivity**: measures percentage of actual positives that were correctly predicted: (TP)/(TP + FN). For example, of the people who actually had cancer, how many did we detect?
- **Specificity**: measures percentage of actual negatives that were correctly predicted: (TN)/(TN + FP). For example, of the people who did not actually have cancer, how many did we predict did not have cancer?

These four metrics each entail trade-offs. What if your model requires risk aversion (e.g. high prediction of threats, even if including false alarms), but only up to a limit (not *too many* false alarms)?

For such real-world interests, combinations are often used to balance various trade-offs. The chief example, which we will study in a future session, is the **AUC-ROC curve**, which finds an optimal point between sensitivity and specificity (learn more about this curve at 5):

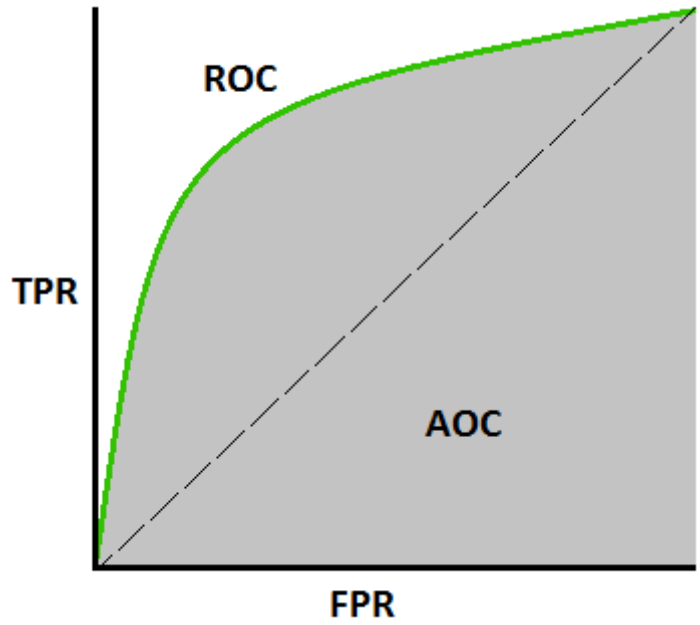

Figure 3: A model AUC-ROC curve. Credit: Sarang Narkhede, Towards Data Science.

- **The problem with accuracy in classification of imbalanced datasets**: Let's return to the lending problem of situation 1a). Here, we almost certainly will have *imbalanced* data: far more instances of no default than of default. If only one in a hundred loan recipients defaulted in the training dataset, then no classification whatsoever would yield a 99% accuracy (correct classifications).

  The problem is similar for a range of classification problems, from spam filters to cancer detection. The case of cancer detection makes clear yet another challenge, this time chiefly ethical: **how risk tolerant should we be in detecting cancer**?

  Most respondents instinctively seek to minimize risk. However, when we are forced to put numbers to it, *minimal* risk suggests that if there is *any* percentage chance of cancer, we flag the case as cancerous, just in case. Few are likely to see zero percent chance, given the ambiguity and chance involved; meanwhile, cancer incidence is in reality quite low, likely less than one percent in healthy populations. This would move many if not most individuals to further rounds of testing. However, this is likely to be both impractical and not the ethically ideal scenario. What threshold, then, is acceptable, so that the hospital neither misses likely cases nor wantonly informs everyone they may have cancer? Part of the ethical decision is where to draw the line; another is to consider whether the model is sufficient to deploy at all. Both may have reasonable alternatives; but that is a decision that must be hard-coded, a joint effort by the policy personnel and the engineers. (For further discussion, see 6).

Closing Comments

Real-world applications of supervised learning are subject to an array of challenges and uncertainties. But with a critical awareness of data and model evaluation, such models can produce remarkable results. From a policy perspective, understanding the data requirements and available models for a given application empowers clear communication and coordination

with engineers, helping to achieve the given objectives with appropriate trade-offs. Such communication and coordination are essential elements of effective AI policy and strategy.