Data Structures

Outcomes:

1. Become familiar with data structures in Python
2. Familiarize yourself with Lists, Dictionaries, and Arrays.
3. Python programming tools for data management
4. Apply the knowledge in processing, analyzing, and modifying datasets for targeted outcomes.

Basic Data Structures

- Lists
- Dictionaries
- Arrays

Lists are ordered sequences of data.

Elements: numbers, strings, lists of lists, etc.

Operations: Add, remove, change/modify

Example

myList= [math.pi, 3.14, 'apple', 'How are you?', [3, 'ok']]

Try the following and see what happens to the list? (Exam question)

```
print (myList[1])
myList.remove('How are you?')
myList.append(123)
```

Operations on Lists

append()	Adds an element at the end of the list
clear()	Removes all the elements from the list
copy()	Returns a copy of the list
count()	Returns the number of elements with the specified value
extend()	Add the elements of a list (or any iterable), to the end of the current list
index()	Returns the index of the first element with the specified value
insert()	Adds an element at the specified position
pop()	Removes the element at the specified position
remove()	Removes the item with the specified value
reverse()	Reverses the order of the list
sort()	Sorts the list

Dictionary Object

Dictionary lets you store items in “pairs”
* [key, value] pair in python

MyDict ={‘key1’: ‘value1’, ‘key2’: ‘value2’} Notice “Key:Value” pairs!

print (MyDict [‘key2’]) #returns value1

MyDict[‘key3’] = ‘value3’

MyList.append(123)

MyList.length()

[3.14, 'apple', [3, 'ok'], 123]

Operations on Dictionary Object

Method	Description
clear()	Removes all the elements from the dictionary
copy()	Returns a copy of the dictionary
fromkeys()	Returns a dictionary with the specified keys and value
get()	Returns the value of the specified key
items()	Returns a list containing a tuple for each key value pair
keys()	Returns a list containing the dictionary's keys
pop()	Removes the element with the specified key
popitem()	Removes the last inserted key-value pair
setdefault()	Returns the value of the specified key. If the key does not exist: insert the key, with the specified value
update()	Updates the dictionary with the specified key-value pairs
values()	Returns a list of all the values in the dictionary

Arrays are collections of elements in a single data type

We use numpy for array support in python.

Arrays provide contiguous data store support: Efficient on store/retrieve.

Examples

```
data = np.array([[3,6], [9,12]])
print (data/3)
[[1,2], [3,4]]
print (data[1,1])
?
```

Follow Tutorials

Basic Data Structures

Basic Language Syntax

Functions

Loops and Control Structures

Object Oriented Programming

Learning Objectives

- Data-centered programming
- Python is OOP
- Writing Functions and Classes
- Properties of OOP

Object Oriented Programming at Work

```
(base) rxv441@UM-F3Y4FYK6FC PIBS792 % python3
Python 3.9.12 (main, Apr  5 2022, 01:53:17)
[Clang 12.0.0 ] :: Anaconda, Inc. on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import os
>>> import Bio
>>> from Bio import SeqIO
>>> for record in SeqIO.parse('gencode.v40.pc_transcripts.fa', 'fasta'):
...     print (record)
```

Writing a Function

- Read sequence and find nucleotides
- How many Cs and Gs?
- Total GC content?

Sub-sequence:
'CCCAGATCTCTTCAGTTTTTATGCC
TCATTCTGTGAAAATTGCTGTAGTC
TCTT'

```
# nucleotides in a sub-sequence
```

```
def CGcount(fragment):
    Ccount = fragment.count('C')
    Gcount = fragment.count('G')
    totalCGcount = 100*(Ccount + Gcount)/len(fragment)
    return totalCGcount
```

String Object in Python

myString = “Hello Class, Welcome to CSCE 4300”

Methods on String:

.capitalize()	Converts first character to Capital Letter
.upper()	Converts whole string into uppercase letters
.lower()	Converts whole string into lowercase letters
.len()	length of the string

What do you Observe?

Practice Assignment

Python Terminal

- Using Terminal Window, run the following commands
- >>> import os
- >>> os.getcwd()
- >>>os.listdir()
- >>>os.mkdir('c5300')
- >>>os.rename('c5300', 'd5300')
- >>>os.chdir('C:\\Users', 'C:\\Users\\d5300')
- >>os.rmdir('d5300')

Jupyter Notebook

Use the sub-sequence:
CCCAGATCTCTTCAGT
TTTTATGCCTCATTCT
GTGAAAATTGCTGTA
GTCTCTT'

Run the function Cgcount (fragment) and report %Cs %Gs and totalCGcount.

Writing User Defined Class

Python Class Structure:

```
class name(object):
        def __init__(self):
                …
                …
        def __operation1(self, a, b):
                self.a = a
                self.b = b
                return a (operation) b
```

Example: Triangle Class

```
Class triangle(object):
    def __init__ (self, base, height):
        self.base = base
        self.height = height
    def getBase(self):
       return self.base
    def setBase(self, b):
       self.base = b
```

Practice 2: Implement Triangle Class in Jupyter Notebook

Writing Circle Class

```
class Circle (object):
        pi = 3.14
        def __init__ Area(self, r=1):
                self.r = r
                return pi*self.r**2

x = Cirlce()
x.Area()                    # 3.14
```

Tutorials

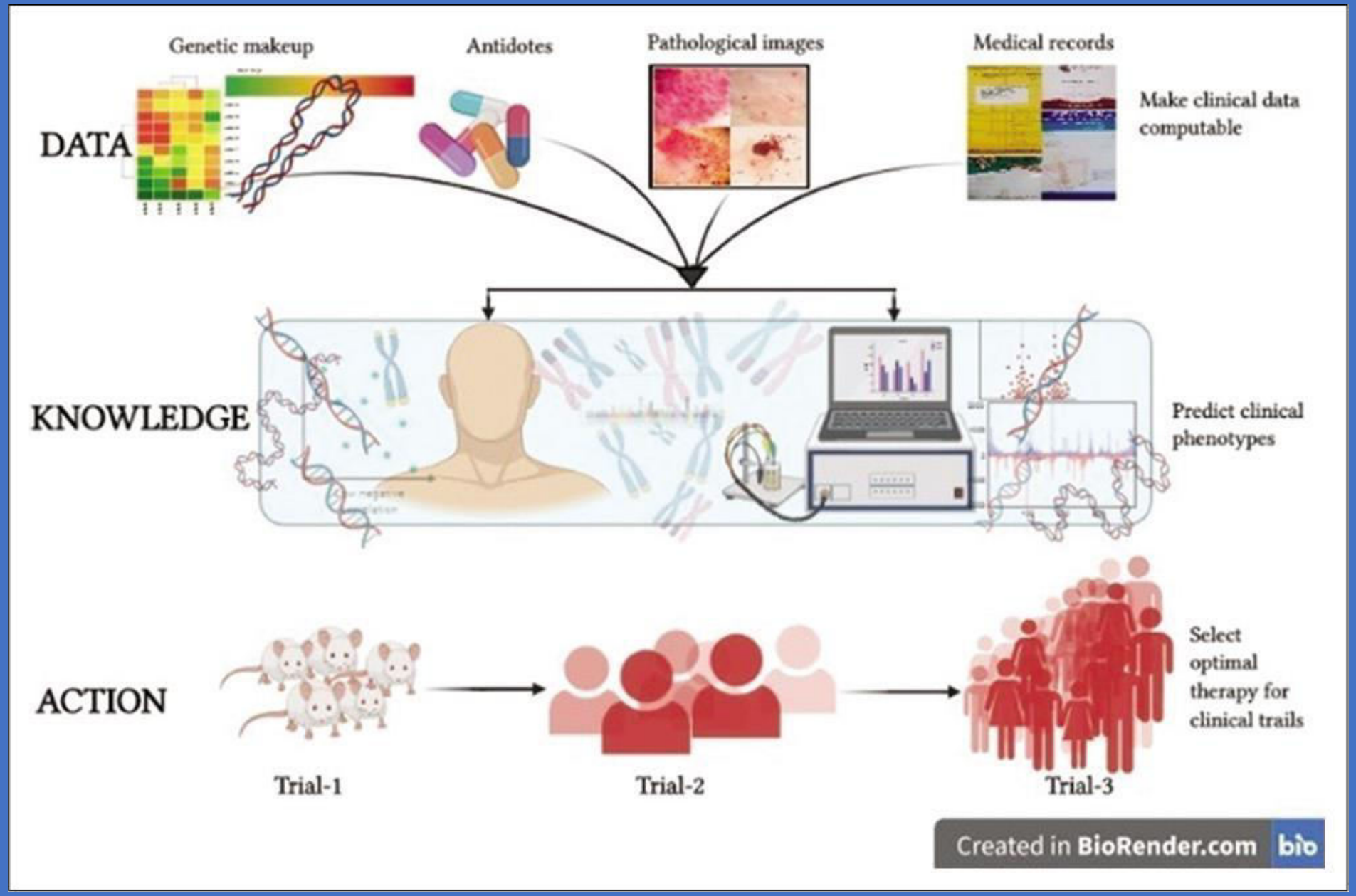
- Object Oriented Programming

Bioinformatics: Computation and Analysis of Genetics and Genomics

- **Sequencing** involves understanding the makeup of our DNA (A, T, G, C)
  - Genes?
  - How are they organized?
  - To help diagnose diseases and develop treatment strategies, etc.
- **Pattern recognition** (from other genomes such as mice, insects, plants, etc.)
  - At least 80% protein coding similarities between mice and humans.
  - What can we learn from patterns different genomes?
  - Involves massive computation, disparate data (fragmented sub-sequences, etc.)
- **Pick differences in DNA**
  - How cells compare?
  - mutations

Biopython: Mature codebase, extensive functionality

SeqIO – Sequence I/O interface for Biopython



https://dx.doi.org/10.25259/IJMIO_28_2020

Biopython

- Library built on python package
- Supports sequencing (nucleotide --> protein) strategies and tools
- Supports various file formats.

- Blast output – both from standalone and W
- Clustalw
- FASTA
- GenBank
- PubMed and Medline
- ExPASy files, like Enzyme and Prosite
- SCOP, including 'dom' and 'lin' files
- UniGene
- SwissProt

```
conda install biopython
import Bio
from Bio import SeqIO
```

Jupyter Notebook: Sequencing Data File I/O

```
In [1]: !pip3 install biopython

Requirement already satisfied: I
/lib/python3.9/site-packages (1
Requirement already satisfied: I
/python3.9/site-packages (from I
```

```
In [2]: import numpy as np
import Bio
from Bio import SeqIO
```

```
!wget https://ftp.ebi.ac.uk/pub/databases/gencode/Gencode_human

--2022-06-28 23:48:47--  https://ftp.ebi.ac.uk/pub/databases/ge
ripts.fa.gz
Resolving ftp.ebi.ac.uk (ftp.ebi.ac.uk)... 193.62.193.138
Connecting to ftp.ebi.ac.uk (ftp.ebi.ac.uk)|193.62.193.138|:443
HTTP request sent, awaiting response... 200 OK
Length: 46681619 (45M) [application/octet-stream]
Saving to: 'gencode.v40.pc_transcripts.fa.gz'

gencode.v40.pc_tran 100%[===================>]  44.52M   579KB/

2022-06-28 23:50:07 (576 KB/s) - 'gencode.v40.pc_transcripts.fa
```

```
import gzip
gz = gzip.open ('gencode.v40.pc_transcripts.fa.gz', 'rb')
content = gz.read()
print (content)
```

Command Line: Sequencing Data File I/O

```
(base) rxv441@UM-F3Y4FYK6FC PIBS792 % python3
Python 3.9.12 (main, Apr  5 2022, 01:53:17)
[Clang 12.0.0 ] :: Anaconda, Inc. on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import os
>>> import Bio
>>> from Bio import SeqIO
>>> for record in SeqIO.parse('gencode.v40.pc_transcripts.fa', 'fasta'):
...     print (record)
```

Pattern Recognition

- Read sequence and find nucleotides
- How many Cs and Gs?
- Total GC content?
- Check for ‘CGAC’ and print yes!
- …

Sub-sequence:
‘CCCAGATCTCTTCAGTTTTTATGCCTCATTCTGTGAAAATTGCT
GTAGTCTCTT’

```
# nucleotides in a sub-sequence
```

```
def CGcount(fragment):
    Ccount = fragment.count('C')
    Gcount = fragment.count('G')
    totalCGcount = 100*(Ccount + Gcount)/len(fragment)
    return totalCGcount
```

Lab Write Gene Class (for fun!)

Purpose: The Gene class reads a subsequence and finds percentage of Cs and Gs and calculates total CG percentage in the subsequence.

Argument: fragment
Methods: Set/Get

Approach: Read “fragment” into string, find # of Cs and # of Gs using .count() method for strings.

Note: NCBI sequence files often have mixed letters – upper and lower case. Make sure the program handles it!

Finding a base pair in a sub-sequence

- Use “re module” (re: regular expression)
- Two methods: re.search(), re.finditer(), and more …

```
import re
subseq = “AGCGCTTTTTACGACCA”
match = re.search(‘GC’, subseq)
if match:
   print(match.start(), match.end()) #returns start, end positions of ‘GC’

Find all occurrences of ‘GC’ in the subsequence
If match:
  for i in match.finditer(‘GC’):
     print(match.start(), match.end(), subseq(match.start(), match.end))
```

Pipelines

```
match2 = re.search(‘GC|TA’)
If match2:
  for j in match2.finditer(‘GC|TA’):
     print(match2.start(), match2.end(),
subseq(match2.start(), match2.end())
```

Tutorials

- Biopython tutorial and cookbook
- https://data-flair.training/blogs/python-directory/

- https://www.youtube.com/watch?v=OQlwLMCinhs

Bioinformatics is divided into five areas:

1. **Functional genomics**. How do genes and intergenic segments contribute to metabolic pathways (aka gene expression patterns)? Examples include new species
2. **Structural genomics**. Understand 3D structures of proteins encoded by the genes.
3. **Comparative genomics**. Compare complete genetic material of one organism with another to better understand how species evolved and the role of coding and non-coding genes
4. **DNA microarrays**. A lab technique to find if a DNA of an individual has gene has mutation
5. **Medical informatics**. Converge medical and CS tools to improve patient care.

Three Branches of Bioinformatics:

1. Computational biology. Data-based solutions in bioinformatics
2. Genetics. Study and variation of inherited characteristics
3. Genomics. Branch of molecular biology concerned with structure, function, evolution, and mapping of genomes.

Biopython is a library of bioinformatics tools that you can use for predictive models and analytics.

Building Blocks of Bioinformatics

Tissue. Section of an organ that consists of a largely homogeneous population oof cell types. Organs are multifunctional and therefore cells are specialized to perform different functions.

Why homogeneous cell type? Finding a section of organ with homogeneous cell types ensures that the gene expression profiles extracted from the cells will accurately represent the class of cells that makeup the tissue.

Cell. Basic unit of a living organism. It's a product of the expression of its genes. A cell contains DNA, RNA, small molecules of nutrients and metabolites.

Gene. Specific segment of DNA on a chromosome that tells the cell how to function. Basic physical and functional unit of heredity. Some genes act as instructions to make molecules called proteins. And others (non-coding) assist in regulating the gene expression.

Gene Expression. The process by which information encoded in the gene is used to assemble a protein molecule. It's a combination of transcription and translation.

Transcription. The process by which cells make an RNA copy (called messenger RNA or mRNA) of a piece of DNA. This mRNA carries the genetic information needed to make proteins in a cell.

Translation. Translating the sequence of mRNA into a sequence of amino acids during protein synthesis

ACGT. A DNA molecule has four types of bases. They are Adenine (A), Cytosine (C), Guanine (G), and Thymine (T).

Base Pairs. DNA molecule consists of two strands wound around each other with each strand held together by bonds between the four bases ACGT. These bases form specific pairs (A with T and G with C) called base pairs.

Human Genome. A human genome has **three billion** base pairs of DNA distributed across 23 chromosomes.

Chromosome. Made up of proteins and DNA and organized into genes. Chromosomes ensure that DNA is copied exactly during a cell division. Human chromosomes range from 50 million to 300 million base pairs.

5' and 3' in DNA. Each end of DNA has a number. One end is referred to as 5' (five prime) and the other end is 3' (three prime). The 5' and 3' represent # of carbon atoms in deoxyribose sugar molecule to which a phosphate group bonds.

KOH-don (Condon). A Condon signals start/stop of translation. It is a sequence of three consecutive nucleotides in a DNA or RNA molecule that codes for specific amino acids.

DNA vs RNA (Key Differences).
DNA: Several million Base Pairs, fairly stable, self-replicating
RNA: several thousand Base Pairs, reactive, synthesized by transcription

Gene vs. Genome. Gene is a specific segment of DNA that tells Cells how to function. A genome is the entirety of the genetic material inside an organism. A human genome consists of 20K-25K genes.

Human Genome. A human genome consists of 20K to 25K genes. Every person has two copies of a gene (one inherited from each parent). Most of the genes are common across all people. Less than 1% of the genes are slightly different in different people and these small differences contribute to each person's unique features.

Diseases. Variations in genes (from less than 1% that differ) can influence health, appearance, and risk of developing certain diseases. Example. Sickle cell disease.

Trends in Bioinformatics.
Single-cell RNA-sequencing (scRNA-seq). It captures global state of all mRNA expressions within a tissue up to a single-cell resolution. The scRNA-seq assists in understanding

1. How does a tissue composition and function change during development of a disease?
2. The heterogeneity and complexity of RNA transcripts (coding for proteins or non-coding for other functions) within individual cells.
3. Composition of different cell types and functions within a tissue.

RNA-seq provides "average" of the expression profiles of 1000s of cells.

Disease. Genetic variation may contribute to a disease largely through misregulation of gene expression. A gene expression implies the information encoded in a gene is turned into a function. This occurs via transcription of RNA molecules that code for proteins or non-coding RNA molecules for other functions within a cell. Goals of scRNA-Seq:

1. Measure the distribution of expression levels for each gene across a population of cells.
2. Measure transcriptional differences across and within groups of cells
3. Resolve single-cell heterogeneity.

Applications: analysis of cancer evolution and mechanisms of therapeutic resistance.

DataFrames Using Pandas

Introduction to Pandas Package

- Python Library - Pandas

  ```
  import pandas as pd
  ```

- DataFrame is an Object
  - Represent data as table
  - Manipulate data in a table
  - Performing grouped calculation
  - Handling missing data
  - Reading and writing data frames

Tabular Data

- Frequently, data are held in tables.
- A **table** is an arrangement of data in rows and columns.
- **DataFrame** object is designed for common data analysis operations on rows or columns.
- ***Pandas*** is a well-maintained Python library for tabular data management.

Benefits of Using Pandas

- Pandas is an open-source library built on top of NumPy.
- It allows for fast analysis and data cleaning and preparation.
- It also has built-in visualization features.
- It can work with data from a wide variety of sources.

More about Pandas DataFrames

- Go through the tutorial to learn how to:
  - Create a data frame from scratch
  - Select rows or columns
  - Delete rows or columns
  - Create new columns
  - Perform grouped calculations
  - Handle missing data
  - Read and write data in CSV format

Pandas for Data Management

```
import pandas as pd
data = {'Name': ['Raj', Khan', 'David', 'Laura'],
        'Age':[20, 25, 35, 18], 'City':['Dallas', 'Orlando', 'Detroit', 'Arlington'],
        'State':['TX', 'FL', 'MI, 'VA'] }
#Create a DataFrame
df = pd.DataFrame(data)
 #print two columns Name, City

print (df[['Name', 'City']])
```

#print first/last row using iloc command

```
print (df.iloc[0])
print(df.iloc[-1])
```

#print by col name

```
df ['Name']
```

 # delete col

```
df.drop(['Name'], 1)
```

#get rows that meets a condition

```
new_df = df [df ['Age'] > 18]
print (new_df)
```

Missing Data

Some Raw Data for Example

```
raw_data = {'First Name': ['Jason', np.nan, 'Tina', 'Jake', 'Amy'], 'Last Name': ['Miller', np.nan, 'Ali', 'Milner', 'Cooze'], 'Age': [42, np.nan, 36, 24, 73], 'Sex': ['m', np.nan, 'f', 'm', 'f'], 'Pre Test-score': [4, np.nan, np.nan, 2, 3], 'Post Test-score': [25, np.nan, np.nan, 62, 70]} df = pd.DataFrame(raw_data, columns = ['First Name', 'Last Name', 'Age', 'Sex', 'Pre Test-score', 'Post Test-score'])
```

#Drop NaN

```
df_no_missing = df.dropna()
```

#Replace NaN with 0

```
df_fill_values = df.fillna(0)
```

#replace NaN with mean values

```
mean_value = df[‘pre-test-score’].mean()
```

#update the dataframe

```
df[‘pre-test-score].fillna(value=mean_value, inplace=True)
```

#write into a new dataframe

```
df_mean_value = df[‘pre-test-score’].fillna(value=mean_value, inplace=False)
```

Tutorials

Data Frames

Data Visualization

Matplotlib

Why Data Visualization

“A picture is worth a thousand words”

- First step of data analysis.
- Provides an intuitive understanding of data.
- Assists in picking patterns and outliers.
- Visual representations enhances the human cognitive process.

Benefits of Data Visualization

- Allows different perspectives of data
- Cognitive processing and prediction strategies
- Interpret vast amounts of data

Matplotlib

- Matplotlib is a python package 2D visualization (aka arrays)
- Built on NumPy arrays and designed to work with the broader SciPy stack.
- Supports both quantitative (pie, bar, histogram) and rate of change (line, scatter)

- from matplotlib import pyplot as plt *or*
- import matplotlib.pyplot as plt

Objectives

- Data visualization:

2D:

- Pie Chart
- Bar Plot
- Grouped Bar Plot
- Histogram
- Line Plot
- Scatter Plot
- Subplot

3D

- Contour Plot
- Surface Plot
- Mesh Plot

What's so special about
Each of these plot types?

Pie Charts

Homo sapiens - NCBI - NLM (nih.gov)

If you have several parts of something whole, you can demonstrate in one pie chart.

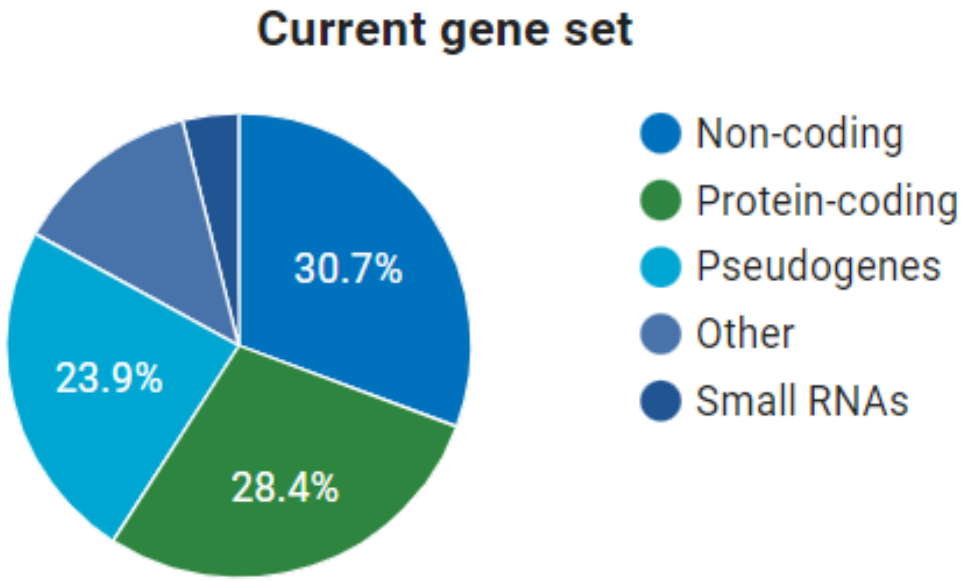
In a pie chart, the area of each slice is proportional to the quantity it represents.

NCBI Reference genome GRCh38.p14

Genome size	3.1 Gb
Contig N50	57.9 Mb
Genes	59,265

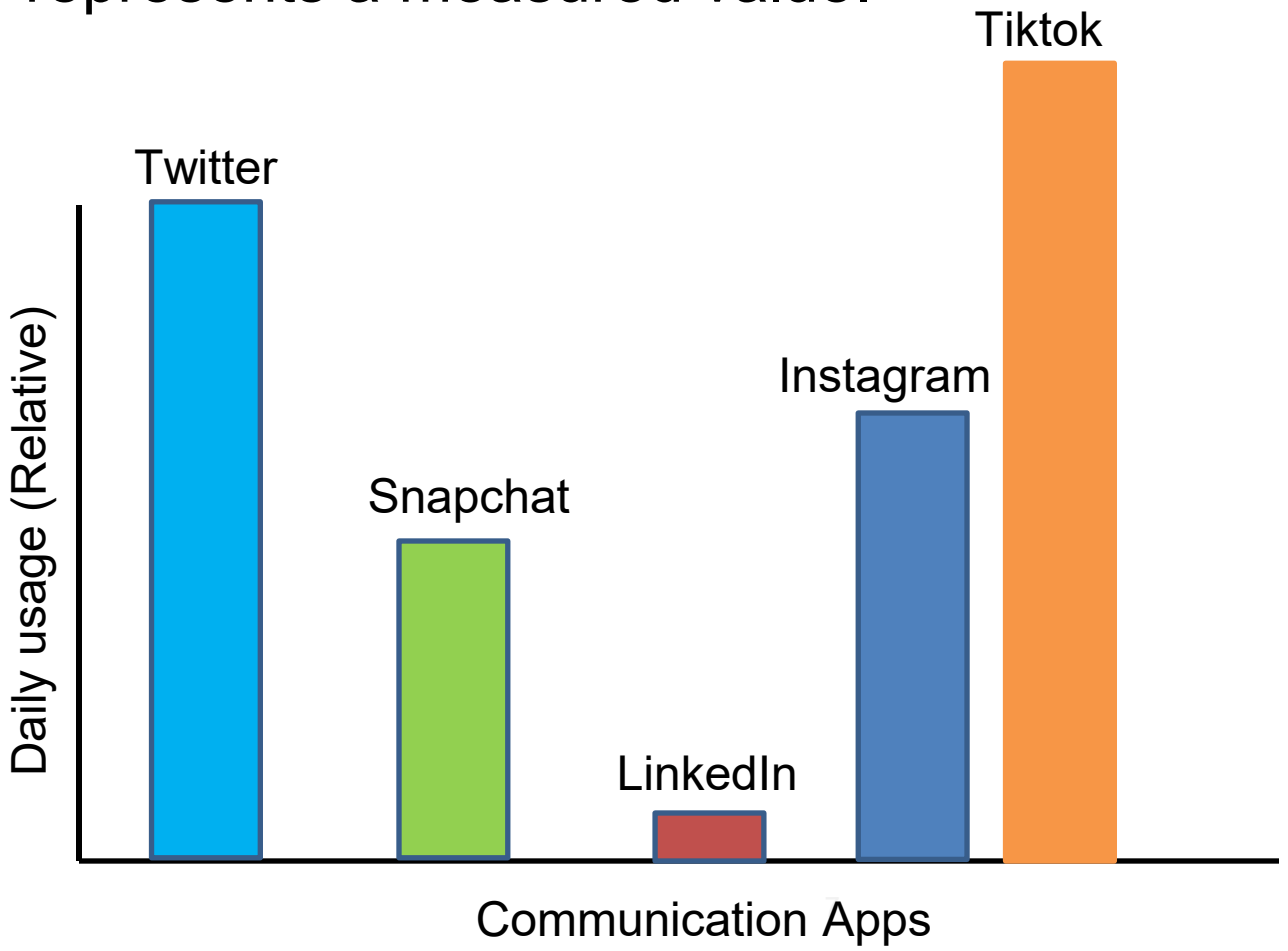
NCBI Annotation Release 110 Feb 25, 2022

View in **Genome Data Viewer (GDV)**



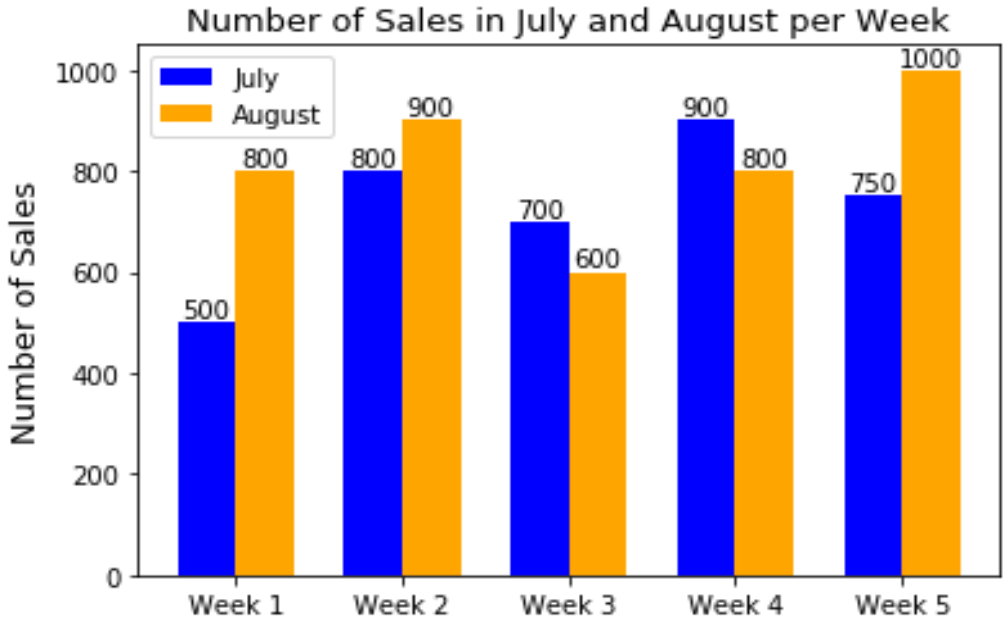
Bar Charts

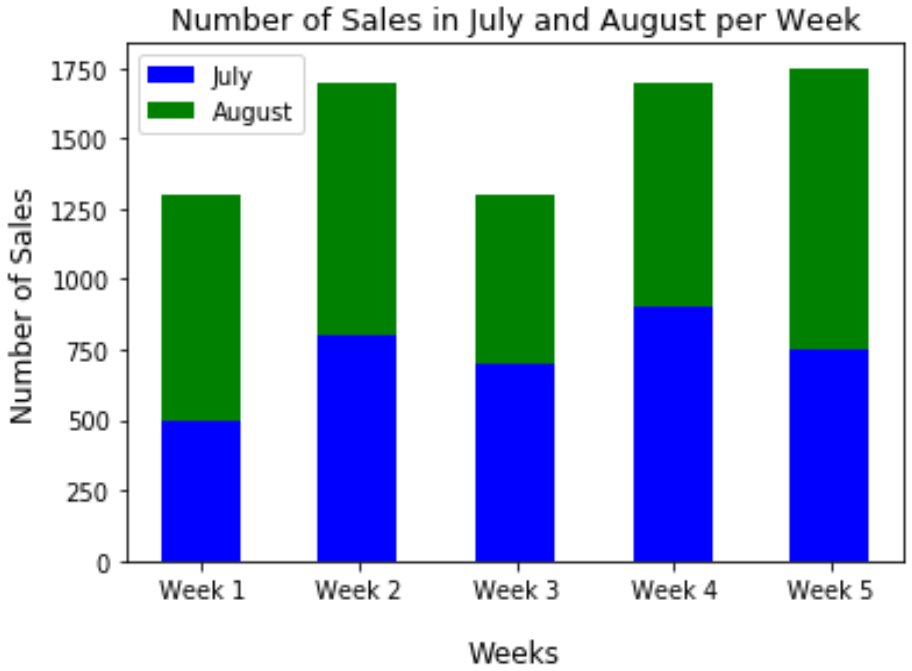
- A bar chart shows comparisons among discrete categories.
- One axis of the chart shows the specific categories being compared, and the other axis represents a measured value.



Grouped and Stacked Bar Plots

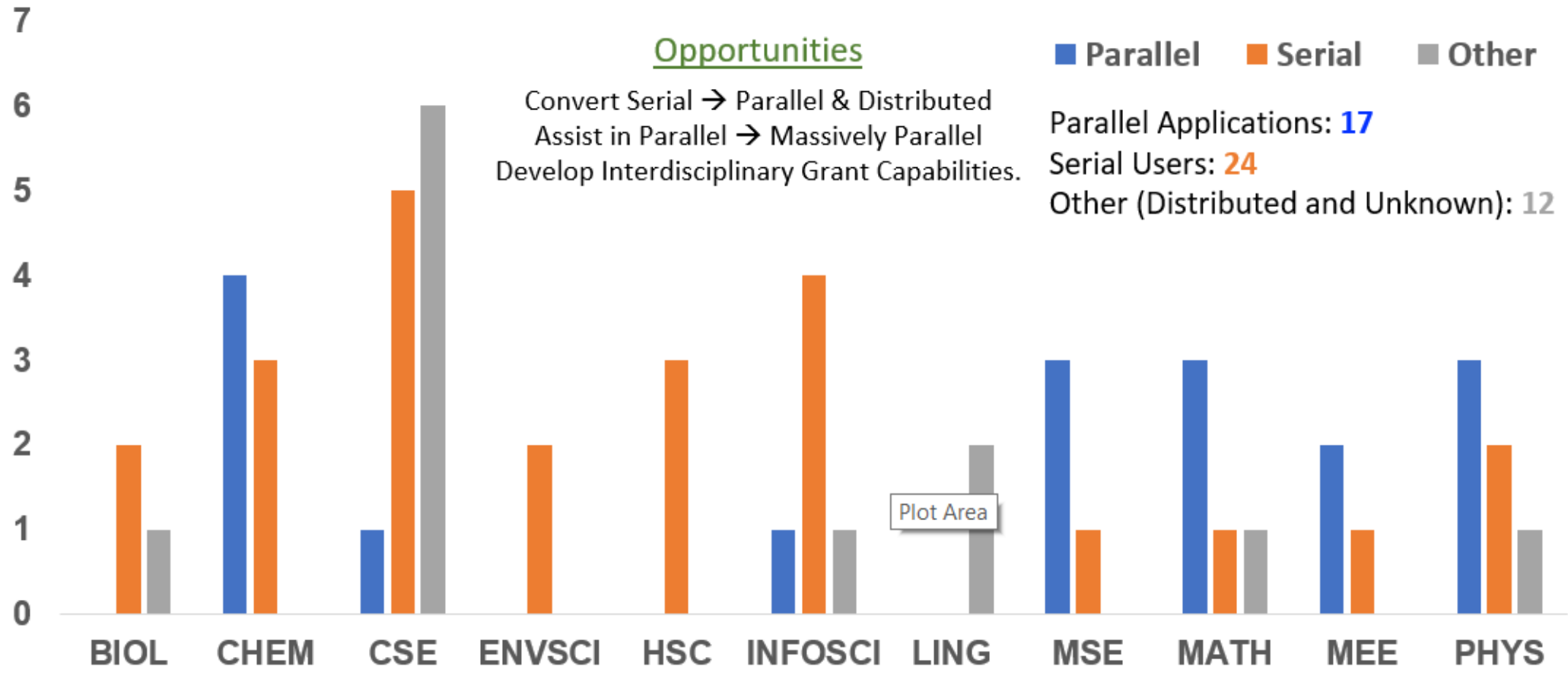
- Grouped Bar plots are used for comparing sizes of individual groups of categorical data
- Stacked Bar plots are used to show total size of the groups along with individual players in the group.





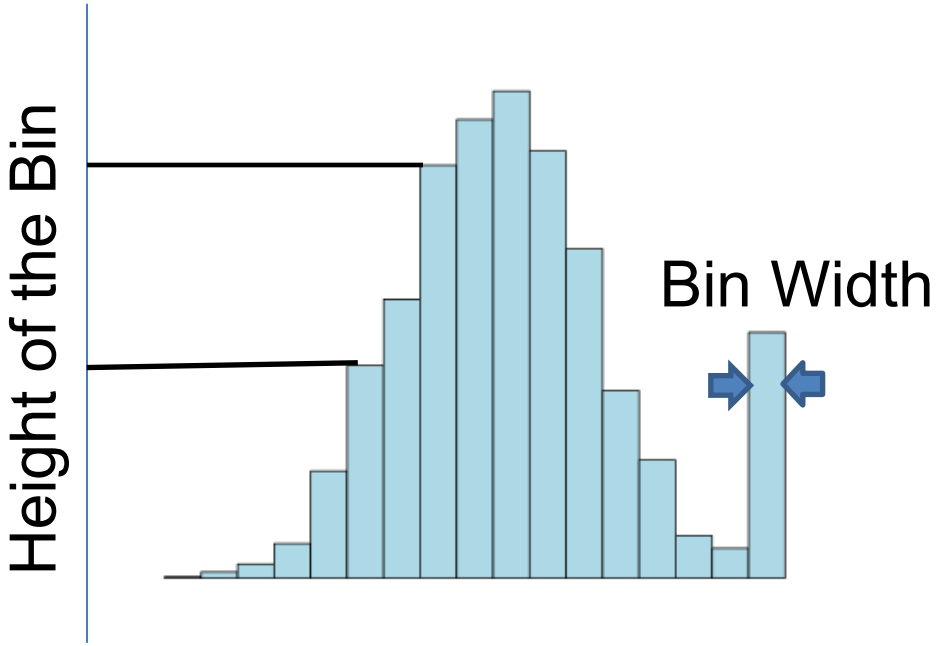
Courtesy Dr. Ting, UNT COI

Stacked Bars for Business Development



Histograms

- A histogram displays numerical data of a parameter and is grouped by equal number of bins.
- Height of the bin represents how many data points are in that bin.
- Bins are also sometimes called intervals, buckets, or classes.



Square Root Method: $k = \sqrt{n}$

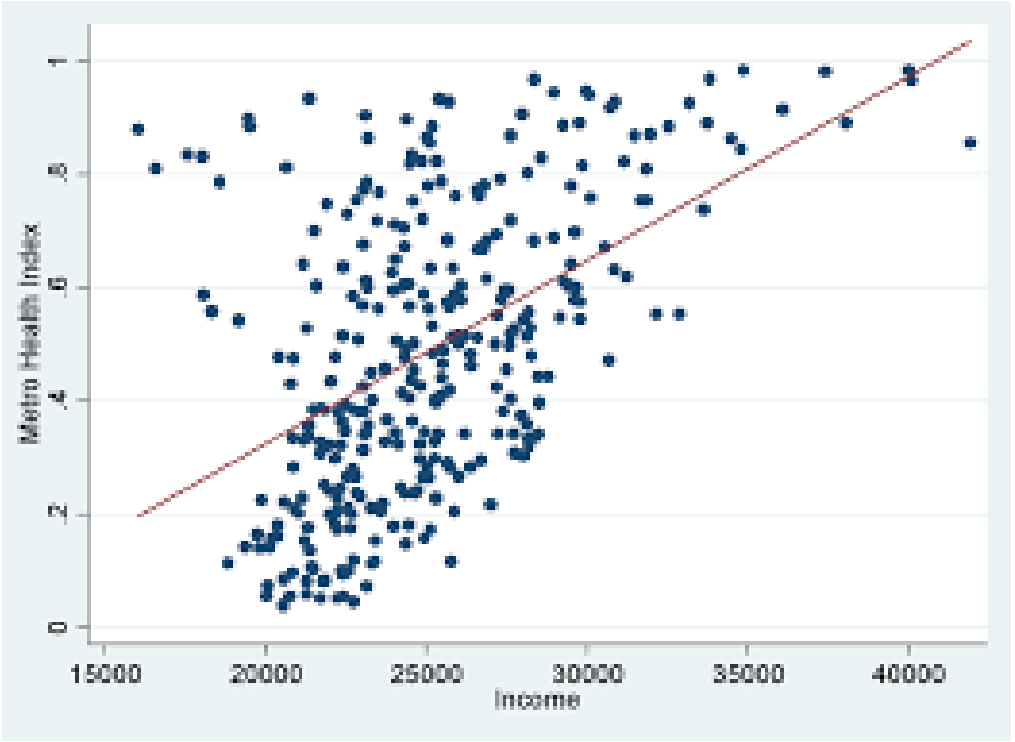
Sturges Formula $k = [log_2 n] + 1$

Rice Rule $k = 2n^{1/3}$

How to calculate the bin width?

Scatter Plot

- Scatter plot represents a relationship between two sets of data
- Goal is to understand (rather intuitively) their relationship as a starting point



Minnesota Public Health dataset showing income inequalities in health outcomes

https://www.health.state.mn.us/communities/practice/resources/phqitoolbox/scatterplot.html

Matplotlib for 2D plots

```
import matplotlib.pyplot as plt
```

Line plot:	plt.plot(x, y)
Scatter plot:	plt.scatter(x, y)
Subplot:	plt.subplot(nrows, ncols, index)
Bar:	plt.bar(x,y)
Histogram:	plt.hist (y)

Object Oriented Method

- A better way to create plots

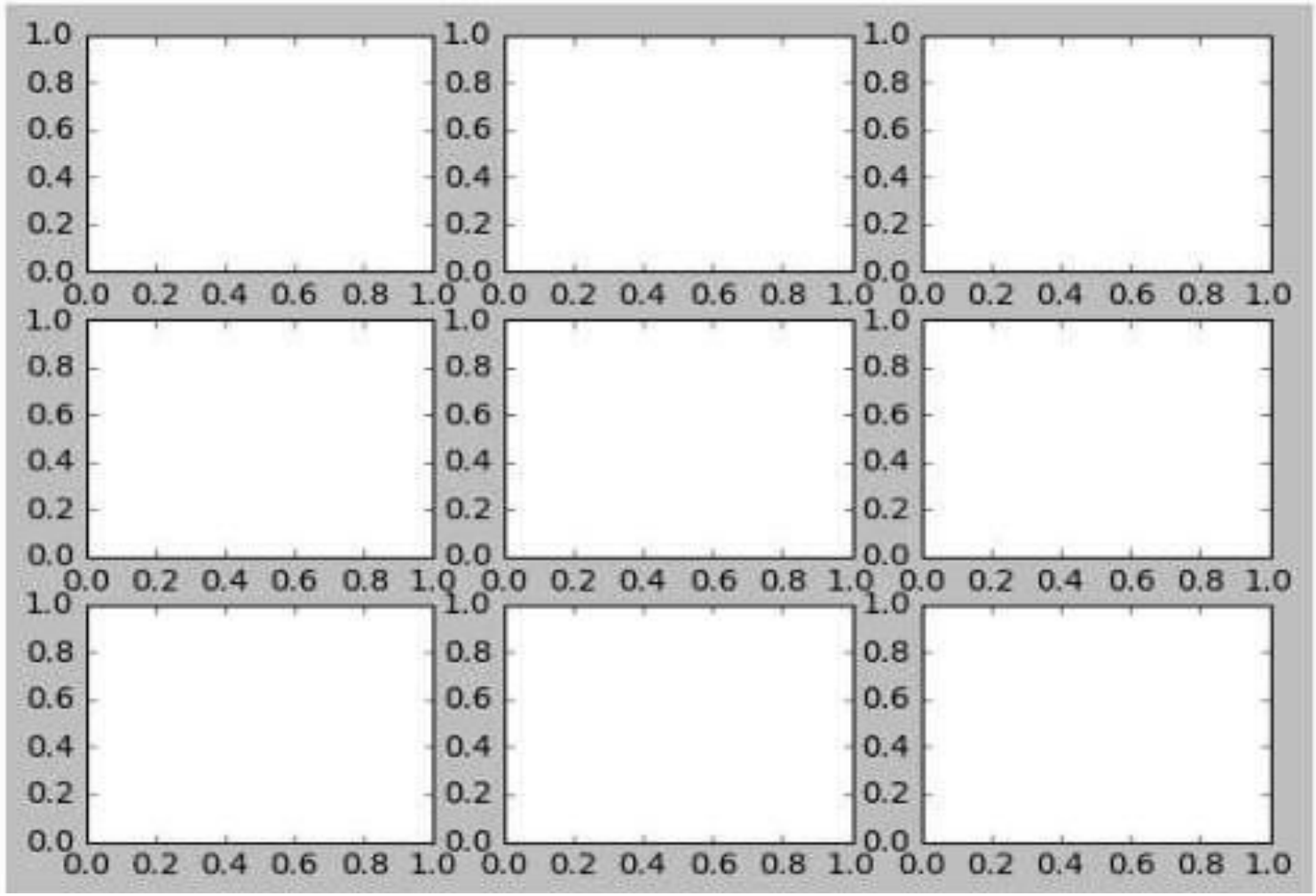
```
import matplotlib.pyplot as plt

fig = plt.figure()   # create a figure object

ax1 = fig.add_subplot(221)  # create a subplot object
ax1.set_ylabel('y label')
ax1.set_xlabel('x label')
ax1.scatter(x,y)
```

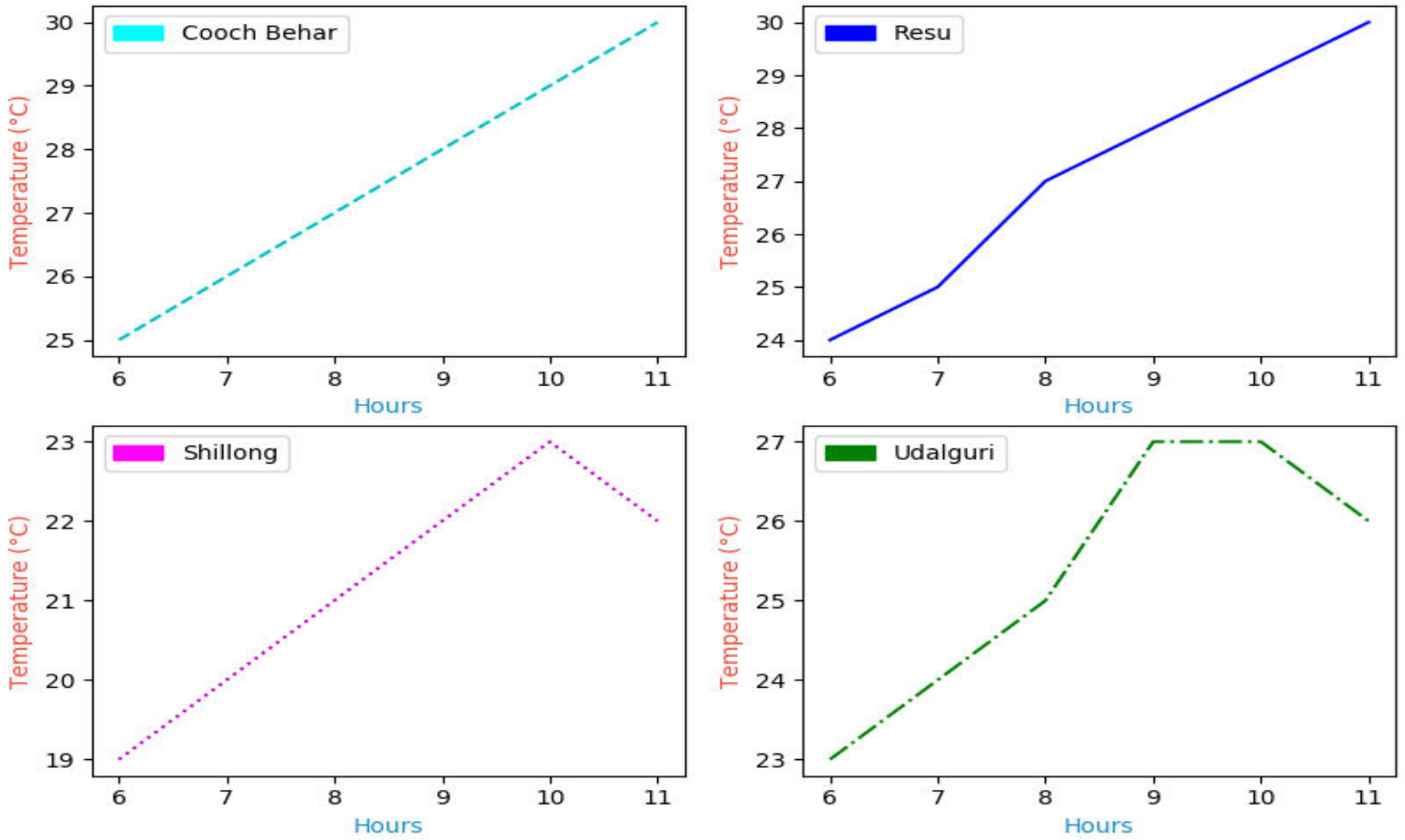
Matrix of subplots

```
In [30]: # Empty canvas of 3 by 3 subplots
         fig, axes = plt.subplots(nrows=3, ncols=3)
```



Subplots

- Show multiple plots on a single figure
- Compare different views of the data side by side.



Courtesy Dr. Ting, UNT COI

References

- https://www.simplifiedpython.net/data-visualization-python-tutorial/
- https://matplotlib.org/
- https://mode.com/blog/python-data-visualization-libraries/
- https://towardsdatascience.com/top-6-python-libraries-for-visualization-which-one-to-use-fe43381cd658

Tutorials

- 2D plots

Data Visualization

Outcomes

- Contour Plots
- Surface Plots
- Publication quality graph(ics)

Libraries Needed

- matplotlib.pyplot
- numpy
- meshgrid()
  - Creates a rectangular grid from an array of x and y values.
  - Replaces loops in python with a vectorized operations
- Countour()
  - Represent a 3D in a 2D map
  - Changes are represented by spacing between contours.
- Contourf() (for filled contours)
  - Draws filled contours
  - Different colors help see changes better (e.g., heatmap analysis)

Functions()

Anti-aliasing: Removing signal components that can't be resolved. Such as minimizing image distortion effects when high resolution image is read by a low-resolution device. Computationally Expensive!

Mesh Plots

```
from matplotlib import cm
from matplotlib.ticker import LinearLocator
surf = ax.plot_surface(X, Y, Z, cmap=cm.coolwarm, linewidth=0,
antialiased=False)
```

Surface Plot

```
from mpl_toolkits.mplot3d import Axes3D
ax = plt.axes(projection='3d')
ax.plot_surface(X,Y,Z) #X,Y,Z are 2D array data values
```

Heatmaps

```
color_map = plt.get_cmap('hot')
surfplot = ax.plot_surface(X,Y,Z, cmap=color_map, edgecolor='none')
fig.colorbar(surfplot,ax=ax, shrink=0.5, aspect=5)
```

Objectives

- Data visualization tasks:
  - Pie Chart
  - Bar Plot
  - Grouped Bar Plot
  - Histogram
  - Line Plot
  - Scatter Plot
  - Subplot
  - Contour Plot
  - Surface Plot
  - Mesh Plot

What/Why Contour Plots?

- Contour plot captures changes as f(X,Y) in 2D graph!
- Look for spacing between contours!
- Space is tight: change is rapid!

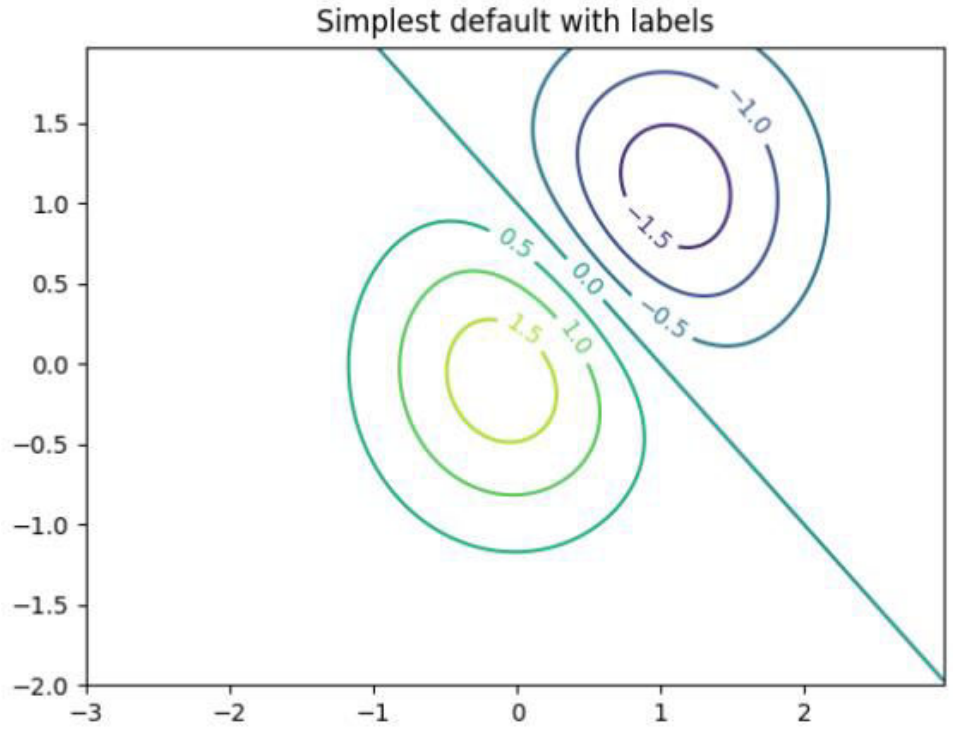
- How to make publication quality figures

Contour plot with Python

```
import numpy as np
import matplotlib.cm as cm
import matplotlib.pyplot as plt

delta = 0.025
x = np.arange(-3.0, 3.0, delta)
y = np.arange(-2.0, 2.0, delta)
X, Y = np.meshgrid(x, y)
Z1 = np.exp(-X**2 - Y**2)
Z2 = np.exp(-(X - 1)**2 - (Y - 1)**2)
Z = (Z1 - Z2) * 2
```

```
fig, ax = plt.subplots()
CS = ax.contour(X, Y, Z)
ax.clabel (CS, inline=True, fontsize=10)
ax.set_title ('Simplest default with labels')
```



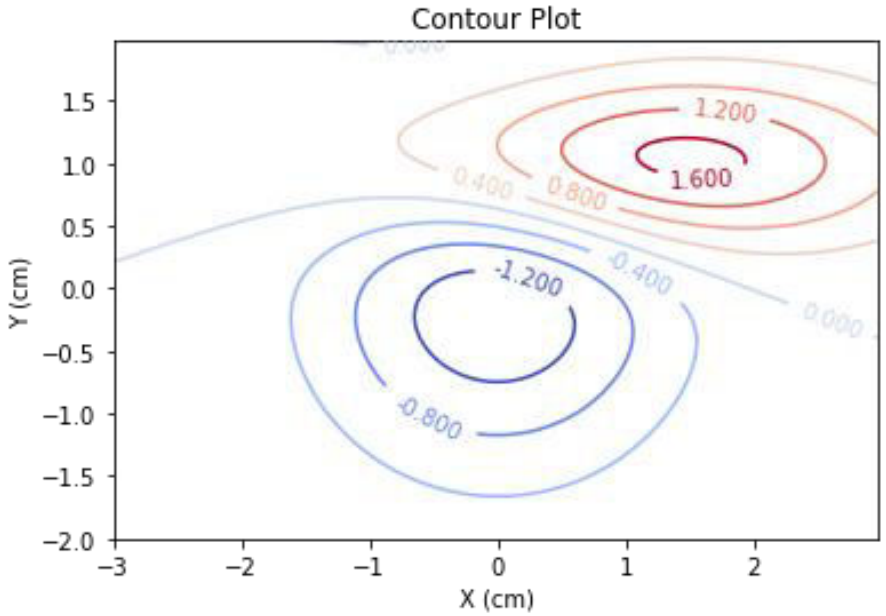
Clabel:
Controls data labels from overlapping on contours

CS: ContourSet object returned by Contour().

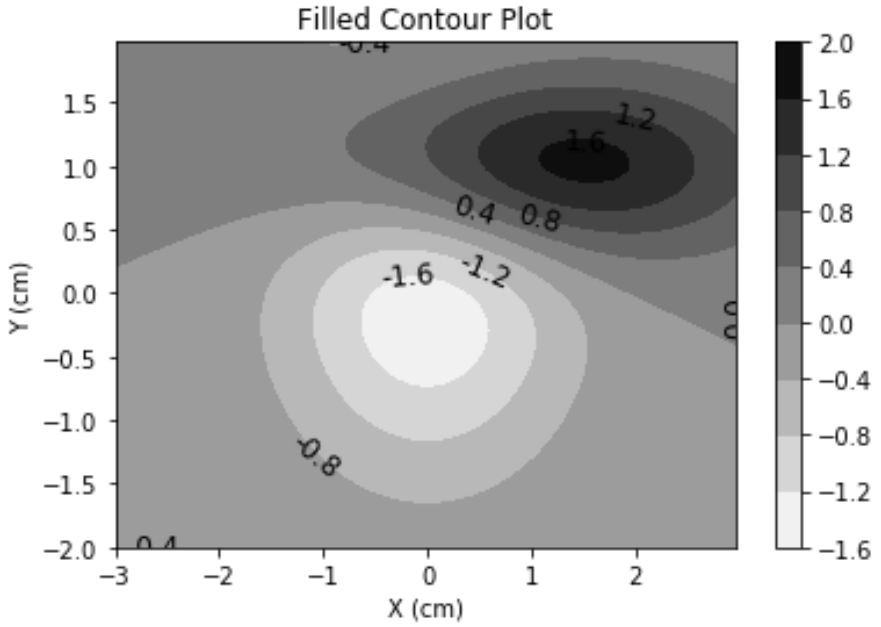
Contour Plots

import matplotlib.pyplot plt
plt.contour()

- Show a 3D surface on a 2D plane.
- Each contour line is shown in an X-Y plot and has a constant Z value.
- Look for peaks, valleys in contour plots!

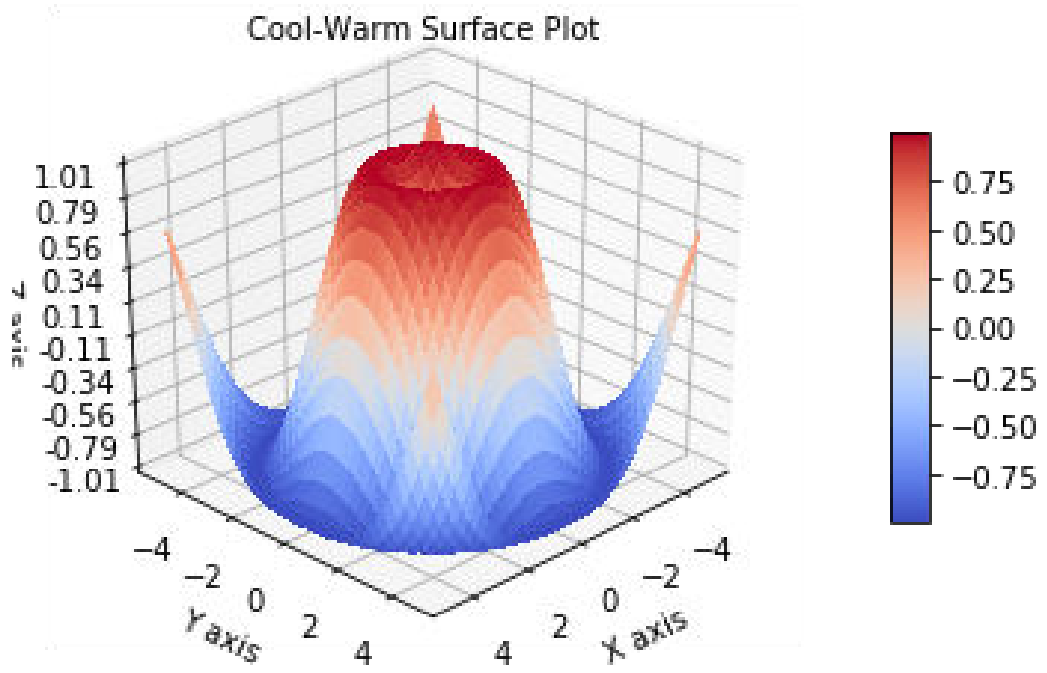


plt.contourf(X,Y,Z)



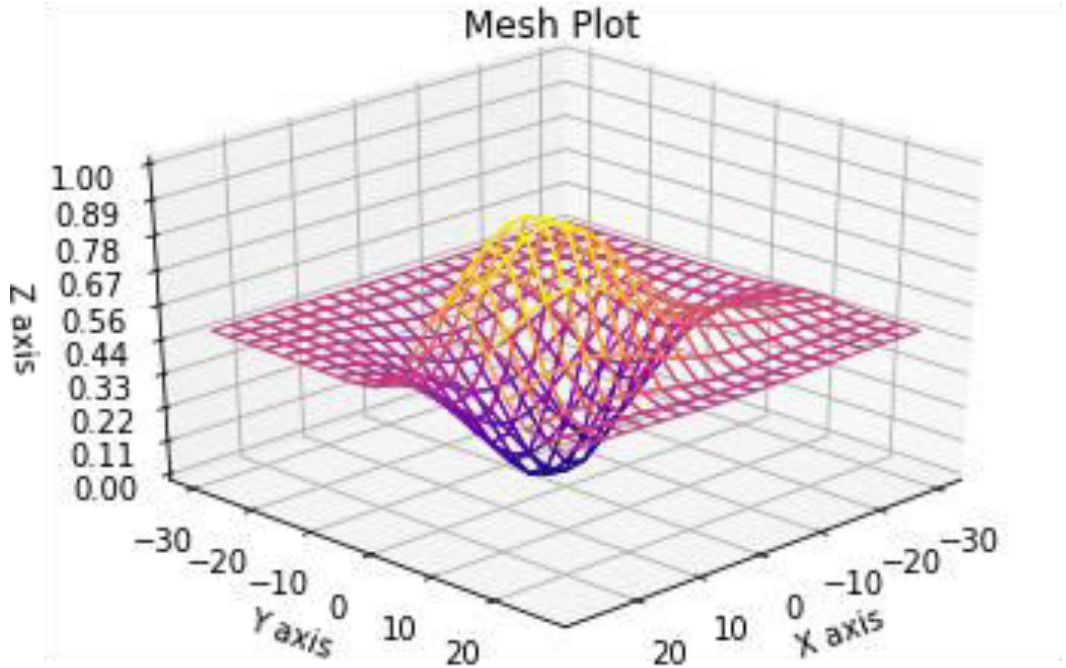
Surface Plots

- Plot 3D data
- Shows a functional relationship between a designated dependent variable (Z), and two independent variables (X and Y).



Mesh Plots

- Like Surface plots (represents 3D data)
- Displays surfaces that color only the lines connecting the defining points.



How to make publication quality figures

Making Publication Quality Figures

Make sure the figure fits with the context of where it is going to be used.

- **Figure outline**: What data do you want to show and how?
- **Purpose**: poster or paper?
- **Submission instructions**: size, resolution, etc.

Making Publication Quality Figures

2. Get a reasonable, rough visual

Pilot/prototype the plot:

- Pick plots that represent your data well.
  - Examples: pie, histogram, surface, etc.
- Configure the plot area to highlight the areas of interest

Choose the right form: pie chart or bar chart, scatter plot or line plot, a single plot or subplots
(**Goal:** Represent data relationships suitably)

Making Publication Quality Figures

3. Add the basics to make figures speak for themselves

- a figure title
- x-axis label (with units)
- y-axis label (with units)
- a legend
- a figure caption
- critical text or shape annotation

Making Publication Quality Figures

4. Format Your Plot (for range, expression, fonts, etc.)

- Fix Limits: X-axis and Y-axis
- Change the figure size
- Change line colors, types, and data point markers
- Make major tick marks minimal to give room for labeling
- Change font sizes
- Change the aspect ratio
- ...

Making Publication Quality Figures

5. Save in a vectorized format (.svg, .pdf, .eps) rather than a raster format (.jpg, .png, .gif, etc)

- **Raster graphics (.jpg, .png, .gif, etc)**: These formats save the image as a collection of X by Y pixels.

- **Vector graphics (.svg, .pdf, .eps)**: These formats define the image using geometric shapes - lines, circles, text by fonts and sizes, etc.

Making Publication Quality Figures

6. Finishing touches with image editing software

- **Example raster graphics editors**:
  - Gimp (free!)
  - Adobe Photoshop

- **Example vector graphics editors**:
  - Inkscape (free!)
  - Adobe Illustrator

Tutorials

- 3D plots
- Publication Quality Figures

Statistics & Terminologies for Data Science

What is Data Science?

- Source: Large and diverse or disparate datasets
- Exploration, prediction, and inference
  - Exploration: "**story**" of this data? Data patterns?
    - Visualization and descriptive statistics
  - Prediction involves "**guessing**" what it means to us! What we wish we knew!
    - Machine Learning and optimization
  - Inference involves establishing a degree of "**certainty**" of our guess work!. Is this information new! Can we quantify?
    - Statistical tests and models

Intro to Statistics

- **Population:** Set of data sources (from which data is to be collected).
- **Sample** is a subset of population
- **Variable** is a data item in the sample. Variable represents any characteristics, value, or a number that can be measured or counted.
- **Statistical model** provides inference on a family of probability distributions such as mean, median, mode of a population.
- **Categories in statistics**
    - Descriptive Statistics: Helps organize data and focuses on visualizations
    - Inferential Statistics: allows you to conclude or predict about the population based on the sample data.

Mean: Average of all values

Median: Middle value of an ordered value of the sample

Range: largest – smallest

Mode: The value most recurrent in the sample.

Intro to Statistics for Data Science

Model sampling process on data not yet collected? Or already collected?

- **Exploration, Prediction, Inference**
  - Descriptive Statistics
  - Distributions
  - Hypothesis Testing
  - Regression
- **Data Modeling**
  - Bayesian Thinking vs frequentists
  - Conditional Probability
  - Prior and Posteriors
  - P-value

Company is rolling out a new product? How many stores? Where?

Experimental design

Predict individual product stockings

Regression

You are testing multiple ML models?
Can you relax your assumptions on input data?

Data Transformation

Data Models

- Bayesian Thinking vs frequentists
  - **Frequentists**: Assign probabilities for data already collected!
  - **Bayesians**: use probability to quantify uncertainty before collecting data
    - Predict likelihood of certain events occurring in the future.
    - Probabilistic models based on conditional dependencies between variables.
  - **Prior probability**: is the level of uncertainty before collecting the data
  - **Posterior probability** is the level of uncertainty after data is collected

How to Learn Statistics for Data Science, The Self-Starter Way (elitedatascience.com)

Data Collection & Modeling

- Lab experiments
- Random sample
- External sources

Why random?

How do you know soup is ready to serve?

To avoid bias and
To generalize the underlying concept

What is the good first step on data?

Best plot option for categorical data!

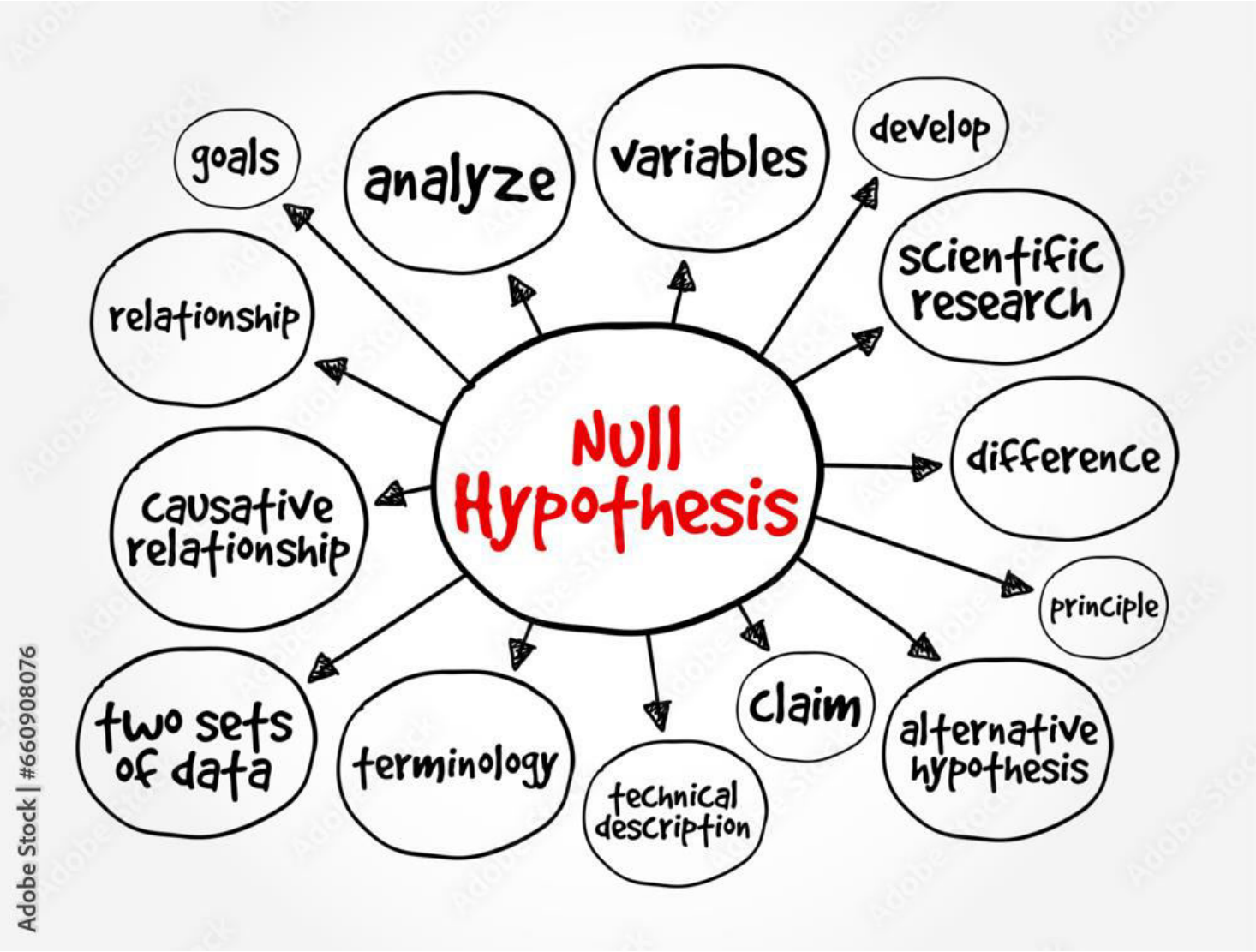
Quantitative data with 2 variables

Statistical inference: Make judgement about the population parameters!

Statistics For Data Science | Data Science Tutorial | Simplilearn - YouTube

Null Hypothesis

Null Hypothesis: The Big Picture



Hypothesis: What is it?

- A hypothesis is a supposition (or proposed explanation) of the behavior or relationship of the data based on preliminary or limited evidence.
- Additional experiments are to be conducted to validate, affirm, or negate the hypothesis.
- Radom Factors. Factors that can not be controlled.
- A plant growth doesn’t solely depend on fertilizer or soil condition only. It may depend on weather to genetics.
- Recovery from a disease may depend how prepared the patient is. Factors include immunity, stress, exercise, food habits, genetics, etc.

Null Hypothesis: What Is It?

- The change between hypothesis and alternative hypothesis is zero.

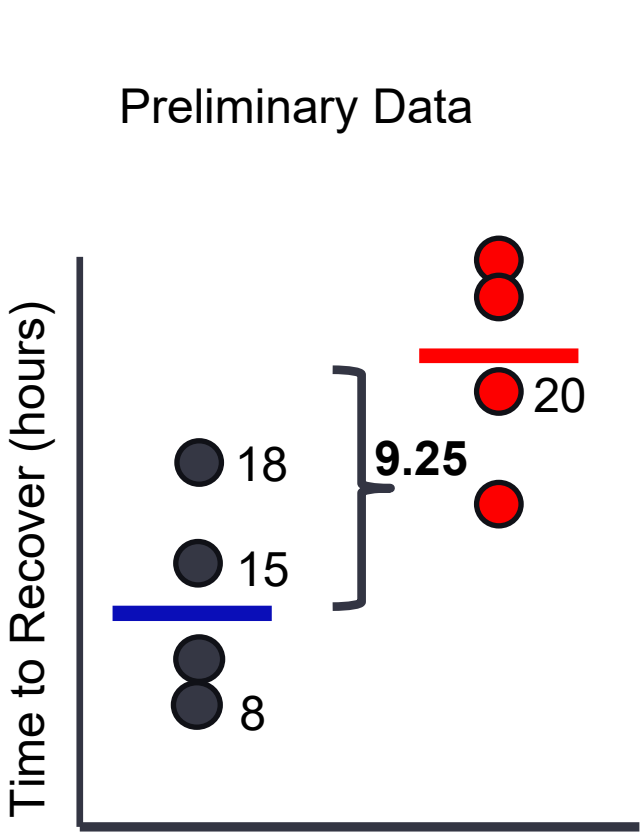
  Example:
  - Plants require fertilizer for healthier growth (Hypothesis)
  - Plants do not require fertilizer for healthier growth (Alternative Hypothesis)

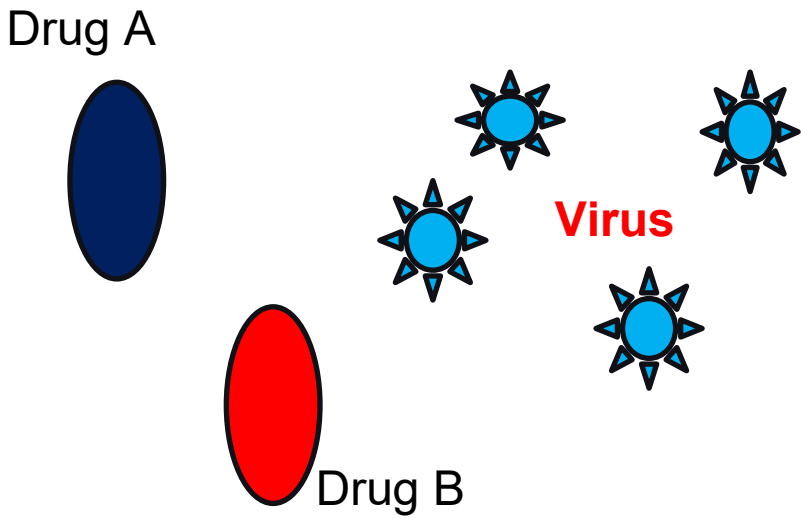
- **Run experiments** on hypothesis and its alternative hypothesis.
- If the probability distribution difference (p-value) between the two is:

P-value <= 0.05 (or 5%):
- Reject the null hypothesis. Original hypothesis may not be valid.

P-value > 0.05 then:
- Accept the null hypothesis ➔ alternative hypothesis is valid

Hypothesis

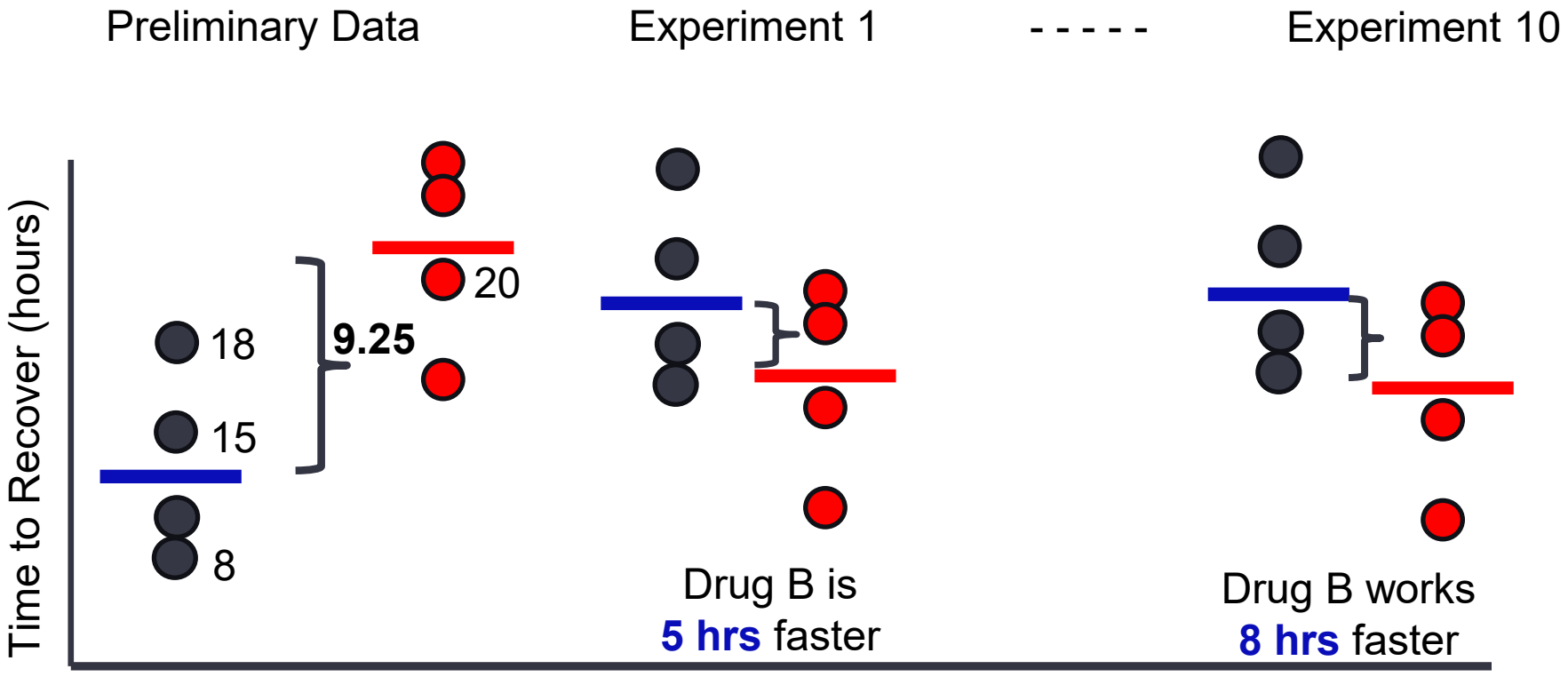




Hypothesis:

Patients tested for Drug A recovered 9.25 hours faster than those tested for Drug B

Hypothesis (Testing)



Reject Hypothesis

Hypothesis Testing in Two-Steps

- Create a distribution (H_0) showing what the statistic look like **when there is no effect!**
- Show that the statistic you observed is **unlikely** to come from the null distribution! (H_1)

Hypothesis:
Patients recovered 15 hours sooner
with Drug A than Drug B

10 experiments showed that Drug B is better than Drug A

Comparison of
Drug A and Drug B on same sample showed no difference

Random Factors that may affect outcome: Patient immunity, eating habits, stress, exercise, family history, genetics, etc.

Recipe for Null Hypothesis

Hypothesis (H_1) Application of bio-fertilizer 'x' increases plant growth.

Null Hypothesis (H_0) Application of bio-fertilizer 'x' does not increase plant growth.

Radom Factors: seed quality, wind, sun light, environment, etc.

Validation

- Conduct experiments
- Check H_1 for each experiment
- Does the experiment contradict with the null hypothesis?
- **Yes:** Reject Hypothesis
- **No:** Fail to reject the hypothesis

- A p-value is a measure of probability that an observed difference could have occurred just by random chance.
- Lower the p-value, greater the statistical significance of the observed difference.

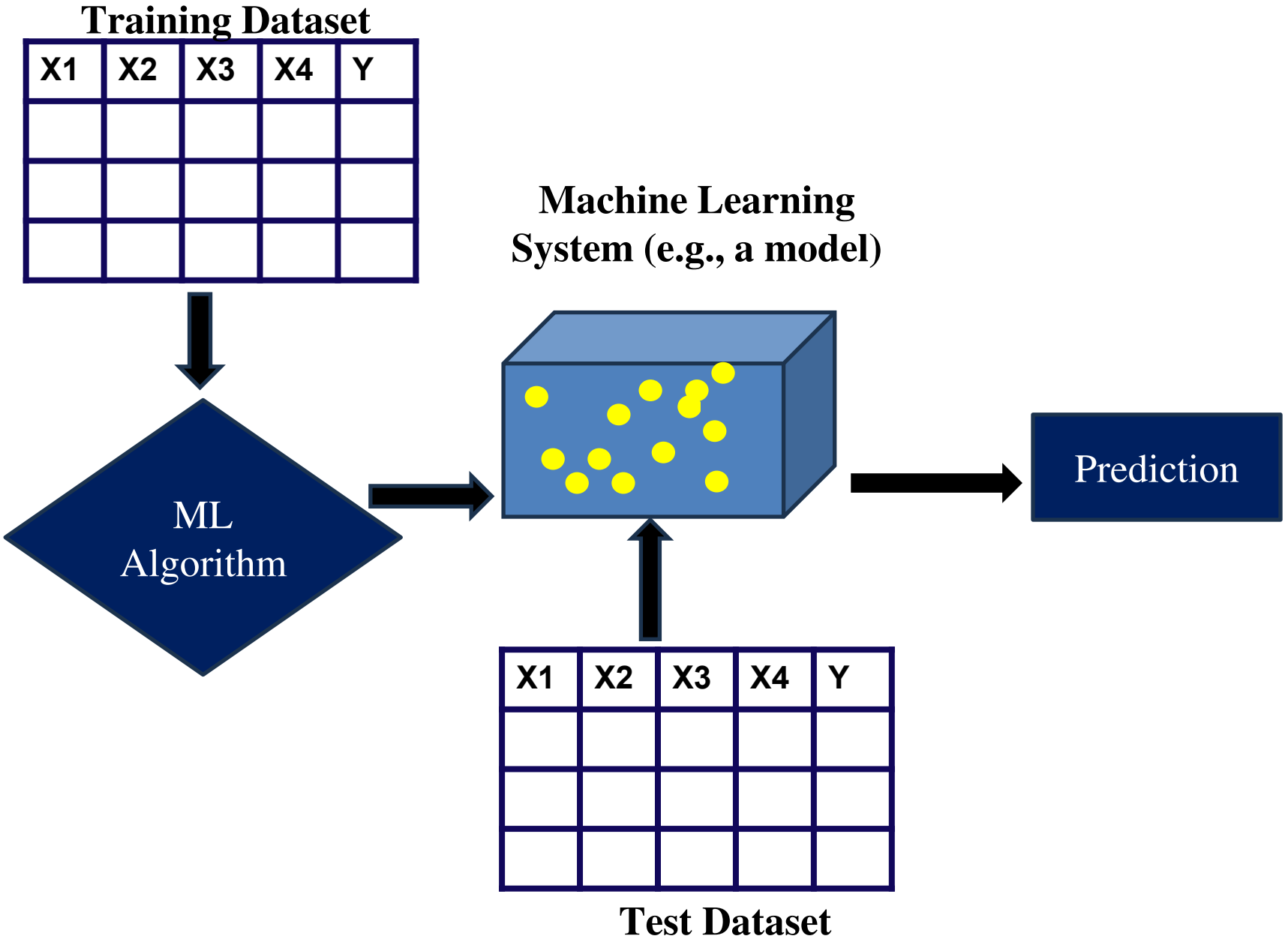
p-value <= 0.05
Not statistically significant.
Accept Null Hypothesis

p-value > 0.05
Statistical significance.
Reject Null Hypothesis

Introduction to Machine Learning

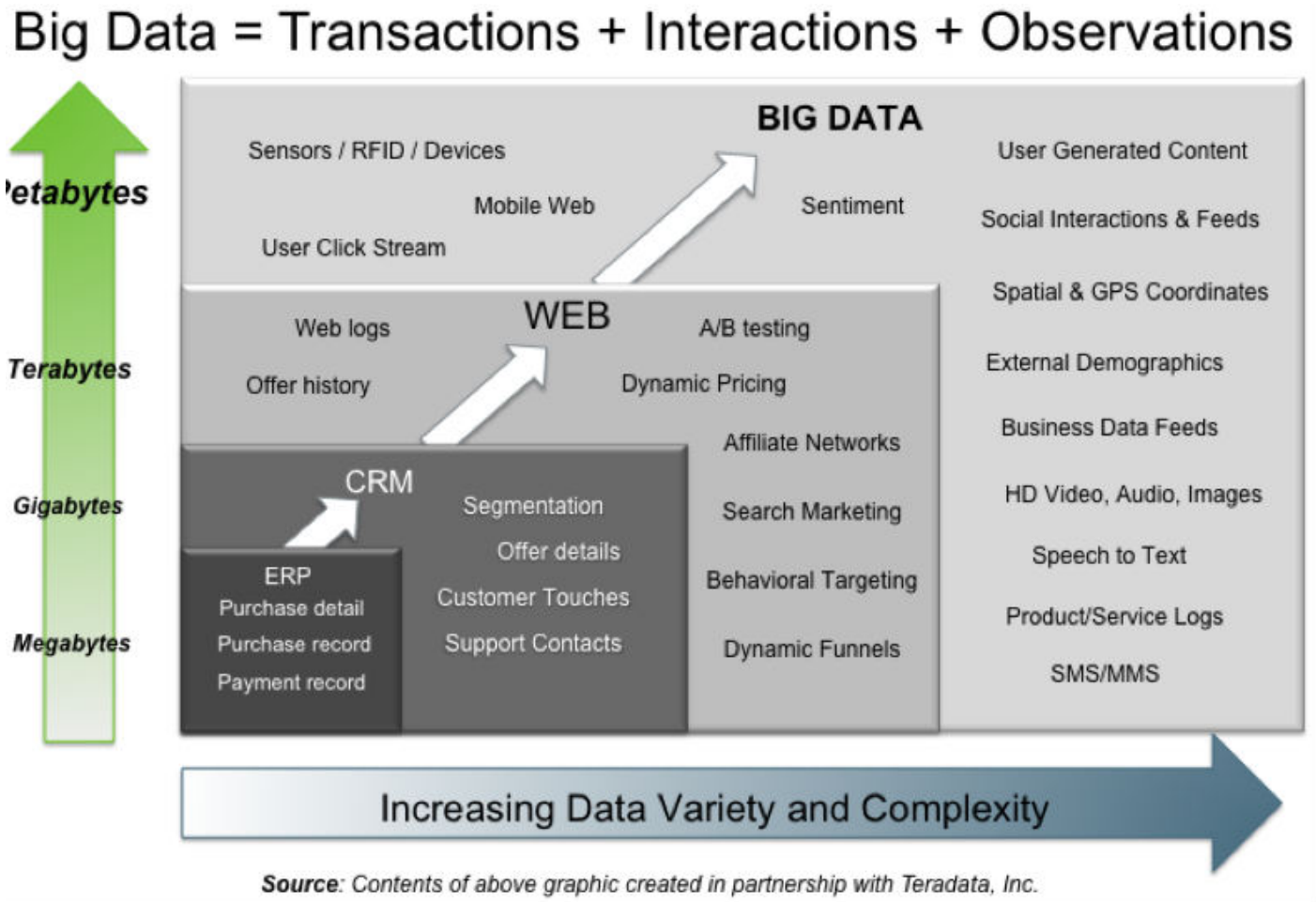
- What is Machine Learning?
- Pros and Cons
- Algorithms and Scenarios
- How to Get Started?

Machine Learning Workflow



What is Machine Learning?

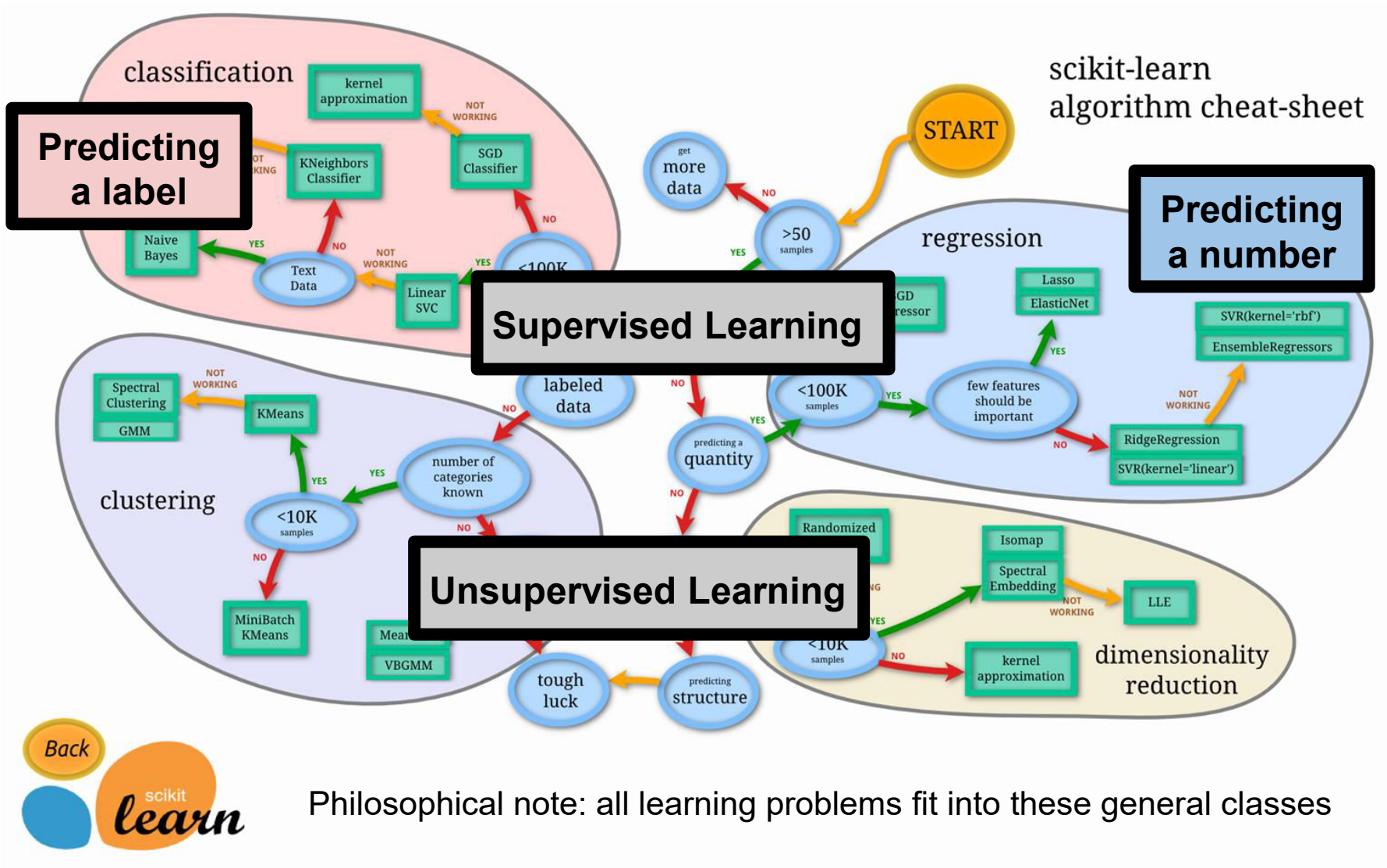
- Machine Learning is a process of learning from Data
- What can we learn from Big Data? How to extend KB?



***Source**: Contents of above graphic created in partnership with Teradata, Inc.*

Courtesy Dr. Jin, Kent State.

Machine Learning Models

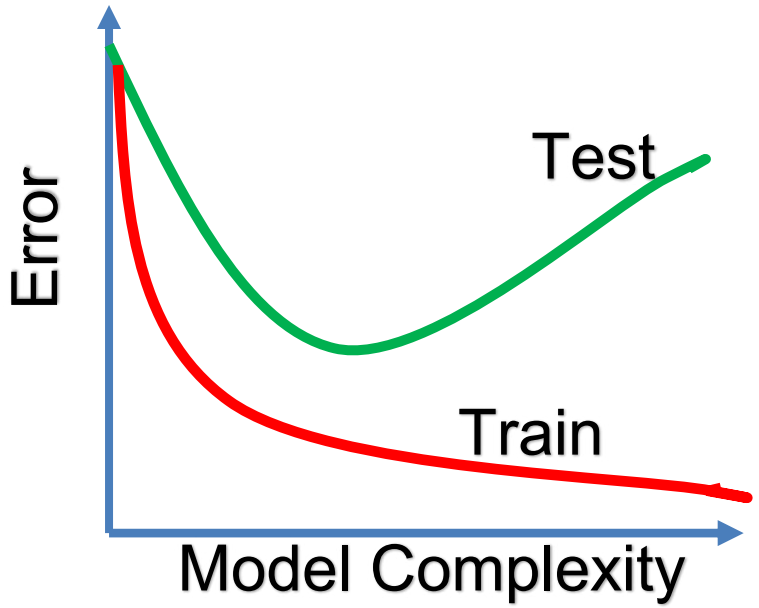


Philosophical note: all learning problems fit into these general classes

Image provided by Scikit Learn

Recipe for Machine Learning

- Training Data: Train the model
- Test Data: Test the model performance
- Accuracy?
- Doesn't work, train again



Think abouts?

- Can I use all data for training?
- Best split for train and test dataset?
- What is the goodness of a fit? And how do I measure it?
- What if the fit is not good or acceptable?

Terminology

- **Hyperparameters**: Parameters that are used to control the learning process of a machine learning algorithm
- **Underfit**: Simple fit that may lead to lower correction
- **Overfit:** Complex fit that may lead to over correction
- **Feature Selection**: aka hyperparameter
- **Outliers**: point or group of points that don't follow the trend
- **Curse of Dimensionality**: Parameter-to-data proportionality or lack there of.

Machine Learning Algorithms

Machine Learning algorithms are applied to predict a class or label (classification) or a number (regression)

Polynomial Regression (relationship between dependent and independent variables)

Logistic Regression (categorical or label-based classifier)

K-Nearest Neighbor (distance-based classification)

Big Data →Data Science

- Autonomous Cars → Self-driving capability
- Electronic Records → Text mining, Speech Recognition
- Stock (SKUs, manufacturing, etc.) → Business Intel.
- Population → Precision Medicine, Cost of Care, etc.
- Infrastructure → Smart * (*communities, cities, etc.)
- So on..

Common Attributes

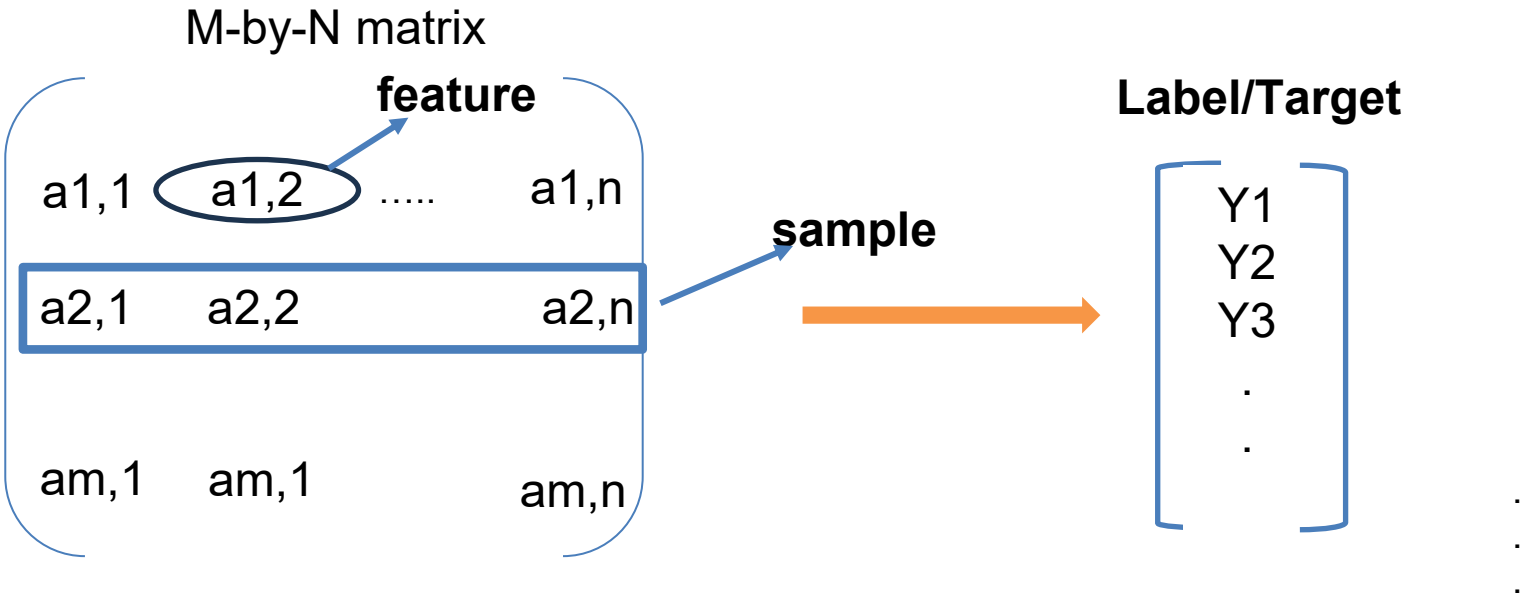
- Big Data: Volume, Variety, Velocity (3Vs)
- Features: Curse of Dimensionality
- Feature Selection/Engineering: Machine Learning
- Data types: numbers, audio/video, images, strings, etc.

What's in It for ME?

Data Intelligence

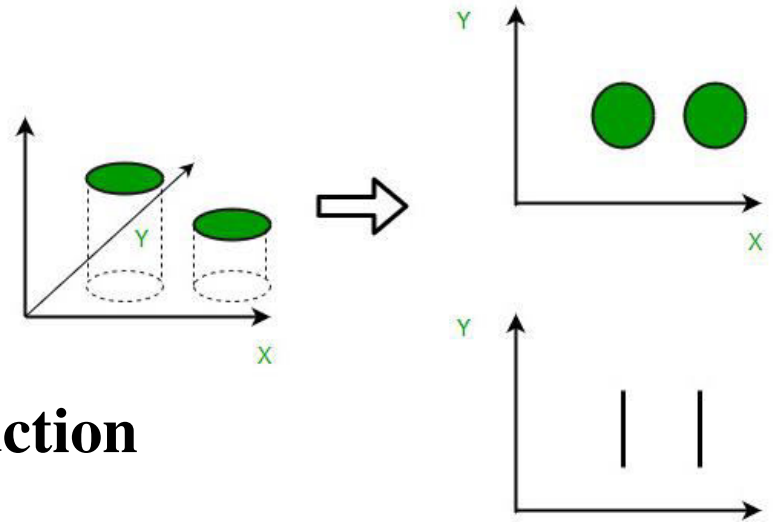
Computing for Machine Learning

- **Rows** – Instance (aka sample) of your data (SKU in supply chain, people in population, image in images, etc.)
- **Columns** - features (observation or attribute) of your data (SKUs: how many SKUs, what type of SKUs, People: sex, disease, height)
- **Target** - what you are trying to get your system to predict (business intel, cost of care, etc.)



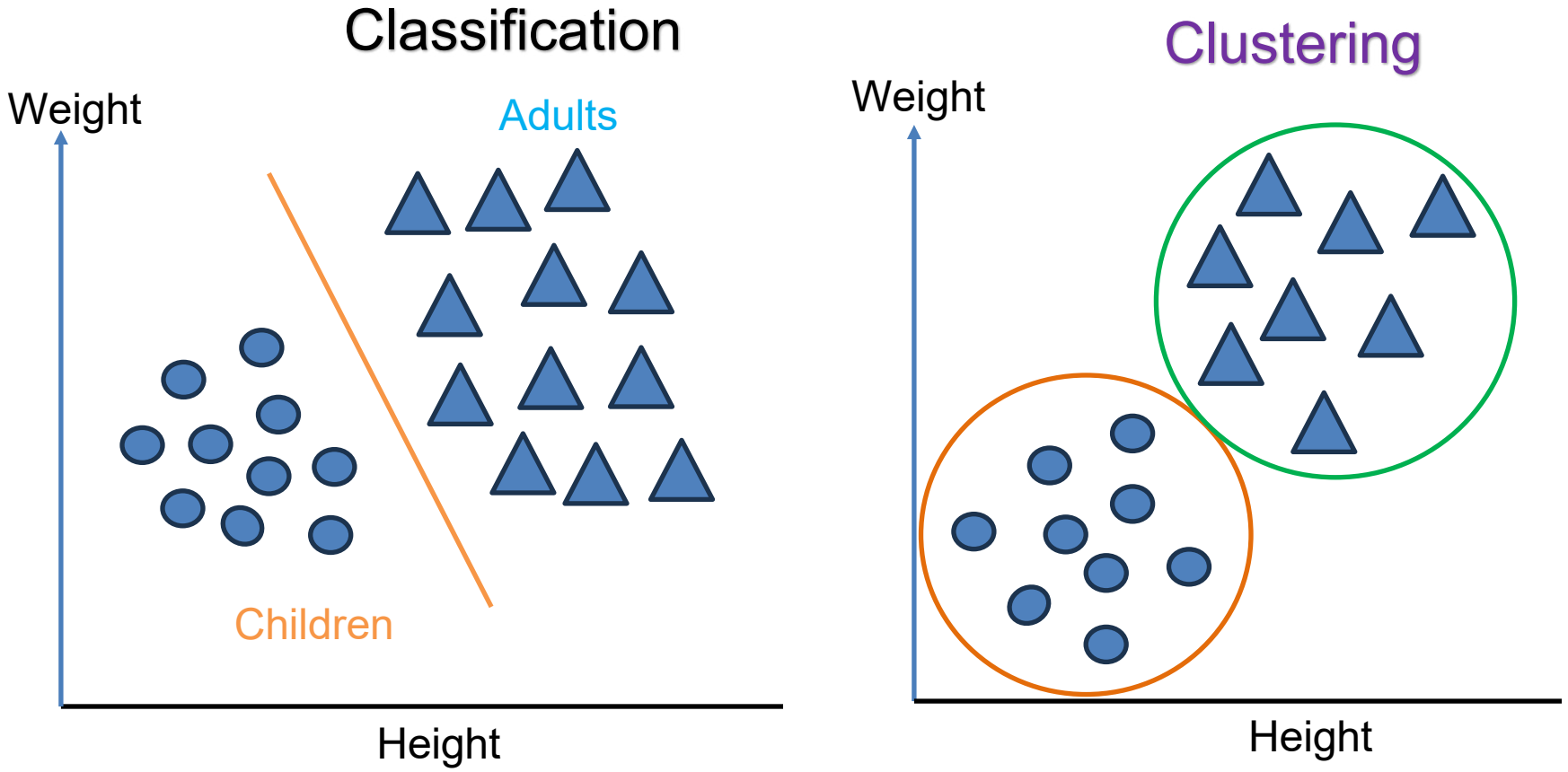
Data Intelligence

- Pick good features (by hand)
- Find more data (on chosen features)
- Improve Data Intelligence
  - Extract meaningful relationships between data (sample → features)
  - **Dimensionality reduction** (can fewer features → same outcome)
  - Clustering (group related data)
    - **Naïve (Bayes) clustering** group the data based one feature
    - **Better clustering** group the data based on more features



Dimensionality Reduction

Classification Vs. Clustering



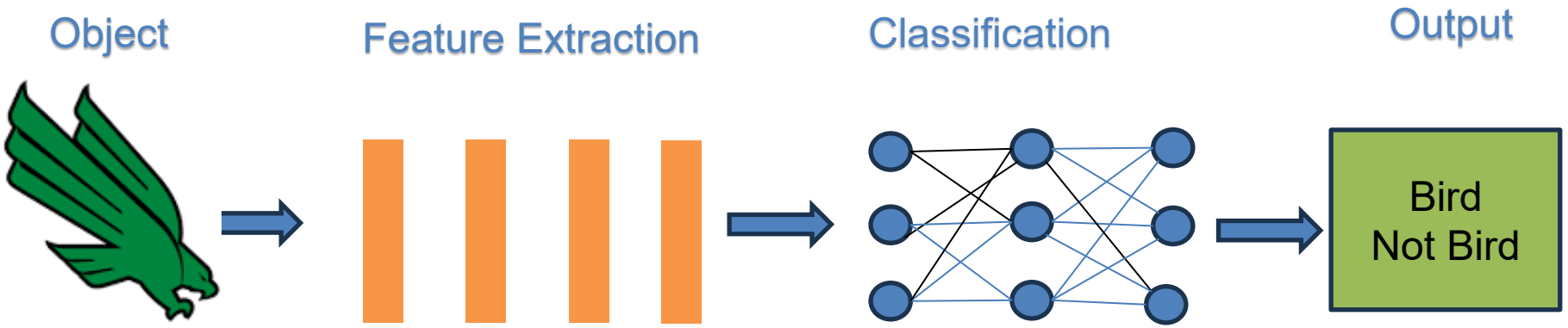
Classification groups targets of a prediction (aka label data): Examples include adult, child, types of infections, etc.

Clustering groups similar instances together (sample data): Examples include behaviors, types of infections specific to child or adult, etc.

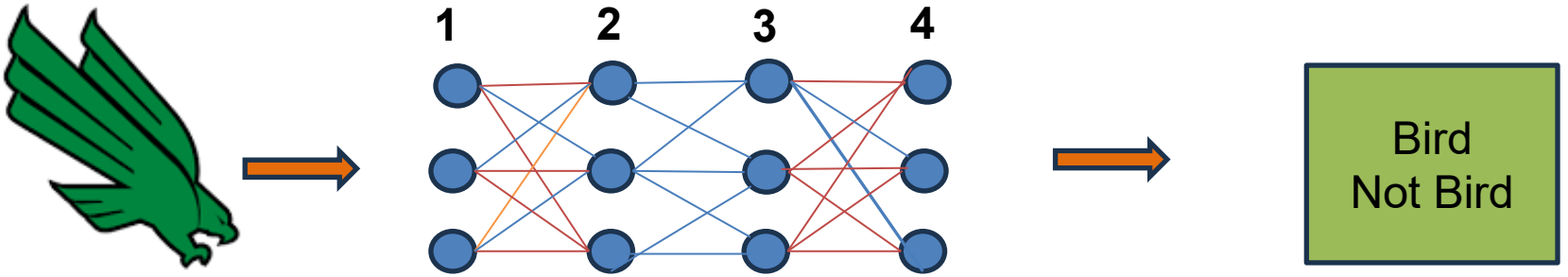
Feature Engineering

- Feature engineering involves creating more or better features from raw data.
- Assists in dimensionality reduction (e.g., Principal Component Analysis)
- BMI is good enough to replace two features: height and weight.
- New features may be a result of intuition, data knowledge, or research.

Machine Learning



Deep Learning



Deep Learning ➔ ("deep" = multilevel neural network)

Requires massive amounts of data and employs hidden variables (see network segments 2 and 3 above)

Recipe for Machine Learning

Step 1: Divide data into train and test datasets

Step 2: Feature Selection. Explore dimensionality reduction and feature engineering strategies

Step 3: Divide data into training and test sets

Step 4: Apply machine learning models classification (for label data), regression (for numerical data).

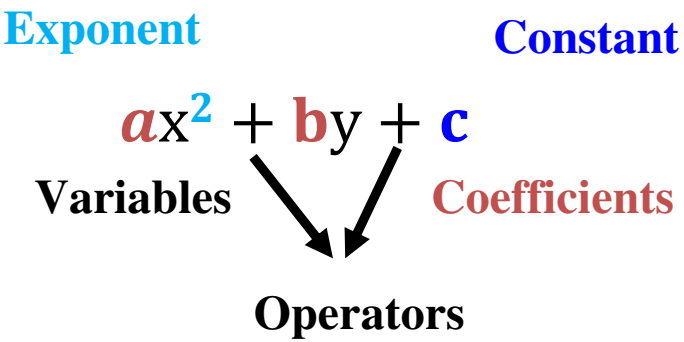
Step 5: Validate the model accuracies on test data

Step 6: If model accuracy is not acceptable go to STEP 3

Polynomial Regression

What is a Polynomial?

- A polynomial is an algebraic expression
- An equation representing relationship between variables
- Written as a mathematical expression of variables, constants, and exponents .
- Number of roots (or solutions) is less than or equal to the highest power of a variable.
- Assists in predicting outcomes



What is not a polynomial?

An algebraic expression that contains

- fractional exponents
- negative exponents (example: $3X - 4X^{-2}$)
- Division by a variable (example: $3/x + 4X^2$)
- Radicals (an integer under * root, * square, cube, etc.)

N-th degree Polynomial

$$Y = a_0 + a_1 * x + a_2 * x^2 + a_3 * x^3 + \ldots + a_n x^n$$

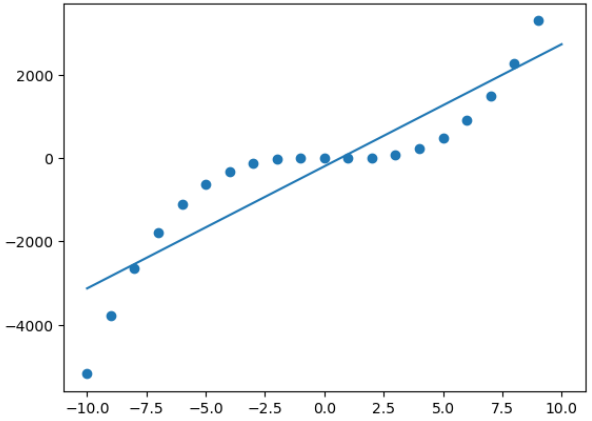
The power "n" of the polynomial Y is the degree of the polynomial

n = 1 (linear):	$Y = a_0 + a_1 * x$
n = 2 (quadratic):	$Y = a_0 + a_1 * x + a_2 * x^2$
n = 3 (cubic):	$Y = a_0 + a_1 * x + a_2 * x^2 + a_3 * x^3$
…	

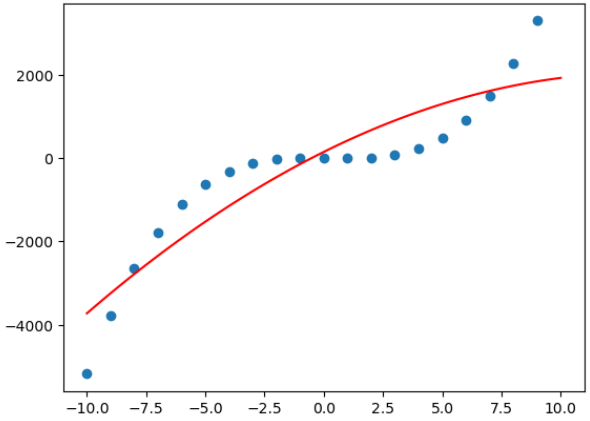
Goal: pick the **appropriate n** to fit the data.

Accuracy of a Fit

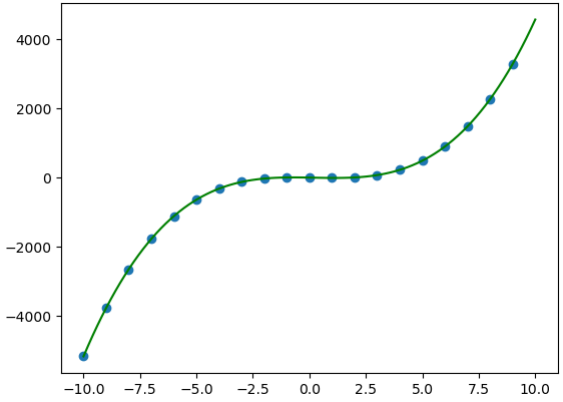
$$Y = 5x^3 - 3x^2 - 12x$$



Degree n = 1



Degree n = 2



Degree n = 3

MSE (Mean Squared Error). The average squared difference between predicted and actual values

RMSE (Root Mean Squared Error) Square root of MSE. RMSE is more commonly used because its in the same units as your prediction.

Mean Square Error Computation

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \tilde{y}_i)^2$$

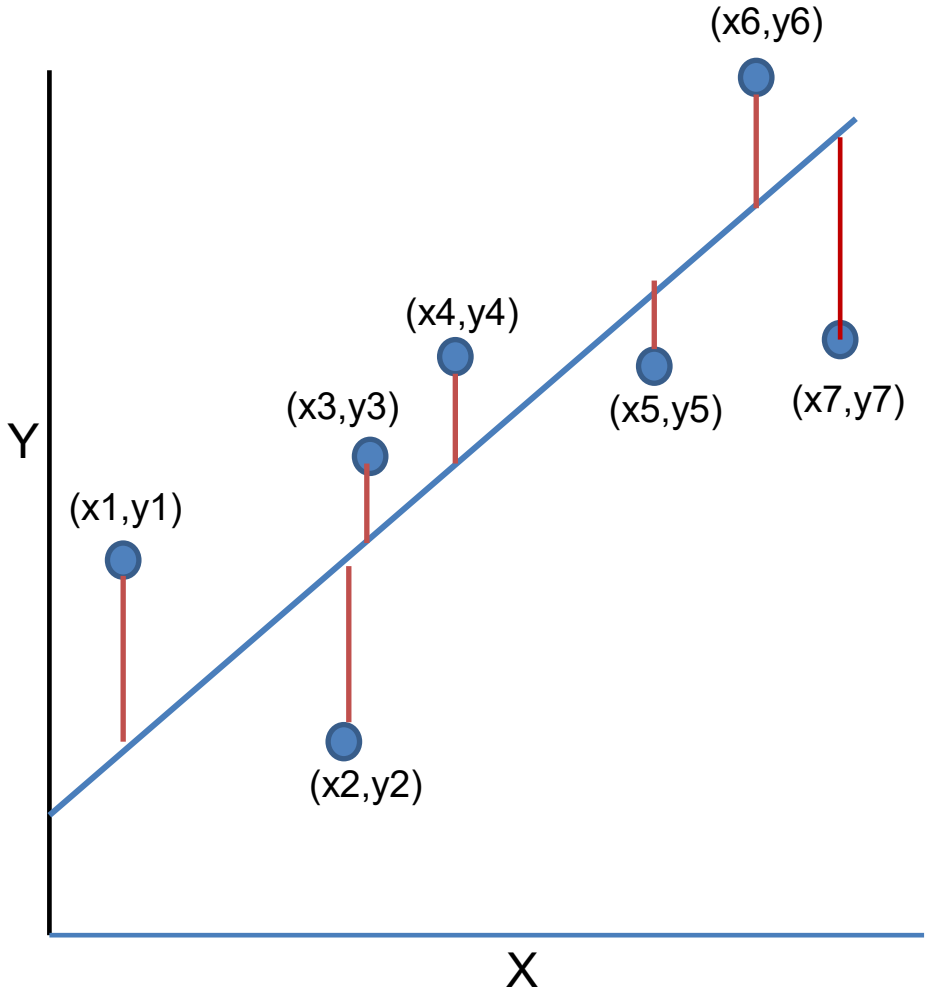
RMSE = Sqrt (MSE)

Intercept: b

$$b = \overline{y} - m\overline{x}$$

Slope: m

$$m = \frac{\overline{xy} - \overline{x}\,\overline{y}}{\overline{x^2} - (\overline{x})^2}$$



Average of terms and cross-terms

$$\frac{x_1 + x_2 + \ldots + x_n}{n} = \overline{x}$$

$$\frac{x_1^2 + x_2^2 + \ldots + x_n^2}{n} = \overline{x^2}$$

$$\frac{y_1 + y_2 + \ldots + y_n}{n} = \overline{y}$$

$$\frac{y_1^2 + y_2^2 + \ldots + y_n^2}{n} = \overline{y^2}$$

$$\frac{x_1 y_1 + x_2 y_2 + \ldots + x_n y_n}{n} = \overline{xy}$$

Tutorials

Polynomial Regression

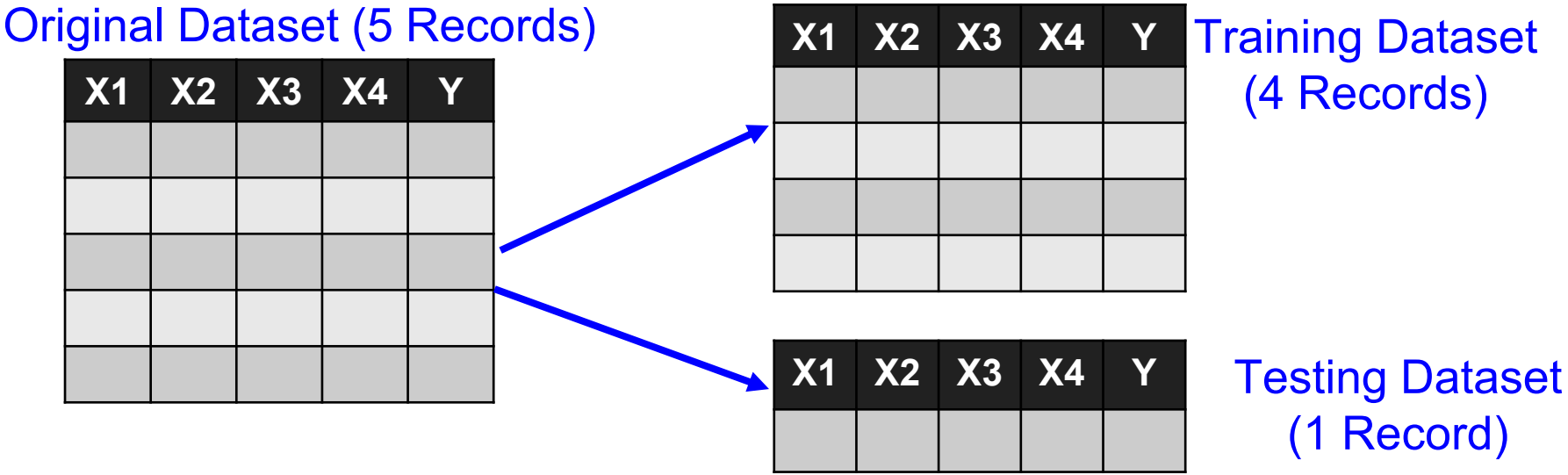
Logistic Regression

Logistic Regression (A Linear Regression Technique)

- **Classification** Method
- Statistical method for predicting binary classes
- Outcome is always one of the two classes
- Examples: 0/1, yes/no, T/F, etc.
- Calculates the probability of event occurance
- Example: Healthcare decisions
- A special case of Linear Regression where target variable is catagorical in nature

Logistic Regression Model Evaluation

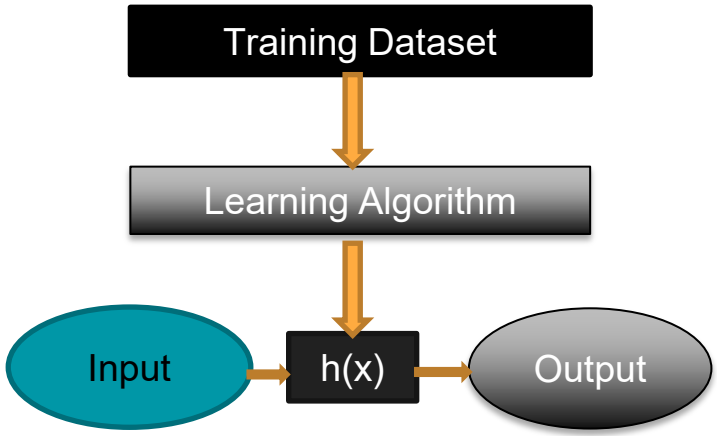
- Divide data into features (X1, X2, …) and target (Y)
- Divide the data into train (X_train, y_train) and test datasets (X_test, y_test)



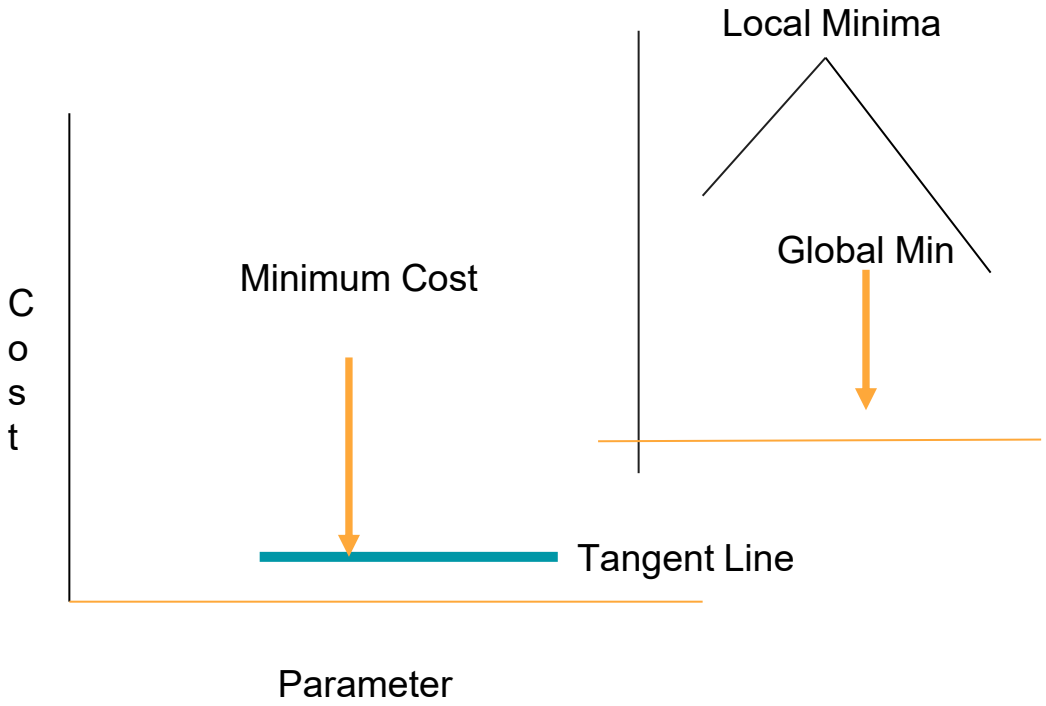
X_train, X_test, y_train, y_test = train_test_split(X, y, **test_size=0.2**)

Logistic Regression Workflow

Find the best parameter that gives least error in predicting the output



h: cost function



Logistic Regression Algorithms

- Quadratic (n=2) or higher order n>2 functions best represent the cost function
- How many derivatives?
- How do we get to the tangent line?

Newton Method

- Computes 2nd derivatives
- Stores all updates
- Computationally Expensive
- Memory intensive

Libfgs

- Approximates 2nd derivative
- Stores last few updates only and therefore, saves memory.
- Superfast with small datasets

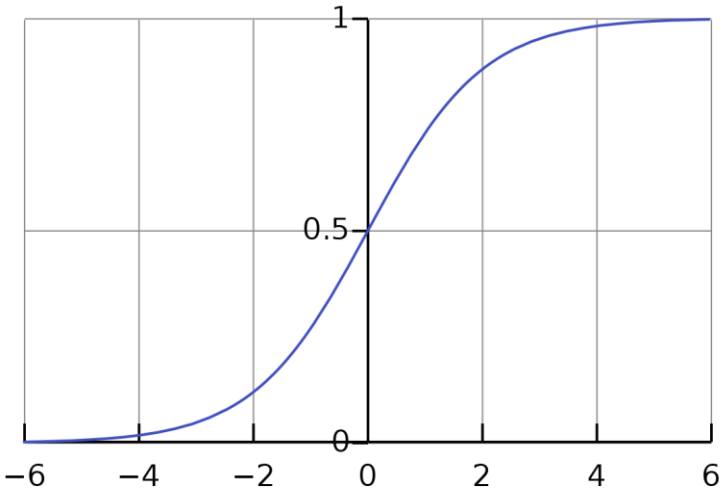
LibLinear

- Library of large Linear Classifiers
- Minimize multivariate function by solving iteratively
- Start at a point, move towards minimum, one at a time.
- Can't run in parallel / multicore processors

Logistic + Polynomial for Better Regression

- Logistic function takes in any value and outputs is in the interval [0,1].
- Sigmoid function Φ(x)
- How to represent continuous data?
- Overlay polynomials

$$\phi(z) = \frac{1}{1 + e^{-z}}$$



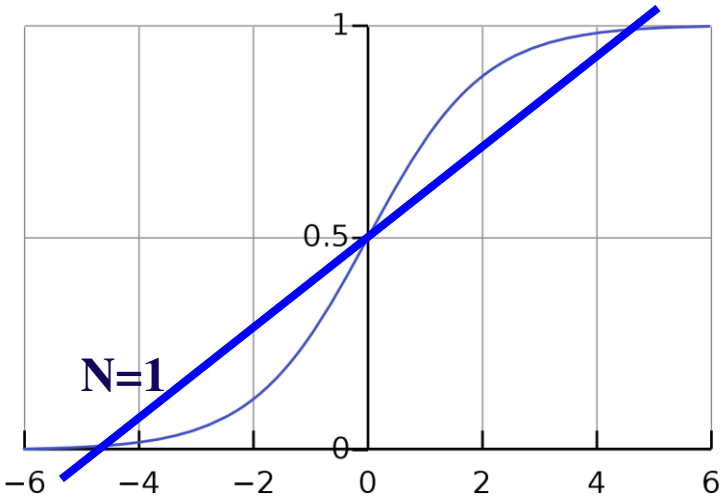
HOW?

What would be z = ? for a degree=3 polynomial?

Polynomial represents Logistic Regression

- Substitute $z = b_0 + b_1 x$ into sigmoid function as $\phi(z)$.

$$\phi(z) = \frac{1}{1 + e^{-z}}$$



Logistic Model with Polynomial of degree=1

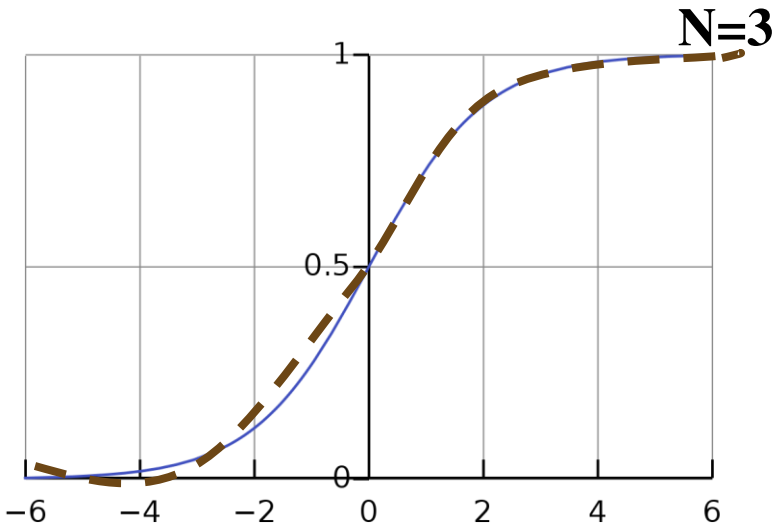
$$\Phi(z) = \frac{1}{1 + e^{-(b_0 + b_1 x)}}$$

Polynomial represents Logistic Regression

What would be z = ? for a degree=3 polynomial?

Polynomial of degree=3

$b_0x^3 + b_1x^2 + b_2x + b_3$



$$\phi(z) = \frac{1}{1 + e^{-z}}$$

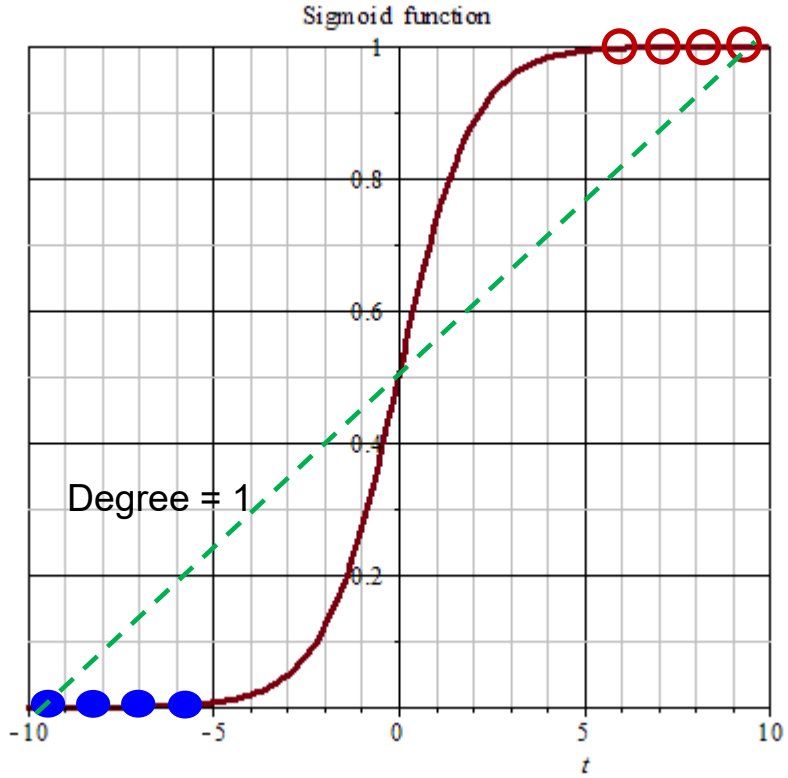
SkLearn-Logistic Regression

LogisticRegression(*penalty='l2', *, dual=False, tol=0.0001, C=1.0, fit_intercept=True, intercept_scaling=1, class_weight=None, random_state=None, solver='lbfgs', max_iter=100, multi_class='auto', verbose=0, warm_start=False, n_jobs=None, l1_ratio=None*)

- Penalty. Increasing parameter weights versus over/underfitting
- tol. Tolerance of the fit (stopping criteria for the model)
- C. Inverse regularization strength. Smaller the value, larger is the penalty
- fit_intercept. Bias
- Solver. lbfgs, liblinear, and newton-CG

```
model = LogisticRegression(solver='liblinear', C=10.0, random_state=0).fit(x,y)
```

Linear Fit? For Logistic Regression



Methods available in Logistic Regression

decision_function(X)	Predict confidence scores for samples.
densify()	Convert coefficient matrix to dense array format.
fit(X, y[, sample_weight])	Fit the model according to the given training data.
get_params([deep])	Get parameters for this estimator.
predict(X)	Predict class labels for samples in X.
predict_log_proba(X)	Predict logarithm of probability estimates.
predict_proba(X)	Probability estimates.
score(X, y[, sample_weight])	Return the mean accuracy on the given test data and labels.
set_params(**params)	Set the parameters of this estimator.
sparsify()	Convert coefficient matrix to sparse format.

Confusion Matrix

n=165	Predicted: NO	Predicted: YES	Reality
Actual: NO	TN = 50	FP = 10	60
Actual: YES	FN = 5	TP = 100	105
	55	110	

Basic Terminology:

- True Positives (TP). Cases in which predicted “yes” and actually “yes”
- True Negatives (TN). Cases in which predicted “no” and actually “no”
- False Positives (FP). Predicted “yes”, actually “no”: Type I error
- False Negatives (FN). Predicted “no”, actually “yes”: Type II error

Accuracy of the Model

n=165	**Predicted: NO**	**Predicted: YES**	
Actual: NO	TN = 50	FP = 10	60
Actual: YES	FN = 5	TP = 100	105
	55	110	

Accuracy:

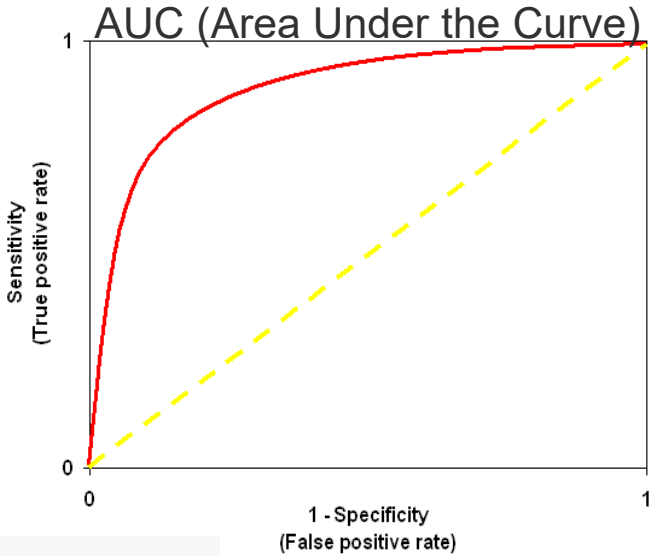
- Overall, how often is it **correct**?
- (TP + TN) / total = 150/165 = 0.91

Misclassification Rate (Error Rate):

- Overall, how often is it **wrong**?
- (FP + FN) / total = 15/165 = 0.09

Metrics for Classification Models:

- Area Under Curve (AUC). How well the model fits the data.



FPR: False Positive Rate
TPR: True Positive Rate

```
#AUC and ROC computation
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.metrics import roc_auc_score

X_train, X_test, y_train, y_test = train_test_split(x,y,test_size=0.2, random_state=0)
y_pred = model.predict_proba(X_test)[::,1]
auc = roc_auc_score(y_test, y_pred, multi_class='ovr')
print (auc)
```

```
0.4492753623188406
```

Receiver Operating Characteristic (ROC) curve

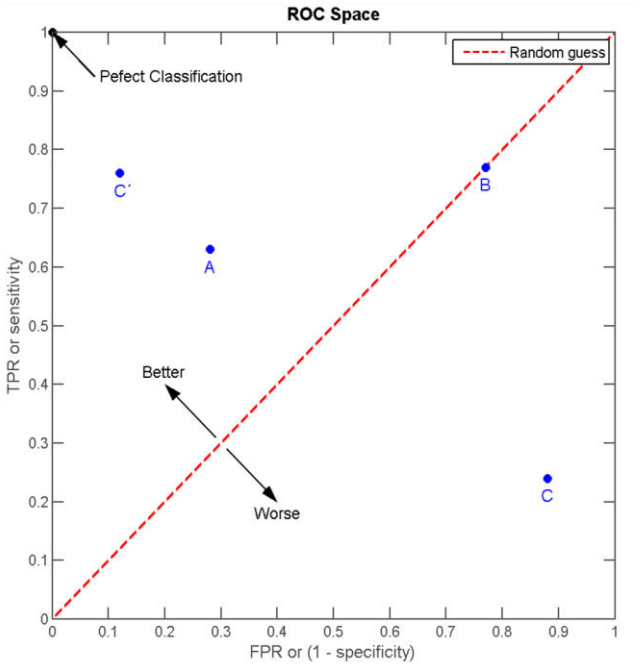
- ROC is a plot of sensitivity vs FPR (1-specificity)
- ROC is for tests which produce results on a numerical scale, rather than binary (+ vs - results)
- Helps to generate decision threshold or cut off values, confidence intervals for sensitivity and specificity and likelihood ratios.

Pros:

- The closer the apex of the curve toward the upper left corner, the greater discriminatory ability of the test
- Simple method for determining optimal cut off values

Cons:

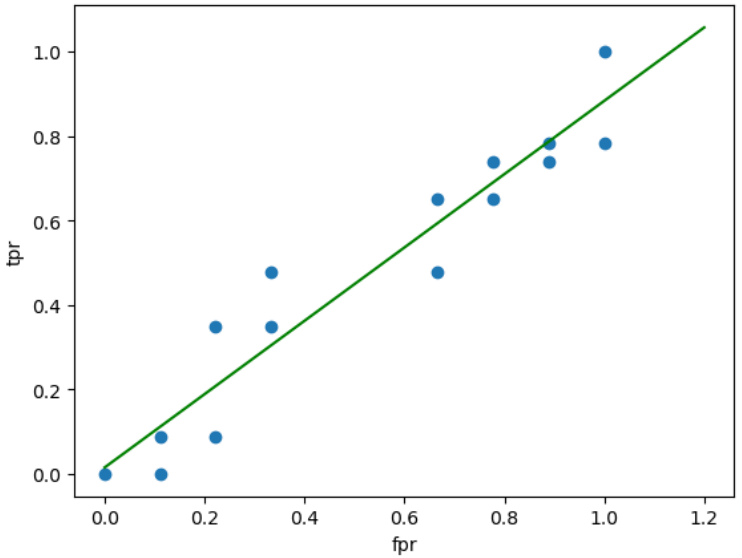
- As the sample size decreases, the plot becomes more jagged
- ROC calculation is cumbersome and needs specialized software



ROC for Logistic Reg on Binary Digits

```
#AUC and ROC computation
from sklearn.metrics import roc_curve, roc_auc_score
x = np.arange(159).reshape(-1,1)
y = np.array([0,0,0,0,1,1,1,1,1,1,0,0,0,0,1,1,1,1,1,1,
              0,0,0,0,1,1,1,1,1,1, 0,0,0,0,1,1,1,1,1,1,
             .....])
X_train, X_test, y_train, y_test = train_test_split(x,y,test_size=0.2, random_state=0)
y_pred = model.predict_proba(X_test)[::,1]
auc = roc_auc_score(y_test, y_pred, multi_class='ovr')
#the following will be run once for each algorithm
fpr, tpr, threshold = roc_curve(y_test, model.predict_proba(X_test)[:,1])
print ('roc info for logistic regression', fpr, tpr, threshold)
```

```
#plot ROC
my_polyfit = np.poly1d(np.polyfit(fpr,tpr,1))
my_line = np.linspace(0,1.2,20)
plt.scatter(fpr,tpr)
plt.plot(my_line, my_polyfit(my_line),'g')
```



Accuracy and Classification Report

```
#Measure Model performance accuracy = correct predictions/total data
from sklearn.metrics import classification_report
score = model.score(x_test,y_test)
print (score)
print(classification_report(y_test,y_predict))
```

```
0.5333333333333333
              precision    recall  f1-score   support

           0       0.48      0.71      0.57        14
           1       0.56      0.31      0.40        16

    accuracy                           0.50        30
   macro avg       0.52      0.51      0.49        30
weighted avg       0.52      0.50      0.48        30
```

$$\text{Accuracy} = \frac{\Sigma\ TP + \Sigma\ TN}{\Sigma\ \text{total population}}$$

Score returns mean accuracy on the test data

Score = TP/Total Predictions

Precision = TP/(TP+FP)

Recall = TP / (TP+FN)

f1-score = 2*(Recall*Precision)/(Recall + Precision)

Precision: What percentage of your predictions are correct?

Recall: Percentage of positive cases you caught (#of positive cases correctly identified

f1-score: what percentage of positive predictions are correct

Support: # of actual occurrences of the class in the specified dataset

Key Terms

Accuracy Score. Measure the correct prediction of the classifier across the sample.

AUC-ROC curve. Visualize performance of the classification model based on its rate of correct/incorrect classifications. Key classifier of all models

Precision, Recall, and F1 score. Metrics from confusion matrix based on TRUE and FALSE classifications. (See Classification_Report)

- **Precision**: What percentage of your predictions are correct
- **Recall**: Percentage of positive cases you caught (#of positive cases correctly identified
- **f1-score**: what percentage of positive predictions are correct
- **Support**: # of actual occurrences of the class in the specified dataset

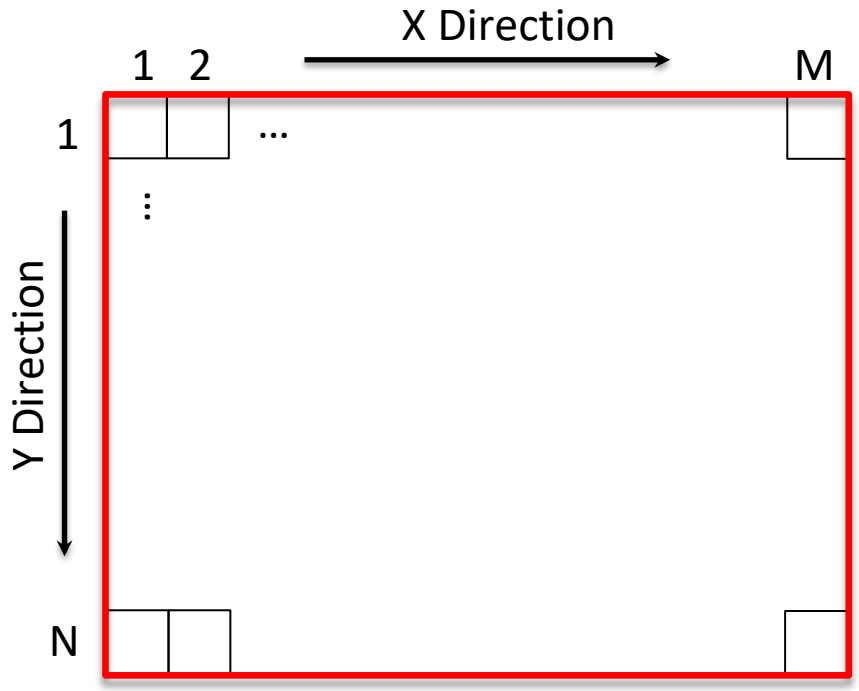
Image Classification

```
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import numpy as np
from PIL import Image
```

Pixels

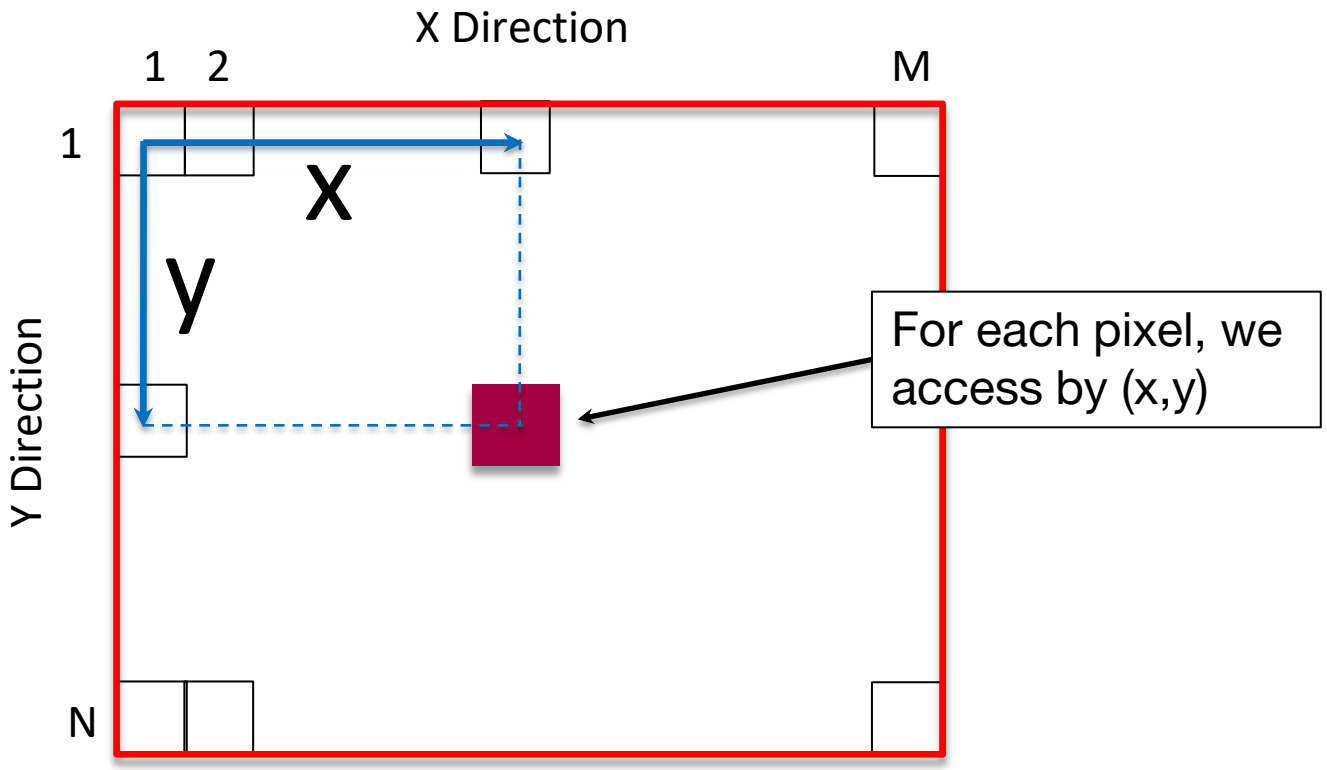
- A pixel is the smallest little square that can be displayed on the screen or monitor.
- A (raster-based) image is defined by an array of pixels with dimensions of width and height.
- For example:

  M x N pixels



Accessing each pixel

```
import matplotlib.image as mpimg
img = mpimg.imread('sample_image.jpg')
pixel = img[x,y]
```



Pixel values (grayscale)

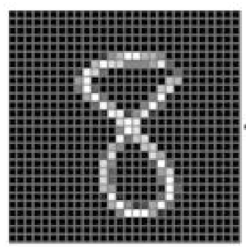
- A grayscale image of M x N pixels is represented by an M x N array, each element representing the brightness of the pixel.

- Pixel values:

  Black: 0

  White: 255

  Gray: in between



28 x 28
784 pixels

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	4	62	146	182	254	254	181	176	139	15	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	34	186	253	217	208	136	136	136	166	232	99	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	61	242	208	111	3	0	0	0	0	0	18	32	107	43	0	0	0	0	0	0
0	0	0	0	0	0	0	0	156	242	23	0	0	0	0	0	0	0	13	191	181	6	0	0	0	0	0	0
0	0	0	0	0	0	0	0	121	255	98	3	0	0	0	0	0	8	194	225	12	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	169	253	120	3	0	0	0	0	128	247	51	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	3	111	244	169	19	0	14	131	249	117	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	59	241	235	72	142	229	66	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	25	218	254	231	36	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	133	253	221	33	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	19	237	111	196	217	19	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	174	138	0	23	193	204	18	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	96	224	0	0	0	25	218	169	3	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	215	138	0	0	0	0	86	253	99	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	215	97	0	0	0	0	3	162	214	11	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	215	97	0	0	0	0	0	118	253	68	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	185	157	0	0	0	0	0	40	254	98	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	50	244	61	0	0	0	0	112	244	58	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	174	251	142	59	83	167	244	111	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	6	133	253	253	253	169	61	3	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Pixel values (color)

- A color image is represented by an M x N x 3 array.

  - Three components
    - M = {R, G, B}

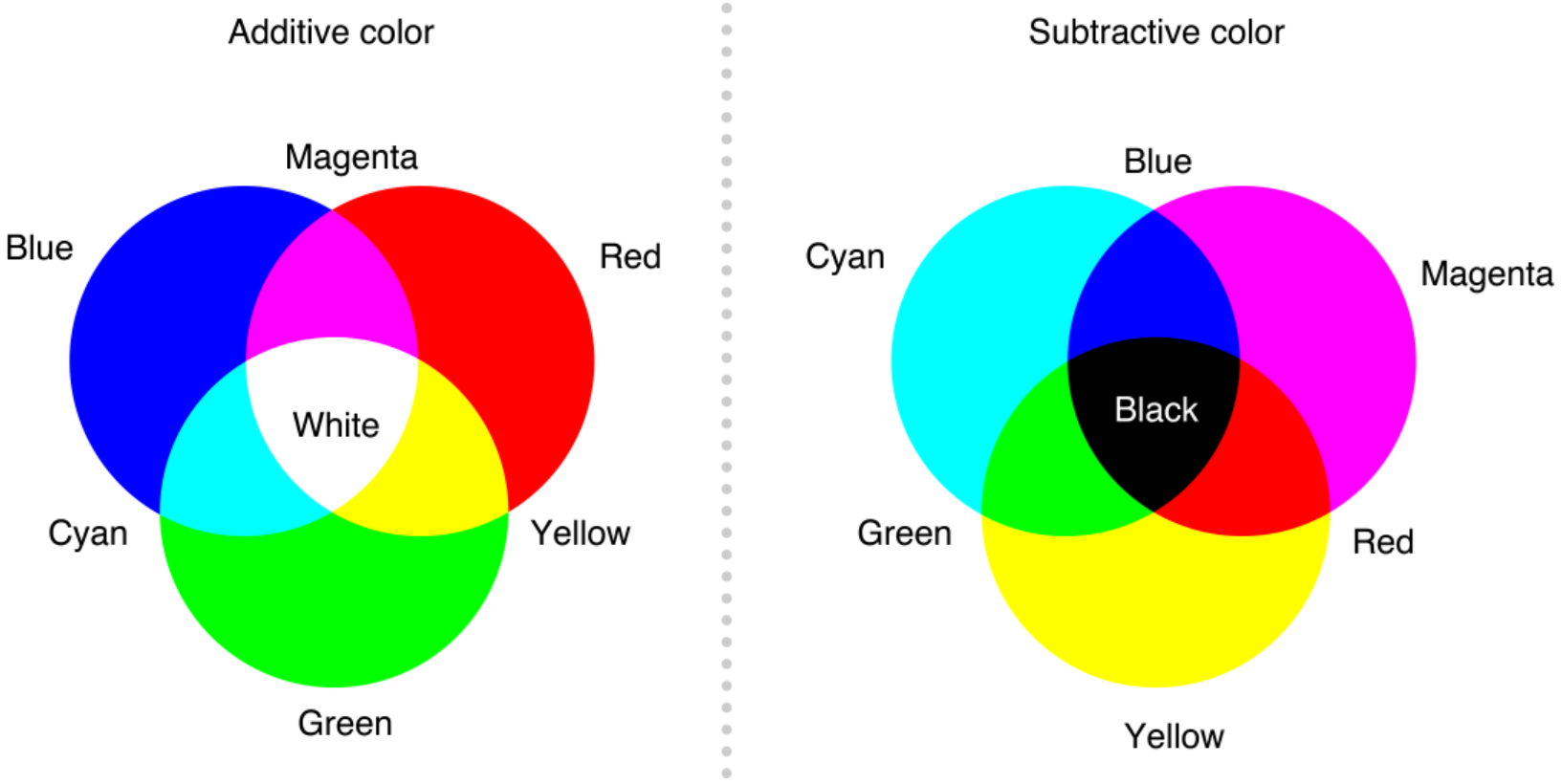
$$R = \begin{bmatrix} 73 & \cdots & 87 \\ \vdots & \ddots & \vdots \\ 27 & \cdots & 17 \end{bmatrix}, G = \begin{bmatrix} 66 & \cdots & 98 \\ \vdots & \ddots & \vdots \\ 36 & \cdots & 13 \end{bmatrix}, B = \begin{bmatrix} 31 & \cdots & 61 \\ \vdots & \ddots & \vdots \\ 36 & \cdots & 14 \end{bmatrix}$$

Pixel values (color + transparency)

- Some types of images are represented by M x N x 4 arrays, where the 4th number (alpha) defines the amount of transparency in the images.

Additive vs subtractive colors

- RGB values in defining pictures are additive colors, which behave differently in combination than mixing the same colors with paints (subtractive colors).

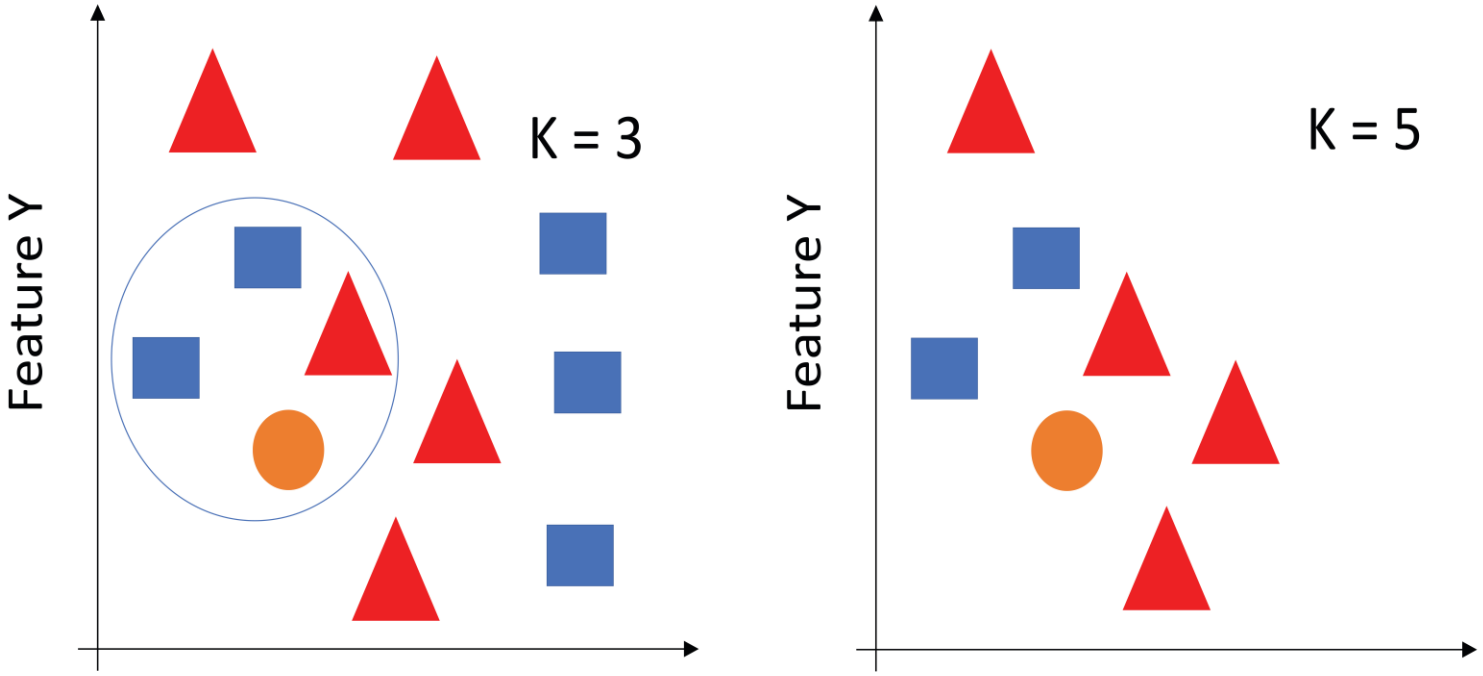


K-Nearest Neighbor Classifier

- **Bayes Classifier:**
- Error rate is minimized.
- Assigns observations to a class based on predictor values.
- Impossible to know clear boundary!

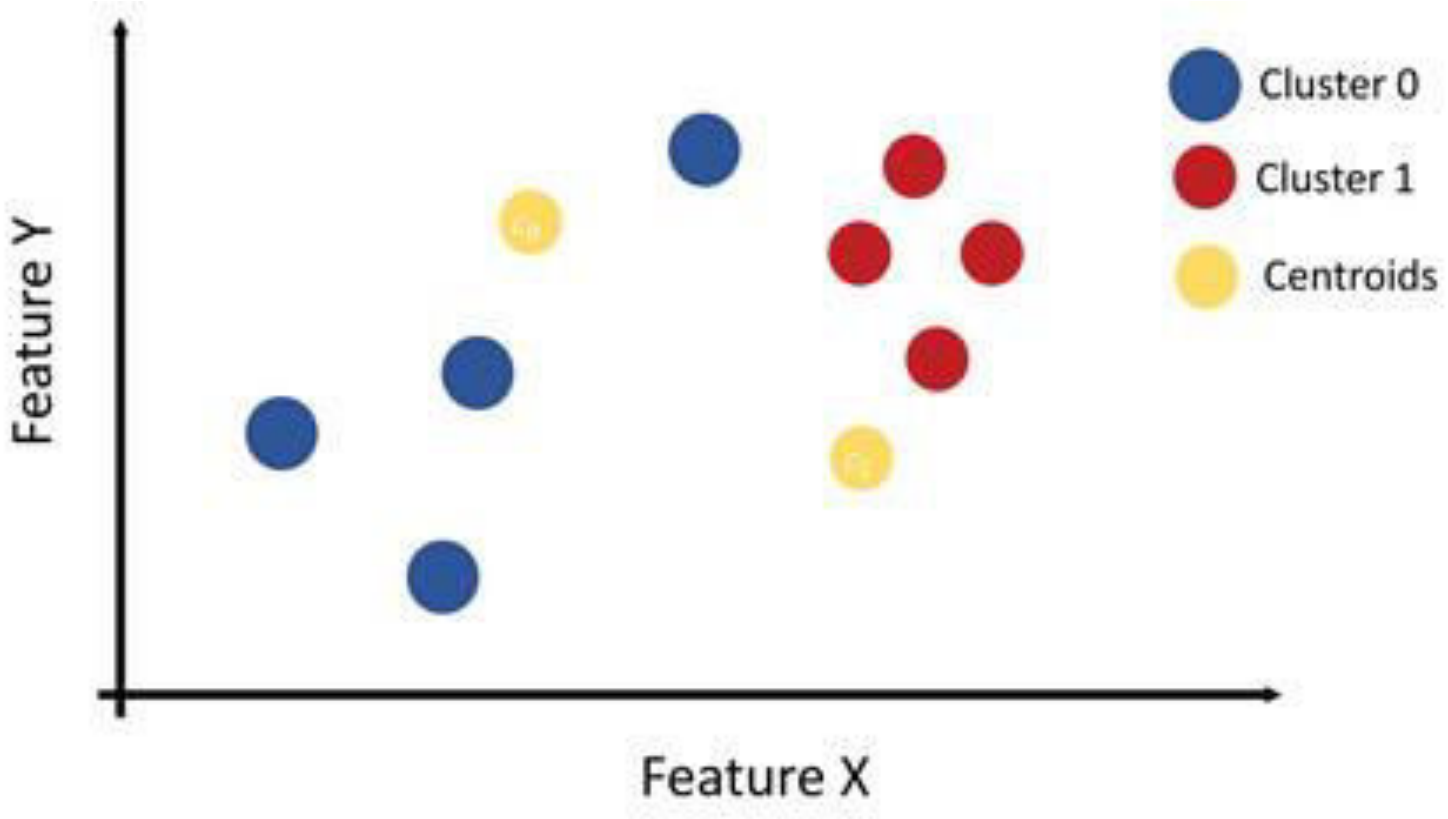
- **KNN classifier** estimates
  - conditional distribution of Y=f(X)
  - Assign to class with highest estimated probability.

K-Nearest Neighbor Classification



CreativeCommons.org

K-Nearest Neighbor Classification



CreativeCommons.org

Building Confusion Matrix

Chest Pain	Good Blood Circulation	Blocked Arteries	Weight	Heart Disease
Yes	No	No	125	No
No	Yes	Yes	160	Yes
Yes	Yes	No	120	No

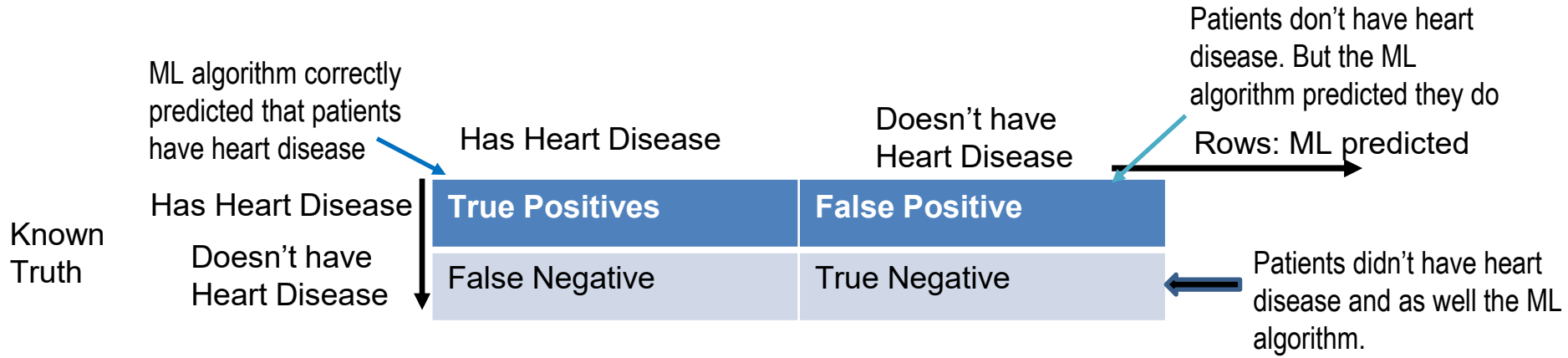
ML Algorithm 1

142	**22**
29	110

ML Algorithm 2

107	**53**
64	79

Confusion Matrix for Heart Disease



Confusion Matrix for Accuracy of Models

		predicted condition	
	total population	prediction positive	prediction negative
true condition	condition positive	**True Positive (TP)**	**False Negative (FN)** (type II error)
	condition negative	**False Positive (FP)** (Type I error)	**True Negative (TN)**

K-Nearest Neighbor Classification

- No learning required
- Model stores the entire dataset and classifies datapoints based on points in question
- The prediction is based on training data only!

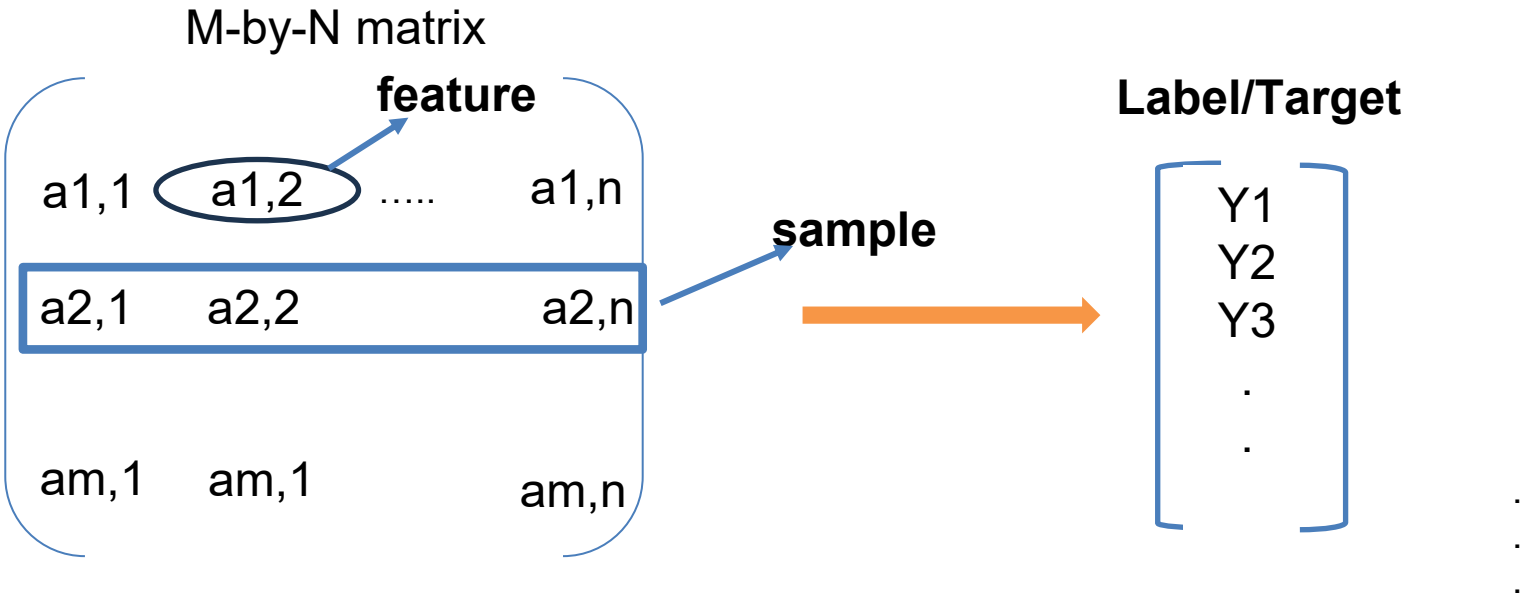
Algorithm:

1. Calculate the distance between data points.
2. Store distance array in ascending order (numpy argsort)
3. Select first k-elements in the sorted list
4. Use majority voting to classify new datapoint group.

Metric: Euclidian Distance
Get nearest neighbors
Make predictions

K-Nearest Neighbor Algorithm for Image Processing

- **Rows** – Instance (aka sample) of your data (SKU in supply chain, people in population, image in images, etc.)
- **Columns** - features (observation or attribute) of your data (SKUs: how many SKUs, what type of SKUs, People: sex, disease, height)
- **Target** - what you are trying to get your system to predict (business intel, cost of care, etc.)



k-NN with images

Libs

```
import matplotlib.pyplot as plt; matplotlib.image as mpimg
from PIL import Image; import numpy as np
```

Create your classifier:

```
from sklearn import neighbors
k1 = neighbors.KNeighborsClassifier(n_neighbors=1, weights='distance')
```

Train your classifier:

```
k1.fit(training_data, training_target)
```

Predict the class of the test images:

```
k1_pred = k1.predict(test_data)
```

(Note: you need to put the test data into the standard form as well)

KNN with images

- Follow our tutorials
  - Image processing
  - k-NN Classification

Installing PyCharm

For CSCE 4300/5300

Ravi Vadapalli, Ph.D.

PyCharm

- PyCharm is an integrated development environment (IDE) for python programming language.
- You can use community edition free of cost
- PyCharm environment supports code analysis, graphical debugger, and various other features.
- The best benefit of PyCharm is that it doesn’t incur the same level of memory overhead like Jupyter Notebook.

Installing PyCharm on Local Computing Environment

- PyCharm is free for .edu: Students and Teachers.

1. To get a copy of PyCharm for your educational use go to https://www.jetbrains.com/shop/eform/students
   - Pick apply with “University Email Address”
   - Select Status: “Student”
   - Enter your email id: john.doe@my.unt.edu
   - Provide First and Last Name
   - Country/Region: United States
   - Accept the terms and conditions
2. Check your email (that you provided in one of the above questions) for link to download a copy of PyCharm.
3. Follow the link in the email and sign-up for JetBrains using your UNT email and a password (choose forgot password to pick a password even for the first-time users)
4. Then login to JetBrains Account using your UNT email and the password you set in the previous step.
5. Now download PyCharm package (depending your native operating system: Windows or MacOS. Note: for MacOS: Make sure it’s Intel or Apple Silicon (aka M1 or M2 processors))
6. You may need up to 3GB disk space to install PyCharm.
7. Once you download PyCharm, you may need to provide UNT email and password (you setup in step 4) to activate the license for your student/educational/personal use.