APS1070

Foundations of Data Analytics and Machine Learning

Winter 2022

Week 3:

- *End-to-end Machine Learning*
- *Data Retrieval and Preparation*
- *Plotting and Visualization*
- *Making Predictions*
- *Decision Trees*

Sinisa Colic and Samin Aref

Agenda

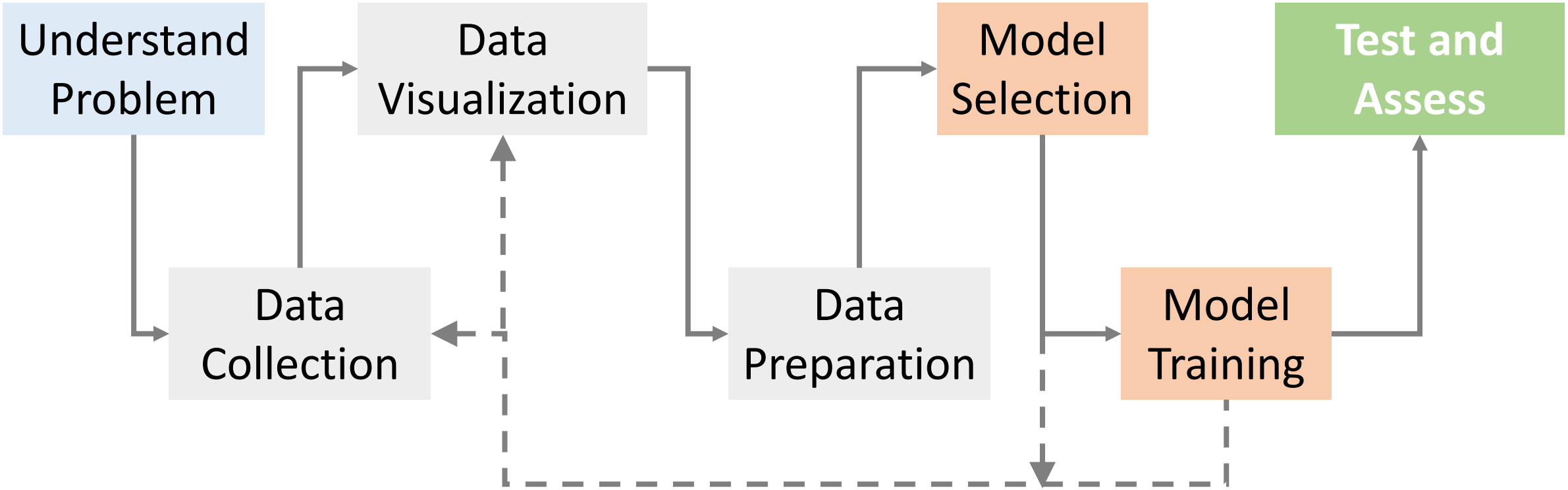
- Today's focus is on **Foundations of Learning**
  1. End-to-end machine learning
  2. Python Libraries
     - NumPy
     - Matlplotlib
     - Pandas
     - Scikit-Learn
  3. Decision Trees

Part 1
End-to-End Machine Learning

End-to-End Machine Learning

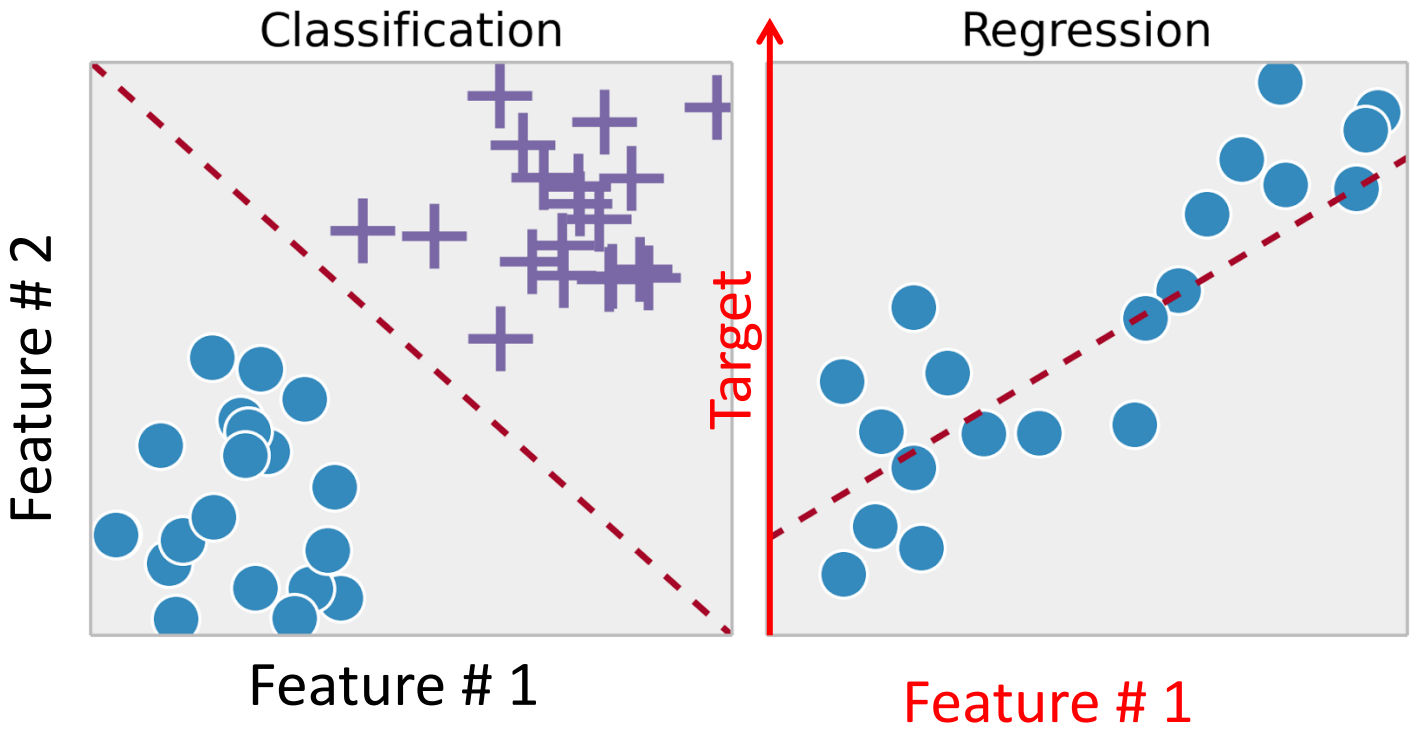
1. Understand the problem
2. **Retrieve the data**
3. **Explore and visualize the data to gain insights**
4. **Prepare the data for the algorithm/model**
5. **Select and train the algorithm/model**
6. **Fine-tune your algorithm/model**
7. Present your solution
8. Launch, monitor, and maintain your system

End-to-End Machine Learning



Classification vs. Regression

- **Classification:** Discrete target
  - Separate the Dataset
    - Apples or oranges?
    - Dog or Cat?
    - Handwritten digit recognition
- **Regression:** Continues Target
  - Fit the dataset
    - Price of a house
    - Revenue of a company
    - Age of a tree



Understand the Problem

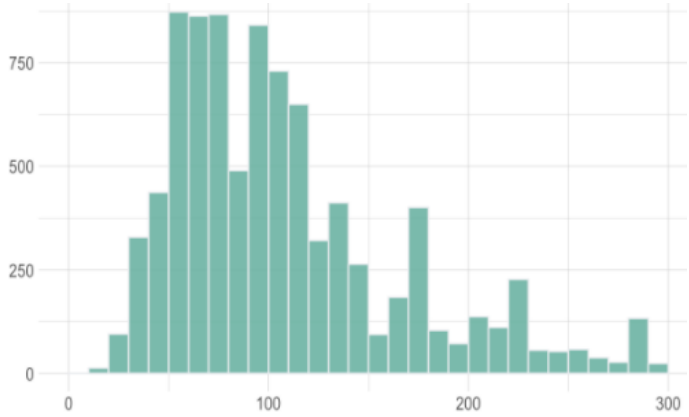
- Often, we need to make some sort of decisions (predictions)
- Two common types of decisions that we make are:
  - Classification
    - Discrete number of possibilities
  - Regression
    - Continuous number of real-valued possibilities

	Supervised	Unsupervised
Discrete	classification	clustering
Continuous	regression	dimensionality reduction

Understand the Problem

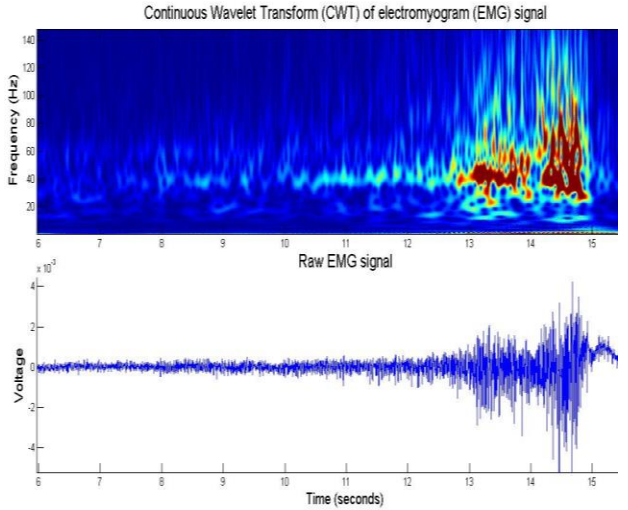
Input data is represented by features that can come in many forms:

- Raw pixels
- Histograms
- Tabular data
- Spectrograms
- …



	Gulf Respondents			Non-Gulf Respondents		
	Sample			Sample		
	Eats Oysters	Does Not	Census Range	Eats Oysters	Does Not	Census Range
N =	444	72		269	35	
Age 65+ (%)	0.23	0.21	0.08–0.16	0.20	0.29	0.10–0.14
Male (%)*,†	0.47	0.32	0.50–0.53	0.67	0.63	0.49–0.53
White (%)	0.65	0.68	0.33–0.73	0.69	0.77	0.30–0.85
Persons per household	2.48	2.29	2.30–2.70	2.58	2.49	2.10–2.80
High school graduate (%)	0.96	0.96	0.72–0.93	0.95	1.00	0.80–0.93
College graduate (%)*	0.50	0.43	0.24–0.49	0.58	0.46	0.22–0.58
Household income category*	$40,000 –$49,999	$40,000 –$49,1000	$30,858 –$52,971	$75,000 –$84,999	$60,000 –$74,999	$34,800 –$78,378

Notes: Asterisk (*) indicates significant difference between sample Gulf and non-Gulf oyster eaters. Dagger (†) indicates significant difference between sample Gulf oyster eaters and noneaters.
Source: U.S. Census Bureau (2016).

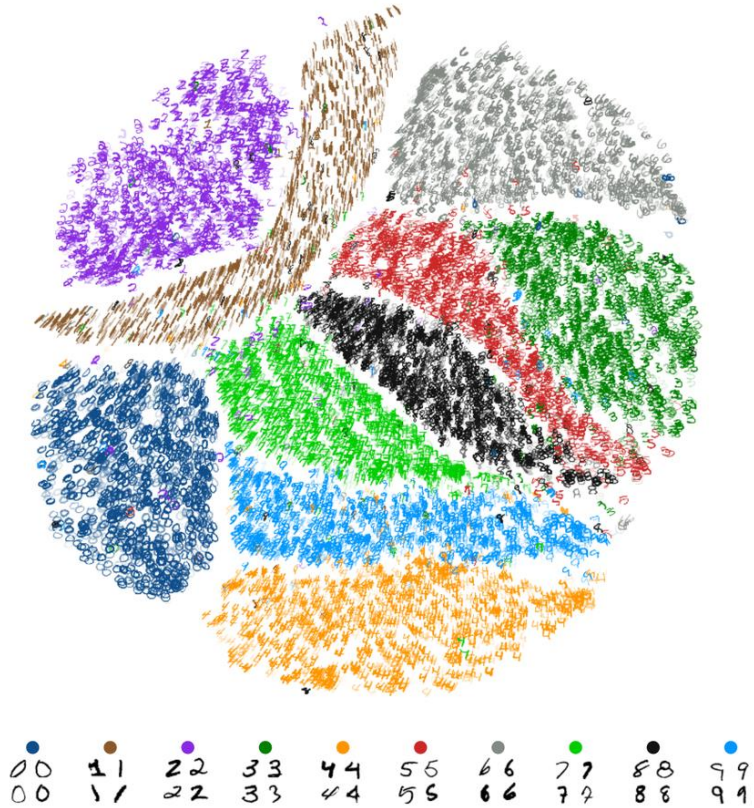


Data Exploration

- Understand your data through visualization
- Assess the difficulty of the problem

- You have a data set D = $\{(x^{(i)},y^{(i)})\}$
- You want to learn y = f(x) from D
  - more precisely, you want to minimize error in predictions

- What kind of model (algorithm) do you need?



MNIST low-dimensional projection

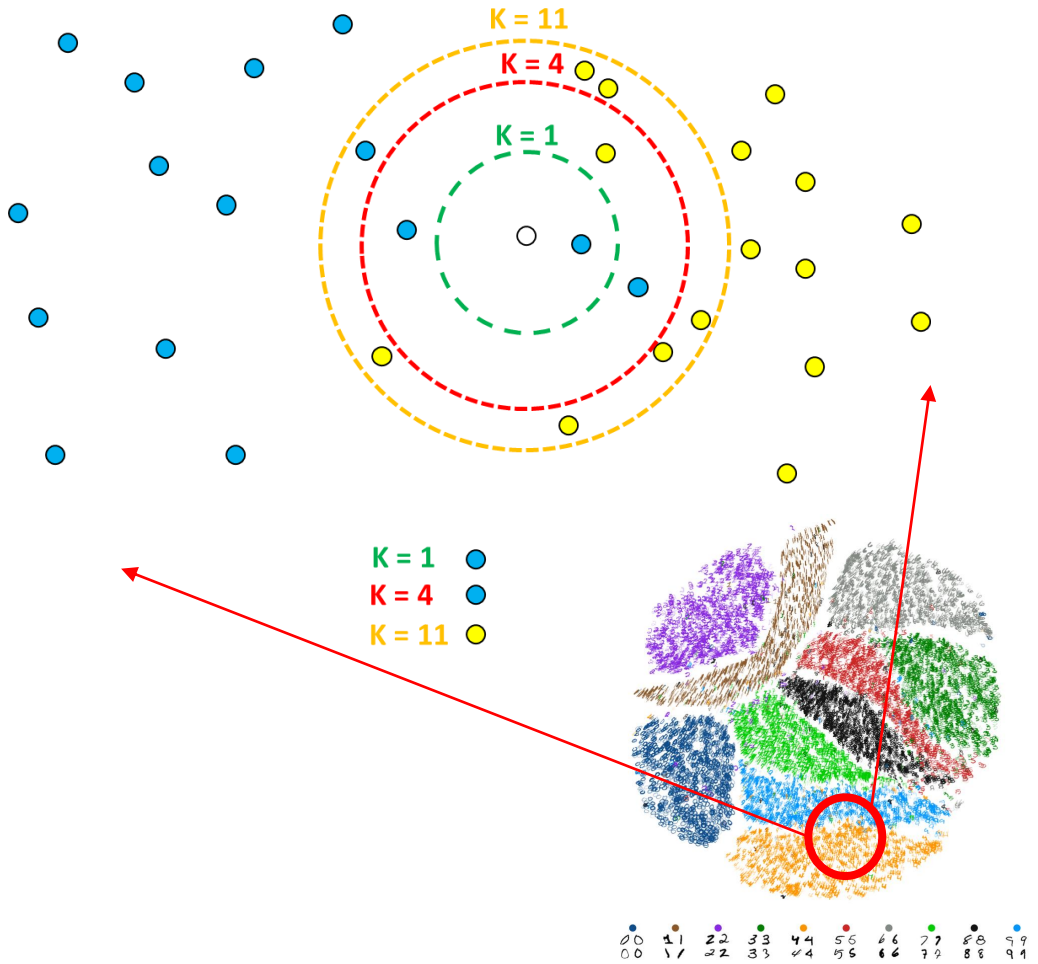
Model Selection

Many classifiers to choose from

- Support-Vector Machine (SVM)
- Logistic Regression
- Random Forests
- Naive Bayes
- Bayesian network
- K-Nearest Neighbour
- (Deep) Neural networks
- Etc.

Model Selection

- Often the easiest algorithm to implement is k-Nearest Neighbours
- Match to similar data using a distance metric



Q: What happens as we increase #data?

Q: What about as #data approaches infinity?

Test and Assess

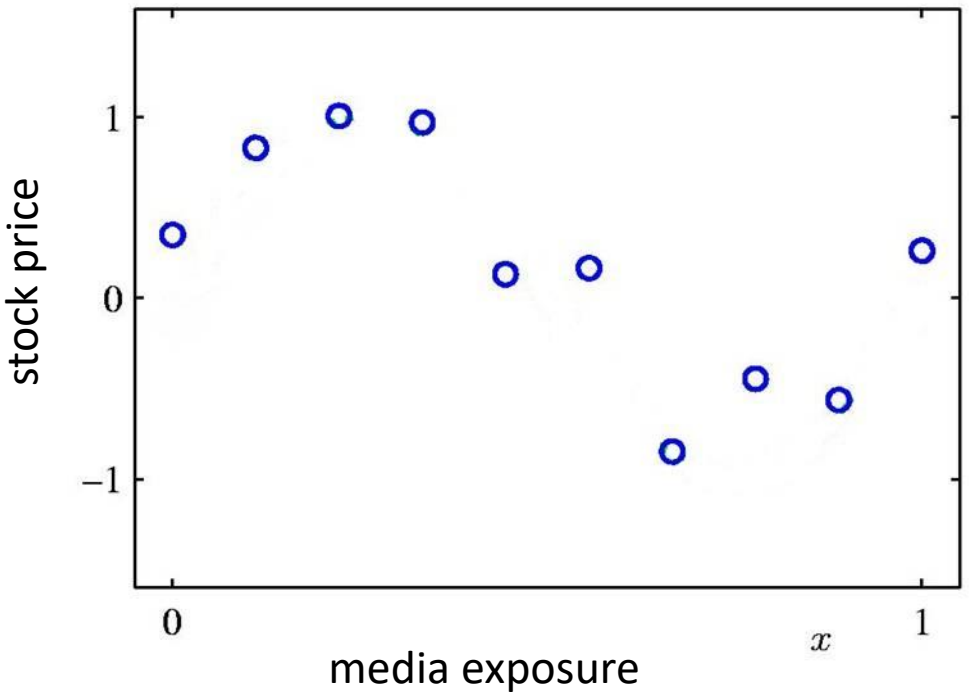
- Unlike us, computers have no trouble with memorization.
- The real question is, how well does our algorithm make predictions on new data?

- We need a way to measure how well our algorithm (model) **generalizes** to new, never before seen, data.

Regression Example

- Let's look at a more concrete example…
- Given noisy sample data (blue), we want to find the polynomial that generated the data

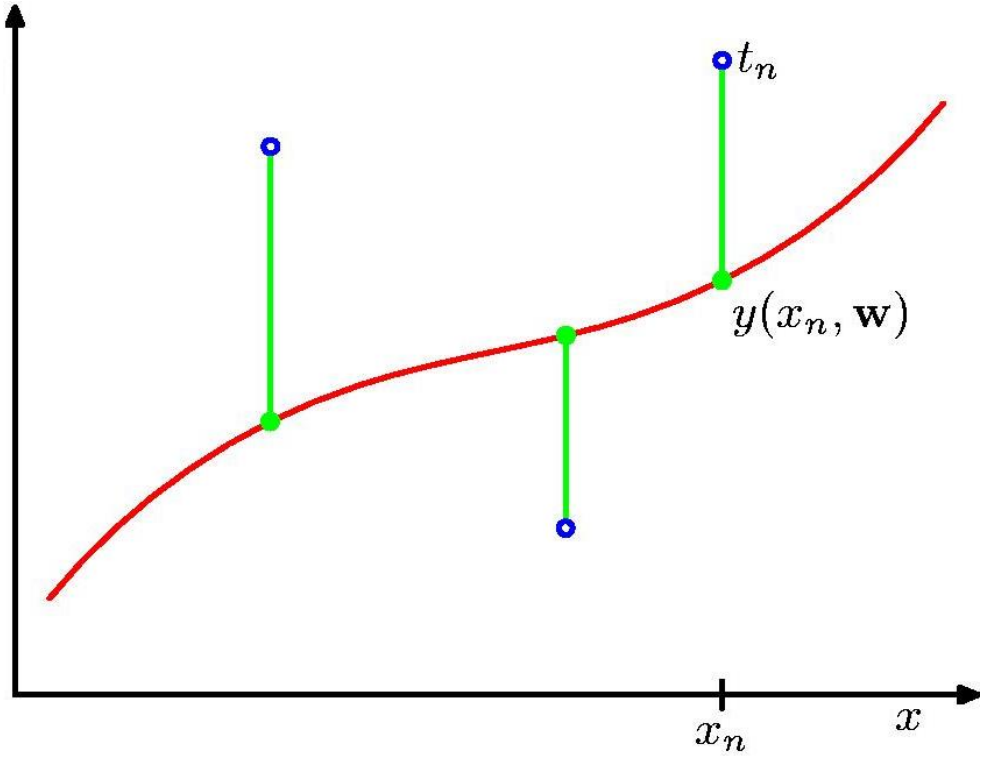
- Q: What kind of a problem is this?



From PRML (Bishop, 2006)

Mean Squared Error

- Need to first define our error term, in this case we can use the **mean squared error (MSE)**:
- Error is measured by finding the squared error in the prediction of y(x) from x.
  - The error for the red polynomial can be measured based on the mean of the squared vertical errors

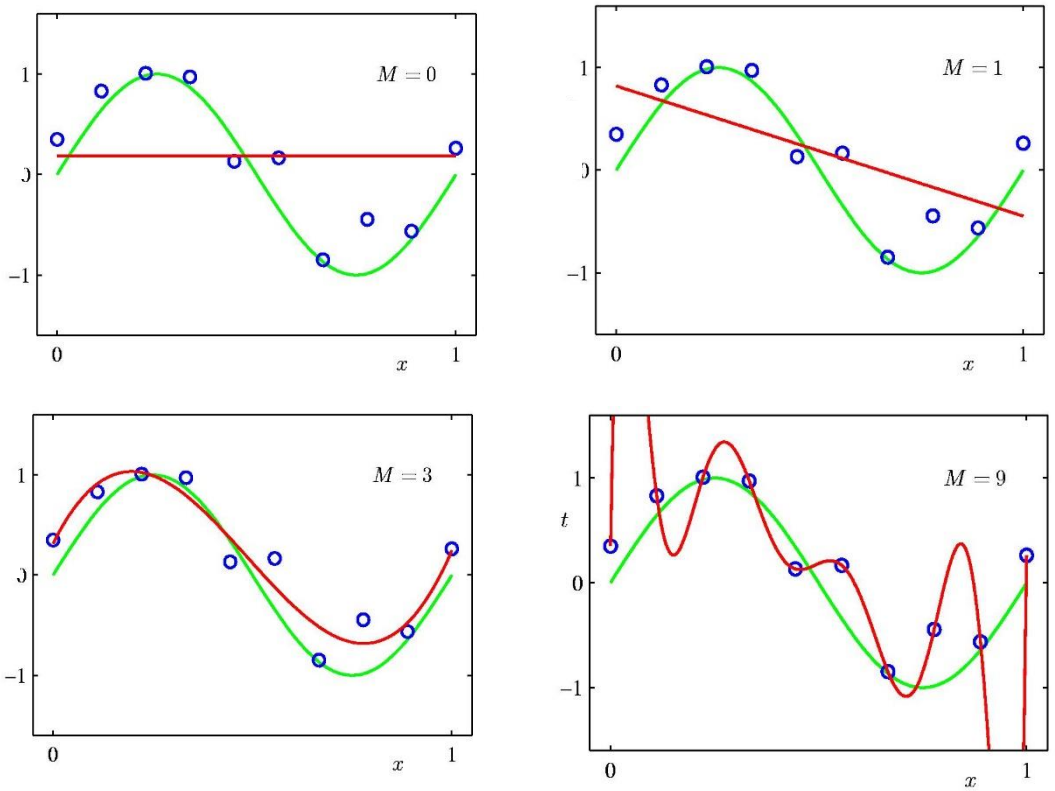


From PRML (Bishop, 2006)

Fitting the Data

Q: Which polynomial fits the data best?

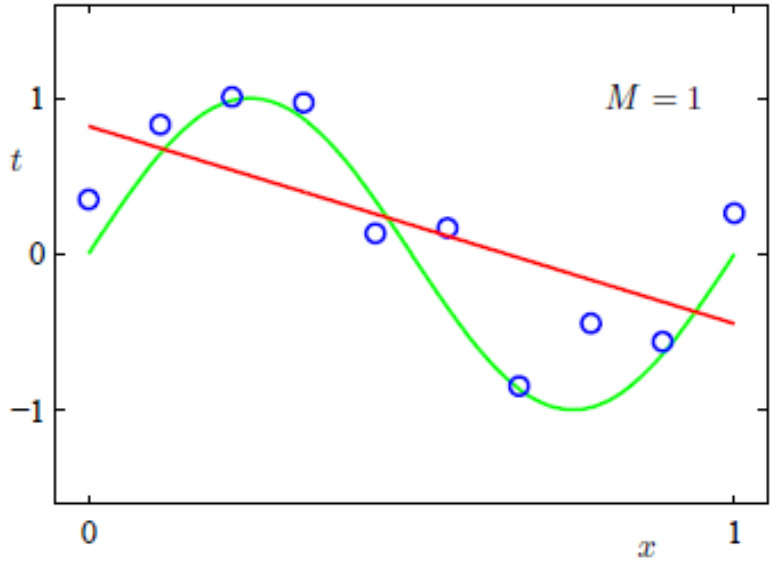
- based on training data?
- based on test data?



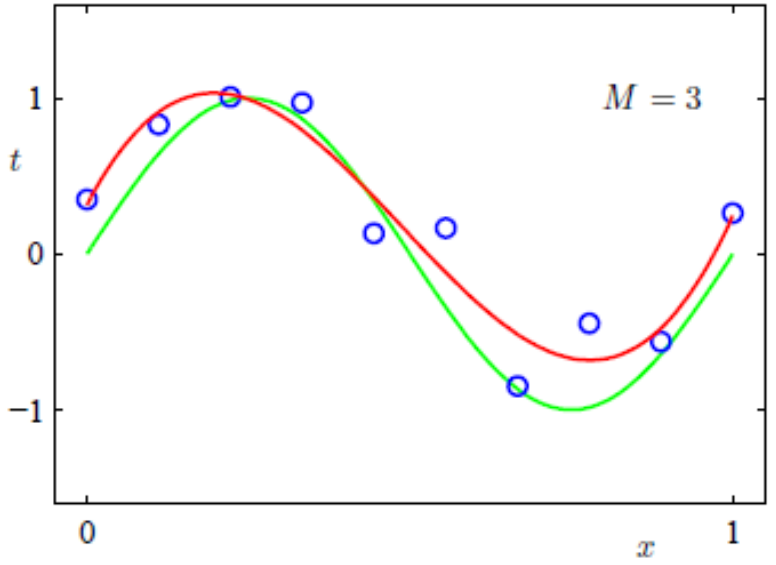
From PRML (Bishop, 2006)

Overfitting vs Underfitting

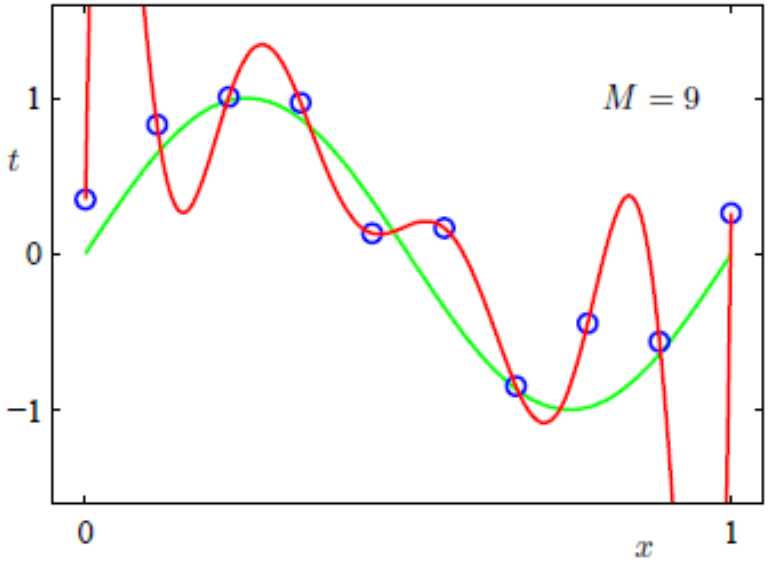
High training error
and high test error (Underfit)



Acceptable training error
and test error



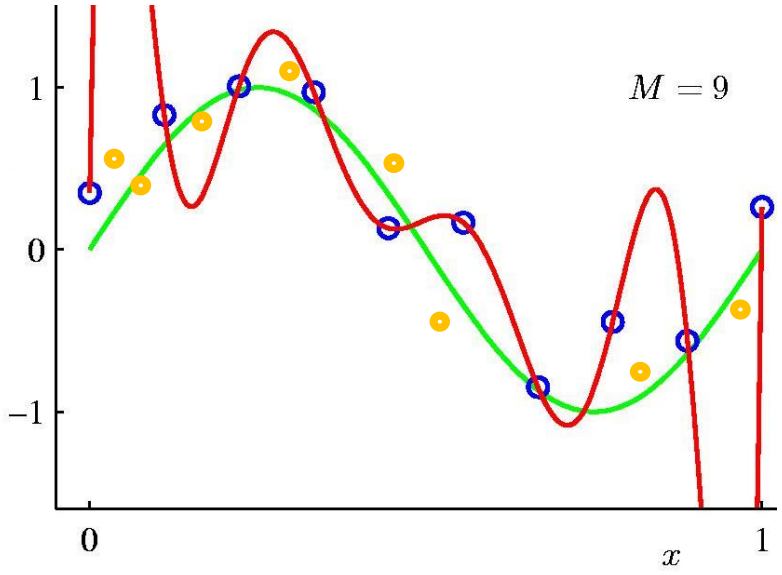
Perfect training error (zero)
and high test error (Overfit)

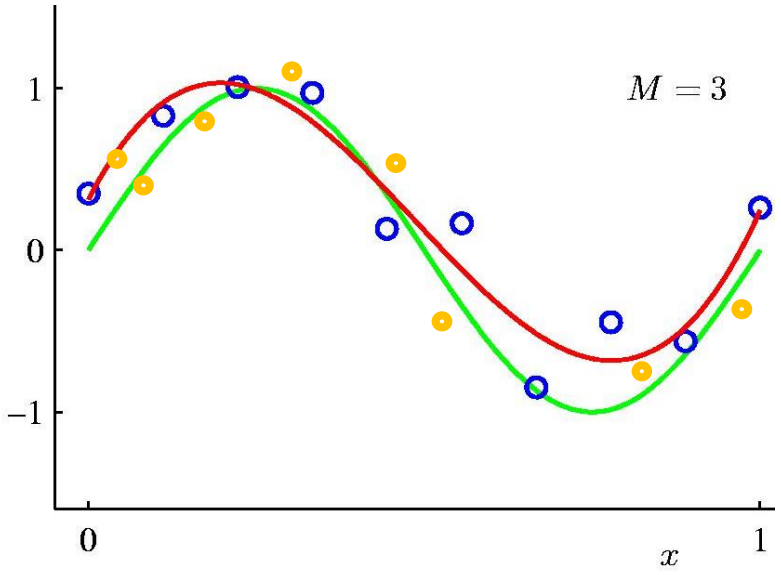


From PRML (Bishop, 2006)

Generalization

- Giving the model a greater capacity (more complexity) to fit the data… does not necessarily help
- How do we evaluate the model performance?





From PRML (Bishop, 2006)

Verify model on New Data

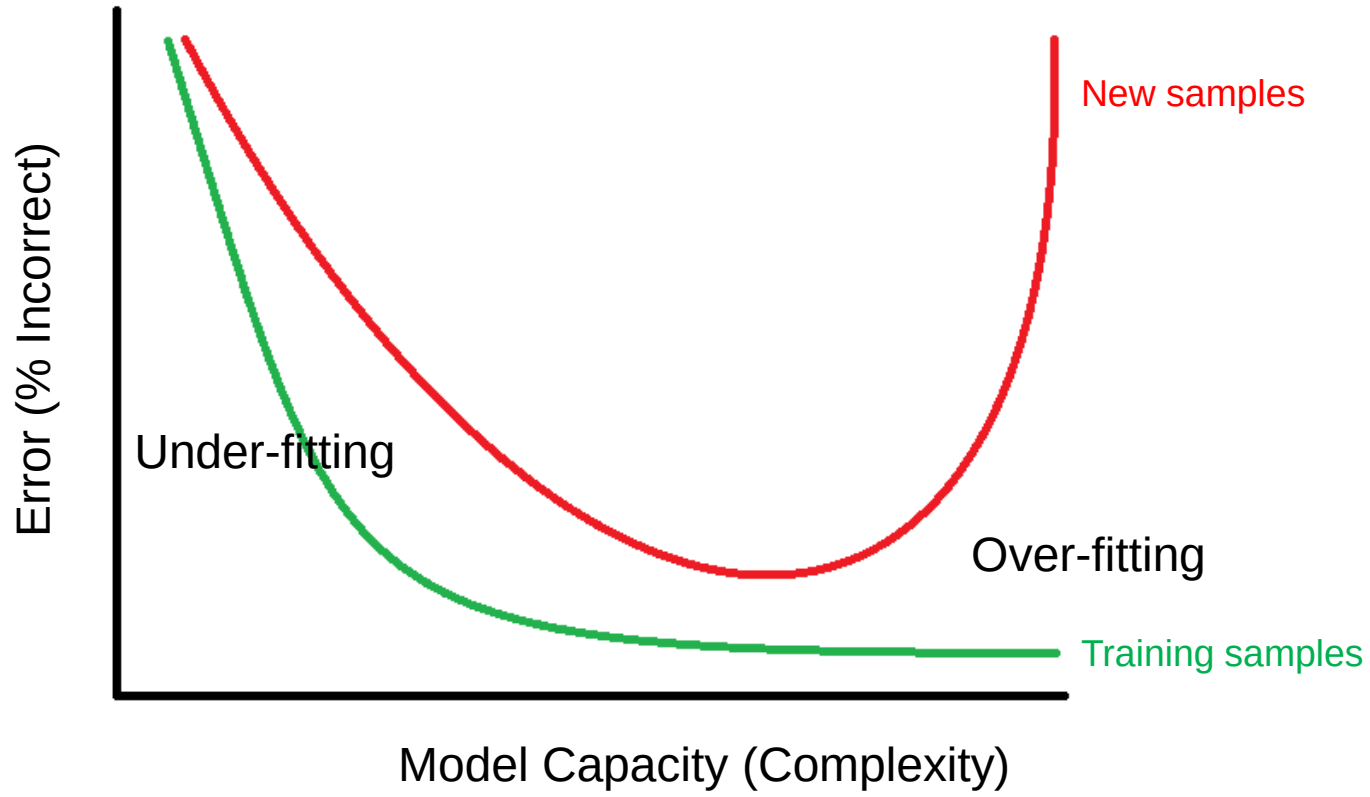
Overfitting

- In brief: fitting characteristics of training data that do not generalize to future test data

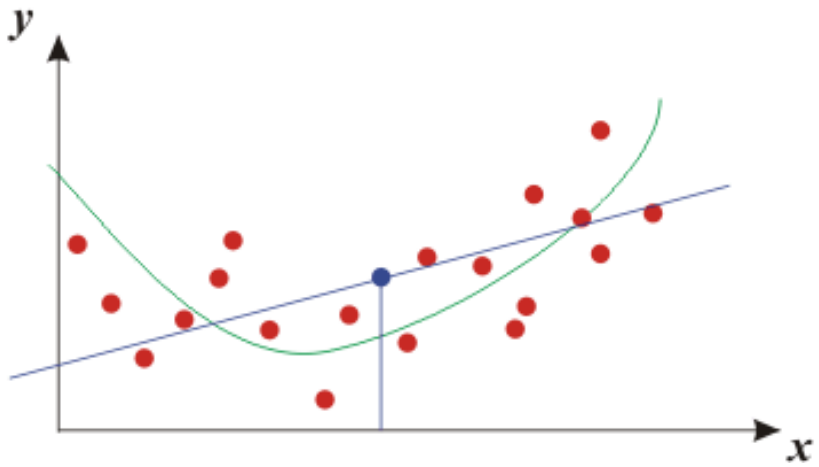
- Central problem in machine learning
- Particularly problematic if #data << #parameters
- … don’t have enough data to “identify” parameters

Generalization

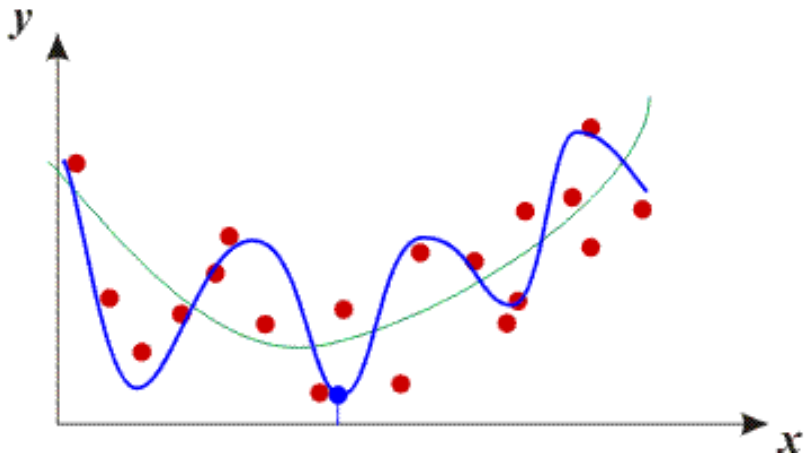
- Machine learning is a game of balance, with our objective being to **generalize** to all possible future data



Bias-Variance Trade-off



- Models with too few parameters are inaccurate because of a **large bias** (not enough flexibility).



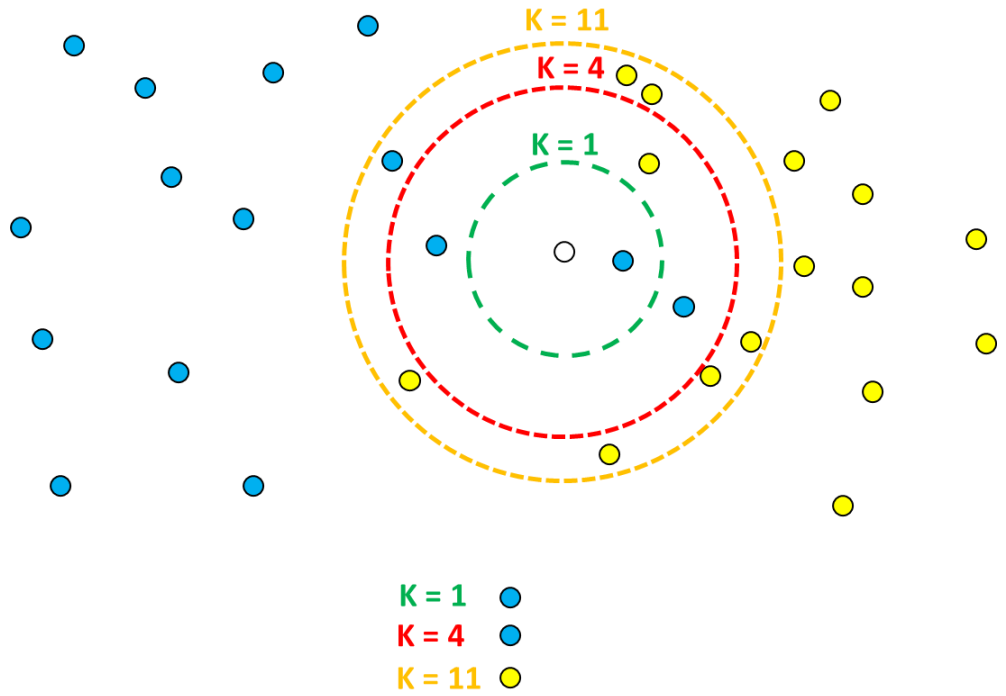
- Models with too many parameters are inaccurate because of a **large variance** (too much sensitivity to the sample).

Inductive Bias

- Let's avoid making assumptions about the model (polynomial order)
  - Assume for simplicity that D = $\{(x^{(i)},y^{(i)})\}$ is noise free
  - $x^{(i)}$'s in D only cover small subset of input space x
- Q: What's the best we can do?
  - If we've seen $x=x^{(i)}$ report $y=y^{(i)}$
  - If we have not seen $x= x^{(i)}$, can't say anything (no assumptions)
- This is called rote learning… boring, eh?
  - Key idea: you can't generalize to unseen data w/o assumptions!
- Thus, key to ML is generalization
  - **To generalize, ML algorithm must have some inductive bias**
  - Bias usually in the form of a restricted model (hypothesis) space
  - Important to understand restrictions (and whether appropriate)

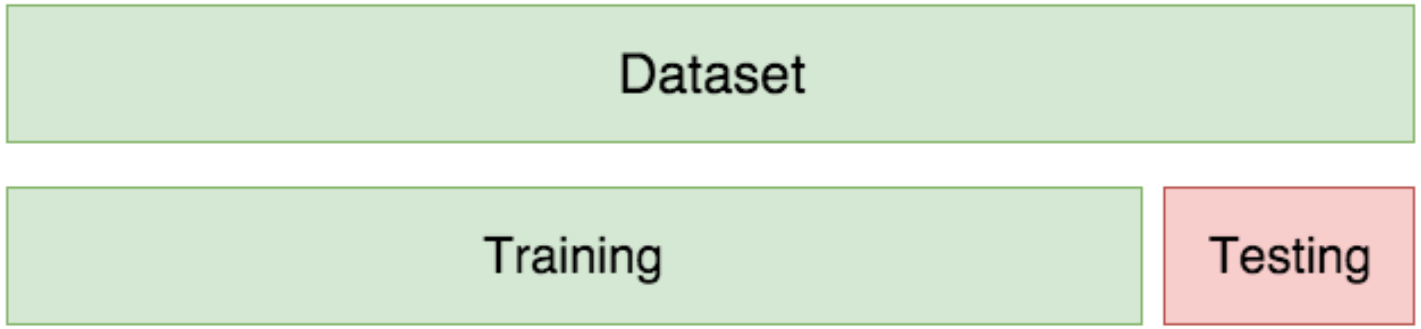
Inductive Bias

- Example: Nearest neighbors
  - We suppose that most of the cases in a small neighborhood in feature space belong to the same class. Given a case for which the class is unknown, we assume that it belongs to the same class as the majority in its immediate neighborhood.
  - This is the bias used in the k-nearest neighbors algorithm.
  - The assumption is that cases that are near each other tend to belong to the same class.



Training and Testing Data

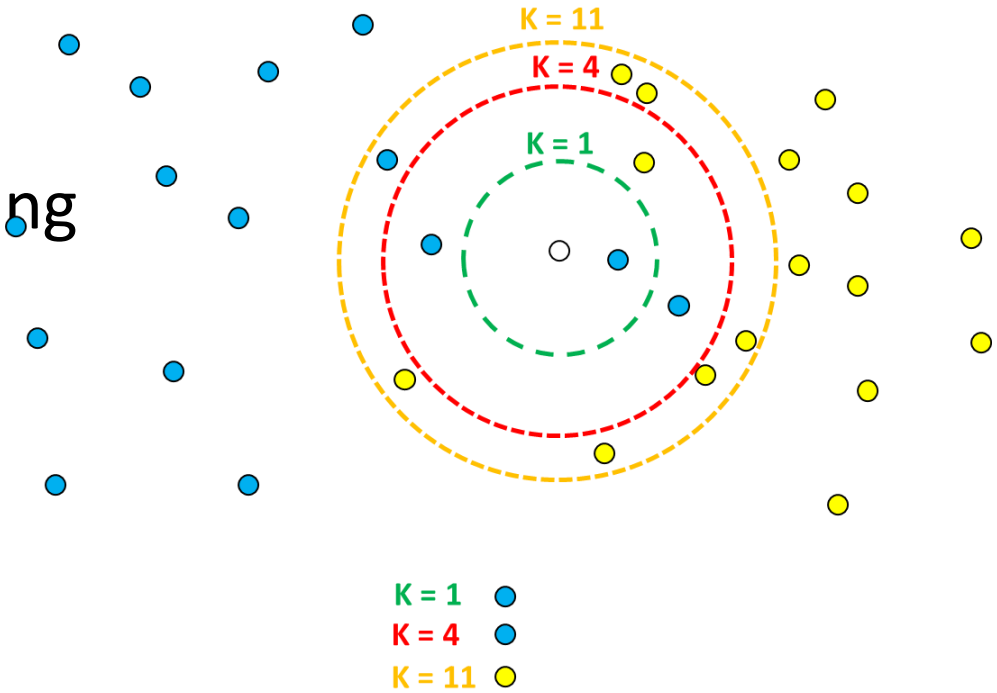
- Track generalization error by splitting data into training and testing
  - 80% training and 20% testing
- More data = better model
  - Would like to use all our data for training, however we need some way to evaluate our model



The problem with tracking test accuracy

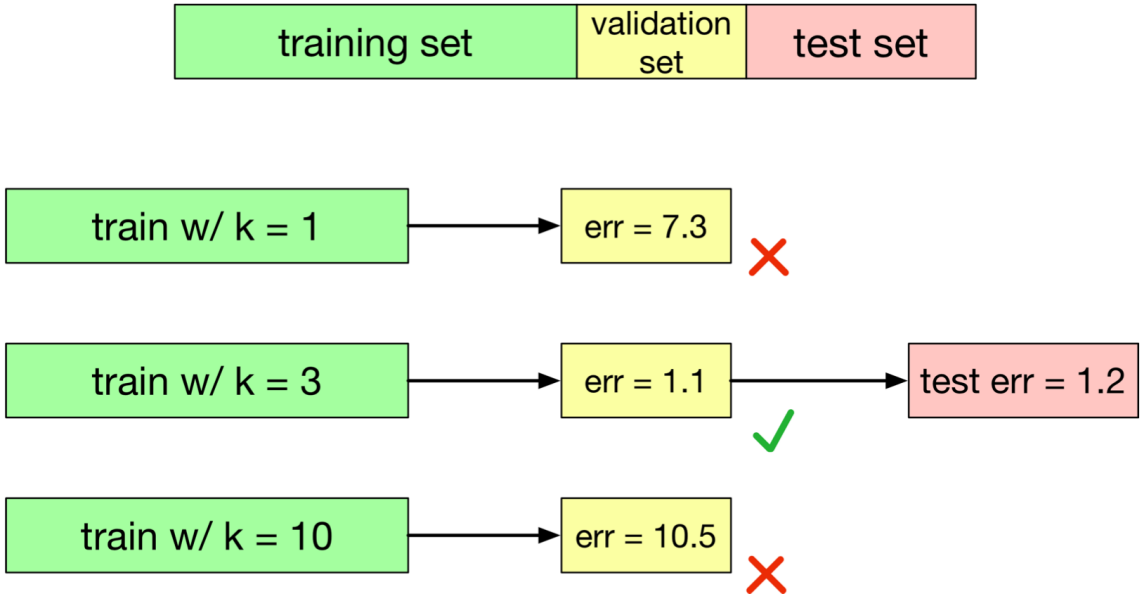
- What K should be?

- If we track test error/accuracy in our training curve, then:
  - We may make decisions about model architecture using the test accuracy and make the testing meaningless.
  - **The final test accuracy will not be a realistic estimate of how our model will perform on a new data set!**



Validation Set

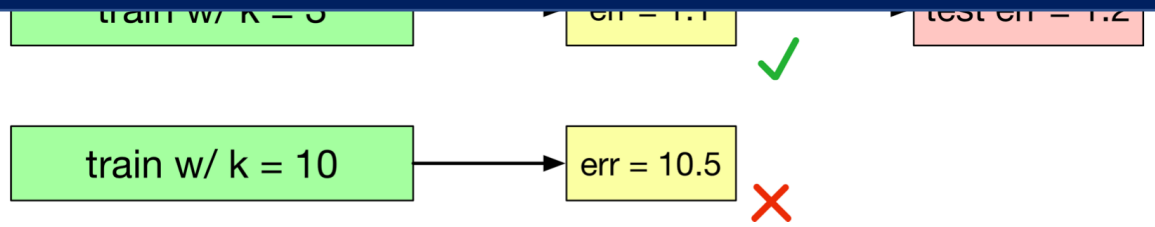
- We still want to track the loss/accuracy on a data set not used for training
- **Idea**: set aside a separate data set, called the **validation set**
  - Track validation accuracy in the training curve
  - Make decisions about model architecture using the validation set



Validation Set

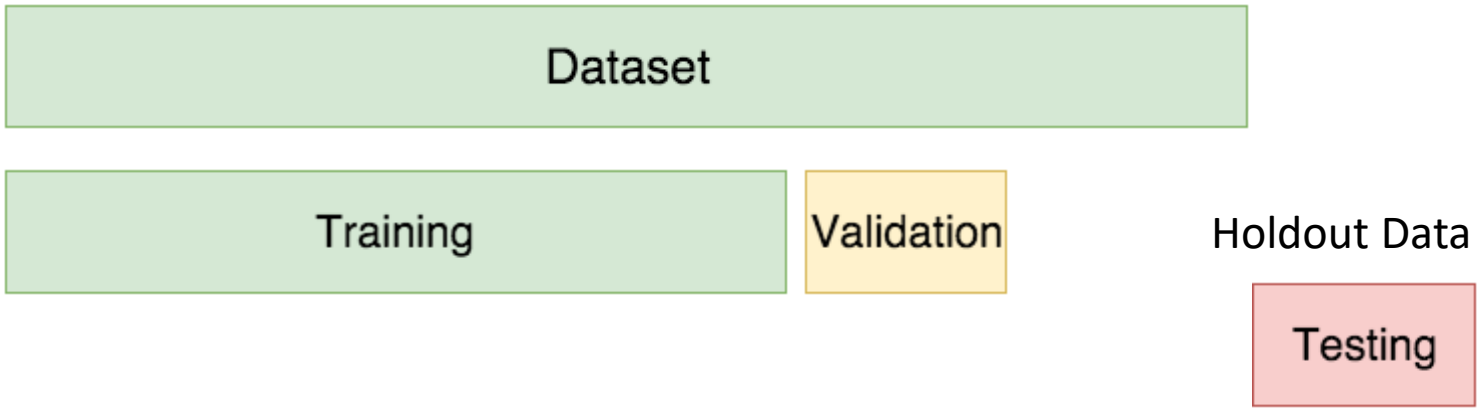
- We still want to track the loss/accuracy on a data set not used for training
- **Idea**: set aside a separate data set, called the **validation set**

K is a hyperparameter.
We tune hyperparameters using the validation set



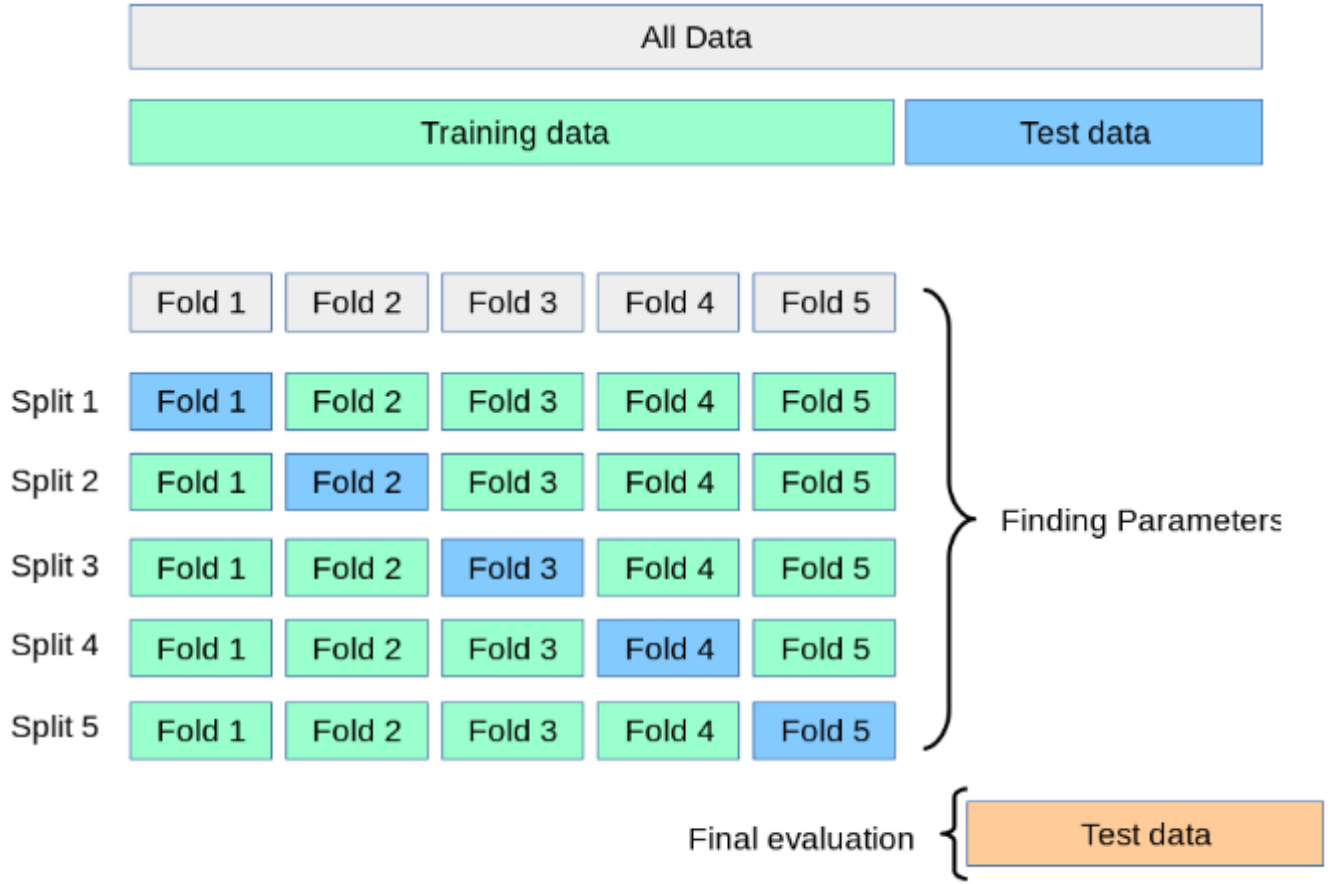
Validation and Holdout Data

- Training, Validation and Testing Data
  - Less data for your training model
- Ideally use the holdout data only once
  - **Requires a great deal of discipline** to not look at the holdout data



Cross-Validation

- Splitting training and validation data into several folds during training
- This is known as **k-fold Cross-Validation**
- Model parameters selected based on average achieved over *k* folds



Source: [scikit-learn]

Data Processing

- Q: You test your model on new data and you find it fails to predict certain samples. Why could be happening?

Training Data

Test Data

Data Augmentation

- For example, how can your algorithms (models) predict on rotations if it has never seen a rotated sample?

- Apply Data Augmentation!
  - translation,
  - scaling,
  - rotation,
  - reflection,
  - ...

Linear Algebra to the Rescue!

Source: https://morioh.com/p/928228425a08

More Data Processing

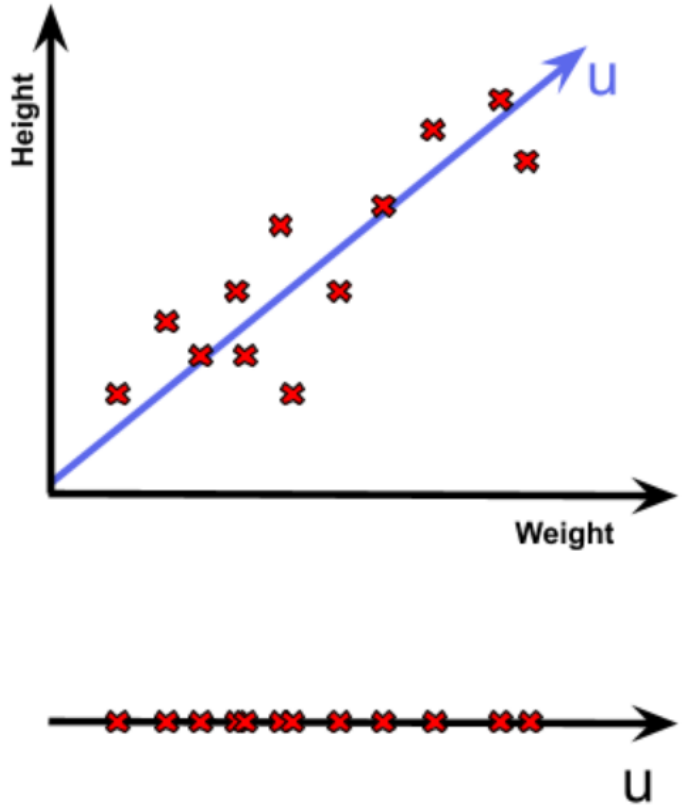
- Q: Large input feature size (short and wide data) is problematic? Why do you think that is?

- **Curse of dimensionality**!
  - As features grow you require more model capacity (complexity) to represent the data
  - Models of greater complexity require exponentially more training data

Dimensionality Reduction

Solution:

- Reduce the number of features using dimensionality reduction
  - Principal Component Analysis
  - more details provided in weeks 7 and 8

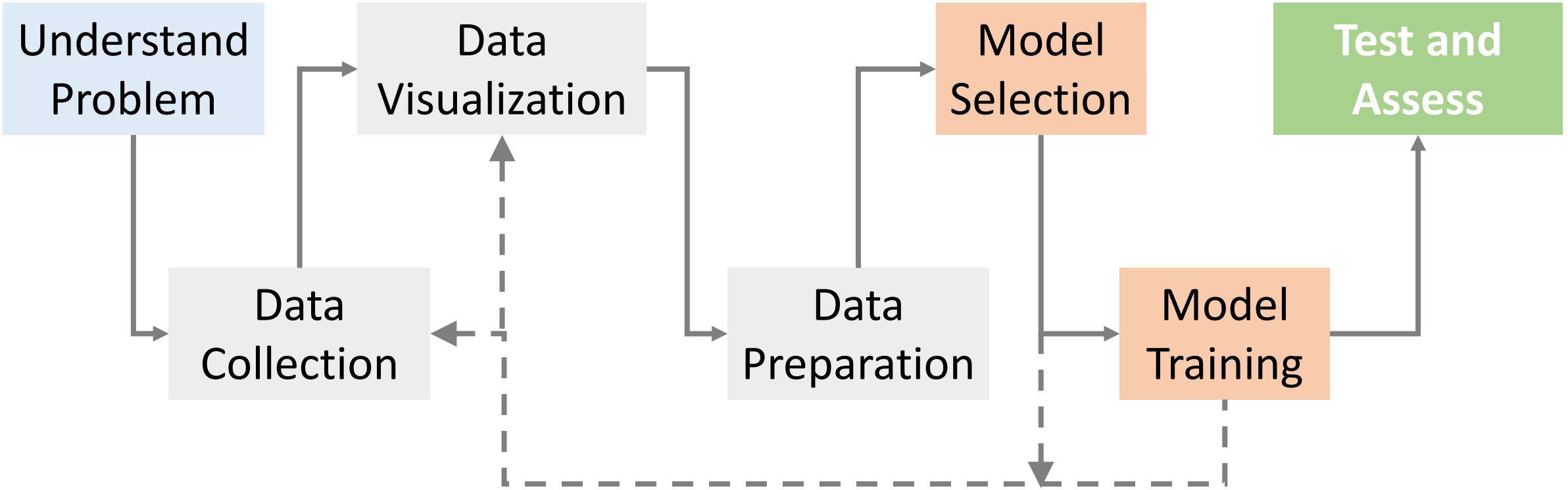


Source: Data Courses

Deep Learning

- Principle Component Analysis (PCA) is limited to **linear transformations**
- Deep Learning techniques can be used to learn and apply **nonlinear transformations** for dimensionality reduction
  - More detail on model-based machine learning techniques in weeks 9 – 11

Roadmap for the rest of APS1070



End-to-end machine learning is just one piece of the pie. The concepts we'll cover in this course have utility that goes far beyond machine learning.

Basic Python Check-up

Tutorials 0 and 1: Python Basics

- Data Types
  - Single: int, float, bool
  - Multiple: str, list, set, tuple, dict
  - index [], slice [::], mutability
- Conditionals
  - if, elif, else
- Functions
  - def, return, recursion, default vals
- Loops
  - for, while, range
  - list comprehension
- Operations
  - arithmetic: +,*,-,/,//,%, **
  - boolean: not, and, or
  - relational: ==, !=, >, <, >=, <=
- Display
  - print, end, sep
- Files
  - open, close, with
  - read, write
  - CSV
- Object-Oriented Programming (OOP)
  - class, methods, attributes
  - __init__, __str__, polymorphism

Other resources for Python

- Toronto-based and internationally popular resources:
  - Kaggle 5-hour course on Python (by Colin Morris)
    https://www.kaggle.com/learn/python
  - U of T MOOC Learn to Program: The Fundamentals
    https://www.coursera.org/learn/learn-to-program
  - U of T MOOC Learn to Program: Crafting Quality Code
    https://www.coursera.org/learn/program-code
  - U of T Coders (student-run group)
    https://uoftcoders.github.io/

- Google is your (BEST) friend?
- APS1070 Piazza Discussion Board

Scientific Computing Tools for Python

- Scientific computing in Python builds upon a small core of packages:
  - NumPy, the fundamental package for numerical computation. It defines the numerical array and matrix types and basic operations on them.
  - The SciPy library, a collection of numerical algorithms and domain-specific toolboxes, including signal processing, optimization, statistics and much more.
  - Matplotlib, a mature and popular plotting package, that provides publication-quality 2D plotting as well as rudimentary 3D plotting
- Data and computation:
  - pandas, providing high-performance, easy to use data structures.
  - scikit-learn is a collection of algorithms and tools for machine learning.

Source: https://www.scipy.org/about.html

NumPy

- Let’s start with NumPy. Among other things, NumPy contains:
  - A powerful N-dimensional array object.
  - Sophisticated (broadcasting/universal) functions.
  - Tools for integrating C/C++ and Fortran code.
  - Useful linear algebra, Fourier transform, and random number capabilities.
  - Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data.
  - Many other python libraries are built on NumPy
  - Provides vectorization of mathematical operations on arrays and matrices which significantly improves the performance

NumPy

- The key to NumPy is the ndarray object, an n-dimensional array of **homogeneous data types**, with many operations being performed in compiled code for performance.
- There are several important differences between NumPy arrays and the standard Python sequences:
  - NumPy arrays have a fixed size. Modifying the size means creating a new array.
  - NumPy arrays must be of the same data type, but this can include Python objects.
  - More efficient mathematical operations than built-in sequence types

NumPy

- To begin, NumPy supports a wider variety of data types than are built-in to the Python language by default. They are defined by the numpy.dtype class and include:
  - intc (same as a C integer) and intp (used for indexing)
  - int8, int16, int32, int64
  - uint8, uint16, uint32, uint64
  - float16, float32, float64
  - complex64, complex128
  - bool_, int_, float_, complex_ are shorthand for defaults.

NumPy

- There are a couple of mechanisms for creating arrays in NumPy:
  - Conversion from other Python structures (e.g., lists, tuples).
  - Built-in NumPy array creation (e.g., arrange, ones, zeros, etc.).
  - Reading arrays from disk, either from standard or custom formats (e.g. reading in from a CSV file).
  - and others …

NumPy

- There are a couple of mechanisms for creating arrays in NumPy:
  - Conversion from other Python structures (e.g., lists, tuples).
  - Built-in NumPy array creation (e.g., arrange, ones, zeros, etc.).
  - Reading arrays from disk, either from standard or custom formats (e.g. reading in from a CSV file).
  - and others …

- In general, any numerical data that is stored in an array-like container can be converted to an ndarray through use of the array() function. The most obvious examples are sequence types like lists and tuples.

SciPy

- Collection of algorithms for linear algebra, differential equations, numerical integration, optimization, statistics and much more
- Part of SciPy Stack
- Built on NumPy

- With SciPy an interactive Python session becomes a data-processing and system-prototyping environment rivaling systems such as MATLAB, IDL, Octave, R-Lab, and SciLab.

SciPy

- SciPy's functionality is implemented in a number of specific sub-modules. These include:
  - Special mathematical functions (scipy.special) -- airy, elliptic, bessel, etc.
  - Integration (scipy.integrate)
  - Optimization (scipy.optimize)
  - Interpolation (scipy.interpolate)
  - Fourier Transforms (scipy.fftpack)
  - Signal Processing (scipy.signal)
  - Linear Algebra (scipy.linalg)
  - Statistics (scipy.stats)
  - Multidimensional image processing (scipy.ndimage)
  - Data IO (scipy.io)
  - and more!

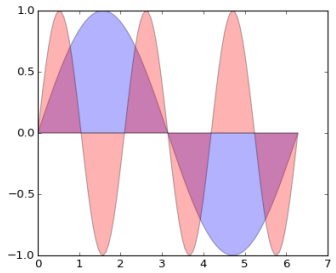
Pandas

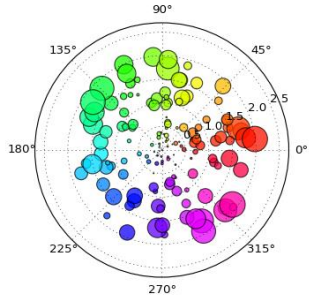
- Adds data structures and tools designed to work with table-like data (similar to Series and Data Frames in R)
- Provides tools for data manipulation: reshaping, merging, sorting, slicing, aggregation etc.
  - **Aggregation** - computing a **summary statistic for groups**
  - min, max, count, sum, prod, mean, median, mode, mad, std, var
- Allows for handling missing data

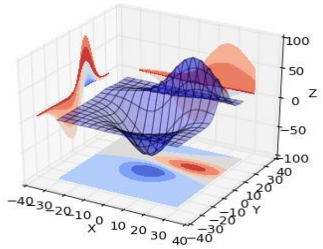
Source: http://pandas.pydata.org/

Matplotlib

- Matplotlib is an incredibly powerful (and beautiful!) 2-D plotting library. It's easy to use and provides a huge number of examples for tackling unique problems.
- Similar to MATLAB



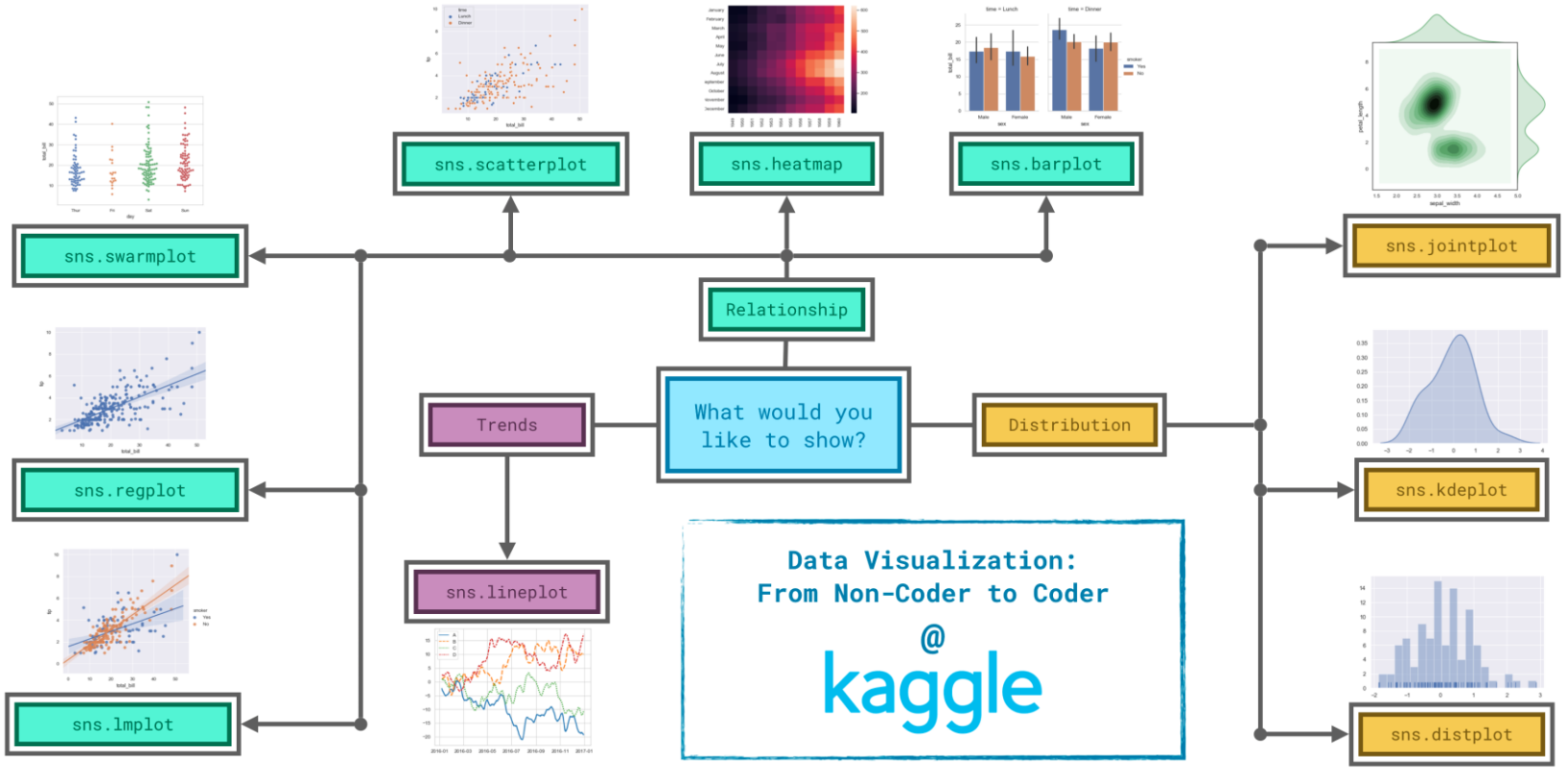




Seaborn

- Seaborn has more convenient commands and options
- Kaggle 4-hour course on information visualization (by Alexis Cook and Dan Becker)

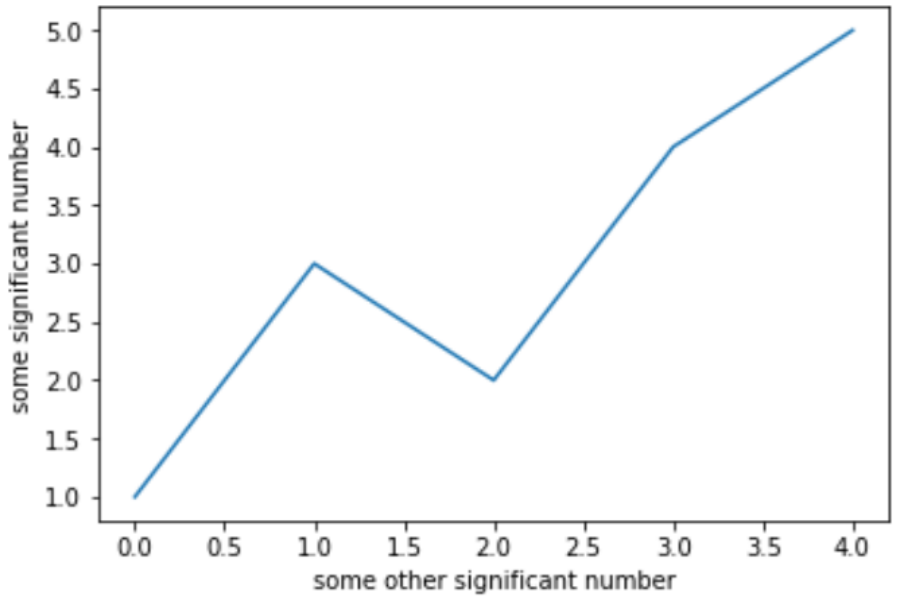
https://www.kaggle.com/learn/data-visualization



pyplot

- At the center of most matplotlib scripts is pyplot.
- The pyplot module is stateful and tracks changes to a figure. All pyplot functions revolve around creating or manipulating the state of a figure.

```
import numpy as np
import matplotlib.pyplot as plt
plt.plot([1, 3, 2, 4, 5])
plt.ylabel('some significant number')
plt.xlabel('some other significant number')
plt.show()
```



pyplot

- The plot function can actually take any number of arguments.
- The format string argument associated with a pair of sequence objects indicates the color and line type of the plot (e.g. ‘bs’ indicates blue squares and ‘ro’ indicates red circles).
- Generally speaking, the x_values and y_values will be numpy arrays and if not, they will be converted to numpy arrays internally.
- Line properties can be set via keyword arguments to the plot function. Examples include label, linewidth, animated, color, etc…

Jupyter Notebook

- All of these libraries come preinstalled on Google Colab
- Google Colab uses a Jupyter notebook environment that runs in the cloud and requires no setup to use
- Runs in Python 3
- Includes all the commonly used machine learning (data science) libraries
  - i.e. NumPy, SciPy, Matplotlib, Pandas, PyTorch, Tensorflow, etc.

- Alternatively, can use Jupyter notebook on your computer

Let's take a look at week 3 Jupyter Notebook

Part 2
Python Libraries and Titanic

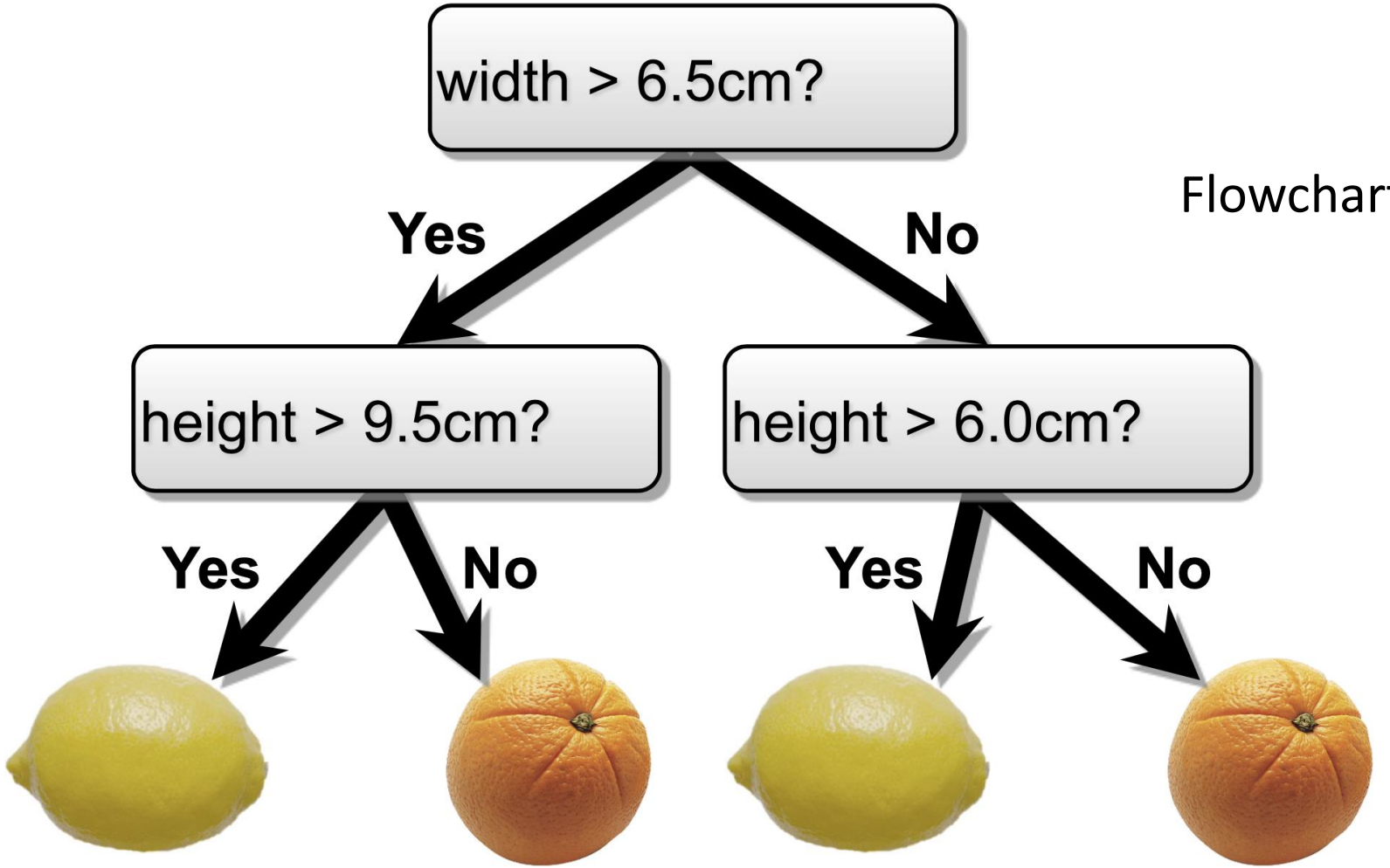
Part 3
Decision Trees

Decision Trees

- A rule-based supervised learning algorithm
- Powerful algorithm capable of fitting complex datasets.
- Can be applied to classification (discrete) and regression (continuous) tasks.
- Highly interpretable!

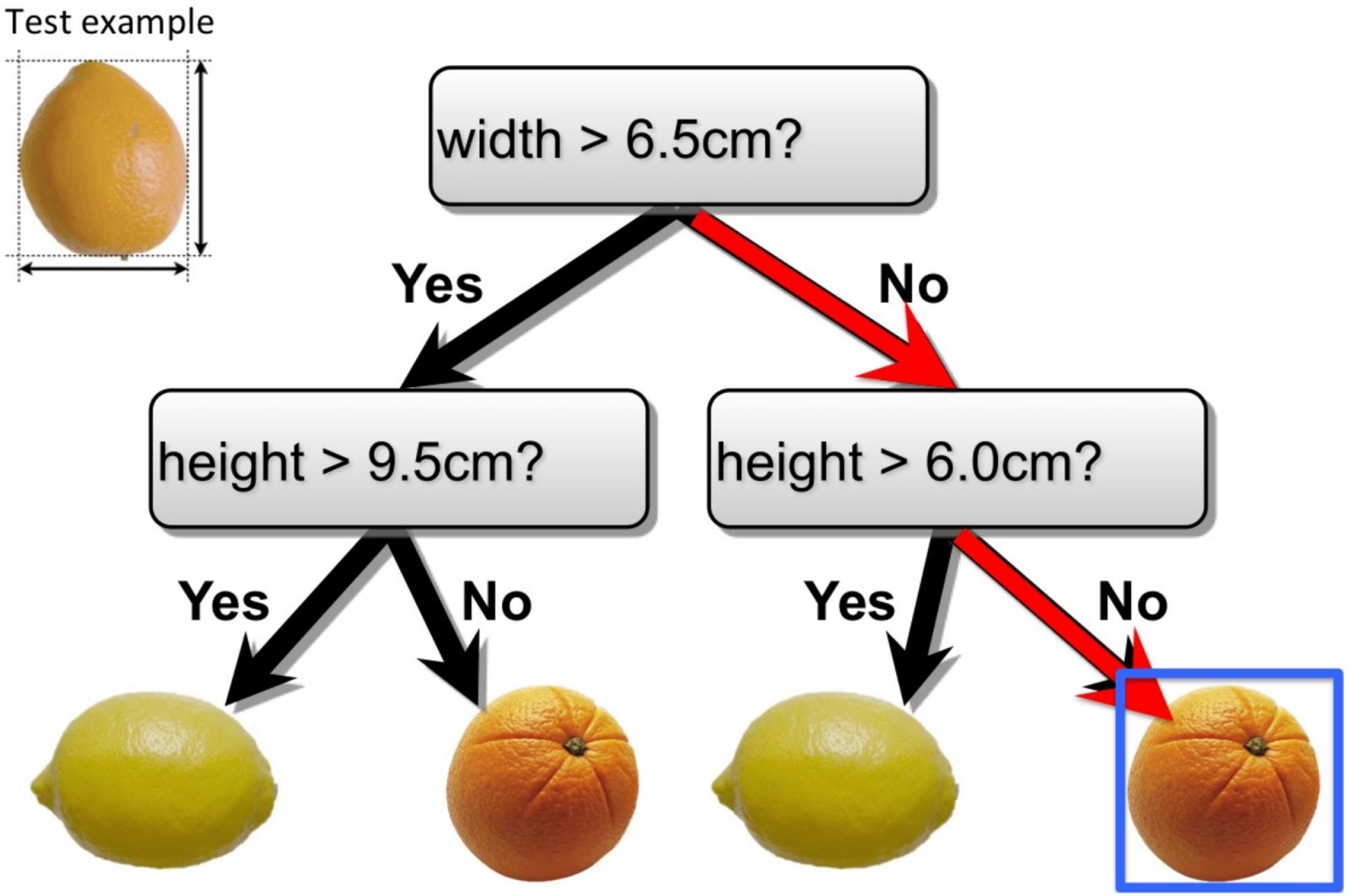
- A fundamental component of Random Forests which are one of the most used Machine Learning algorithms today

Lemon Vs. Orange!



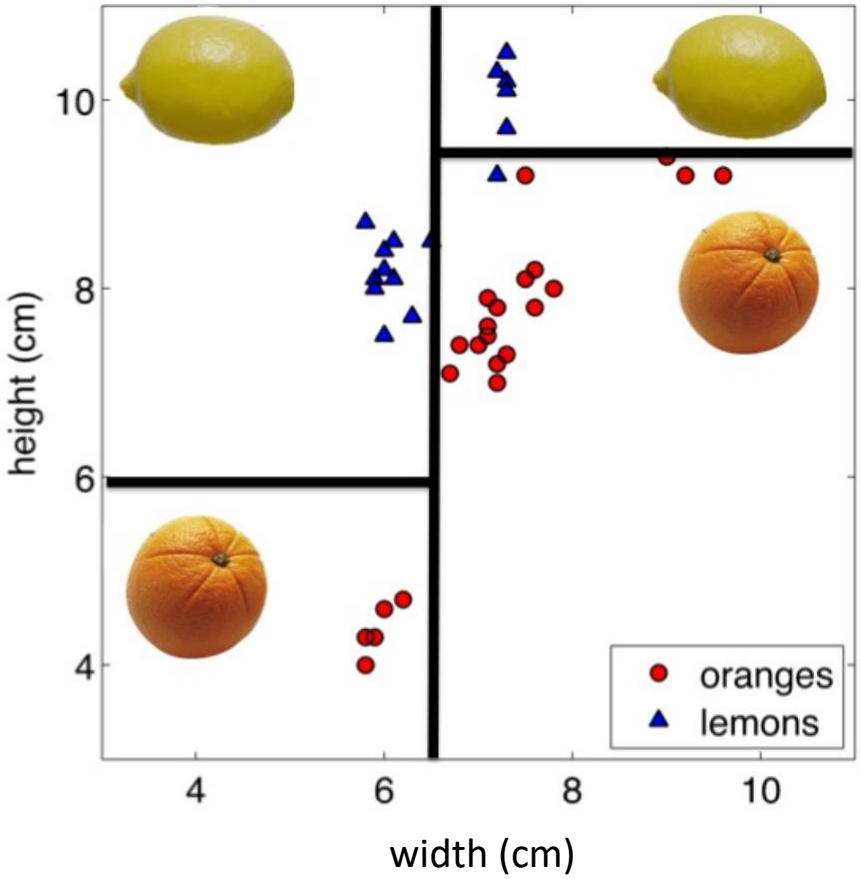
Flowchart-like structure!

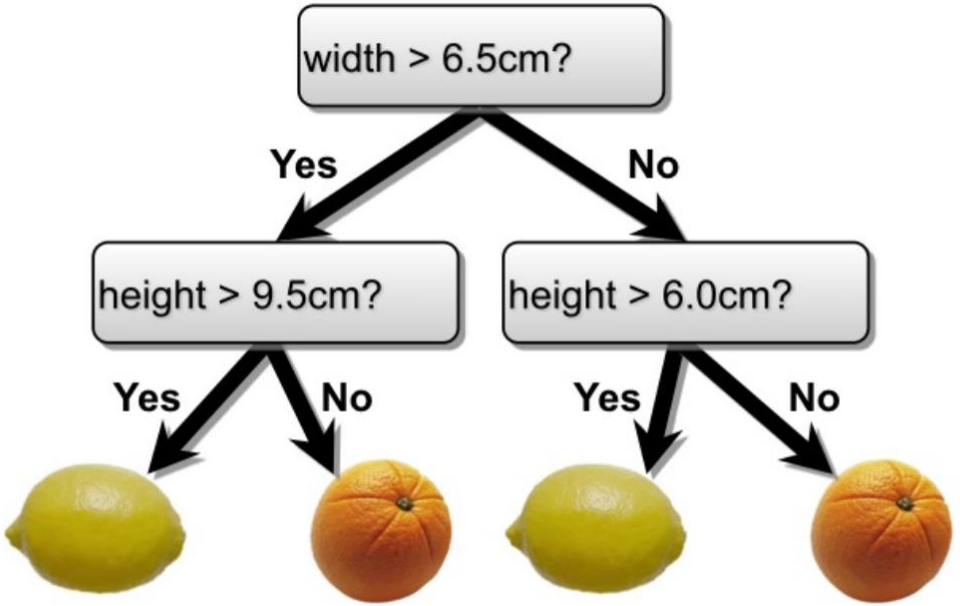
Test example



Constructing a Decision Tree

- Decision trees make predictions by recursively splitting on different attributes according to a tree structure





What if the attributes are discrete?

Example	Input Attributes										Goal
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
$\mathbf{x}_1$	*Yes*	*No*	*No*	*Yes*	*Some*	*$$$*	*No*	*Yes*	*French*	*0–10*	$y_1 =$ *Yes*
$\mathbf{x}_2$	*Yes*	*No*	*No*	*Yes*	*Full*	*$*	*No*	*No*	*Thai*	*30–60*	$y_2 =$ *No*
$\mathbf{x}_3$	*No*	*Yes*	*No*	*No*	*Some*	*$*	*No*	*No*	*Burger*	*0–10*	$y_3 =$ *Yes*
$\mathbf{x}_4$	*Yes*	*No*	*Yes*	*Yes*	*Full*	*$*	*Yes*	*No*	*Thai*	*10–30*	$y_4 =$ *Yes*
$\mathbf{x}_5$	*Yes*	*No*	*Yes*	*No*	*Full*	*$$$*	*No*	*Yes*	*French*	*>60*	$y_5 =$ *No*
$\mathbf{x}_6$	*No*	*Yes*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Italian*	*0–10*	$y_6 =$ *Yes*
$\mathbf{x}_7$	*No*	*Yes*	*No*	*No*	*None*	*$*	*Yes*	*No*	*Burger*	*0–10*	$y_7 =$ *No*
$\mathbf{x}_8$	*No*	*No*	*No*	*Yes*							
$\mathbf{x}_9$	*No*	*Yes*	*Yes*	*No*							
$\mathbf{x}_{10}$	*Yes*	*Yes*	*Yes*	*Yes*							
$\mathbf{x}_{11}$	*No*	*No*	*No*	*No*							
$\mathbf{x}_{12}$	*Yes*	*Yes*	*Yes*	*Yes*							

1. Alternate: whether there is a suitable alternative restaurant nearby.
2. Bar: whether the restaurant has a comfortable bar area to wait in.
3. Fri/Sat: true on Fridays and Saturdays.
4. Hungry: whether we are hungry.
5. Patrons: how many people are in the restaurant (values are None, Some, and Full).
6. Price: the restaurant's price range ($, $$, $$$).
7. Raining: whether it is raining outside.
8. Reservation: whether we made a reservation.
9. Type: the kind of restaurant (French, Italian, Thai or Burger).
10. WaitEstimate: the wait estimated by the host (0-10 minutes, 10-30, 30-60, >60).

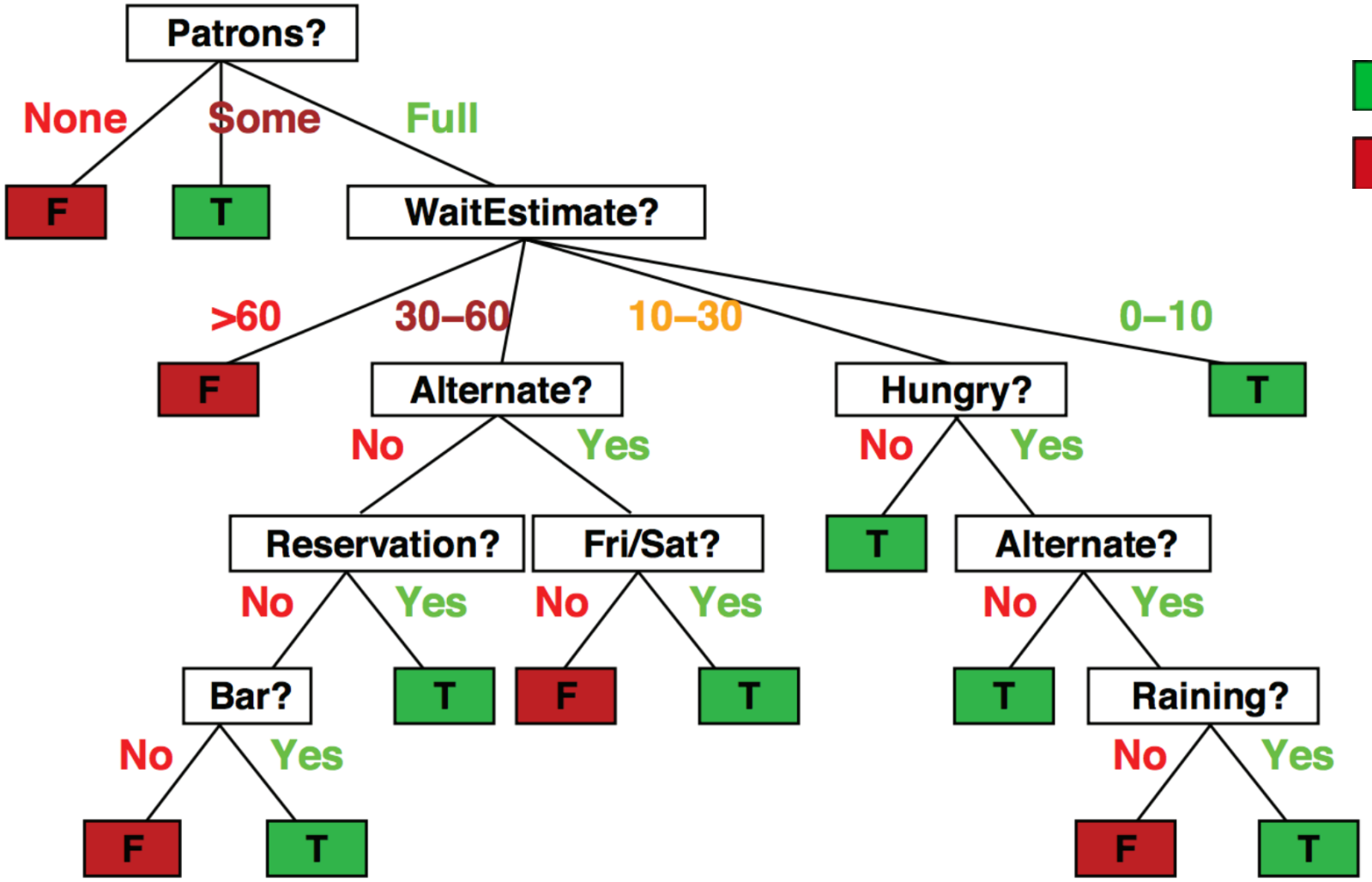
What if the attributes are discrete?

Example	Input Attributes										Goal
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
$\mathbf{x}_{10}$	*Yes*	*Yes*	*Yes*	*Yes*	*Full*						
$\mathbf{x}_{11}$	*No*	*No*	*No*	*No*	*None*						
$\mathbf{x}_{12}$	*Yes*	*Yes*	*Yes*	*Yes*	*Full*						

Attributes: Features (inputs)!
Discrete or Continuous

3.	Fri/Sat: true on Fridays and Saturdays.
4.	Hungry: whether we are hungry.
5.	Patrons: how many people are in the restaurant (values are None, Some, and Full).
6.	Price: the restaurant's price range ($, $$, $$$).
7.	Raining: whether it is raining outside.
8.	Reservation: whether we made a reservation.
9.	Type: the kind of restaurant (French, Italian, Thai or Burger).
10.	WaitEstimate: the wait estimated by the host (0-10 minutes, 10-30, 30-60, >60).

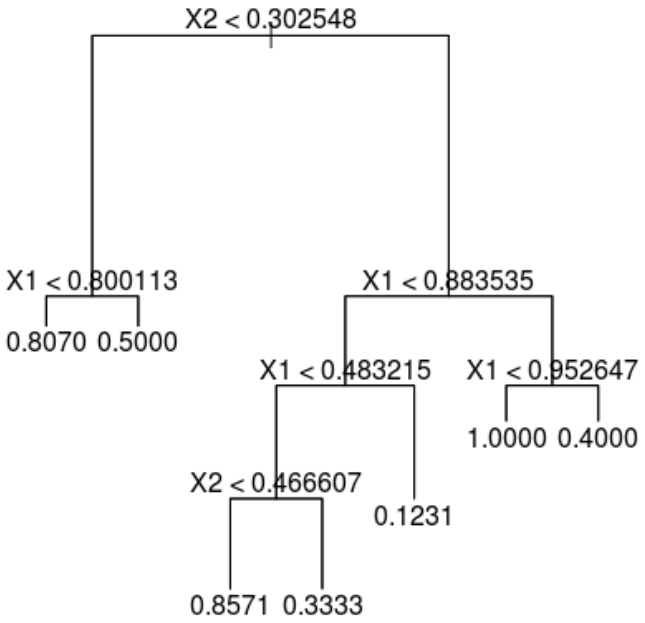
Output is Discrete

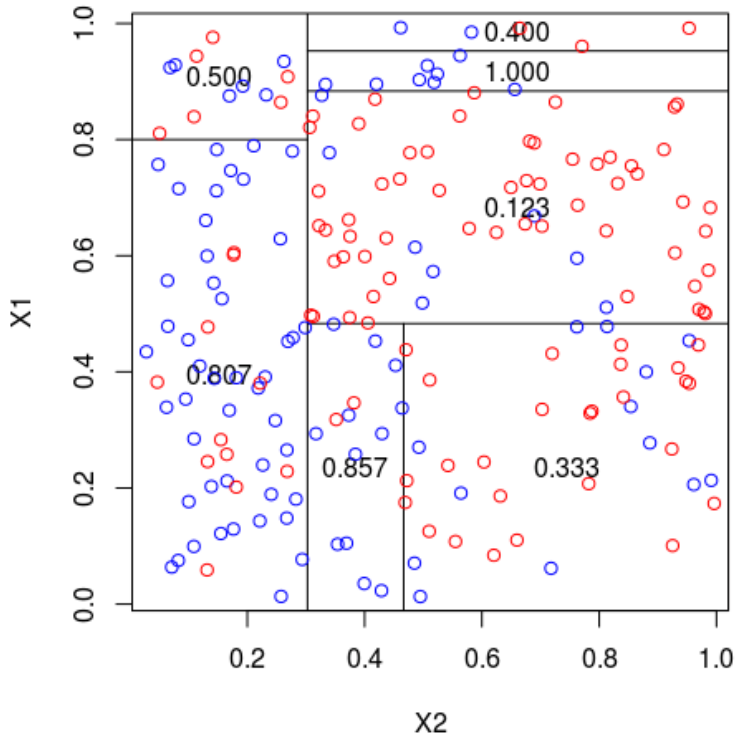


T — Wait for table

F — Go somewhere else

Output is Continuous (Regression)





- Instead of predicting a class at each leaf node, predict a **value based on the average** of all instances at the leaf node.

Source: GDCoder

Summary: Discrete vs Continuous Output

- **Classification Tree**:
  - discrete output
  - output node (leaf) typically set to the **most common value**

- **Regression Tree**:
  - continuous output
  - output node (leaf) value typically set to the **mean value** in data

Generalization

- Decision trees **can fit any function arbitrarily closely**
- Could potentially create a leaf for each example in the training dataset
- Not likely to generalize to test data!

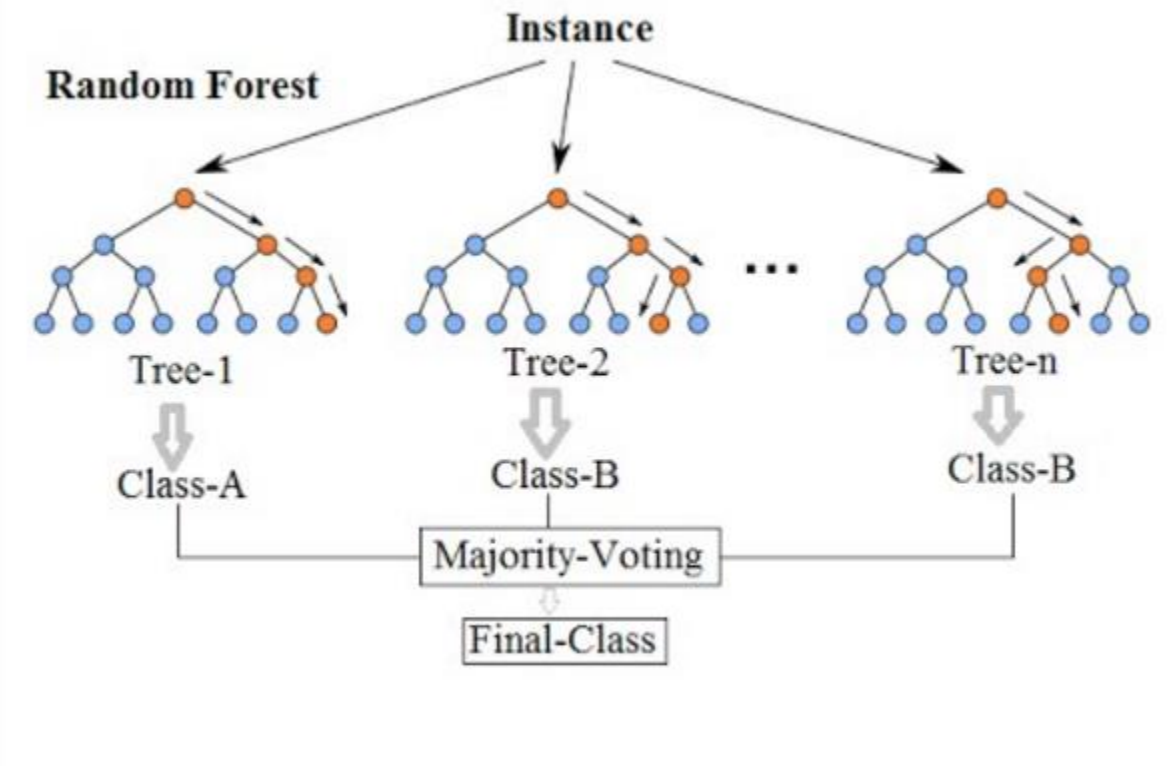
- Need some way to prune the tree!

Managing Overfitting

- Add parameters to reduce potential for overfitting
- Parameters include:
  - depth of tree
  - minimum number of samples

Random Forests

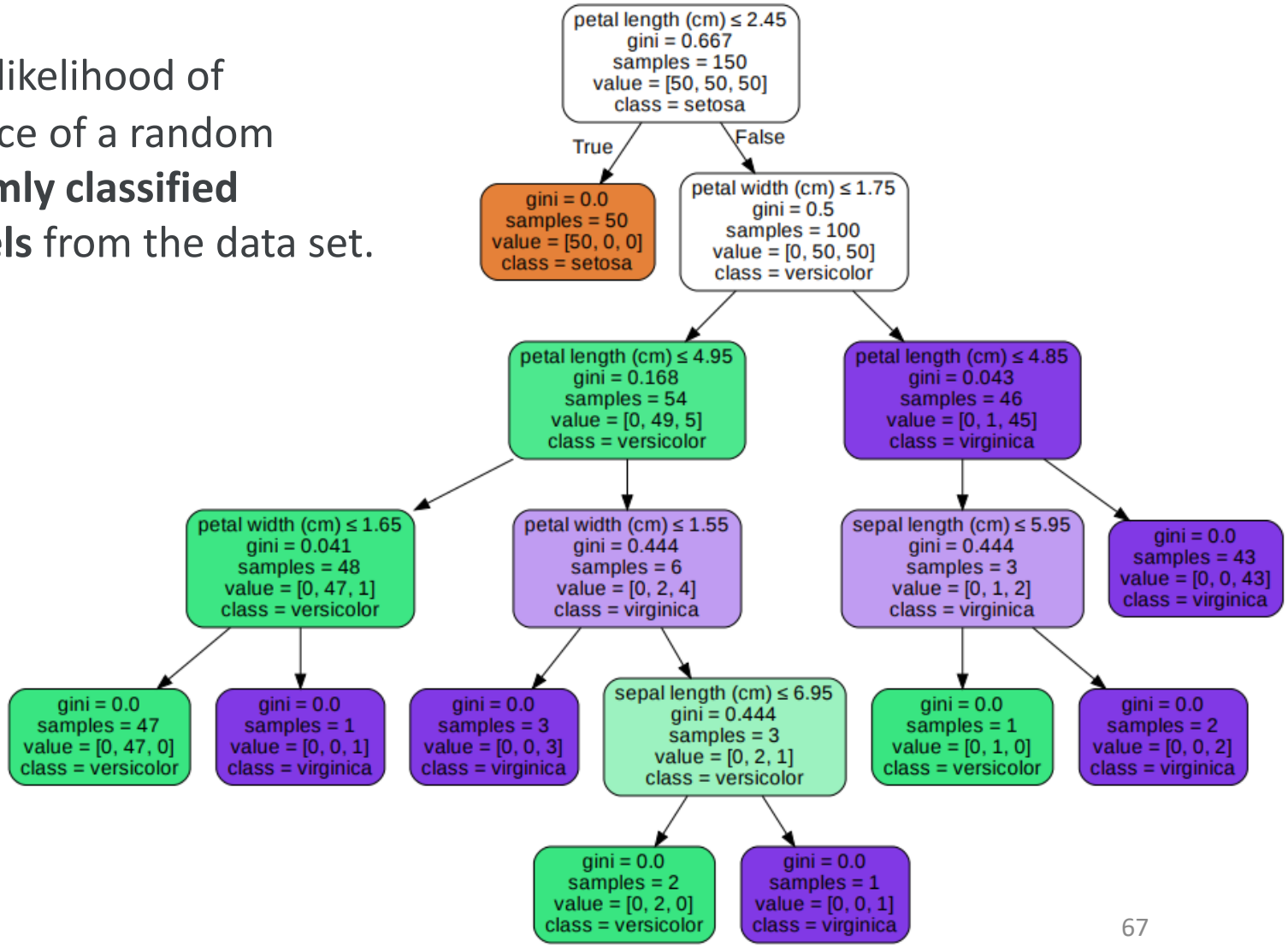
- One of the most popular variants of decision trees
- Addresses overfitting by training multiple trees on subsampling of features among other things
- Majority vote of all the trees is used to make the final output



Source: Venkata Jagannath

Decision Trees are interpretable models

Gini Impurity is a measurement of the likelihood of an **incorrect classification** of a new instance of a random variable, if that new instance were **randomly classified according to the distribution of class labels** from the data set.



Comparison to k-NN

- There are many advantages of Decision Trees over k-Nearest Neighbours:
  - Good with discrete attributes
  - Robust to scale of inputs (does not require normalization)
  - Easily handle missing values
  - Good at handling lots of attributes, especially when only a few are important
  - Fast test time
  - More interpretable
  - Decision trees not good at handling rotations in data
  - Decision trees have limited predictive performance (more advanced tree-based models)

Next Time

- Week 3 Q&A Support Session on Thursday and Friday
  - Help with Python and Project 1
- Reading assignment 3 is due on Monday
- Project 1 is due on 4 February
- Week 4 Lecture – Uncertainty and Performance
  - K-Means Clustering
  - Probability Theory
  - Summary Statistics
  - Multivariate Gaussians
  - Performance Metrics

Decision Trees Code Example (Google Colab)