Machine Learning 520

Advanced Machine Learning

Lesson 1: Introduction to Advanced Machine Learning

Today's Agenda

- Class Introduction
- Class policy and expectations
- How to succeed in this course
- Assignments, quizzes and milestones
- Participation and grading
- Grading expectations and supplementary material
- Introduction to Advanced Machine Learning
- Challenges in Machine Learning and how to overcome them

Learning Objectives

By the end of this lesson, you should be able to

- Implement two strategies for multiclass classification (lab).
- Apply implementation to data set .
- Apply insights gained from exploring a real-world case to thinking about how to address real-world problems, starting with a machine learning problem, then applying techniques.

Course Overview

- Lesson 01: Introduction to Advanced Machine Learning
- Lesson 02: Decision Tree
- Lesson 03: Random Forest & Gradient Boosted Trees
- Lesson 04: Support Vector Machine
- Lesson 05: Stacking and Blending
- Lesson 06: Unsupervised Learning
- Lesson 07: Natural Language Processing
- Lesson 08: Recommendation Systems
- Lesson 09: Forecasting
- Lesson 10: Building Machine Learning Applications

Student Introduction

- Name and current role
- Why enrolled in data science certificate
- Course and program expectations

Course Format

- On average, we spend about 40% on lecture and 60% on hands-on
- Let's have frequent discussions during the session and share diverse perspectives
- Let's take frequent short breaks to fit online format

How to succeed in this course

- Pay attention during class and ask questions
- Run notebooks alongside but don't let it distract you too much
- Learn from peers by participating in discussion boards and by sharing your code after assignment due dates
- Do your best not to fall behind, because lessons build on each other
- Basic to intermediate Python programming concepts should be second nature by now

Assignments, quizzes and milestones

- Assignments are due by 11:59 PM one week after lecture
- I will make no exceptions about assignment due dates
- Questions about the assignments should be posted on the discussion board on Canvas, where instructional team will answer them (or students if they wish)

How to submit a great assignment

- Your code should run from start to finish, so once you finish the assignment, restart the notebook and run again to make sure nothing breaks
- Each step in the assignment should include (1) one code cell with a few comments in the code to point out key parts, and (2) a Markdown cell explaining your reasoning and (3) a Markdown cell explaining your conclusions
- Someone who doesn't know Python could still be able to read through and understand the analysis
- When faced with ambiguity, you are free to make assumptions as long as you (1) clearly state your assumptions and (2) it is a reasonable assumption

let's take a tour through Canvas

Grading expectations

- Please use the discussion boards for questions on assignment
- DO NOT post your assignment code in discussion boards
- When necessary, it's generally OK to make an assumption about the assignment reqs - as long as you state your assumption
- More important to be on time than perfect:
- Meet the requirements with working code
- Write good comments for your code
- Write good explanations showing your line of thinking

Participation and grading

activity	what you need to do	grade
Class and discussion board participation	be active in **discussion boards**	15%
Quizzes		10%
Weekly Lab Assignments	submit by **11:59 PM the week after**	75%

Coding Environment

- We will be using browser-based Jupyter notebooks as our python environment
- Basics of Jupyter notebooks:
  - Code cells and Markdown cells
  - Running and re-running code cells - order matters!

Break time

What Is Machine Learning?

- An algorithm is a self-contained set of **rules or instructions** used to *solve problems*
- **Machine learning** is the field of study that gives computers the ability to learn *without being explicitly programmed*, by using data to learn
- The *problems* ML algorithms try to solve are usually
  - prediction: **supervised learning**
  - finding structure in data: **unsupervised learning**
  - Mimicking human behavior: **reinforcement learning** (not covered)

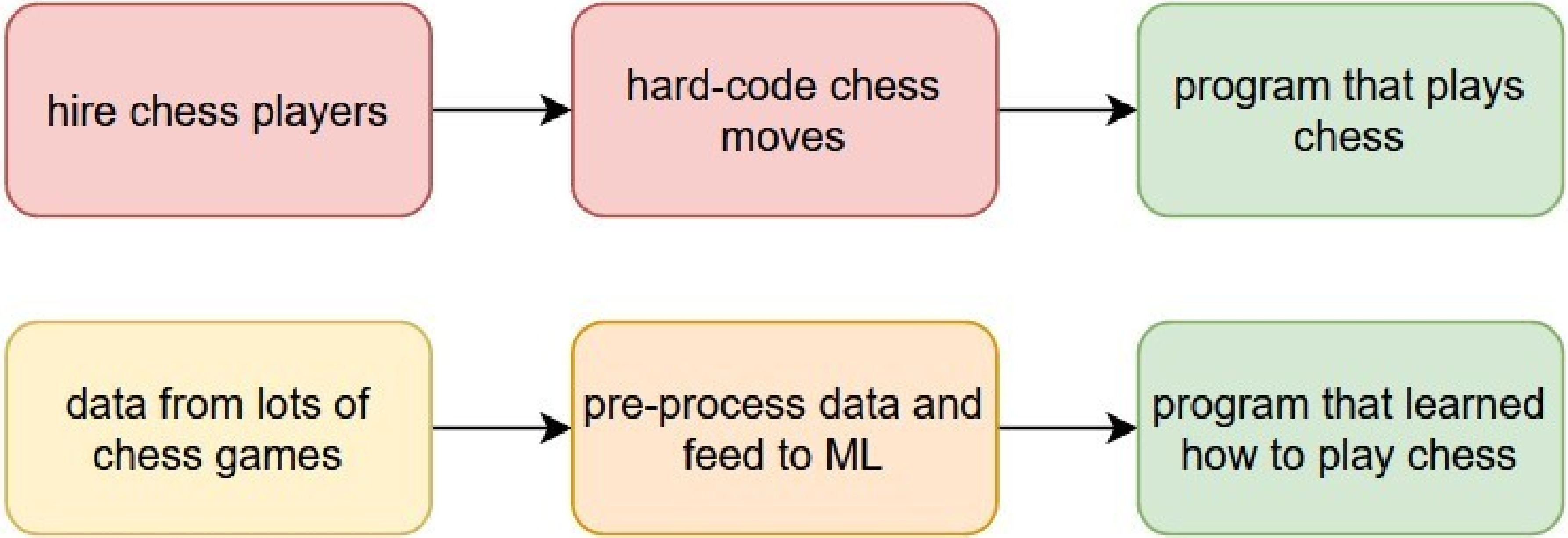
Machine learning algorithm

A machine learning algorithm walks into a bar.

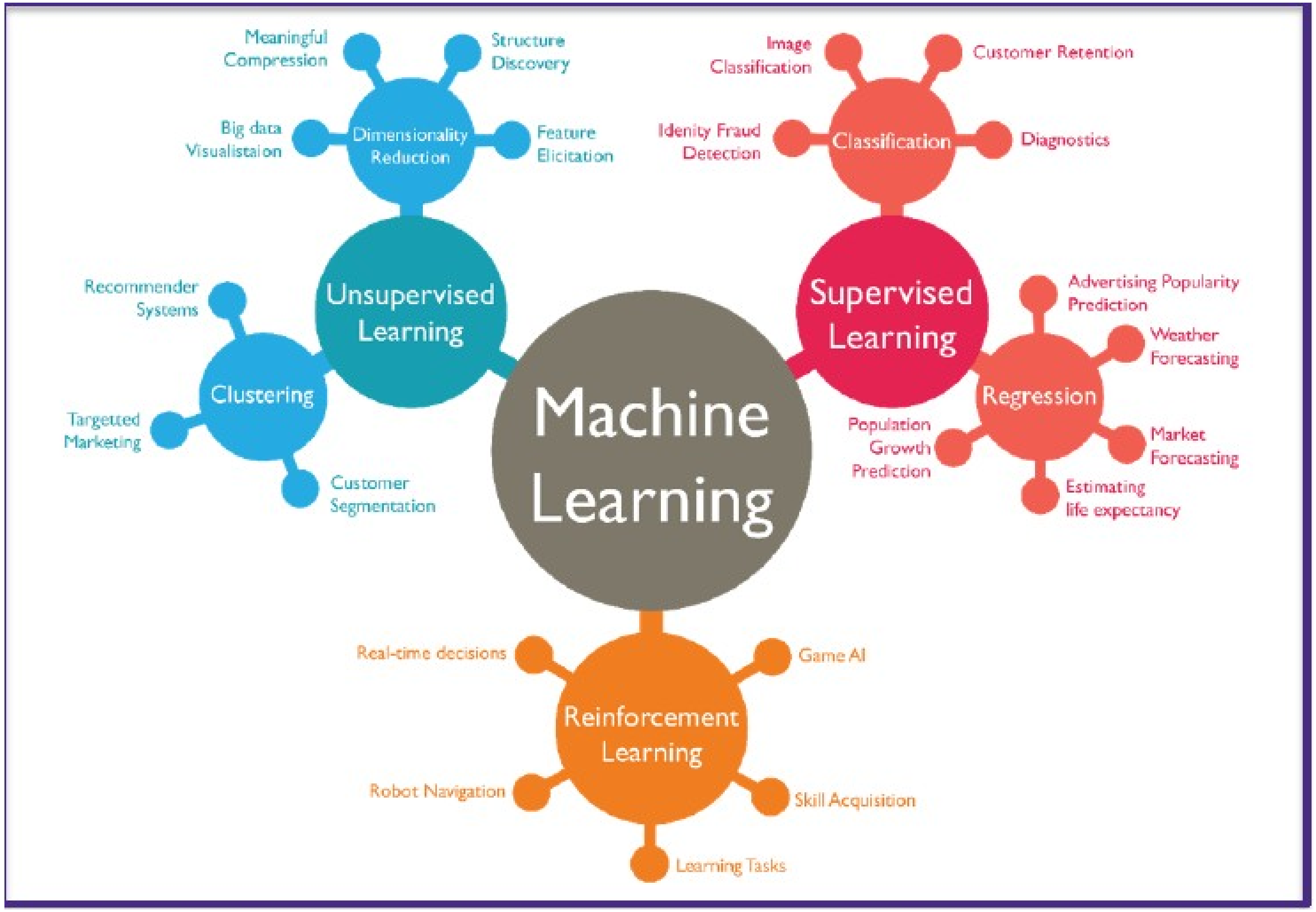
The bartender asks, "What'll you have?"

The algorithm says, "What's everyone else having?

Rule-based Vs Machine Learning?



Types of Machine Learning Algorithms



https://blogs.oracle.com/ai/types-of-machine-learning-and-top-10-algorithms-everyone-should-know

Supervised Learning

- Also called **predictive modeling** or **inference**
- Look at some current examples (**labeled data**) and find a model that can predict future examples (**unlabeled data**)
- The **target variable** or **label** is what we want to predict
    - **Regression** algorithms are used with a **numeric target**
    - **Classification** algorithms are used with a **categorical target**
- By comparing predictions with the actual labels, we can evaluate our model's accuracy (hence the term *supervised*)

Unsupervised Learning

- Also called data-mining / pattern recognition / structure discovery
- Look at **unlabeled data** and find general patterns
- More subjective and difficult to evaluate and interpret, and hence it is far **less common** than supervised learning
- **Clustering** is the most common example
  - K-means clustering
  - Variable clustering / dimensionality reduction
  - Word clouds (kind of)

Reinforcement Learning

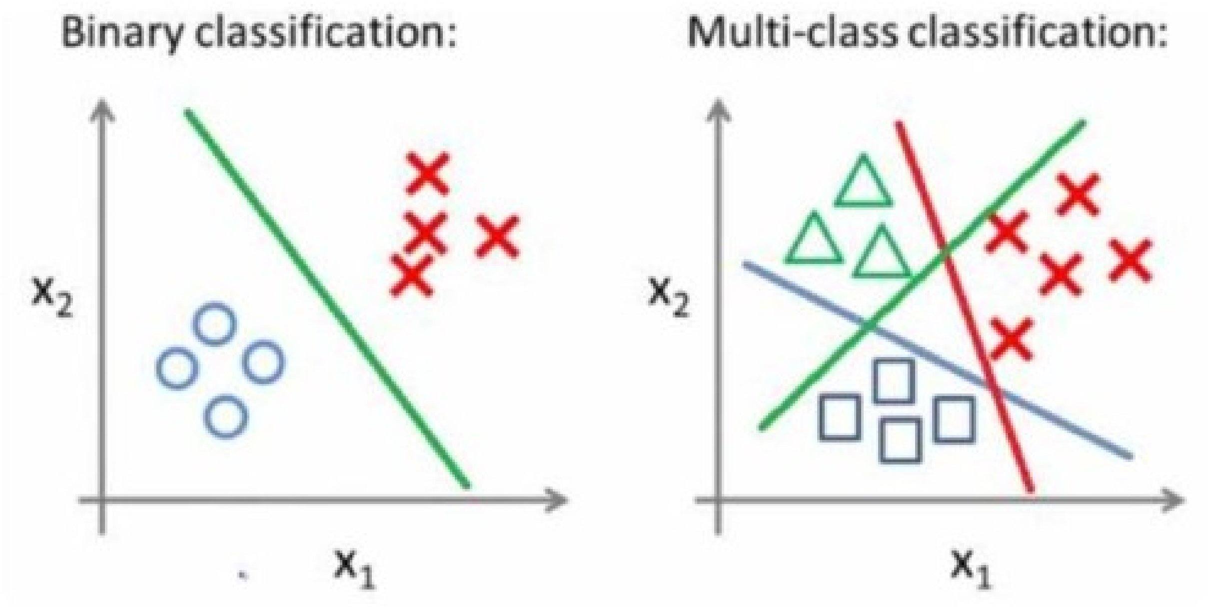
- Learn to take actions in an environment to maximize expected future

Discussion

Task	Learning Type
Housing price estimates on Redfin	?
Product recommendations on Amazon	?
Voice recognition on Alexa	?
Play the game of Go	?
Grouping Air Bnb listings into neighborhoods	?
Predict delivery times for Uber Eats	?
Segmenting US population for targeted advertising	?
Learn to play Starcraft	?
File a tax return	?

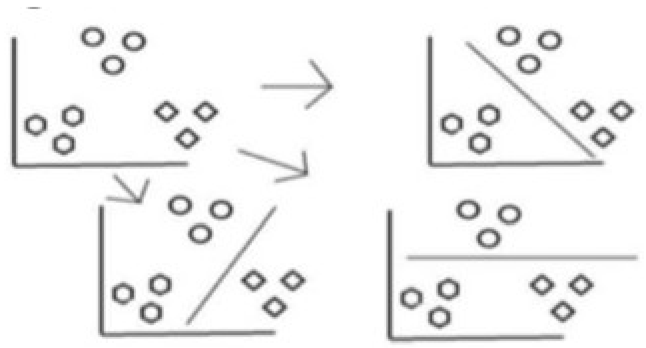
Multi-Class Classification

- Multi-class classification generalizes the problem where classifier
  - is expected to distinguish between more than two classes
  - Hand-written digits (0, 1, …, 9)
  - Image classification (cat, dog, human etc.)
  - Classifying different species of flowers



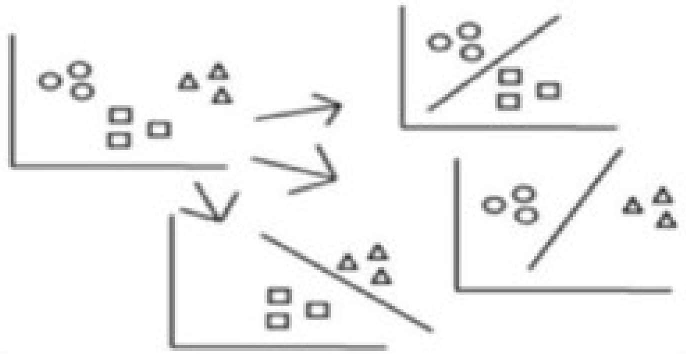
Multi-Class Classification – One vs. All

- One vs. All is a way to handle multi-class classification
- **Training**
  - **Fit a binary classifier per class, with the samples of that class as positive samples and all other samples as negative**
  - **Base classifiers produce a real-valued confidence score for its decision, instead of just a class label**
- **Prediction**
  - **Apply all trained classifiers to the new unseen data point and select the class with highest score**



Multi-Class Classification – One vs. All

- A second strategy is One vs. One
- **Training**
  - Fit a binary classifier for every pair of classes. Therefore, if there are N classes, we need N*(N-1)/2 classifiers
- **Prediction**
  - A voting scheme is applied
  - All N*(N-1)/2 classifiers are applied to an unseen sample and the class that gets the highest number of votes is selected



Underfitting vs. Overfitting

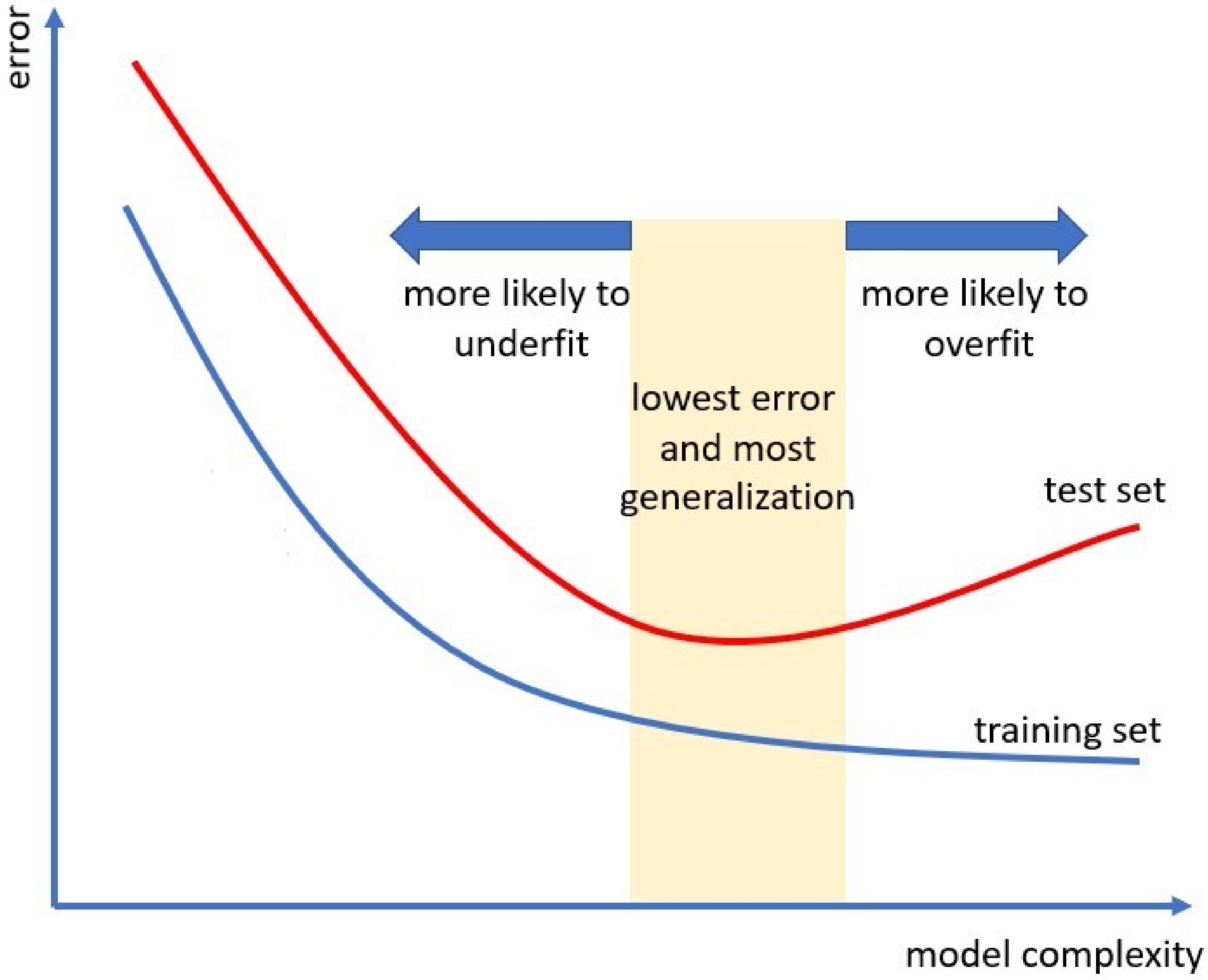
- Fitting means **learning** when we call the `.fit()` method
- If a model performs poorly on the training data, then it almost certainly will perform poorly on the test data as well: we say the model is **underfitting** (not learning enough)
- If a model performs well on the training data, but poorly on the test data: we say the model is **overfitting** (it's learning the signal but also "learning" the noise in the training data, and hence fails to generalize)
- A good model is one that neither underfits nor overfits

Overfitting and Complexity

- More simple models tend to underfit, because they are more likely to oversimplify (not pick up enough signal)
  - Analogy: models can be prejudiced too, we call it **bias**
- Complex models tend to overfit, because they are so eager to pick up any signal that they also grab noise disguised as signal
  - **Analogy:** people who **read too much into** something
- The trick is to find the happy medium

Everything should be made as simple as possible, but not simpler. -**Albert Einstein**

(also look up **Occam's Razor**)


error
more likely to
underfit
more likely to
overfit
lowest error
and most
generalization
test set
training set
model complexity

What Are Hyper-Parameters?

- Almost all algorithm have ways they can be "tuned" through different **hyper-parameter** choices
- Some hyper-parameters are very generic, e.g. **learning rate**
- Most hyper-parameters are algorithm-specific, such as
  - For tree-based algorithms: tree depth, min leaf size
- ML algorithms **cannot** directly learn optimal hyper-parameter values during training
- If we don't specify them, they usually default to "reasonable" values

Model Selection and Validation

- So how do we know what hyper-parameter values to pick?
  - Mostly through trial and error, also called **model selection**
- If we want to tune our hyper-parameters, we also need **validation data** in addition to training and testing

1. Train many models, each with different hyper-parameters, and evaluate their performance on the validation data
2. The model that performs best on the validation data wins
3. Check how the winner model performs on the test data

Regularization

- **Reduces overfitting** and the extent of it can be adjusted or **tuned**
- Instead of minimizing prediction error only, minimize **prediction error + a penalty term** where the penalty term is higher when model coefficients are higher

- Prefers models with **smaller coefficients** (features must be normalized), **unless** they significantly improve prediction

L1 vs L2 Norm Regularization

- has sharp edges
- has round edges
- can drop features

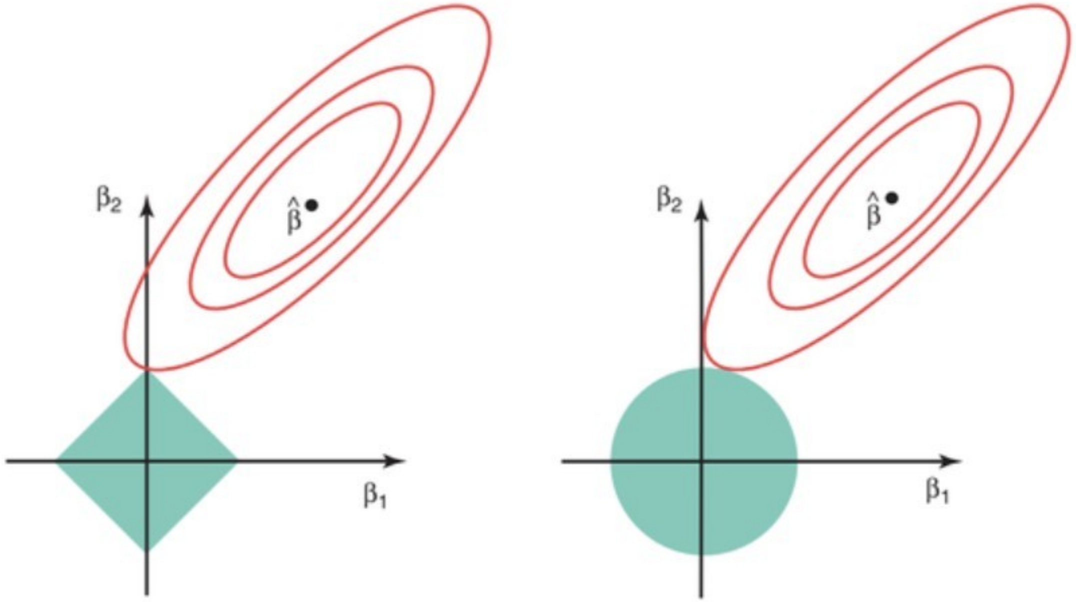


Image source: Elements of Statistical Learning

Types of Regularization

- **L2 regularization** (such as in **ridge regression**) is good when we have fewer and more relevant features and we want to keep all
- **L1 regularization** (such as in **lasso regression**) is good when we have lot's of features and many are probably not relevant
- We can also combine **L1 regularization** and **L2 regularization**, such as in **elastic-net regression**, where we tune in addition to

Test Data vs. Validation Data

So why not combine test data and validation data?

- Because the test data is used **once** with the **final model** to get an **unbiased estimate** of performance (prediction error)
- The validation data is used **many times** so we can compare the performance of models trained with different sets of hyper-parameters, a.k.a. **hyper-parameter tuning**
- If we also use the test data to tune hyper-parameters, we are over-using it and its estimate of performance will not be so unbiased anymore

Cross Validation

- Choose number of folds
- Train k times, each using folds to train and the th to evaluate
- Aggregate the evaluation results from all models into one measure

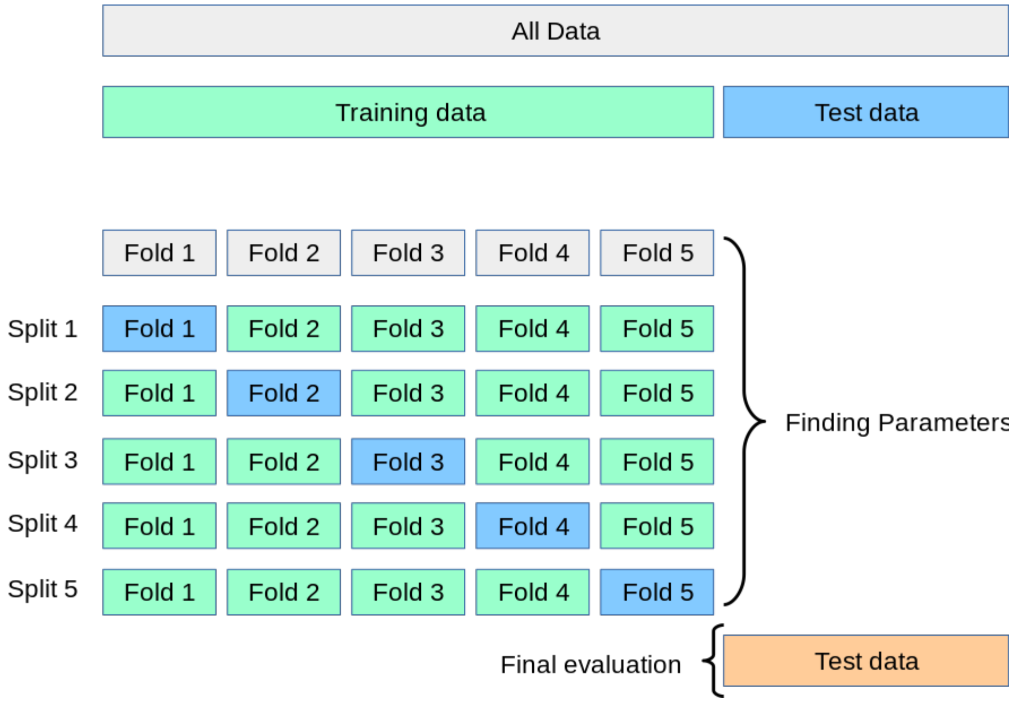
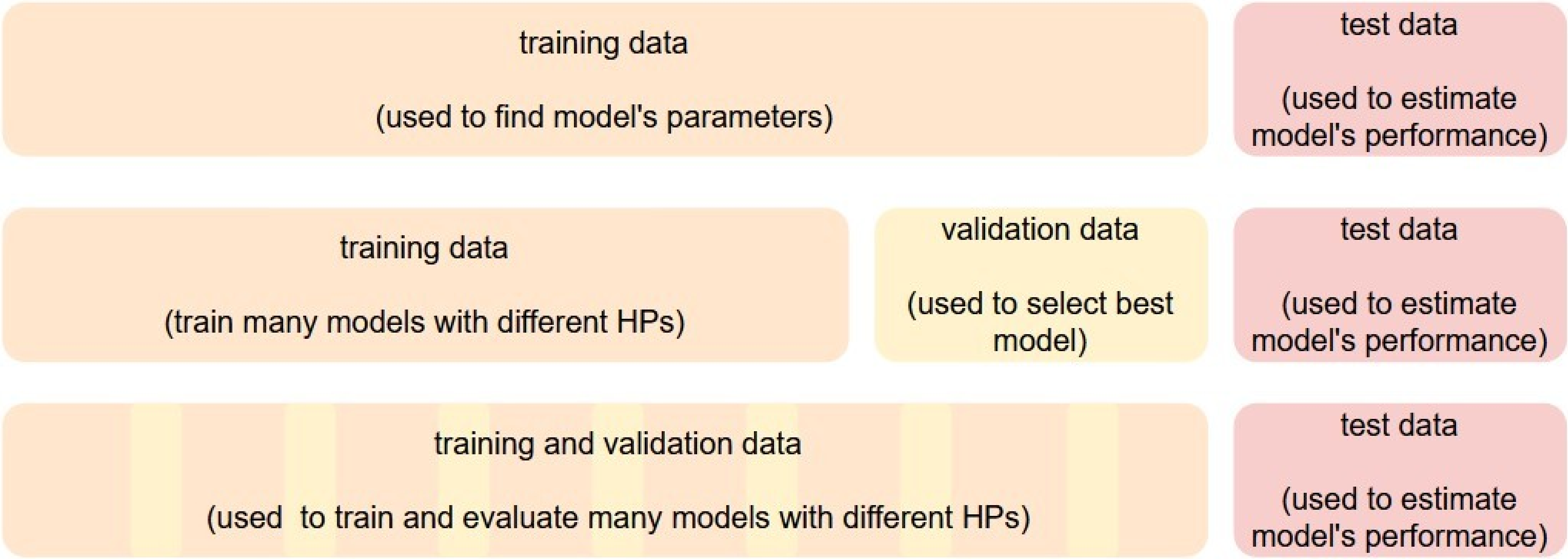
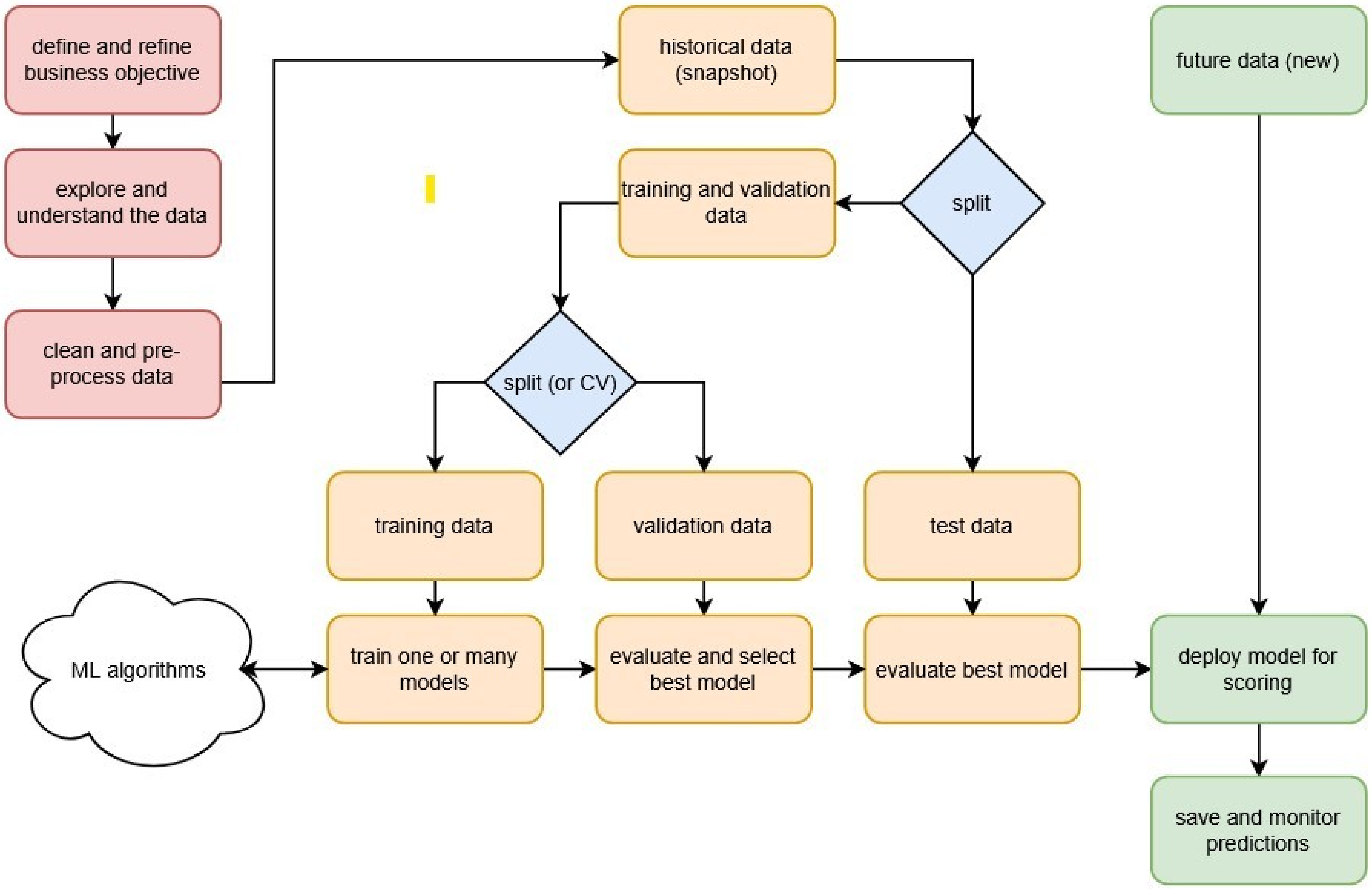


Image source: sciit-learn.org


training data
(used to find model's parameters)
test data
(used to estimate model's performance)
training data
(train many models with different HPs)
validation data
(used to select best model)
test data
(used to estimate model's performance)
training and validation data
(used to train and evaluate many models with different HPs)
test data
(used to estimate model's performance)


define and refine business objective
explore and understand the data
clean and pre-process data
historical data (snapshot)
split
training and validation data
split (or CV)
training data
validation data
test data
future data (new)
ML algorithms
train one or many models
evaluate and select best model
evaluate best model
deploy model for scoring
save and monitor predictions

The Confusion Matrix

It's really not that confusing!

	predicted positive	predicted negative
actually positve	true positive TP	false negative FN
actually negative	false positive FP	true negative TN

For TP/FP/TN/FN

- The second letter indicates what the prediction was
- The first letter indicates if the prediction was right or not

Discussion

- We saw there a binary classification model can make two kinds of errors: FP and FN
- For the following scenarios, say what kind of error is more costly (use common sense)
  - **Credit card fraud detection**: someone impersonates you to use your credit card
  - **Medical diagnosis**: finding out who has a disease
  - **Information retrieval**: finding relevant web pages based on a search query

Precision and Recall

- **Precision** is the percentage of positive predictions that were actually positive
- **Recall** is the percentage of positive cases that we correctly predicted as such

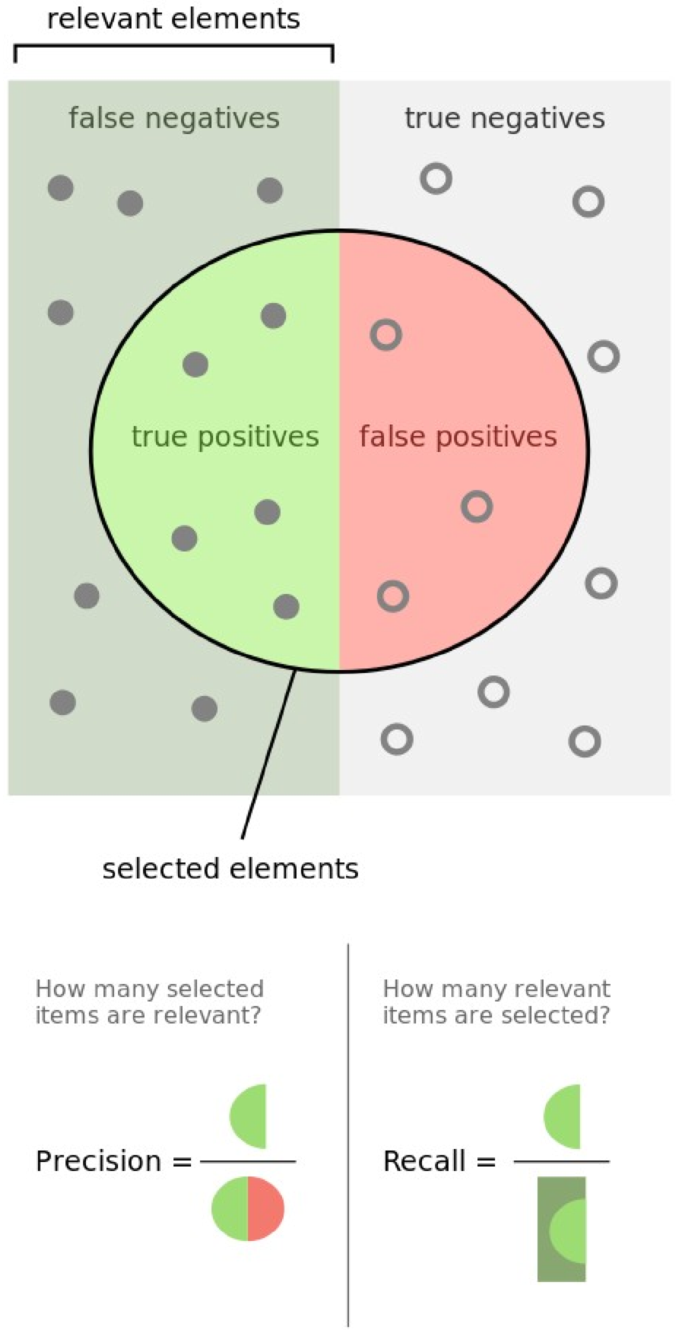


Image source: Wikipedia

Accuracy, Precision and Recall

- For rare events usually TN far exceeds TP, FN, or FP, inflating accuracy, but precision and recall don't have TN in it

Discussion

Here's an analogy that shows why we should evaluate a classification model's accuracy using **both** precision and recall:

- When you stand witness in a court of law, you are asked to tell the truth:
  - **The whole truth**: no lie of omission
  - **Nothing but the truth**: no lies
- Relate the above two statements to precision and recall

Notebook time

The end