Machine Learning

15-110 – Friday 11/12

Announcements

- **Check6-1** is due **Monday at noon**
    - For the programming portion, start from the General Guidelines to pick your project, and start early!
    - **Demo** - how to work with the starter files

Learning Goals

- Identify the three major categories of learning (**supervised**, **unsupervised**, and **reinforcement**) and the three major categories of reasoning (**classification**, **regression**, and **clustering**)

- Decide which combination of **learning** and **reasoning** categories are best used to solve a stated problem

- Describe how **training**, **validation**, and **testing** are used to build a model and measure its performance

Machine Learning and Models

Machine Learning Creates Models

In data analysis and simulation, we designed algorithms and models to explore specific datasets.

Machine learning (ML) goes one step further. Instead of designing a model for a dataset, we apply a general **machine learning algorithm** to the data.

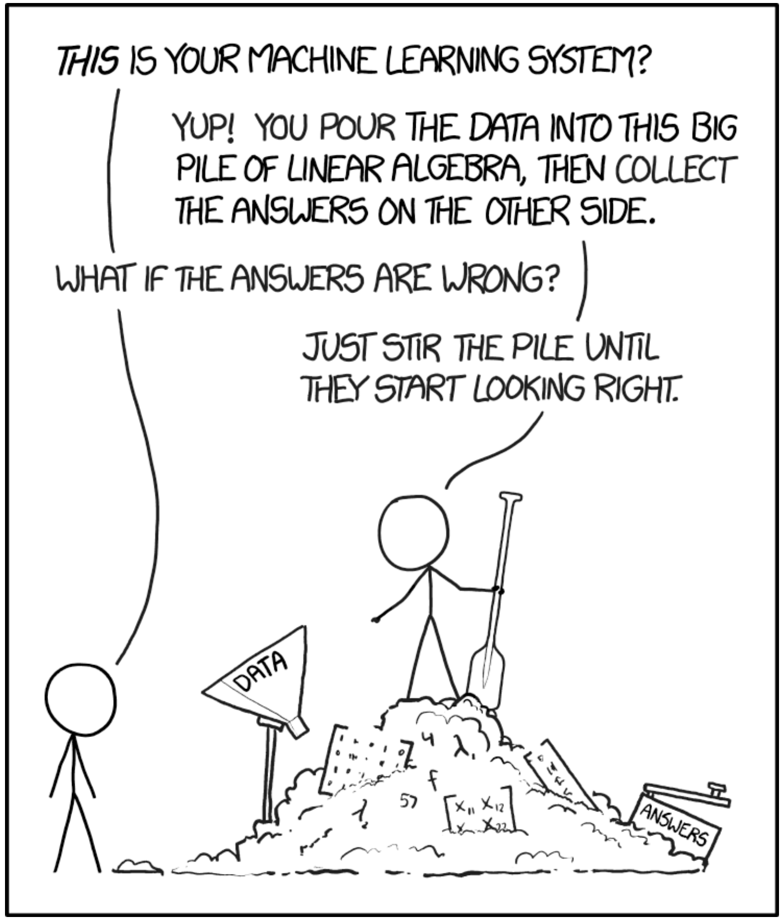
This identifies patterns in the data and generates the model **automatically**. It's like outsourcing the work to the computer!

Machine Learning Finds Patterns

Machine learning algorithms are special because they **reason** over data to find patterns and behavior without having a programmer code those patterns or behaviors into the algorithm. The programmer doesn't teach the model the pattern – they teach the algorithm **how to learn** the pattern.

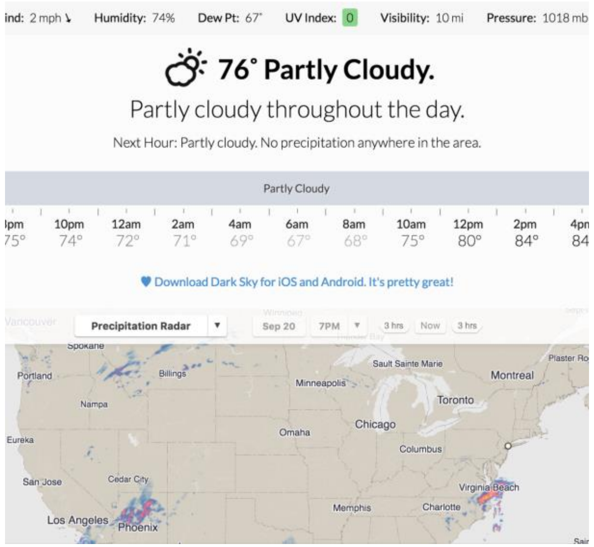
Because the algorithm does its own learning, it is possible for an ML algorithm to **optimize** a model's performance by processing more data and tweaking how the model works.

It's like human learning – you build understanding about a topic over time by practicing and get feedback.


THIS IS YOUR MACHINE LEARNING SYSTEM?
YUP! YOU POUR THE DATA INTO THIS BIG PILE OF LINEAR ALGEBRA, THEN COLLECT THE ANSWERS ON THE OTHER SIDE.
WHAT IF THE ANSWERS ARE WRONG?
JUST STIR THE PILE UNTIL THEY START LOOKING RIGHT.
DATA
ANSWERS

Machine Learning in Many Contexts

Machine Learning generates models that are used in thousands of contexts across the world, including speech recognition, weather prediction, and medical diagnosis.



ind: 2 mph ↘ Humidity: 74% Dew Pt: 67˚ UV Index: 0 Visibility: 10 mi Pressure: 1018 mb

76˚ Partly Cloudy.

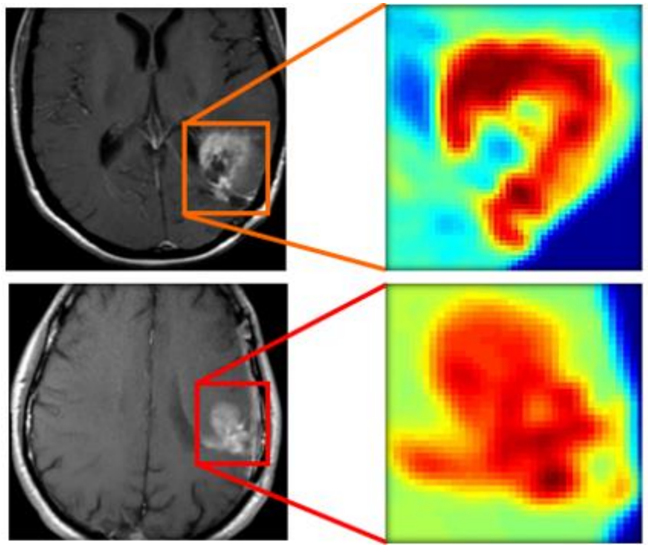
Partly cloudy throughout the day.

Next Hour: Partly cloudy. No precipitation anywhere in the area.

Partly Cloudy

pm	10pm	12am	2am	4am	6am	8am	10am	12pm	2pm	4pn
75°	74°	72°	71°	69°	67°	68°	75°	80°	84°	84

♥ Download Dark Sky for iOS and Android. It's pretty great!



Models

The **model** generated by a machine learning algorithm is used to perform reasoning tasks.

Once a model has been generated, we can analyze it to gain insights about the original dataset. For example, we might build a model to group feedback provided in a country-wide survey, then use those groups to identify common trends.

We can also apply a model to new data to make predictions. A model trained on market behavior could make a prediction about which products produced by a business might be most popular in an upcoming quarter.

Models are Based on Data

To train a model using machine learning, we must start with a large amount of **data**. The algorithm will try to find patterns and correlations in that data. A model is only as good as its dataset!

Each data entry must be broken down into a set of **features**, so the machine learning algorithm can analyze features separately or in groups. If you have a table of data, the features would be the **columns**; in our ice cream data, the three features would be the three chosen flavors.

You can create and add new features the same way you would add new columns in data analysis.

Learning and Reasoning Categories

Many Machine Learning Algorithms

There are a very large number of machine learning algorithms that computer scientists have designed. How do we choose which one to use?

We can break algorithms into groups based on how they **learn** and on how they perform **reasoning**.

Three Ways to Learn

There are three primary ways that machine learning algorithms learn patterns from data. Which type you use depends on what type of problem you're trying to solve.

The three categories are **supervised learning**, **unsupervised learning**, and **reinforcement learning**.

Supervised Learning

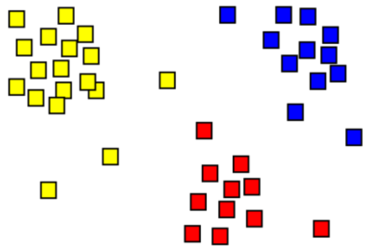
Supervised learning algorithms train models to predict outputs based on inputs. A data entry in the dataset is the input, and a **label** or a **score** is the output.

Example: to train a model to predict a person's favorite ice cream based on other desert preferences, each data entry (set of desert preferences) must be accompanied by a label (that person's favorite ice cream).

Once the model is fully trained, it can be used to make predictions about unlabeled data.

This is like when you learn by completing a homework assignment. You're trying to predict a correct answer (label) which is known by the teacher. This means the teacher (ML algorithm) can check your work.

Unsupervised Learning



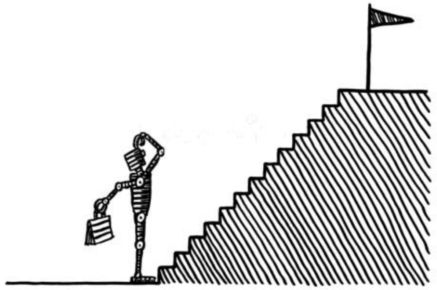
Unsupervised learning algorithms train models that infer the natural structure of a dataset. Which data entries are most similar to each other?

Example: identify common categories of favorite ice cream flavors by grouping together people who have similar desert preferences.

Unsupervised learning is often used when the data is **not** labeled – no true answer is known. That means that a human being will need to look over the model's results to see if they make sense or seem random.

This is like how researchers come up with theories to explain how the world works. Theories are based on data and may make sense, but they are not necessarily how the world actually works.

Reinforcement Learning



Reinforcement learning algorithms train models that help an **artificial intelligence agent** approach an overall goal. Usually this involves taking a sequence of actions to bring the agent closer to the endpoint.

Example: train a robot to make ice cream by constantly changing the temperature and speed of an ice cream machine and measuring the consistency of the product until it's just right.

This is like playing a guessing game where the person with the secret word tells you if your guesses are hot or cold. You adjust your guesses and questions based on the person's feedback.

We'll talk more about how AIs work later in the unit.

Demo: Supervised Learning

We can build an image-recognition model using supervised learning by labeling a bunch of pictures as belonging to specific classes.

Try it out here: https://teachablemachine.withgoogle.com/train/image

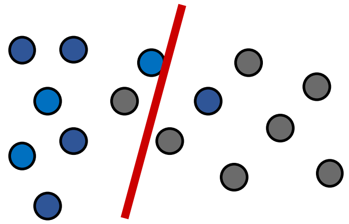
Many Ways to Reason

Once you've identified what type of learning you need to do, there are still lots of options for what type of algorithm you can use! One way to narrow the options down further is to ask what type of **reasoning** the algorithm will need to do.

For supervised learning, common reasoning approaches include **classification** and **regression** algorithms.

For unsupervised learning, common reasoning approaches include **clustering**.

Classification Algorithms

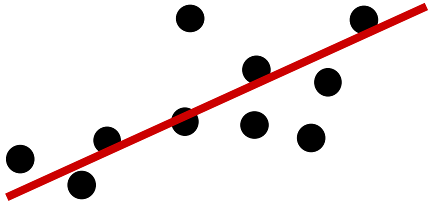


A **classification** algorithm takes labeled data of any type and produces a **model** (called a classifier) which determines which **category** a given data entry most likely belongs to. In other words, it produces a **categorical** (discrete) result.

For example, a classifier might take in data about an email (the sender, title, message content) and identify whether or not it is spam.

Classification is commonly used in image recognition. Examples of classification algorithms include Naive Bayes and Decision Trees.

Regression Algorithms

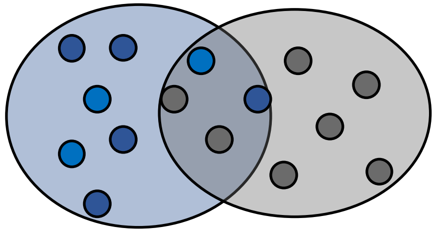


A **regression** algorithm takes numerically-labeled data and produces a function that maps data entries to **numerical** (continuous) results. Instead of predicting a specific category, it gives each data entry a **score**.

For example, a regression algorithm might take information about a house (GPS location, previous sell date, previous sell price, average price of nearby houses) and use it to predict its market value.

Regression algorithms are used to predict things like stock market prices and weather temperatures. A linear regression is one kind of regression algorithm.

Clustering Algorithms



A **clustering** algorithm takes a set of data and produces a model that breaks the data into some number of **clusters** by identifying data entries that are similar to each other. We don't actually know the true categories or scores of the provided data, so it's hard to 'test' the resulting model, as we aren't sure what the actual result should look like.

For example, a clustering algorithm could recommend a new set of plant species by finding similarities between plant features in a given region.

Clustering is used in data compression and in bioinformatics. K-means clustering is one kind of clustering algorithm.

Activity: Choose How to Reason

You do: what combination of learning category + reasoning category would you use to...

- ... predict someone's favorite sport based on their job?
- ... investigate potential groupings of people based on their favorite restaurants?
- ... estimate how many meters a person walks per day based on their demographics?

Many Other Algorithms

Classification, regression, and clustering are just the most common types of reasoning algorithms. There are plenty of others that we don't have time to discuss in this class.

To make a model, though, you just need one algorithm- the best fit for the problem you're trying to solve.

Let's dive a bit deeper into how we can apply machine learning with supervised algorithms specifically.

Creating a Model – Training

ML Process: Decide, Train, then Test

To apply machine learning to a supervised task, follow a simple process

First: **decide** which learning algorithm you'll use and which features you'll train on. Make sure the algorithm matches the feature types!

Second: **train** a model created by the algorithm by providing it with the data. The algorithm will 'learn' from the data the same way a student learns by going over worked examples.

Third: **test** the model on a different set of data. This helps determine how accurate the model actually is.

Training Identifies Key Features

During the training process the algorithm identifies which features contribute the most to the underlying pattern. It does this by picking features that reduce the **error** that results from running the model on known data.

It's rare for a machine learning algorithm to identify a single feature that can definitively be used to answer a question. Usually, the algorithm uses a **combination of several features** which are weighted based on how well they correlate with the correct answer.

The algorithm needs to learn from a **lot** of examples to get a good sense of what the real pattern is.

Example: Is It a Dog?

Let's use a very simplified example to demonstrate how a machine learning algorithm might learn which features best match a label.

We want to train a model to recognize pictures of dogs. What kinds of features might it identify?

(The features we describe here are far too high-level, but the general idea is about right).

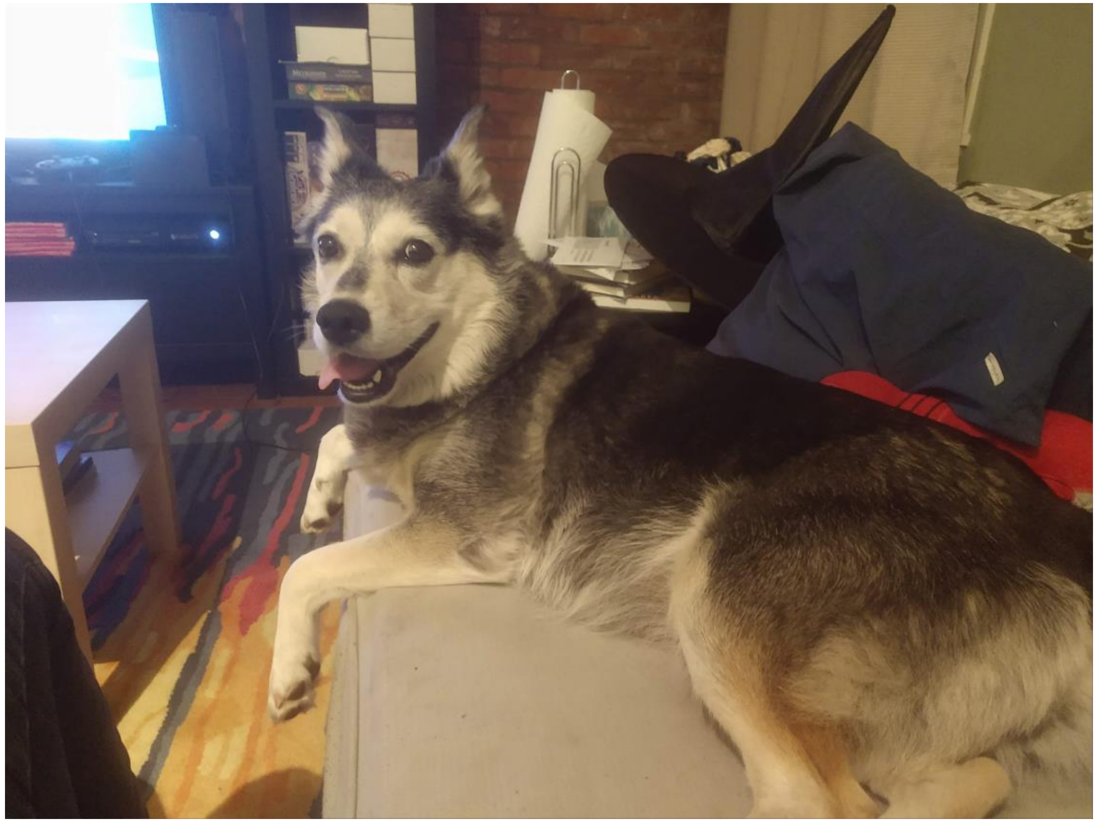
Example: Is It a Dog?

Guess: ?

Answer: ✓

Prior knowledge:

???



Updated knowledge:

- Ear type = pointy
- Has Fur = True
- Screen in background = True

Example: Is It a Dog?

Guess: ✓?   Answer: ✓

Prior knowledge:

- Ear type = pointy
- Has Fur = True
- Screen in background = True

Updated knowledge:

- Ear type = pointy
- Has Fur = True
- Nose length < 6in

Example: Is It a Dog?

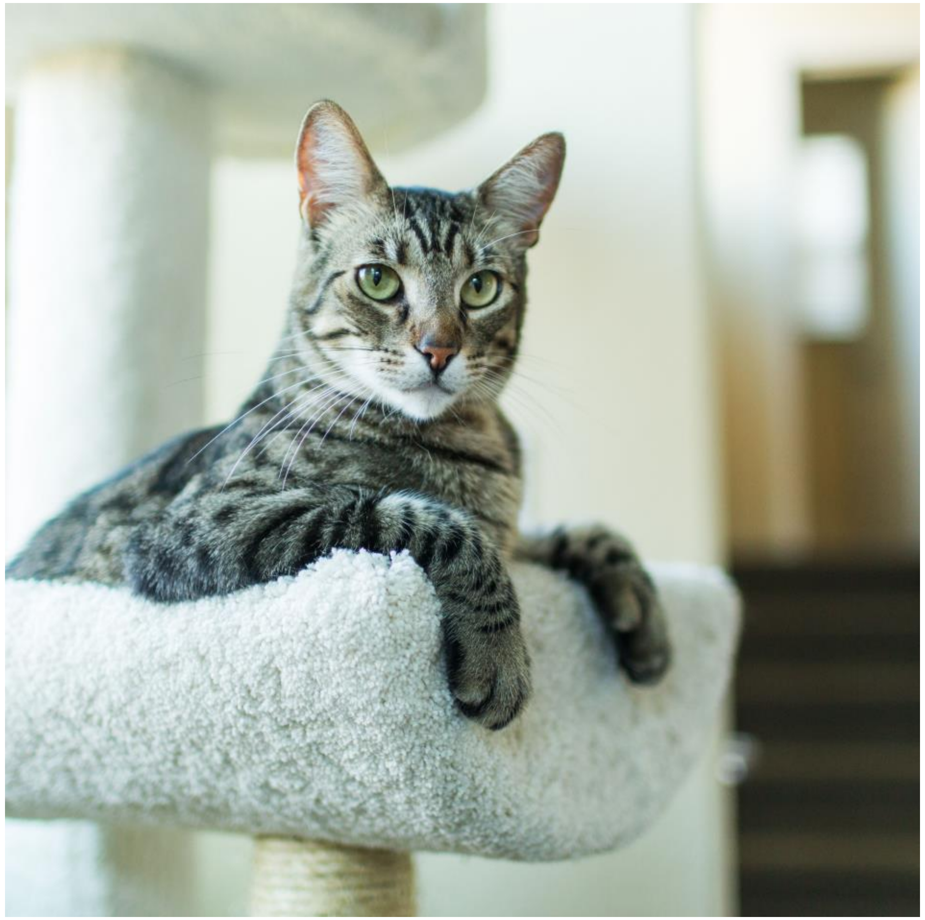
Guess: ✓ Answer: ✗

Prior knowledge:

- Ear type = pointy
- Has Fur = True
- Nose length < 6in

Updated knowledge:

???



Real Example: Favorite Ice Cream Model

Now let's try building a real model with machine learning!

We'll use our ice cream data from the Data Analysis lecture. Our goal is to predict a person's 3rd favorite ice cream based on their 1st and 2nd preferences.

There are a ton of possible flavors, so we'll **hand code** the data to group similar flavors together and reduce the number of options to predict. This results in 6 possible categories: chocolate, coffee/tea, cookie, fruit, vanilla, and other.

Our approach is **supervised** (we already know what people's favorites are) and predicts a **categorical** result (an ice cream flavor). We need to use a classification algorithm.

Machine Learning Libraries

You can implement a machine learning algorithm yourself, but it isn't usually necessary. There are lots of great **external libraries** that have already implement machine learning algorithms for you.

We'll use the library **scikit-learn** here. You can install it with:

```
pip install sklearn
```

(We won't expect you to write code with `sklearn` this semester, but we'll go over it now anyway so you can see how the process works).

Read the Data

Start by reading the data into a 2D list.

We just want the categories, so we'll only copy over the semester and the three categories for now.

```
import csv
# Header:
# Semester,
# #1 Orig,     #2 Orig,     #3 Orig,
# #1 Cleaned,  #2 Cleaned,  #3 Cleaned,
# #1 Category, #2 Category, #3 Category
f = open("all-icecream.csv", "r")
orig = list(csv.reader(f))
allData = []
for line in orig:
    if line[0] != "Semester": # skip header
        # only include coded classes
        categories = line[7:10]
        allData.append([line[0]] +
                       categories)
f.close()
```

Naive Bayes Classifier

Naive Bayes is a simple classification algorithm. The classifier it creates uses **conditional probabilities** to predict categories.

When Naive Bayes is given a new data entry X, it uses the data it has already collected to determine the **probability** that X belongs to each possible category.

For each category C, it checks how **probable** it is that each feature in X occurs with C by checking how often it occurred in past data entries. By multiplying these probabilities together, it can form a general probability that the data entry belongs in that category.

The category with the highest probability wins!

$$P(\mathrm{A}|\mathrm{B}) = \frac{P(\mathrm{B}|A) \cdot P(A)}{P(B)}$$

$$P(C|X) = P(X_1|C) * P(X_2|C) * P(X_3|C) * P(C)$$

Organizing Data

The Naive Bayes implementation in scikit-learn requires that you separate your data into the **inputs** and the **label** for each data point. The inputs are the values used to make the prediction (#1 and #2 favorites); the label is the value being predicted (#3 favorite).

To make predictions with categorical data, we need to map each category to an **integer**. We'll use the `index` method to map each flavor to its index in a list.

```
def mapToIndex(flavor):
    allFlavors = [ "chocolate", "coffee/tea",
                   "cookie",    "fruit",
                   "vanilla",   "other" ]
    return allFlavors.index(flavor)

dataInput = []
dataLabels = []
for point in allData:
    inputValues = [ mapToIndex(point[1]),
                    mapToIndex(point[2]) ]
    outputValue = mapToIndex(point[3])
    dataInput.append(inputValues)
    dataLabels.append(outputValue)
```

Setting Aside Test Data

Actually, we want to split our data into two groups- **train** data and **test** data. That will make it possible for us to check our work later on.

We have information about when the data was collected, so let's just test on the most recent semester for now.

```
dataInput = []
dataLabels = []
testInput = []
testLabels = []
for point in allData:
    semester = point[0]
    inputValues = [ mapToIndex(point[1]),
                    mapToIndex(point[2])]
    outputValue = mapToIndex(point[3])
    if semester == "F21":
        testInput.append(inputValues)
        testLabels.append(outputValue)
    else:
        dataInput.append(inputValues)
        dataLabels.append(outputValue)
```

Training the Model

Actually training the model is easy. Just generate a new instance of the model using `CategoricalNB` (a function from `sklearn`), then call the `fit` method on that model using the data.

We can use the trained model to make predictions on new data points. For example, let's say we want to predict someone's 3rd favorite if their 1st favorite is chocolate and 2nd favorite is cookie. Create a list of the two indexes (0 and 2), put that in another list, and call the `predict` method on the model.

The function returns 0, which corresponds to the chocolate category! A reasonable guess.

```
from sklearn.naive_bayes import CategoricalNB

model = CategoricalNB()
model.fit(dataInput, dataLabels)

print(model.predict([ [0, 2] ])) # [0]
```

Creating a Model – Testing

Building Good Models

Once we've trained a model, we can use that model to make predictions about data. That means we don't want the model to **only** work on the data we provided originally - we want it to work on future data too.

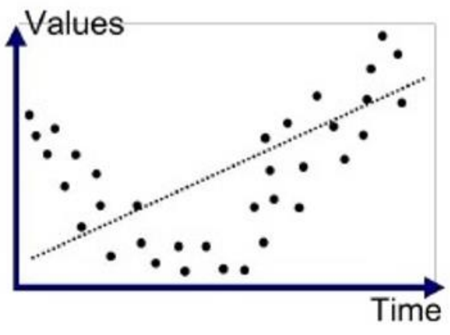
Many machine learning algorithms start out with simple models that become more complex over time as the algorithm tries to eliminate every last bit of error. If we use 100% of our data on training, we'll run into problems.

When you build a model with a machine learning algorithm, you need to separate your data out into three groups: **training** data, **validation** data, and **testing** data. This will let you evaluate your model on 'new' data once it is done.

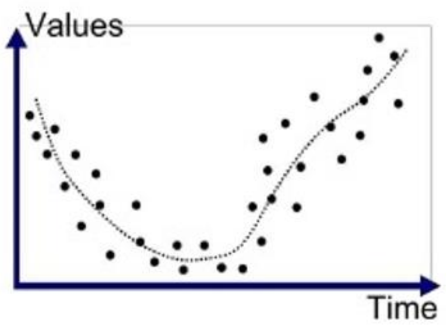
Training Data Can Cause Overfitting

The **training data** is normally composed of the majority (maybe 70%) of the available dataset. This data is run through the machine learning algorithm to produce the model. The more training data there is, the more **accurate** the algorithm's model becomes.

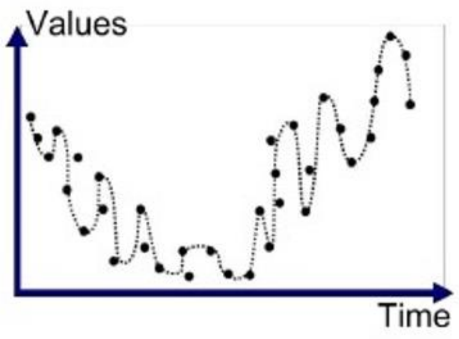
This can go wrong if the algorithm over-optimizes the model. For example, it might identify a pattern that only exists within the training data, not in the general population – the pattern might just be noise. This is called **overfitting**. Overfitting can result in a model performing very well on training data, but poorly on test data.



Underfitted



Good Fit/Robust



Overfitted

Validation Data Identifies Overfitting

To detect and remove parameters in the model that cause overfitting, you can use **validation data**. This is a subset of the data (maybe 15%) that is **not** used when training the model. Instead, it will be used to **validate** the model during training.

The algorithm will repeatedly evaluate the model it has produced on the validation data to see how accurate it is. This makes it possible for the algorithm to try out different sets of features, to see which parameters work best.

Testing Data Provides Final Results

When the algorithm thinks it's achieved an optimal model, the **testing data** is used to determine how accurate that model actually is. This is a portion of the data (maybe 15%) that was set aside at the beginning and never used during the training process.

Unlike the validation data, which is evaluated multiple times, the model is run on the test data only **once**. We measure how close the predicted results are to the actual results. That score is the accuracy of the model.

You cannot train on your testing data if you want a fair test of the model!!!

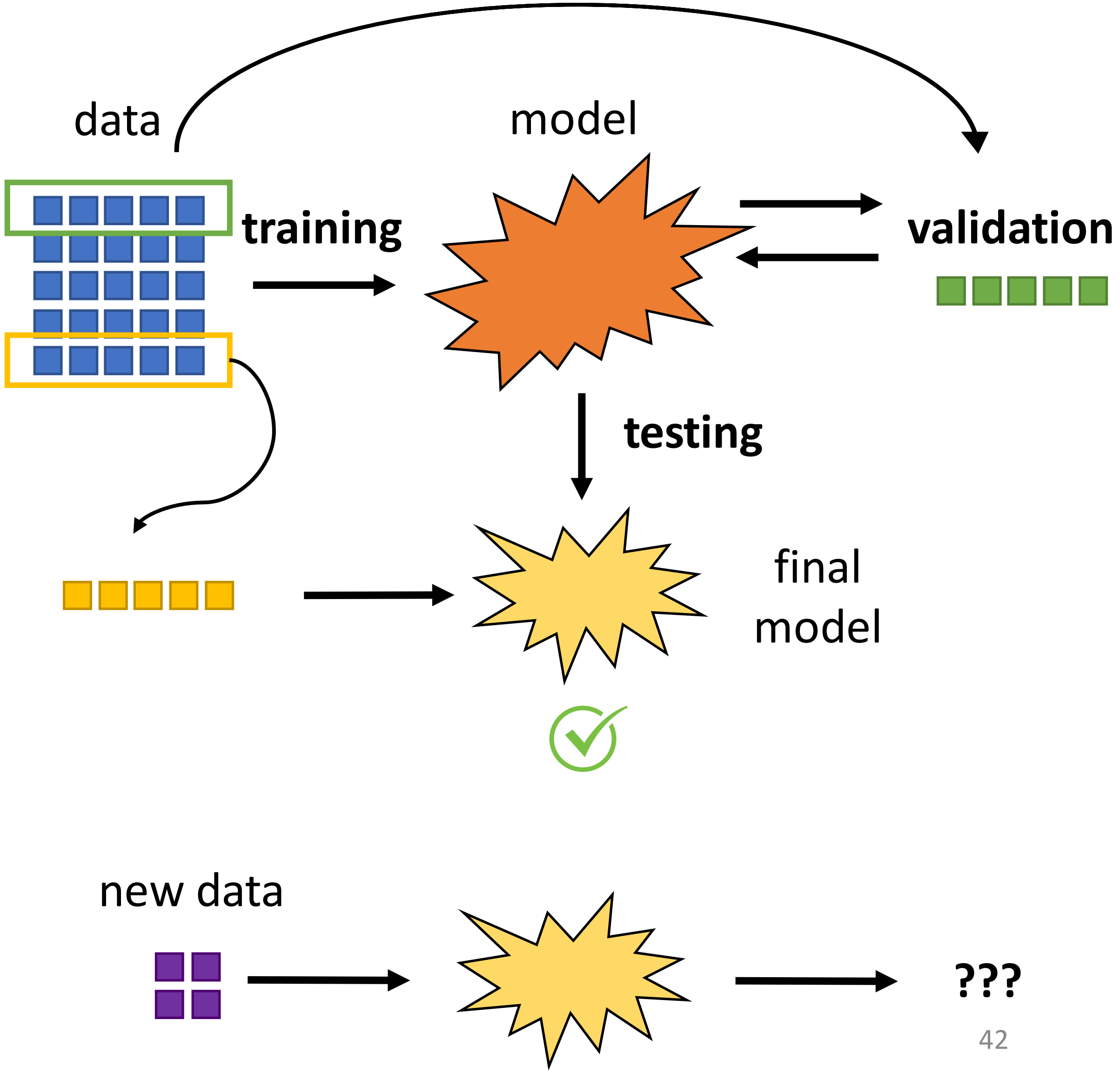
Example: Bad Training Process

What happens if we train on our test data?

The algorithm will get the opportunity to observe patterns in the test data. It will optimize the model to include those patterns.

When the model is tested, it will of course be accurate because the model was optimized to notice the correct patterns.

But if we try to use the model on new, unlabeled data later on, the patterns may no longer be valid. We don't know for sure because all the labeled data was used for training.



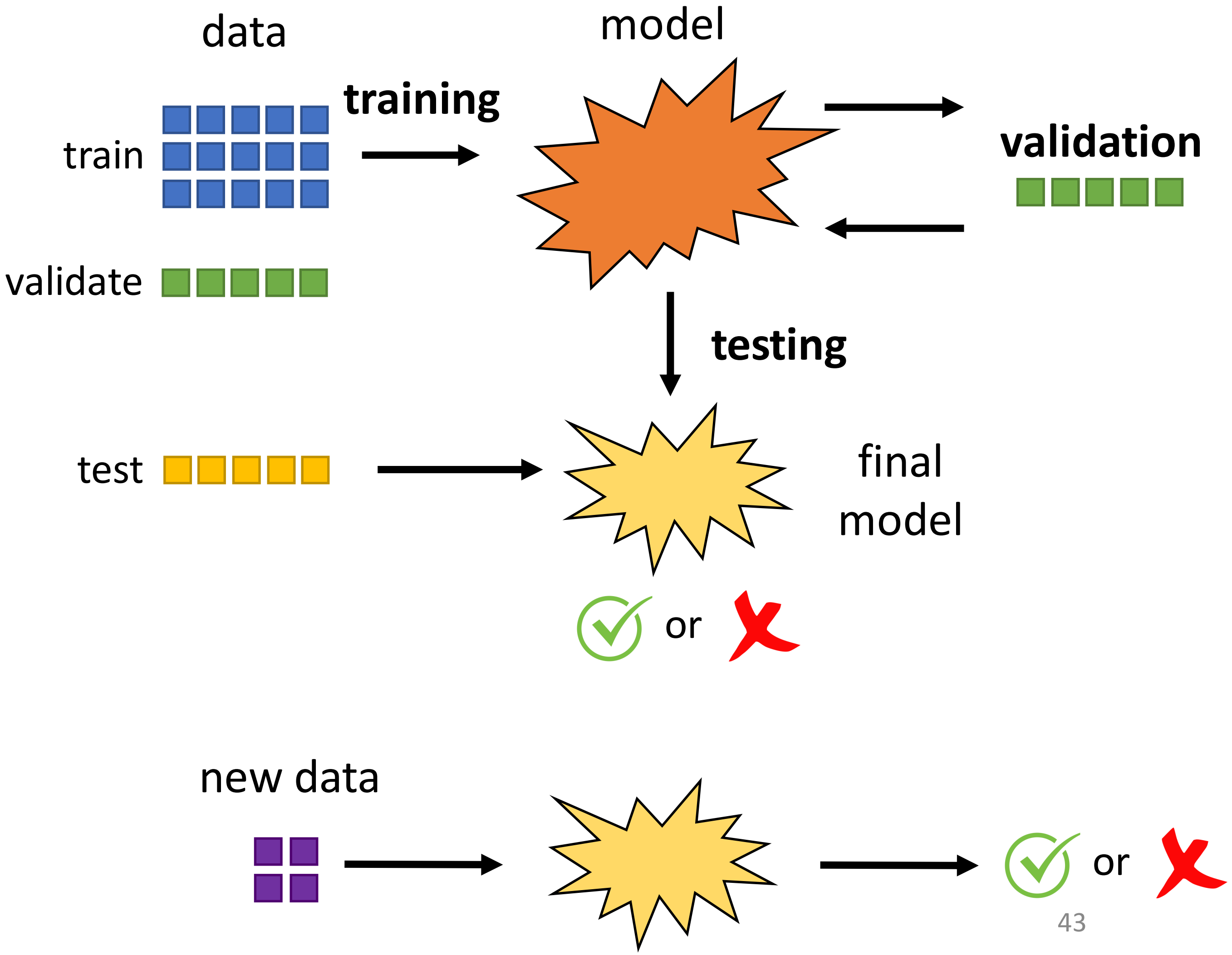
Example: Good Training Process

A better process: split the data into training, validation, and testing sets.

We'll train on the training set and repeatedly test on the validation set. This should remove some of the overfitting from the training data.

When we're done, we'll test on the test set **once**. That produces our final result. It might be good, or it might be bad; it depends on how the model turned out.

However, the new data should have about the same accuracy as the test data, since the model never saw the test data before. There's less uncertainty.



Testing the Model

To test our model in the code example, we just need to call the `score` method on the model using the test data we set aside earlier.

```
accuracy = model.score(testInput, testLabels)

print(accuracy) # 0.2617449664295303
```

The result will be an accuracy between [0, 1]. We get an accuracy of 31.1% - not great, but also not terrible considering the amount of variety in the possible answers.

Note that we had to set aside the test data first for this to work! If we trained on the test data, we'd get a faulty result.

Learning Goals

- Identify the three major categories of learning (**supervised**, **unsupervised**, and **reinforcement**) and the three major categories of reasoning (**classification**, **regression**, and **clustering**)
- Decide which combination of **learning** and **reasoning** categories are best used to solve a stated problem
- Describe how **training**, **validation**, and **testing** are used to build a model and measure its performance

Feedback: **https://bit.ly/110-f21-feedback**