

# APS1070

Foundations of Data Analytics and  
Machine Learning

Fall 2020

## **Week 2:**

- *K-Nearest Neighbor Classification*
- *Cross-validation*
- *Python, NumPy, SciPy, Pandas*
- *Tutorial 1 + Project 1*

Jason Riordon, PhD



# Understanding the risks of international, online learning

- If you are a citizen of another country, and/or accessing your courses at the University of Toronto from a jurisdiction outside of Canada, you remain subject to the laws of the country in which you are residing, or any country of which you have citizenship.
- If you use the VPN (Virtual Private Network) service that UofT has set up in partnership with Alibaba, Alibaba will have access to your credentials for accessing University of Toronto resources creating an additional inherent risk of surveillance.
- I've posted a document on *Understanding the risks of international, online learning* to Quercus (Course Resources Module)

# Survey

- How would you rate your programming abilities? [1=Beginner, 5=Expert] **2.46**
- How would you rate your knowledge of Python and Machine Learning libraries? [1=Beginner, 5=Expert] **2**
- How would you rate your knowledge of Machine Learning concepts? [1=Beginner, 5=Expert] **1.83**

# Top-3 Learning Outcomes

By the end of the course, students will be able to:

1. describe and contrast machine learning models, concepts and performance metrics;
2. perform fundamental linear algebra operations, and recognize their role in machine learning algorithms;
3. apply machine learning models and statistical methods to datasets using Python and associated libraries.

# Posted more old assessments

*Quercus – Modules – Old Evaluations*

# Class start time

*“All U of T classes (and as a result, many U of T club meetings and events) start 10 minutes after the hour. This means you’ll likely spend the first 10 minutes of every class wondering where your prof is or panicking that you’re in the wrong classroom. But don’t worry – soon enough you’ll be in sync with the rest of the U of T student body (and chronically 10 minutes late to everything not U of T-related).”*

# Slide Attribution

These slides contain materials from various sources. Special thanks to the following authors:

- Scott Sanner
- Caitlin Carnahan
- Katia Koleinik
- Ali Hadi Zadeh
- D. Hoiem

# **Nearest-Neighbor Classifier (Supervised Learning)**



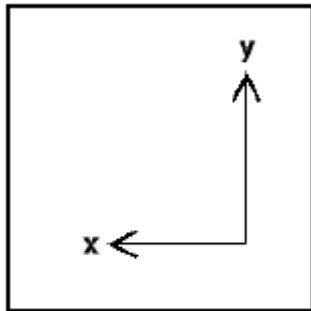


# K-Nearest neighbor classifier

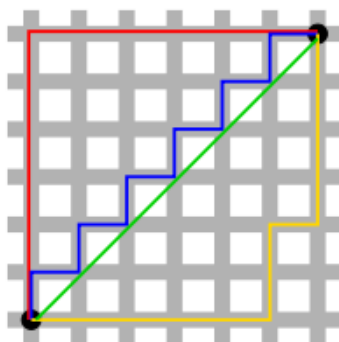
- Output is a class (here, blue or yellow)
- Instance-based learning, or lazy learning: computation only happens once called
- Flexible approach – no assumptions on data distribution

# How else can “distance” be measured?

$p = 1$ , Manhattan Distance

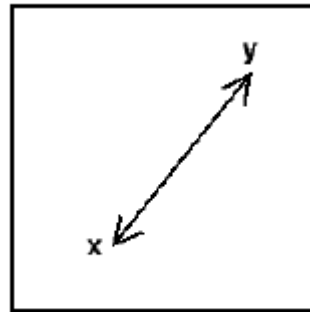


**Manhattan**



Source — Taxicab geometry Wikipedia

$p = 2$ , Euclidean Distance



**Euclidean**

$$d(x, y) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2}$$

Dimension of x or y

$$\left( \sum_{i=1}^n |x_i - y_i|^p \right)^{1/p}$$

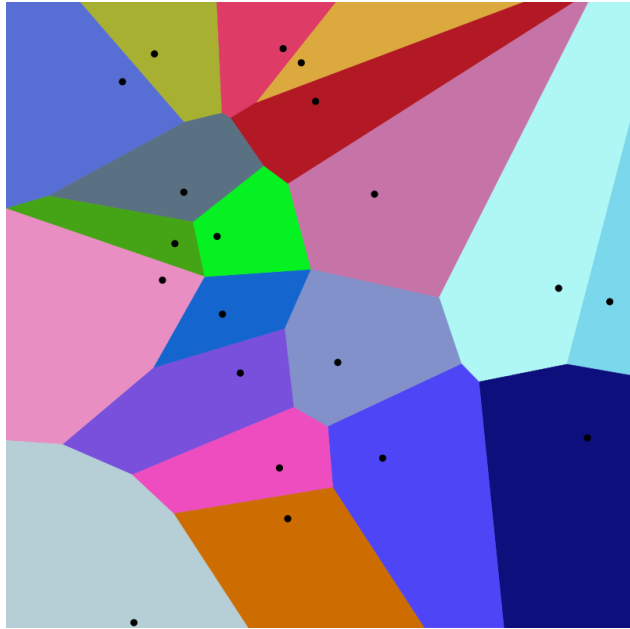
<https://towardsdatascience.com/importance-of-distance-metrics-in-machine-learning-modelling-e51395ffe60d>

<https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.DistanceMetric.html>

# Special case: $k=1$

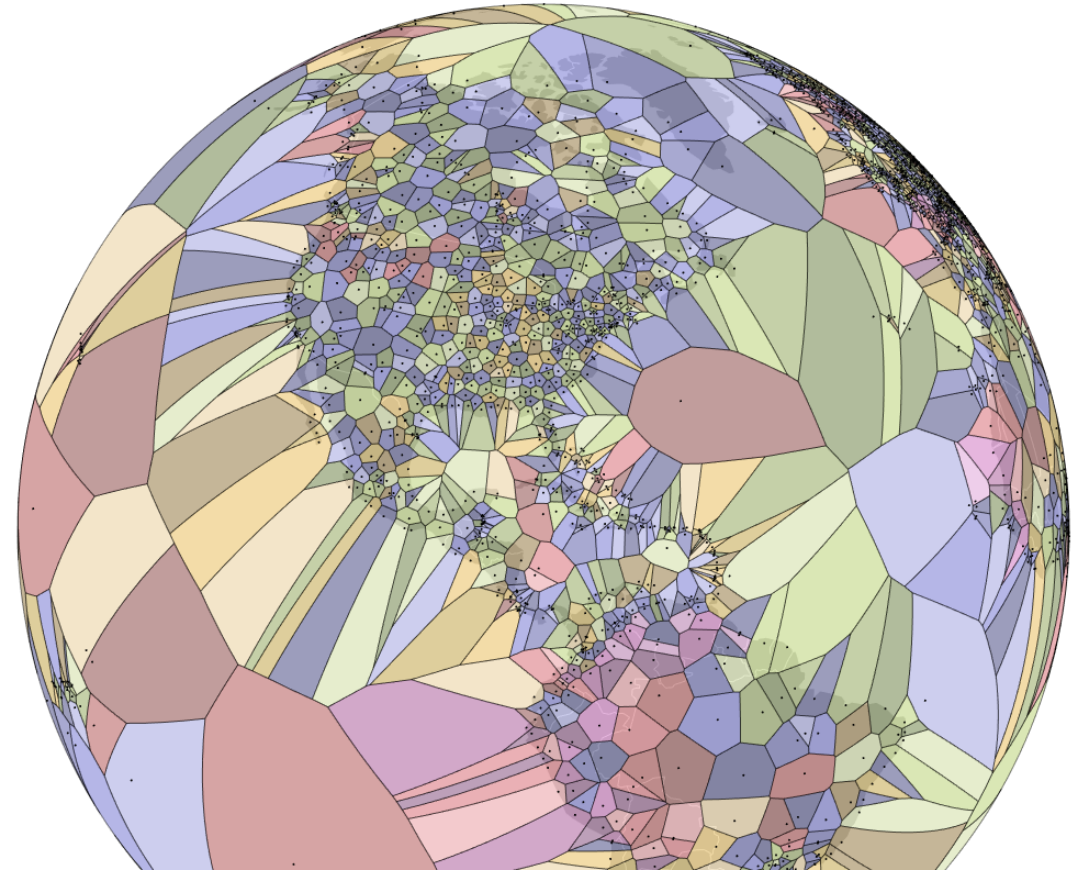
Voronoi diagrams:

Euclidian distance



[https://en.wikipedia.org/wiki/Voronoi\\_diagram](https://en.wikipedia.org/wiki/Voronoi_diagram)

World Airports Voronoi

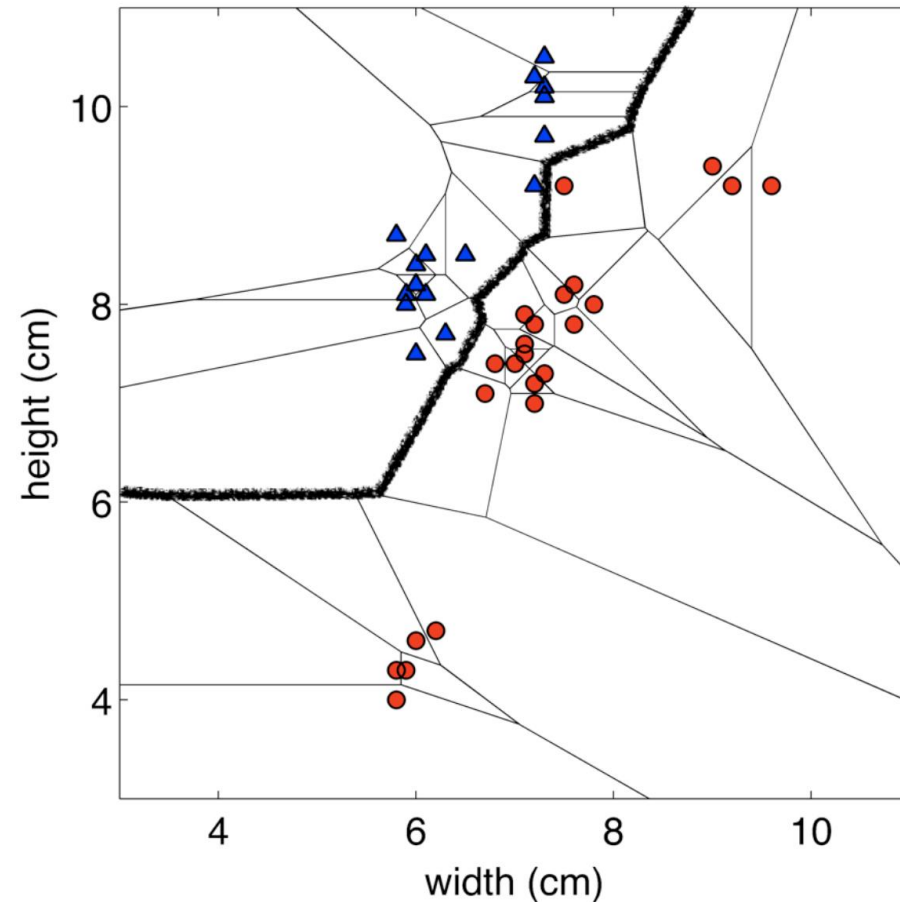


Spherical Voronoi – image from Jason Davies

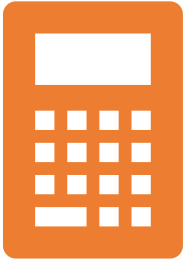
<https://www.jasondavies.com/maps/voronoi/airports/>

# Decision boundary

The boundary between regions of input space assigned to different categories.



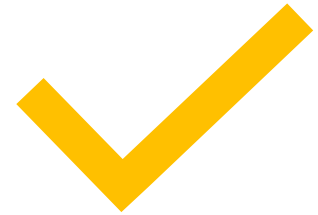
## For a single nearest neighbor



Calculate distance



Obtain the nearest  
neighbour



Determine labels

For a single nearest neighbor

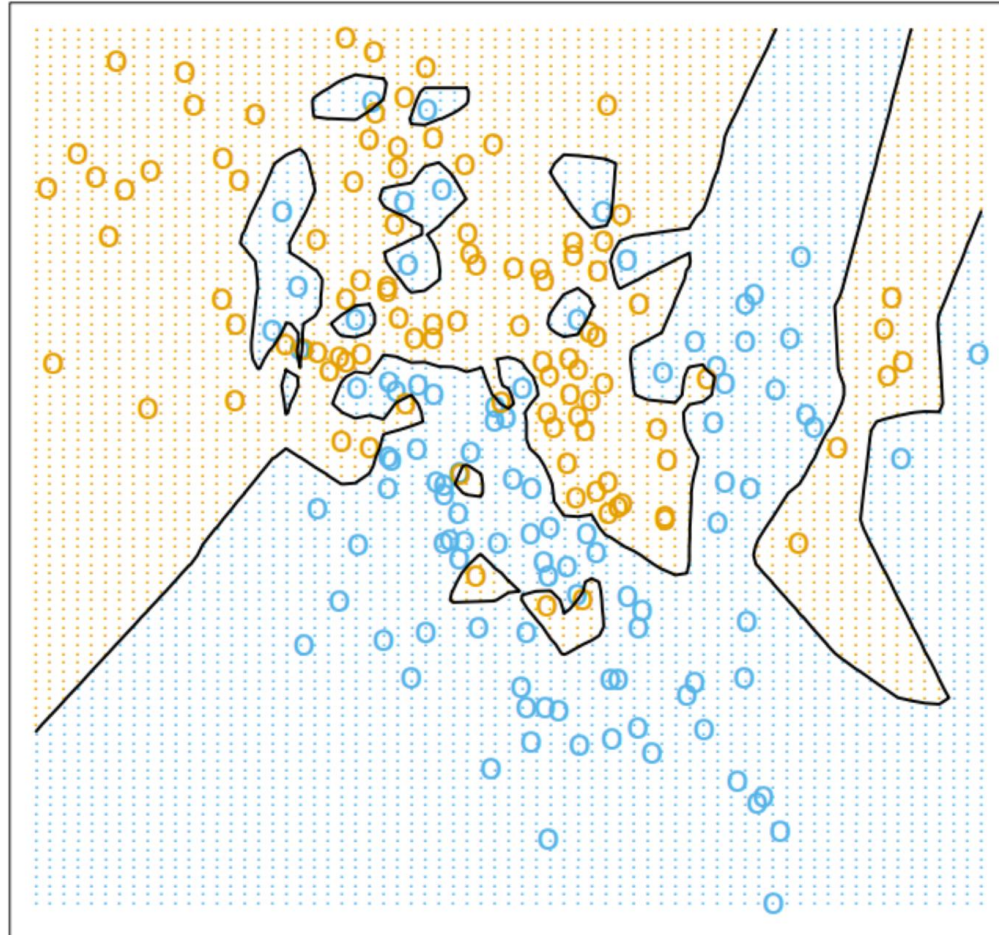
## Algorithm:

1. Find example  $(\mathbf{x}^*, t^*)$  (from the stored training set) closest to  $\mathbf{x}$ .  
That is:

$$\mathbf{x}^* = \operatorname{argmin}_{\mathbf{x}^{(i)} \in \text{train. set}} \text{distance}(\mathbf{x}^{(i)}, \mathbf{x})$$

2. Output  $y = t^*$

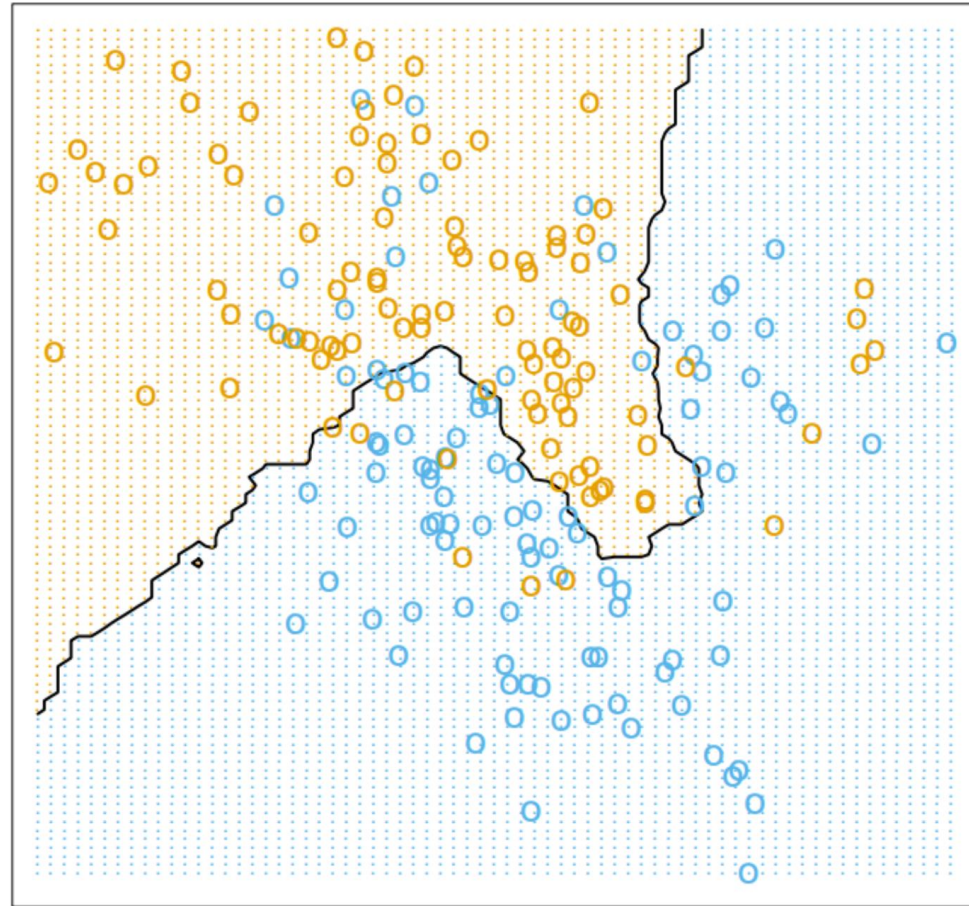
$K=1$



[Image credit: "The Elements of Statistical Learning"]



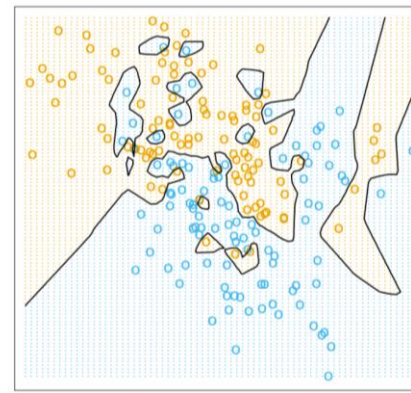
$K=15$



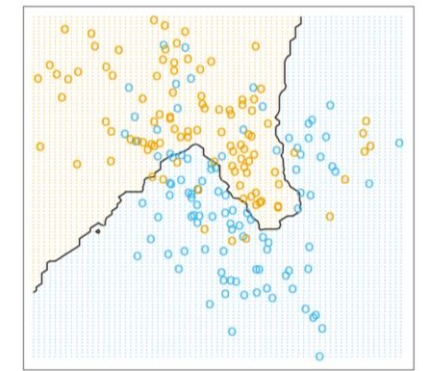
[Image credit: "The Elements of Statistical Learning"]



## Tradeoffs in choosing $k$ ?



[Image credit: "The Elements of Statistical Learning"]



[Image credit: "The Elements of Statistical Learning"]

- Small  $k$

- ▶ Good at capturing fine-grained patterns
- ▶ May **overfit**, i.e. be sensitive to random **NOISE**

Excellent for training data, not that good for new data, too complex

- Large  $k$

- ▶ Makes stable predictions by averaging over lots of examples
- ▶ May **underfit**, i.e. fail to capture important regularities

- Rule of thumb:  $k < \sqrt{n}$ , where  $n$  is the number of training examples

Not that good for training data, not good for new data, too simple

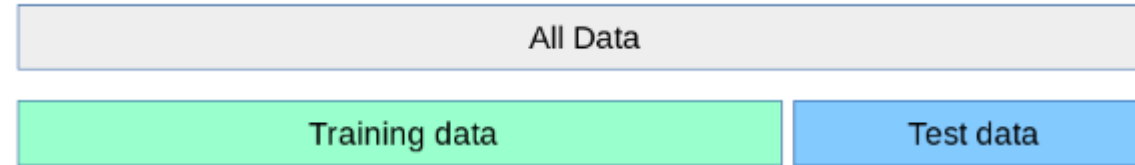
What is the best K?

**K is a hyperparameter**

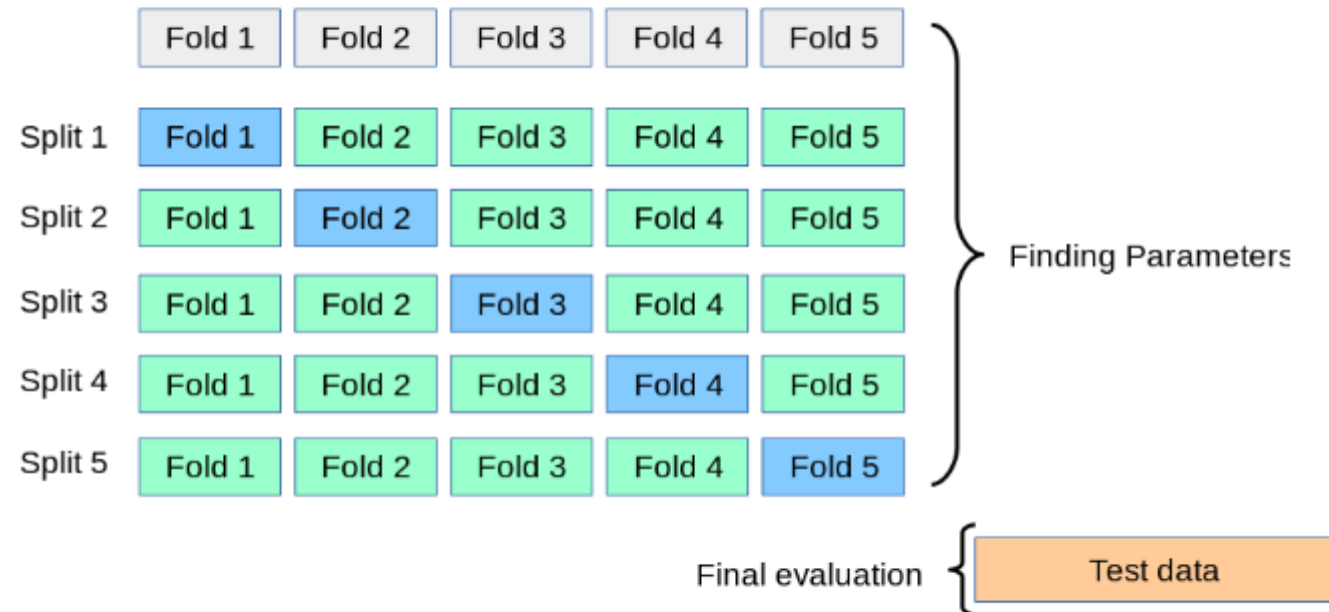
# Cross-validation

# K-Fold cross-validation

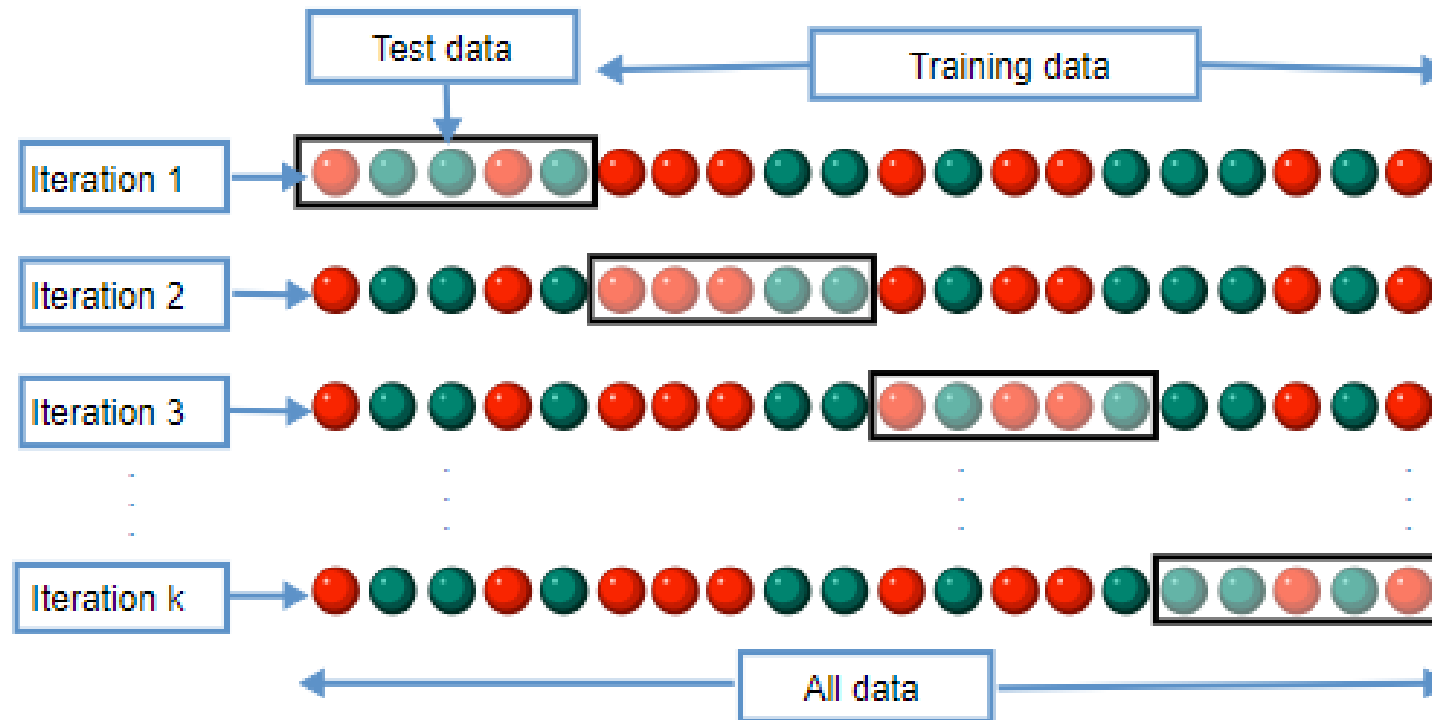
Splitting test and train



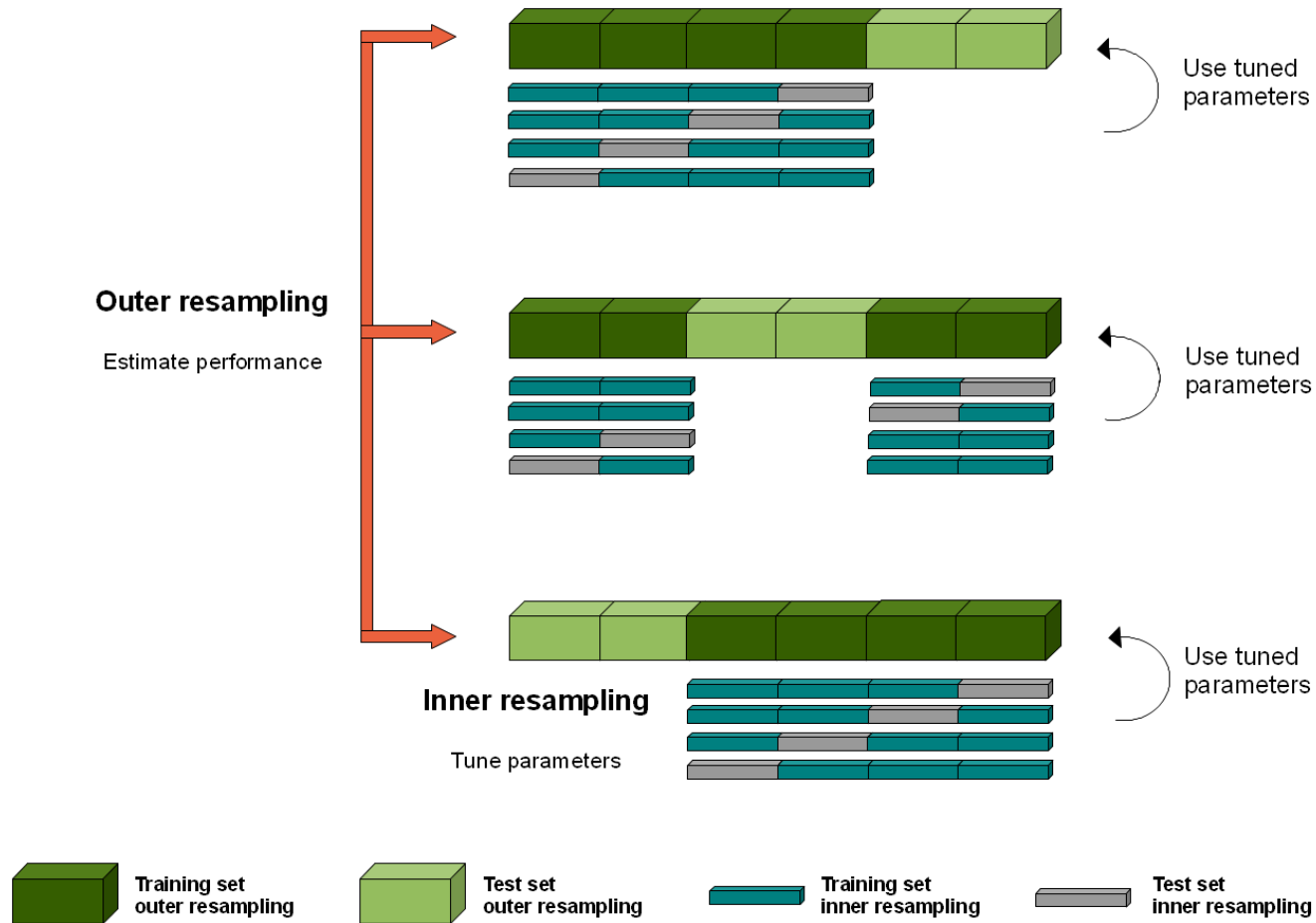
Splitting test and train, with cross-validation during training



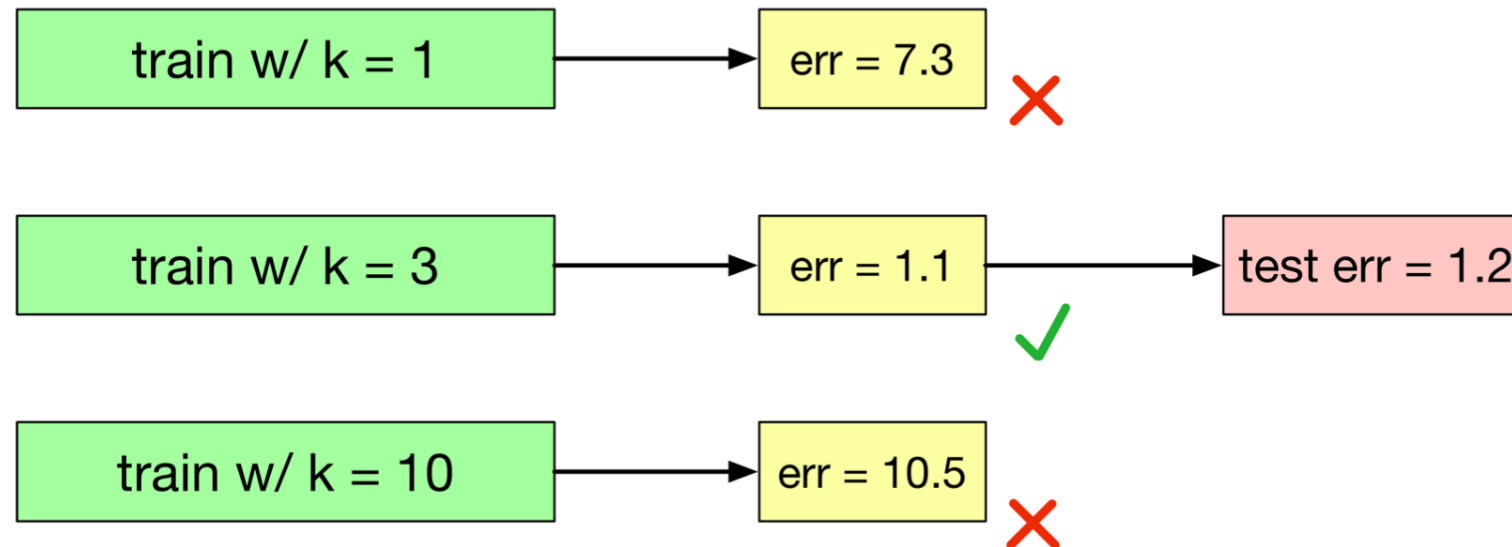
# K-Fold cross validation – great with small # of examples



# Nested cross-validation

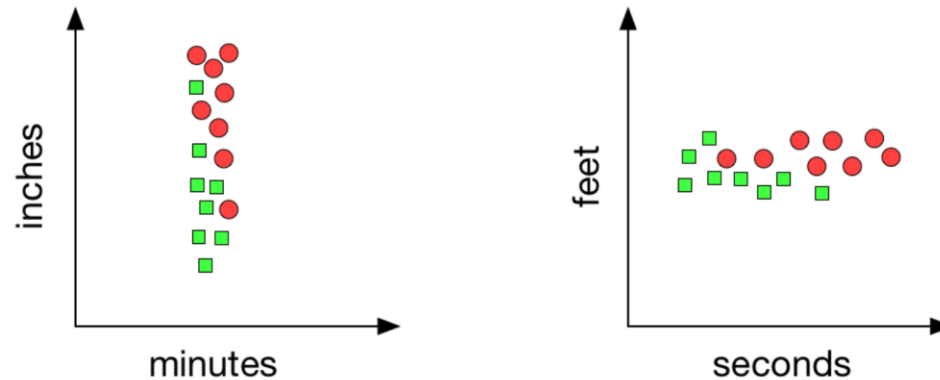


# How to use a dataset?



# Normalization

- Nearest neighbors can be sensitive to the ranges of different features.
- Often, the units are arbitrary:



- Simple fix: **normalize** each dimension to be zero mean and unit variance. I.e., compute the mean  $\mu_j$  and standard deviation  $\sigma_j$ , and take

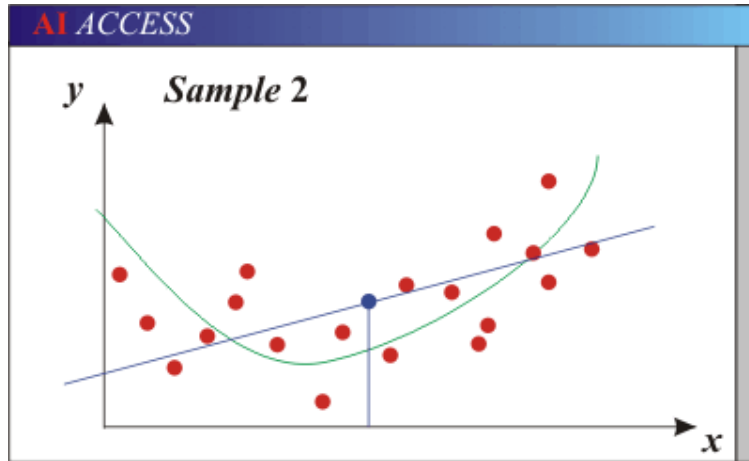
$$\tilde{x}_j = \frac{x_j - \mu_j}{\sigma_j}$$

- Caution: depending on the problem, the scale might be important!

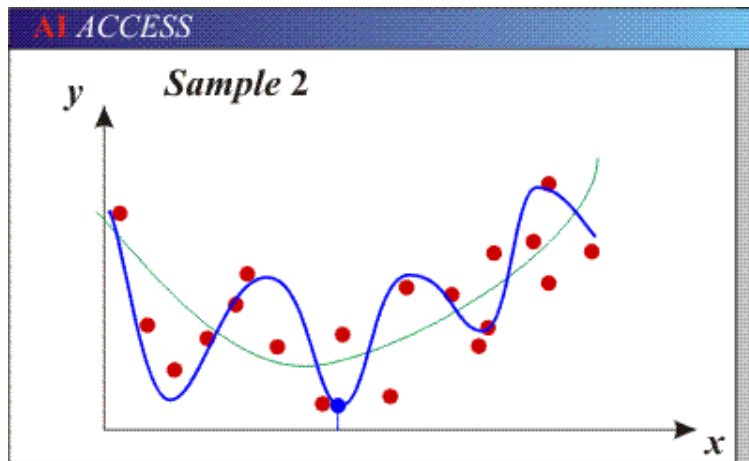




# Bias-Variance Trade-off



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

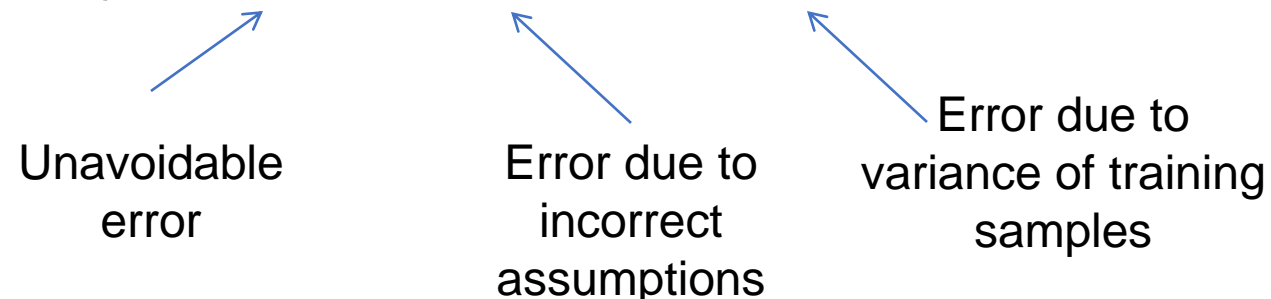


- Models with too many parameters are inaccurate because of a large variance (too much sensitivity to the sample).

# Bias-Variance Trade-off

$$E(\text{MSE}) = \text{noise}^2 + \text{bias}^2 + \text{variance}$$

Unavoidable  
error



Error due to  
incorrect  
assumptions

Error due to  
variance of training  
samples

See the following for explanations of bias-variance (also Bishop's "Neural Networks" book):

- <http://www.inf.ed.ac.uk/teaching/courses/mlsc/Notes/Lecture4/BiasVariance.pdf>

# Generalization



Training set (labels known)



Test set (labels unknown)

- How well does a learned model generalize from the data it was trained on to a new test set?

# Generalization

- Components of generalization error
  - **Bias:** how much the average model over all training sets differ from the true model?
    - Error due to inaccurate assumptions/simplifications made by the model
    - Measure of model rigidity
  - **Variance:** how much models estimated from different training sets differ from each other
- **Underfitting:** model is too “simple” to represent all the relevant class characteristics
  - High bias and low variance
  - High training error and high test error
- **Overfitting:** model is too “complex” and fits irrelevant characteristics (noise) in the data
  - Low bias and high variance
  - Low training error and high test error

# Inductive Bias

- **Let's avoid making assumptions about  $f$** 
  - Assume for simplicity that  $\mathbf{D} = \{(\mathbf{x}_i, \mathbf{y}_i)\}$  is noise free
  - $\mathbf{x}_i$ 's in  $\mathbf{D}$  only cover small subset of input space  $\mathbf{x}$
- **What's the best we can do?**
  - If we've seen  $\mathbf{x}=\mathbf{x}_i$  report  $\mathbf{y}=\mathbf{y}_i$
  - If we have not seen  $\mathbf{x}=\mathbf{x}_i$ , 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 hypothesis space**
  - Important to understand restrictions (and whether appropriate)

# Overfitting

- In brief: fitting characteristics of training data that do not generalize to future test data
- Central problem in machine learning
  - Particularly problematic if  $\#data \ll \#parameters$   
... don't have enough data to “identify” parameters



**Python + ML libraries**

# Common sequence operations

All sequence data  
types support the  
following  
operations

Operation	Result
<code>x in s</code>	True if an item of s is equal to x, else False.
<code>x not in s</code>	False if an item of s is equal to x, else True.
<code>s + t</code>	The concatenation of s and t.
<code>s * n, n * s</code>	n shallow copies of s concatenated.
<code>s[i]</code>	ith item of s, origin 0.
<code>s[i:j]</code>	Slice of s from i to j.
<code>s[i:j:k]</code>	Slice of s from i to j with step k.
<code>len(s)</code>	Length of s.
<code>min(s)</code>	Smallest item of s.
<code>max(s)</code>	Largest item of s.
<code>s.index(x)</code>	Index of the first occurrence of x in s.
<code>s.count(x)</code>	Total number of occurrences of x in s.



# Common sequence operations

Mutable sequence types further support the following operations.

Operation	Result
<code>s[i] = x</code>	Item i of s is replaced by x.
<code>s[i:j] = t</code>	Slice of s from i to j is replaced by the contents of t.
<code>del s[i:j]</code>	Same as <code>s[i:j] = []</code> .
<code>s[i:j:k] = t</code>	The elements of <code>s[i:j:k]</code> are replaced by those of t.
<code>del s[i:j:k]</code>	Removes the elements of <code>s[i:j:k]</code> from the list.
<code>s.append(x)</code>	Add x to the end of s.

# Common sequence operations

and these...

Operation	Result
<code>s.extend(x)</code>	Appends the contents of x to s.
<code>s.count(x)</code>	Return number of i's for which <code>s[i] == x</code> .
<code>s.insert(i, x)</code>	Insert x at position i.
<code>s.pop([i])</code>	Same as <code>x = s[i]; del s[i]; return x</code> .
<code>s.remove(x)</code>	Same as <code>del s[s.index(x)]</code> .
<code>s.reverse()</code>	Reverses the items of s in place.

# 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.

# 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 datatypes

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 arrays

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., `arange`, `ones`, `zeros`, etc.).
- Reading arrays from disk, either from standard or custom formats (e.g. reading in from a CSV file).
- and others ...

# Numpy arrays

- 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

# SciPy

Now we move on to SciPy. In it's own words:

SciPy is a collection of mathematical algorithms and convenience functions **built on the Numpy extension** of Python. It adds significant power to the interactive Python session by providing the user with high-level commands and classes for manipulating and visualizing data. With SciPy an interactive Python session becomes a data-processing and system-prototyping environment rivaling sytems such as MATLAB, IDL, Octave, R-Lab, and SciLab.

- Basically, SciPy contains various tools and functions for solving common problems in scientific computing.

# SciPy

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

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`)
- Compressed Sparse Graph Routines (`scipy.sparse.csgraph`)
- Spatial data structures and algorithms (`scipy.spatial`)
- Statistics (`scipy.stats`)
- Multidimensional image processing (`scipy.ndimage`)
- Data IO (`scipy.io`)
- Weave (`scipy.weave`)

*and more!*

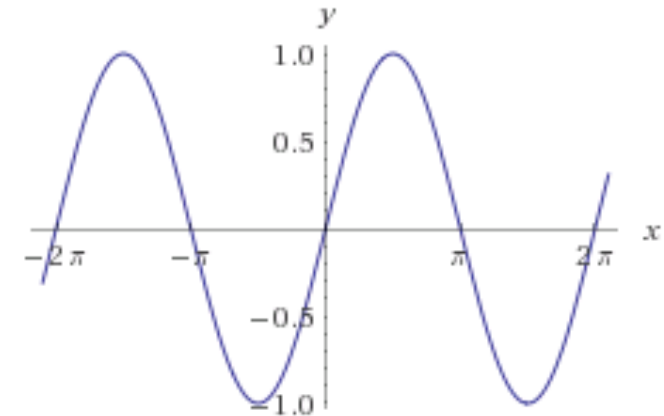
# SciPy

- We can't possibly tour all of the SciPy library and, even if we did, it might be a little boring. So let's just look at some example modules with SciPy to see how it can be used in a Python program.

Let's start with a simple little integration example.  
Say we wanted to compute the following:

$$\int_a^b \sin x \, dx$$

- Obviously, the first place we should look is `scipy.integrate`!



# Scipy.integrate

- **Methods for Integrating Functions given a function object:**
  - `quad` -- General purpose integration.
  - `dblquad` -- General purpose double integration.
  - `tplquad` -- General purpose triple integration.
  - `fixed_quad` -- Integrate `func(x)` using Gaussian quadrature of order `n`.
  - `quadrature` -- Integrate with given tolerance using Gaussian quadrature.
  - `romberg` -- Integrate `func` using Romberg integration.
- **Methods for Integrating Functions given a fixed set of samples:**
  - `trapz` -- Use trapezoidal rule to compute integral from samples.
  - `simps` -- Use Simpson's rule to compute integral from samples.
  - `romb` -- Use Romberg Integration to compute integral from  $(2^{**k} + 1)$  evenly-spaced samples.

# Python Libraries for Data Science

## *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.
- allows handling missing data

Link: <http://pandas.pydata.org/>

# Data Frames attributes

Python objects have *attributes* (think characteristics) and *methods* (functions)

df.attribute	description
dtypes	list the types of the columns
columns	list the column names
axes	list the row labels and column names
ndim	number of dimensions
size	number of elements
shape	return a tuple representing the dimensionality
values	numpy representation of the data



# Data Frames methods

Unlike attributes, python methods have *parenthesis*.

All attributes and methods can be listed with a *dir()* function: `dir(df)`

df.method()	description
head( [n] ), tail( [n] )	first/last n rows
describe()	generate descriptive statistics (for numeric columns only)
max(), min()	return max/min values for all numeric columns
mean(), median()	return mean/median values for all numeric columns
std()	standard deviation
sample([n])	returns a random sample of the data frame
dropna()	drop all the records with missing values