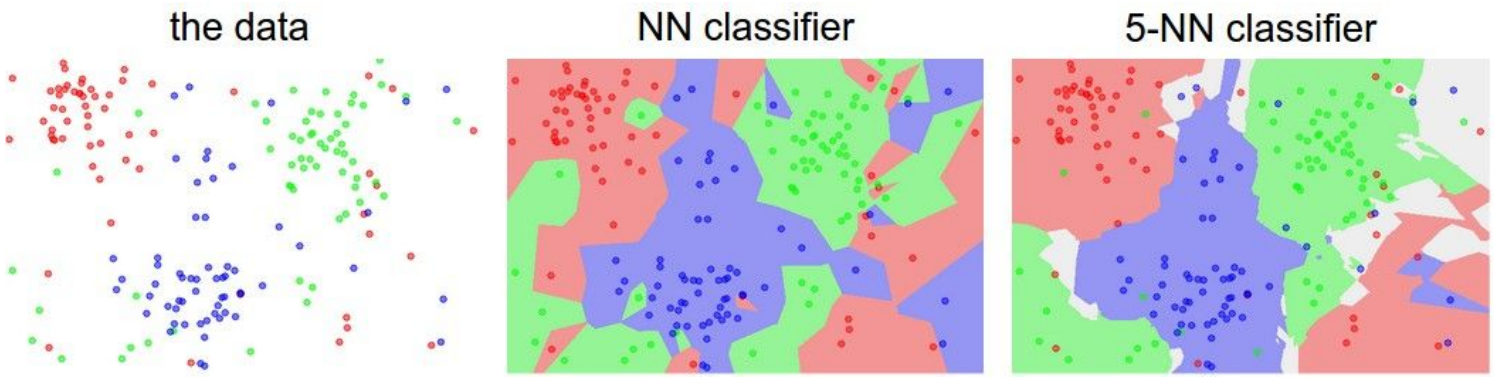


CS 106S Week 4

Cancer Detection with K-Nearest Neighbors

Ben Yan, Winter 2025

Stanford | ENGINEERING
Computer Science

Welcome to Week 4 of Class!

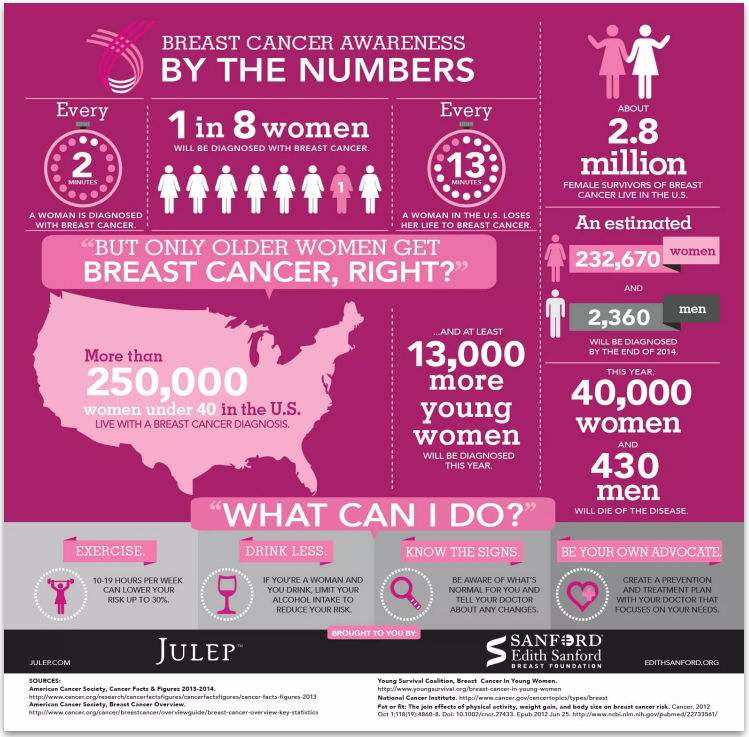
Winter Break

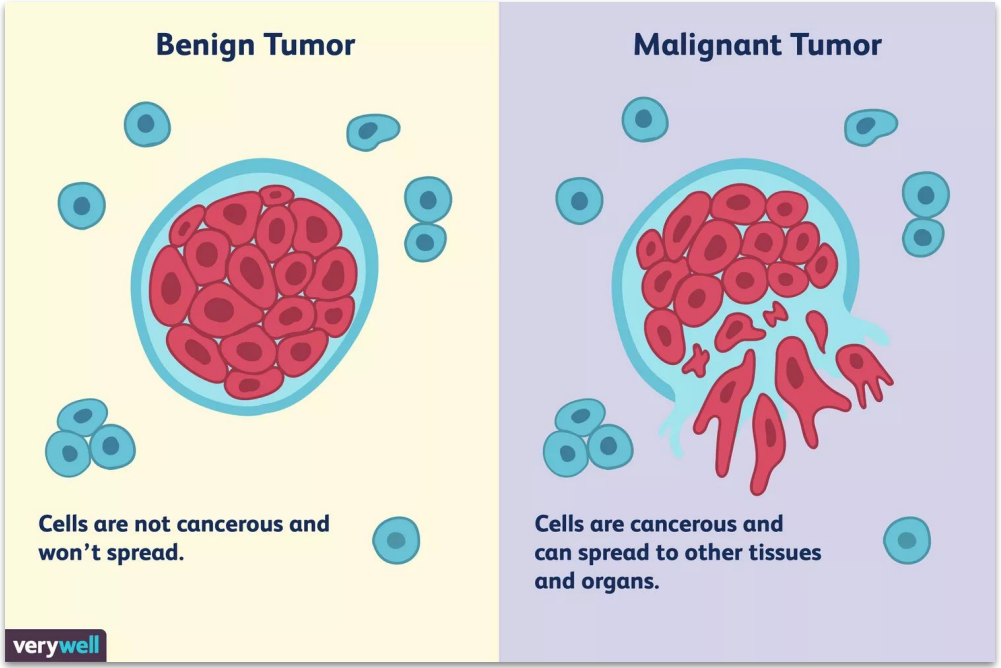
Spring Break

Map for Today

1. Brief overview of machine learning and its paradigms, e.g., supervised vs. unsupervised learning
2. Overview of K-nearest neighbors (KNN) algorithm
3. project: breast tumor classification with KNN
4. implementation & check-off form

The Problem


BREAST CANCER AWARENESS
BY THE NUMBERS
Every
2 MINUTES
A WOMAN IS DIAGNOSED WITH BREAST CANCER.
1 in 8 women
WILL BE DIAGNOSED WITH BREAST CANCER.
Every
13 MINUTES
A WOMAN IN THE U.S. LOSES HER LIFE TO BREAST CANCER.
ABOUT
2.8 million
FEMALE SURVIVORS OF BREAST CANCER LIVE IN THE U.S.
An estimated
232,670 women
AND
2,360 men
WILL BE DIAGNOSED BY THE END OF 2014.
"BUT ONLY OLDER WOMEN GET BREAST CANCER, RIGHT?"
More than 250,000 women under 40 in the U.S.
LIVE WITH A BREAST CANCER DIAGNOSIS.
...AND AT LEAST
13,000 more young women
WILL BE DIAGNOSED THIS YEAR.
THIS YEAR,
40,000 women
AND
430 men
WILL DIE OF THE DISEASE.
"WHAT CAN I DO?"
EXERCISE.
10-19 HOURS PER WEEK CAN LOWER YOUR RISK UP TO 30%.
DRINK LESS.
IF YOU'RE A WOMAN AND YOU DRINK, LIMIT YOUR ALCOHOL INTAKE TO REDUCE YOUR RISK.
KNOW THE SIGNS.
BE AWARE OF WHAT'S NORMAL FOR YOU AND TELL YOUR DOCTOR ABOUT ANY CHANGES.
BE YOUR OWN ADVOCATE.
CREATE A PREVENTION AND TREATMENT PLAN WITH YOUR DOCTOR THAT FOCUSES ON YOUR NEEDS.
JULEP.COM
JULEP
BROUGHT TO YOU BY:
SANFORD Edith Sanford BREAST FOUNDATION
EDITHSANFORD.ORG
SOURCES:
American Cancer Society, Cancer Facts & Figures 2013-2014.
http://www.cancer.org/research/cancerfactsfigures/cancerfactsfigures/cancer-facts-figures-2013
American Cancer Society, Breast Cancer Overview.
http://www.cancer.org/cancer/breastcancer/overviewguide/breast-cancer-overview-key-statistics
Young Survival Coalition, Breast Cancer In Young Women.
http://www.youngsurvival.org/breast-cancer-in-young-women
National Cancer Institute. http://www.cancer.gov/cancertopics/types/breast
Fat or fit: The join effects of physical activity, weight gain, and body size on breast cancer risk. Cancer. 2012 Oct 1;118(19):4860-8. Doi: 10.1002/cncr.27433. Epub 2012 Jun 25. http://www.ncbi.nlm.nih.gov/pubmed/22733561


Benign Tumor
Cells are not cancerous and won't spread.
Malignant Tumor
Cells are cancerous and can spread to other tissues and organs.
verywell

Today's Task

*Given medical data about cell growths, can we accurately classify these tumors as **benign** or **malignant**?*

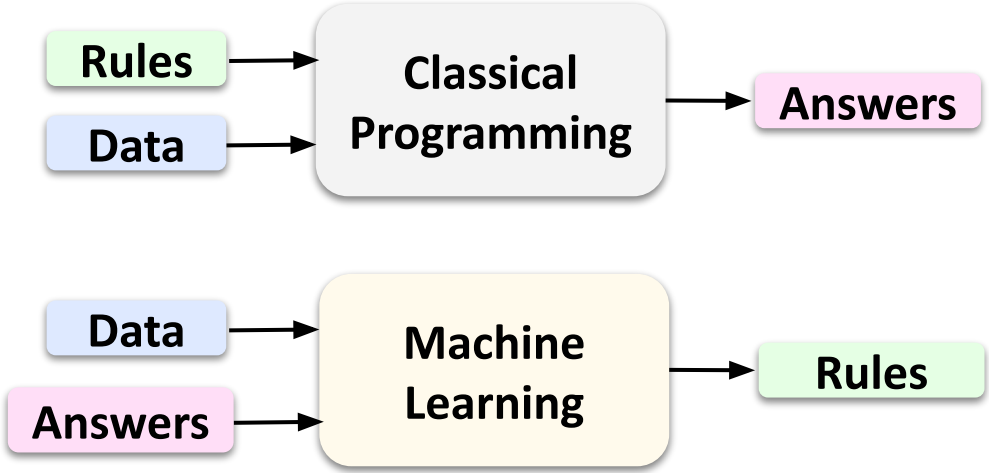
Test Sample ID	Correct?	Actual Label	Predicted Label
666942	✅	Benign	Benign
667204	✅	Malignant	Malignant
673637	✅	Benign	Benign
684955	✅	Benign	Benign
688033	✅	Benign	Benign
691628	✅	Malignant	Malignant

First, an overview of machine learning

Machine Learning

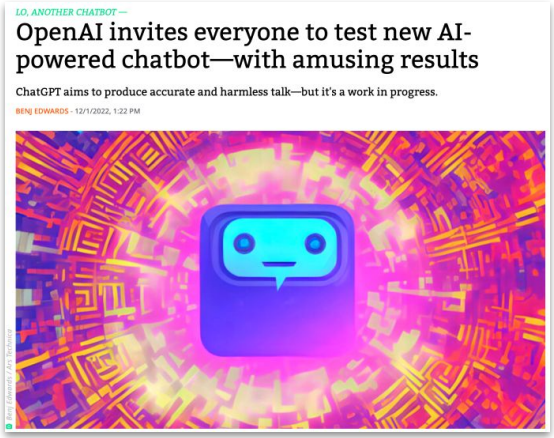
Machine learning is a “field of study that gives computers the ability to learn **without being explicitly programmed”** – Arthur Samuel

Game-Playing AI (AlphaGo)



Deep Learning

Deep learning (neural networks) requires **very large amounts of data** to learn complex rules, and is the core paradigm behind large language models, or *generative pre-trained transformers*.



I **am** **majoring** **in**

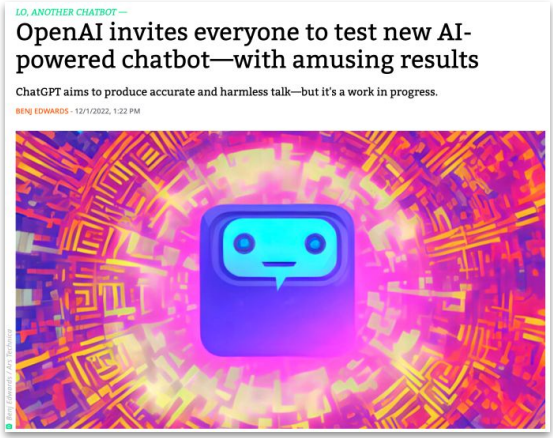
CS	0.25
economics	0.15
SymSys	0.12
biology	0.09
...	...

probability of being next word

Next word prediction—the rule is a **probability distribution** over all possible next words, which is learned from ingesting massive amounts of Internet-based text 🌐📚📗📒

Deep Learning

Deep learning (neural networks) requires **very large amounts of data** to learn complex rules, and is the core paradigm behind large language models, or ***generative pre-trained*** *transformers*.

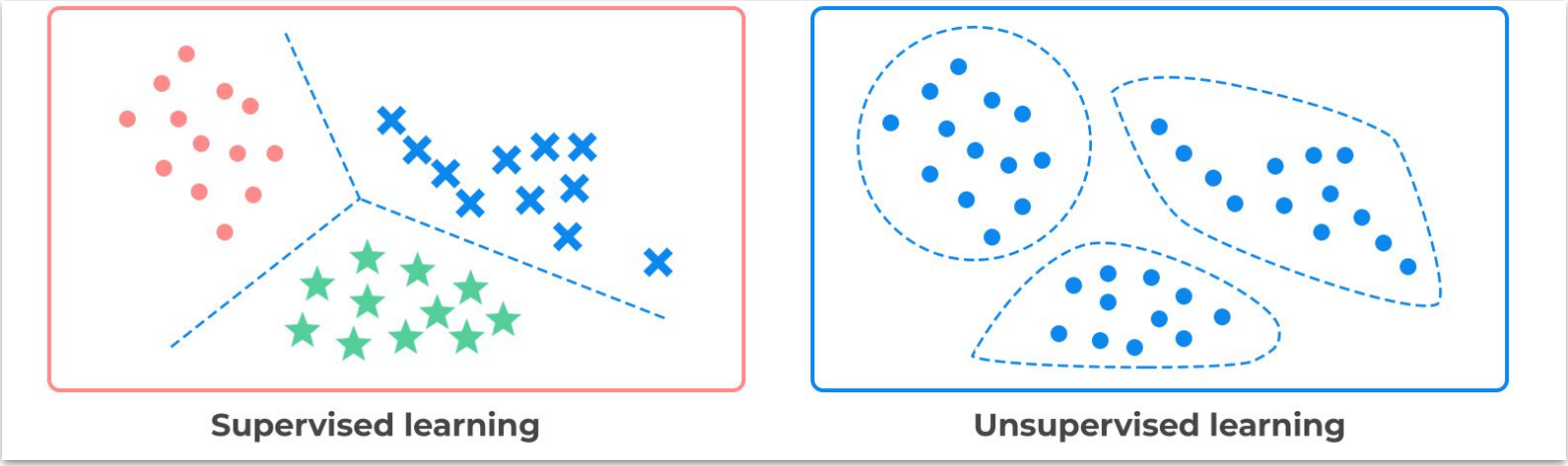


I **am** **majoring** **in** **CS** **because** **of**

passion	0.45
love	0.35
money!	0.10
conformity	0.05
...	...

Next word prediction – the algorithm, after choosing a word, proceeds to choose the next word from another **learned probability distribution**, and so forth iteratively.

Supervised vs Unsupervised Learning

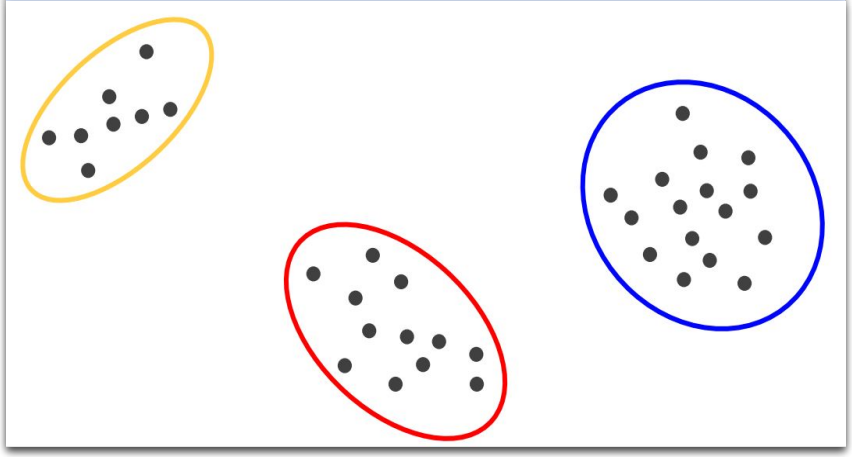


- **Supervised learning** uses **labeled** training data (input features/points and expected outputs) to train an algorithm to predict outputs for new inputs
- **Unsupervised learning** uses **unlabeled data**, and attempts to find patterns/ groupings on input features/points without them being explicitly tagged

Unsupervised Learning: Clustering

A common application of unsupervised learning is **clustering**, which involves grouping a dataset into some number of independent, related clusters.

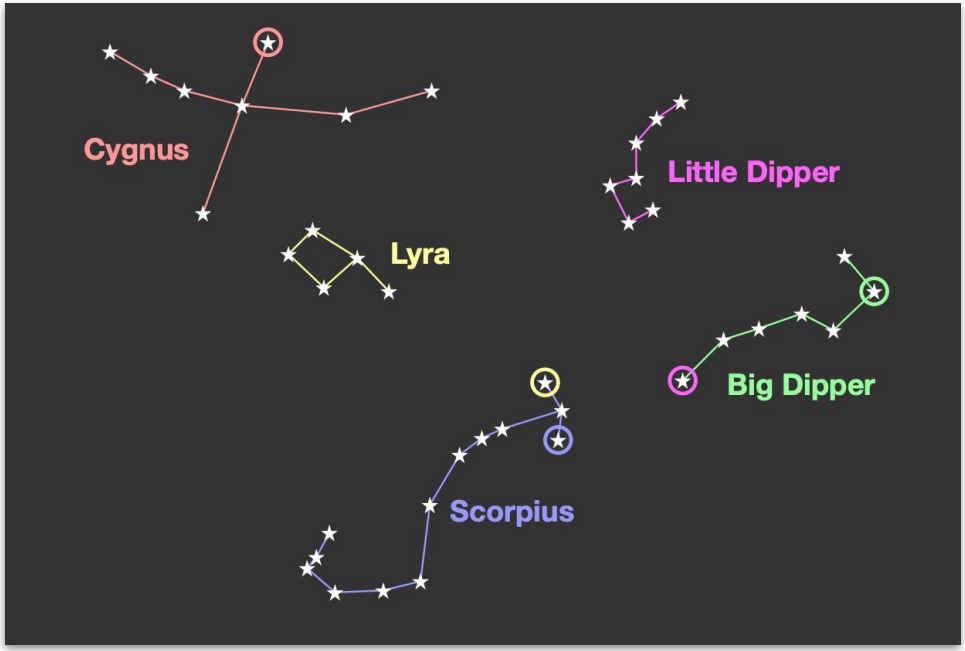
Ex. Based on locality, we can identify 3 groups from the `(x,y)` points below.



Ex. We can group movies based on genre.

Based on CS106AX materials (J. Cain, E. Roberts)

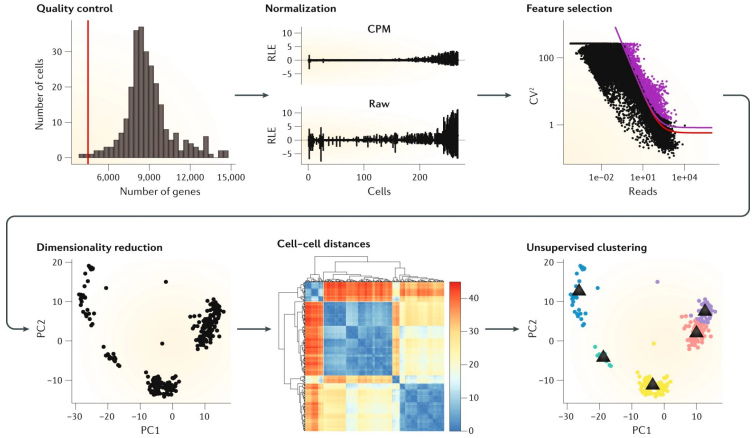
Clustering Example
Stars and Constellations

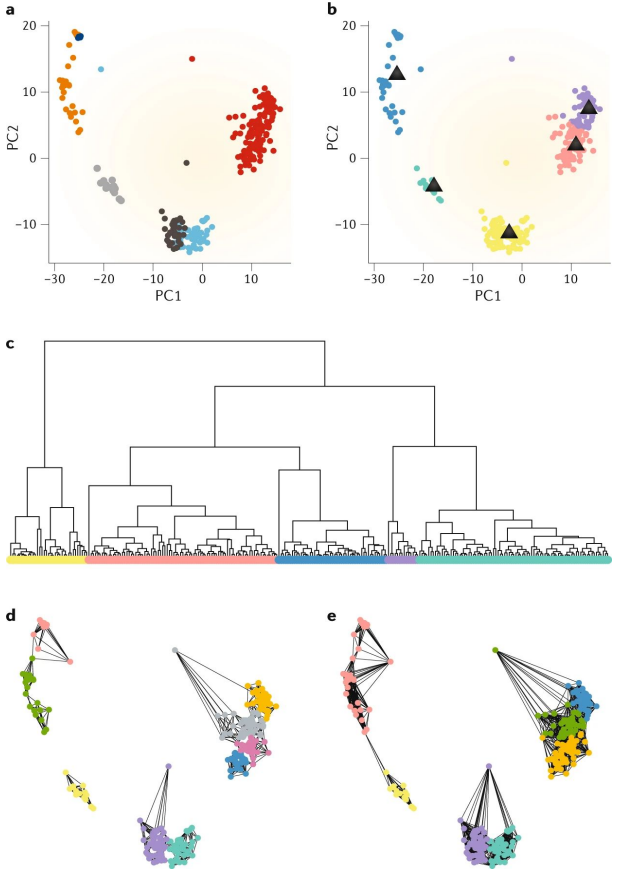


😔 **Star constellations aren't "real"** – in the sense the lines are imaginary, and the stars aren't actually physically connected.

Instead, they're **clusters / patterns** of stars that we see from Earth. It's a matter of perception.

Example Application
clustering scRNA-seq data



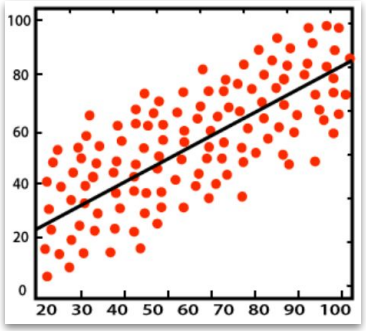


Clusters for scRNA sequences reveal and categorize **different cell types!**

https://www.nature.com/articles/s41576-018-0088-9/figures/1
https://www.nature.com/articles/s41576-018-0088-9/figures/3

Two main types of supervised learning

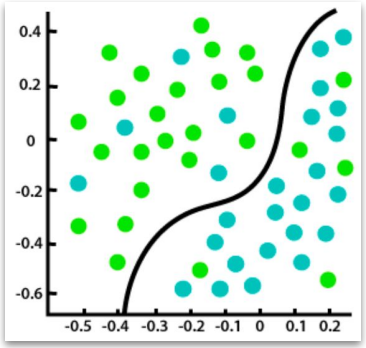
Regression



What numerical value is this data point affiliated with?

Avg Temp. Today 🌡️	Avg Wind Today 💨	Precip. Today ☔	→	Temp (F) Tomorrow 🌡️
51 F	5 mph	1 in	→	48 F?

Classification



Which category / label does this data point belong to?

Avg Temp. Today 🌡️	Avg Wind Today 💨	Precip. Today ☔	→	Weather Type 🥶
-3 F	25 mph	10 in	→	Sunny Cloudy Rainy MN ✅

Our Dataset

	A	B	C	D	E	F
1	666942	1	1	1	1	2
2	667204	7	8	7	6	4
3	673637	3	1	1	1	2
4	684955	2	1	1	1	3

Training: ~630 samples | **Testing: ~70 samples**

~700 samples (rows) in total

💾 Each **row** has 11 columns: a **sample ID #** (col 0), a **binary label** (last col), and **9 input features** (each a value between **1-10**) in between.

- In JavaScript, we represent each row as an **array of numbers**.

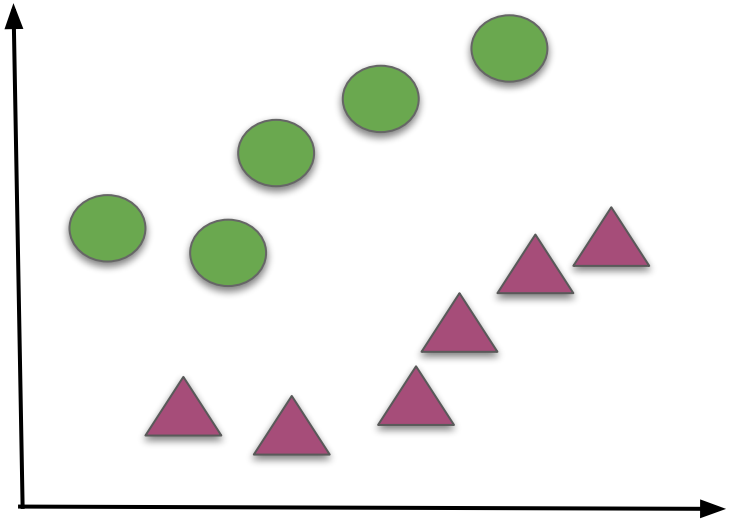
1001	1.0	3.0	4.0	2.0	7.0	6.0	8.0	5.0	1.0	0

9 Input Features

1. Clump Thickness
2. Uniformity of Cell Size
3. Uniformity of Cell Shape
4. Marginal Adhesion
5. Single Epithelial Cell Size
6. Bare Nuclei
7. Bland Chromatin
8. Normal Nucleoli
9. Mitoses

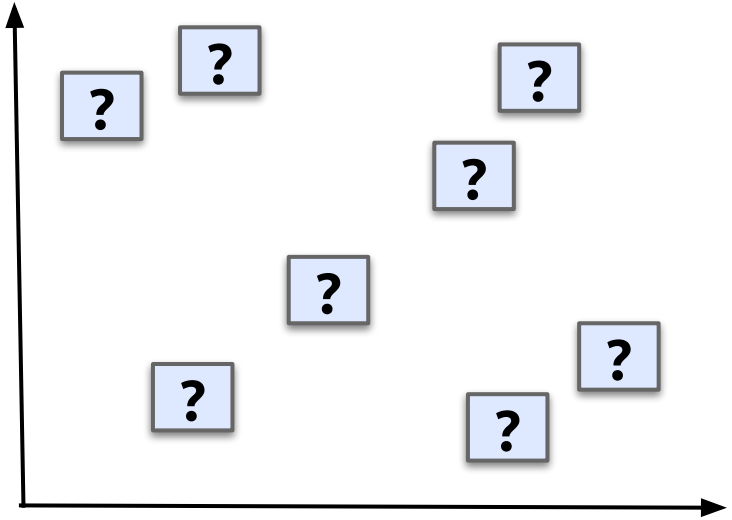
Training / Testing Split

Training Set



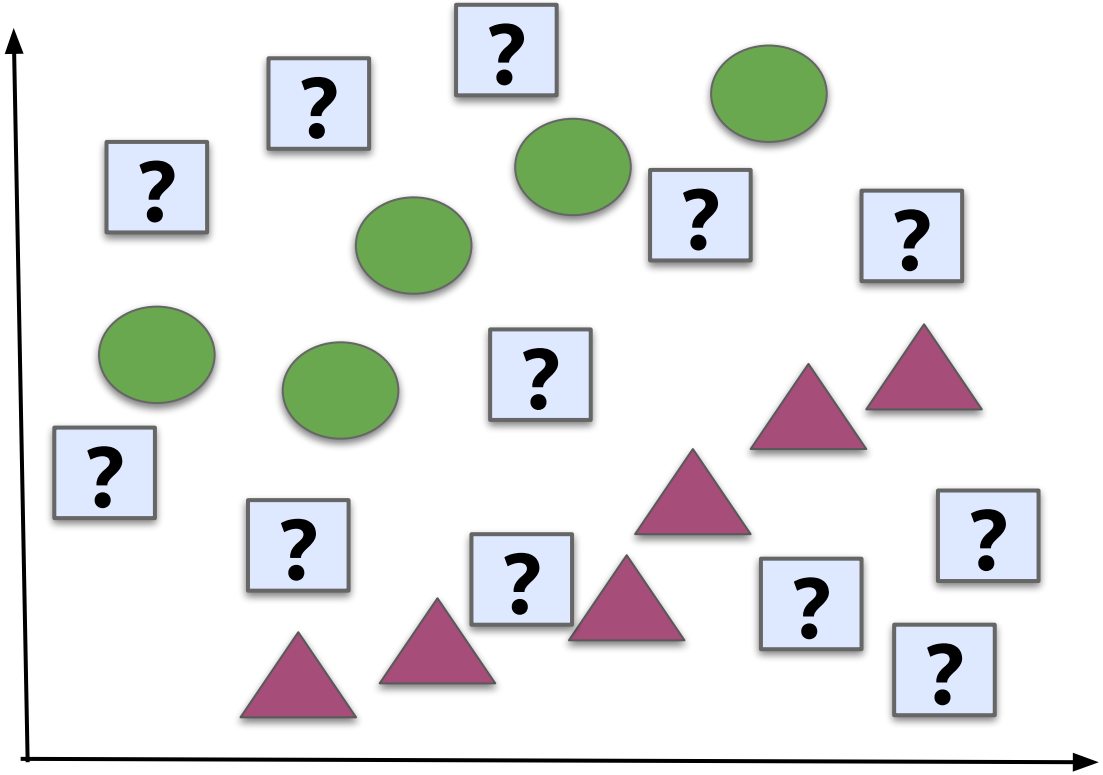
We use this labeled set of tumor samples to train/inform the model

Test Set



We use this to evaluate the model's performance

How to classify the test set?



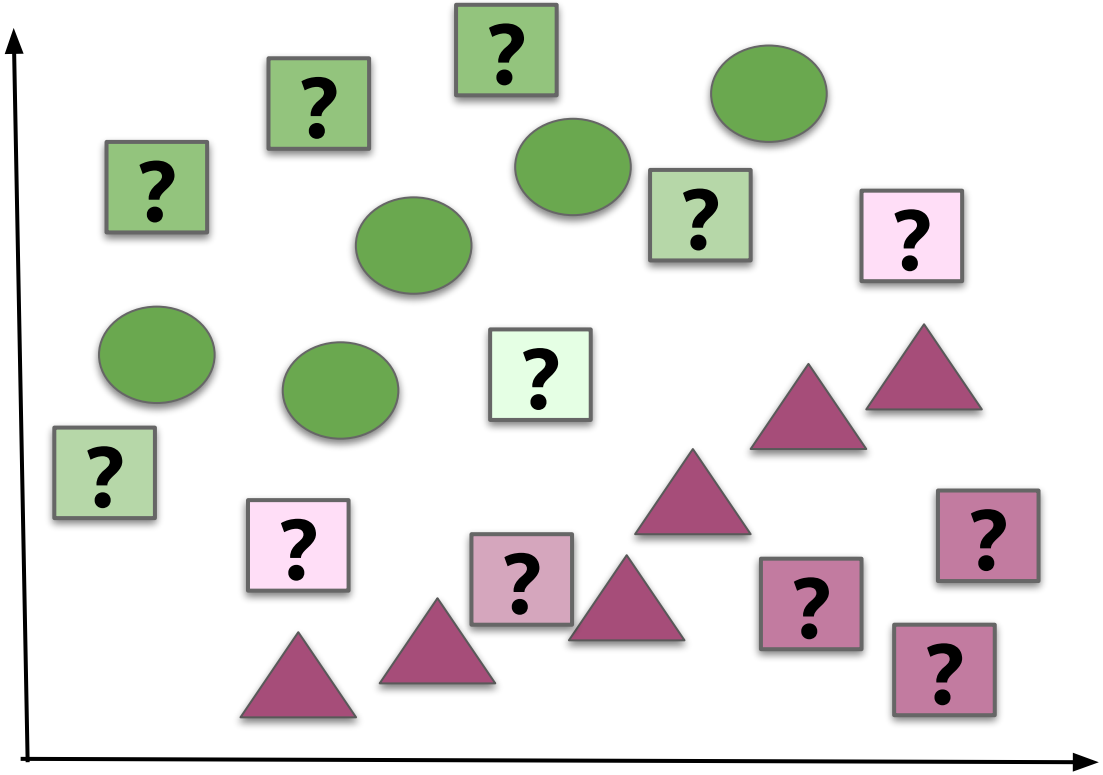
Training Set

Data points whose **label** the model already knows

Test Set

Data points whose **label** the model doesn't know

Maybe like this:



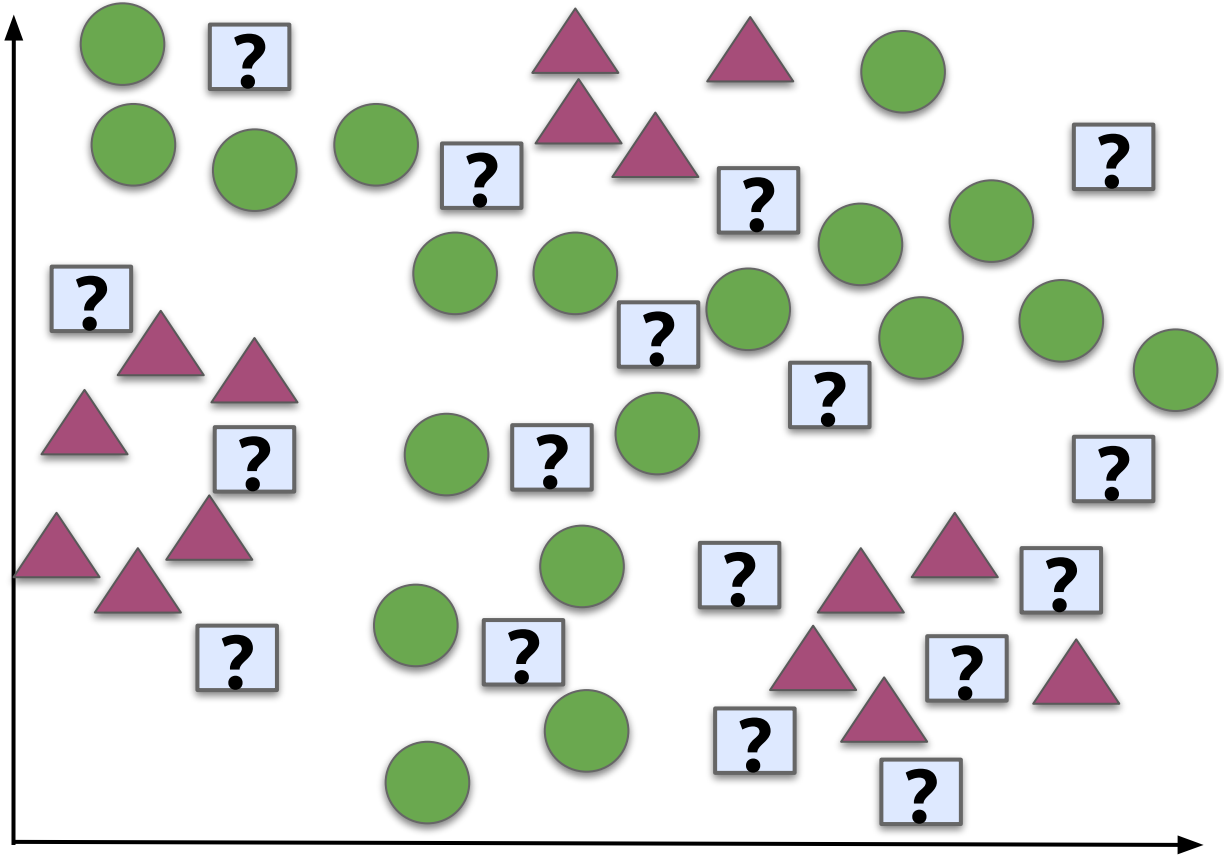
Training Set

Data points whose **label** the model already knows

Test Set

Data points whose **label** the model doesn't know

How about this one? Oof



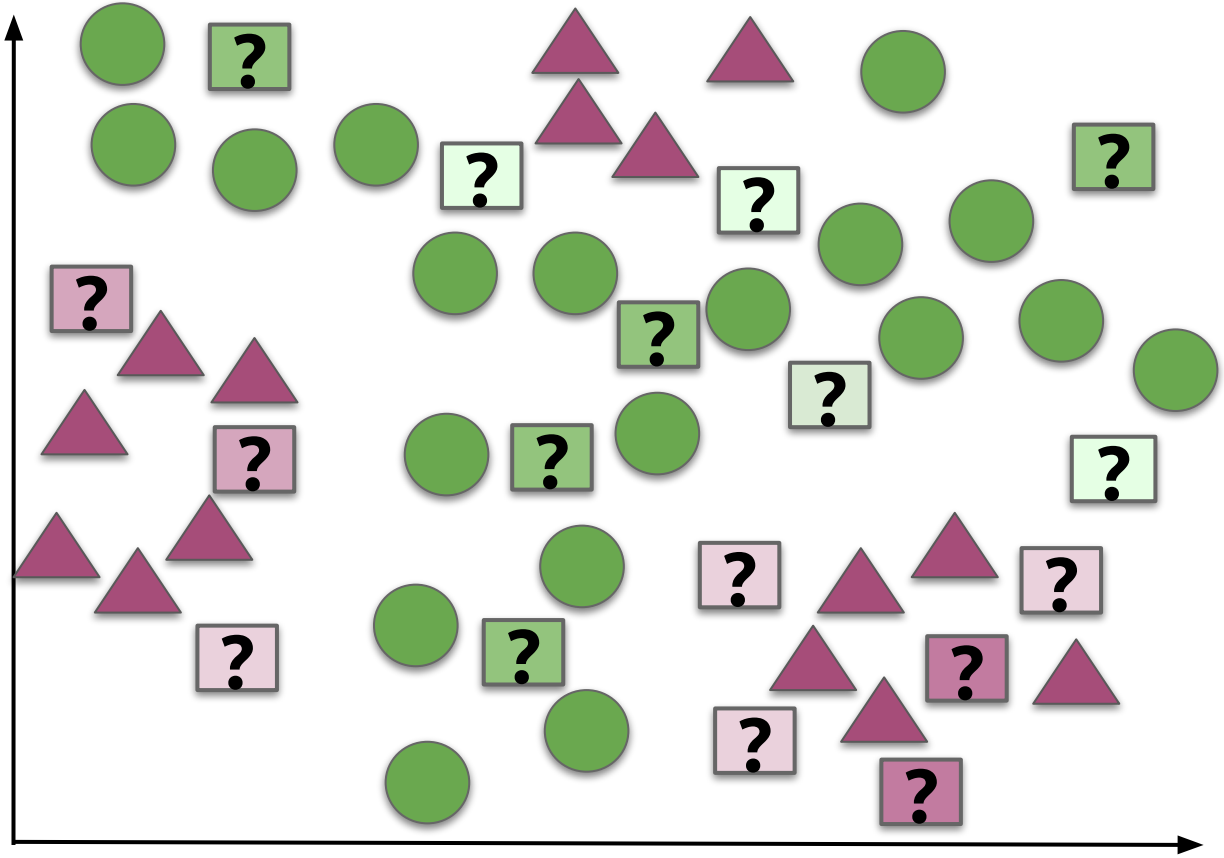
Training Set

Data points whose **label** the model already knows

Test Set

Data points whose **label** the model doesn't know

Maybe like this:



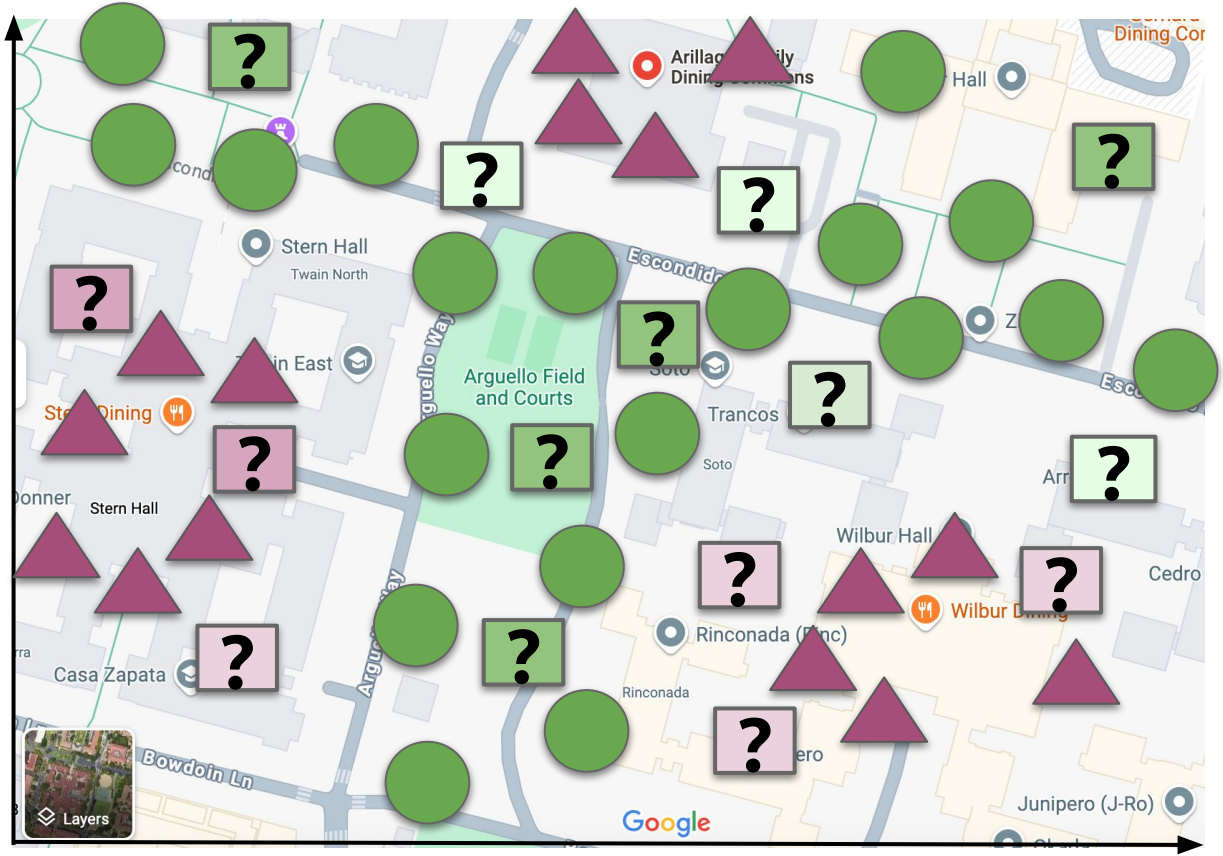
Training Set

Data points whose **label** the model already knows

Test Set

Data points whose **label** the model doesn't know

Real data doesn't look like that…



Within a dining hall, or the surrounding tables outside

Not within a dining hall area

Training Set

Data points whose **label** the model already knows

Test Set

Data points whose **label** the model doesn't know

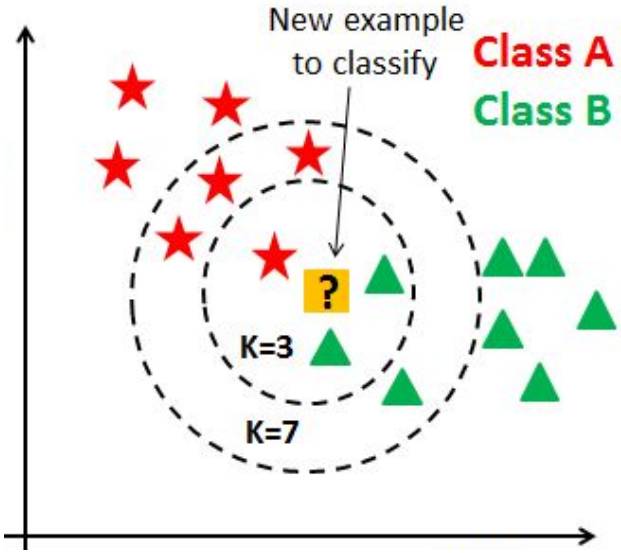
K-Nearest Neighbors Algorithm

Guiding Principle: Use **locality / proximity** in the dataset to predict new points!

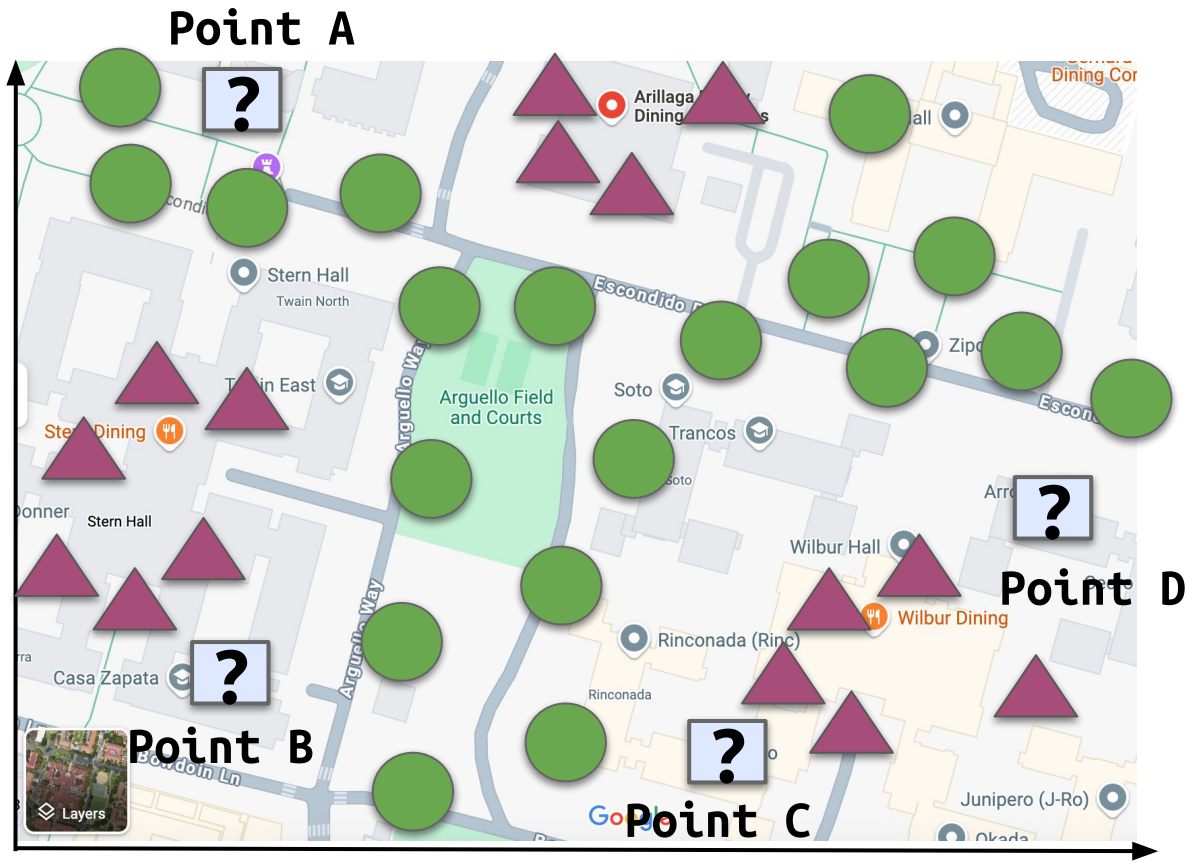
- Points with the **same label** are likely to be **close to each other**
- Points with **different labels** are likely to be **far away** from each other

For each test sample / point to classify:

1. Calculate distance between test sample and every training point.
2. Pick the K training points with the smallest distances to the test sample (i.e. its "nearest neighbors")
3. **Of those K points, do a vote:** how many are classified as **malignant** vs. **benign**?
4. **Majority wins:** pick the majority label



KNN (K=3): Campus Map Example



Point A

3 Nearest Neighbors

Predict

Point B

3 Nearest Neighbors

Predict

Point C

3 Nearest Neighbors

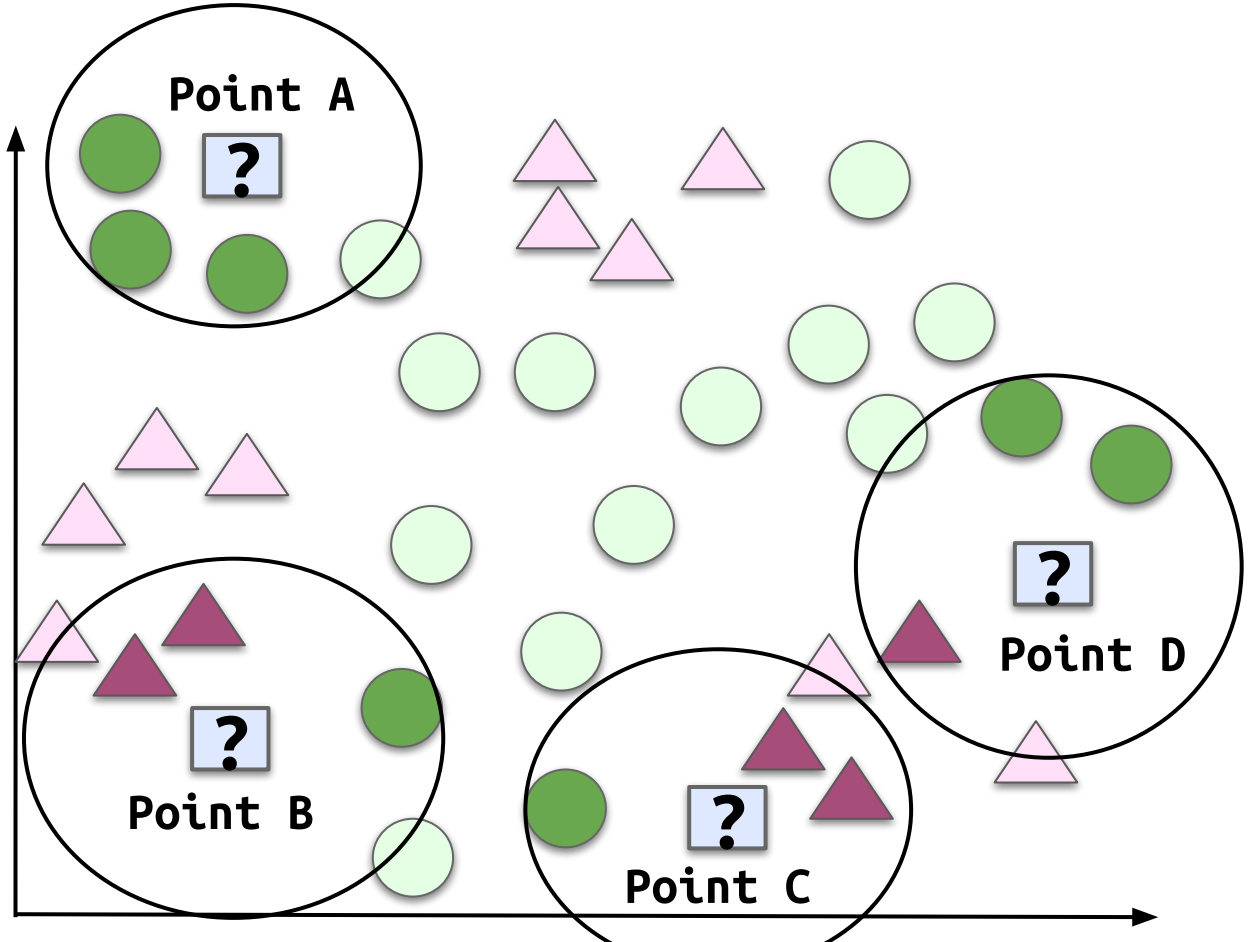
Predict

Point D

3 Nearest Neighbors

Predict

For each point, get nearest neighbors



Point A

3 Nearest Neighbors

Predict

Point B

3 Nearest Neighbors

Predict

Point C

3 Nearest Neighbors

Predict

Point D

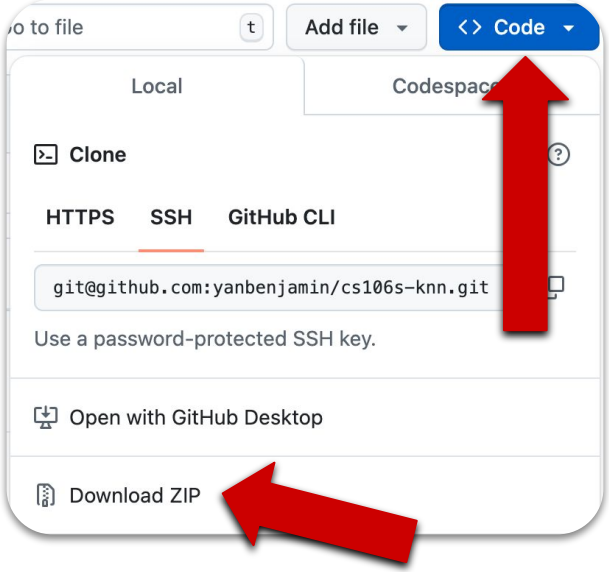
3 Nearest Neighbors

Predict

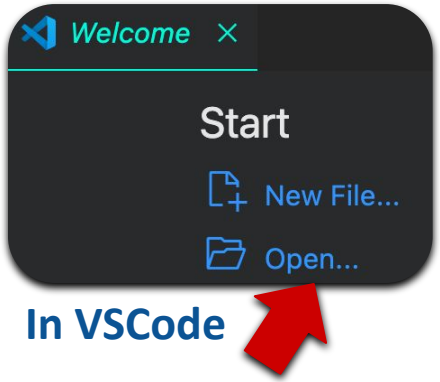
Let's get started!

1 Navigate to Week 4 of the Schedule section of **cs106s.stanford.edu**

Also, at this link: **https://github.com/yanbenjamin/cs106s-knn**



2 Click the bright **"Code"** button, then click "Download ZIP"



In VSCode

3 Unzip the download (clicking .zip file should do the trick) and open the folder / files in your editor

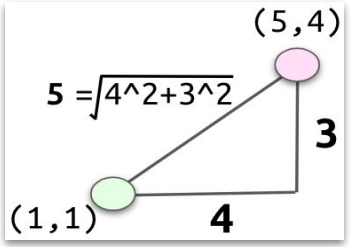
Milestone #1: `classify.js`

`calculateDistance(sample1, sample2)`

Computes Euclidean distance i.e. **sums up the squared differences** between **`sample1[i]`** and **`sample2[i]`** for each index **`i`**, and return the square root of that sum. **However, index 0 (sample ID #) and last index (label) should be ignored, as they're not features.**

sample1	1001	1.0	3.0	4.0	2.0	7.0	6.0	8.0	5.0	1.0	0
sample2	1005	1.0	3.0	6.0	2.0	4.0	4.0	6.0	7.0	1.0	0
index	0	1	2	3	...					sample2.length-1	

$$D = \sqrt{(1.0 - 1.0)^2 + (3.0 - 3.0)^2 + (4.0 - 6.0)^2 + (2.0 - 2.0)^2 + (7.0 - 4.0)^2 + (6.0 - 4.0)^2 + (8.0 - 6.0)^2 + (5.0 - 7.0)^2 + (1.0 - 1.0)^2} = 5$$



Euclidean Distance in 2D

Milestone #2

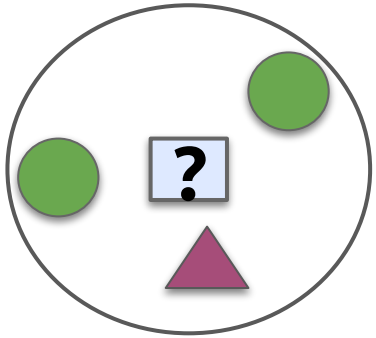
`getClosestKPoints(testSample, trainData, K)`

testSample

1001	1.0	3.0	4.0	2.0	7.0	6.0	8.0	5.0	1.0	?

trainData

2001	1.0	3.0	4.0	2.0	7.0	6.0	5.0	5.0	1.0	0
2002	1.0	7.0	6.0	3.0	6.0	6.0	3.0	7.0	1.0	0
• • •					• • •				• • •	
3000	2.0	3.0	3.0	2.0	4.0	4.0	6.0	9.0	2.0	0



Searches for and returns an array of the **K** closest points in **trainData** to the input **testSample** point.

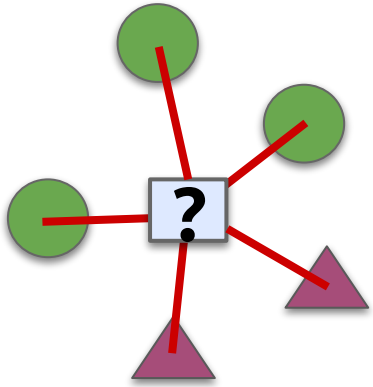
An example return array may look like the one on the right **(K = 3).**

```
[
  {“id”: 2100, “distance”: 1.0, “label”: 1},
  {“id”: 2200, “distance”: 1.4, “label”: 0},
  {“id”: 2604, “distance”: 2.7, “label”: 0},
]
```

Milestone #2 Steps

`getClosestKPoints(testSample, trainData, K)`

testSample	1001	1.0	3.0	4.0	2.0	7.0	6.0	8.0	5.0	1.0	?
trainData	2001	1.0	3.0	4.0	2.0	7.0	6.0	5.0	5.0	1.0	0
	2002	1.0	7.0	6.0	3.0	6.0	6.0	3.0	7.0	1.0	0
	• • •					• • •				• • •	
	3000	2.0	3.0	3.0	2.0	4.0	4.0	6.0	9.0	2.0	0



Step 1: Populate an array **`pointDistances`** by iterating through **`trainData`**: for each point,

- Use **`calculateDistance`** to get the training point's distance with `testSample`
- Add a JS object to **`pointDistances`** storing the point's sample id (index 0), distance from `testSample`, and label (last index).

```
pointDistances = [
  {"id": 2001, "distance": 3.0, "label": 0},
  {"id": 2002, "distance": 16.3, "label": 1},
  {"id": 2002, "distance": 12.5, "label": 1},
  ...
  {"id": 3000, "distance": 9.7, "label": 0}
]
```

Milestone #2 Steps

`getClosestKPoints(testSample, trainData, K)`

Step 2: Sort the points in array **`pointDistances`** from lowest to highest **`distance`** (will talk about sorting arrays in JavaScript next).

This ensures the closest points / nearest neighbors to **`testSample`** are at the front.

```
pointDistances = [
  {"id": 2100, "distance": 1.0, "label": 1},
  {"id": 2200, "distance": 1.4, "label": 0},
  {"id": 2604, "distance": 2.7, "label": 0},
  {"id": 2801, "distance": 3.1, "label": 1},
  ...
  {"id": 2050, "distance": 78.0, "label": 0}
]
```

Step 3: Take a subarray of first **`K`** points, which because of sorting earlier, represents the **closest K training points to `testSample`.**

For the example on the right, **`K=3.`**

```
pointDistances = [
  {"id": 2100, "distance": 1.0, "label": 1},
  {"id": 2200, "distance": 1.4, "label": 0},
  {"id": 2604, "distance": 2.7, "label": 0},
]
```

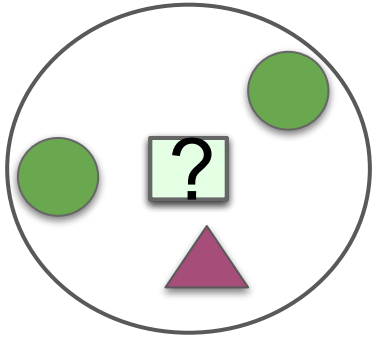
This is what the function should output

Final Milestone

`predictSample(testSample, trainData, K)`

```
closestKPoints = [
  {“id”: 2100, “distance”: 1.0, “label”: 1},
  {“id”: 2200, “distance”: 1.4, “label”: 0},
  {“id”: 2604, “distance”: 2.7, “label”: 0},
]
```

Array returned by `getClosestKPoints`



This functions brings everything together to **predict a test sample’s label.**

- It first makes a call to **`getClosestKPoints()`** to get an array of the K closest points to **`testSample`**.
- Iterate through **`closestKPoints`**, and count how many points are **benign (label 0)** and how many points are **malignant (label 1).**
- Determine the **majority label** among them, and return it.

Array Methods

`arr.push(element,...)`

Adds one or more elements to end of array.

`arr.pop()`

Removes and returns the last element of array.

`arr.slice(start,finish)`

Returns subarray beginning at `index` start and ending **just before** `finish`

Object Methods

To create an **object** / dictionary in JS, you list a sequence of **key-value pairs**, enclosed in curly braces.

```
let bratAlbum = {
    “name”: “Brat”,
    “artist”: “Charli XCX”,
    “year”: 2023
}
```

You can lookup **values** via their **keys!**

```
bratAlbum[“artist”] \\”Charli XCX”

\* alternative syntax *\
bratAlbum.year \\ 2023
```

JavaScript: Sorting

Recall: **JavaScript is a "vibes"-based language** ✨💅


```
> let arr = [3,2,1];
> arr.sort();
> console.log(arr)
[1, 2, 3]
```

Okay makes sense

```
> let arr = [200,60,3,1000];
> arr.sort();
> console.log(arr)
[1000, 200, 3, 60]
```

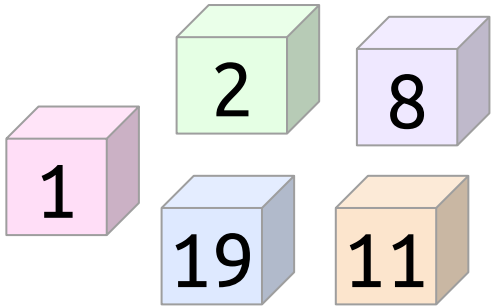
um excuse me what the actual—

Everything is a string if you have the willpower! JavaScript interprets and sort the inputs alphabetically, as though they are strings, so "**2**00" < "**3**", "**3**" < "**6**0"

We gotta teach JavaScript how to sort 😟

We have to define a **comparison function** `compareInputs(a,b)`, and pass that function into `sort()` to guide it. This function should:

- `return 0` if **a** and **b** are of **identical ordering**
- `return <0` (negative) if **a** **should go before** **b**
- `return >0` (positive) if **b** **should go before** **a**



Sorting is really just knowing which items go before which other items.

🖊️How might we write a comparison function to sort numbers from **smallest to largest**?

```
function compareNums(a,b){
    return a - b;
}
```

Note this is negative when a < b. Thus, when a < b, a goes first (**smaller numbers go first**).

```
> let arr = [200,60,3,1000];
> arr.sort(compareNums);
> console.log(arr)
[3, 60, 200, 1000]
```

Okay slay

KNN: Comparing Point Distances

One of the tasks in Milestone #2 is sorting point entries of the form

```
{“id”: 2401, “distance”: 5.6, “label”: 0}
{“id”: 2402, “distance”: 12.2, “label”: 1}
```

from smallest to largest distance from **`testSample`**. Consider how we may adapt the implementation of **`compareNums`** to work for these object types.

```
// function to write-up in classify.js
function comparePoints(pointA, pointB){



} // to get dist, e.g., pointA.distance
```

The function should:

- Return **`0`** if **`pointA`** and **`pointB`** have same distance
- Return **`negative`** if **`pointA`**’s distance is smaller (closer)
- Return **`positive`** if **`pointA`**’s distance is larger (farther)

KNN Predictions

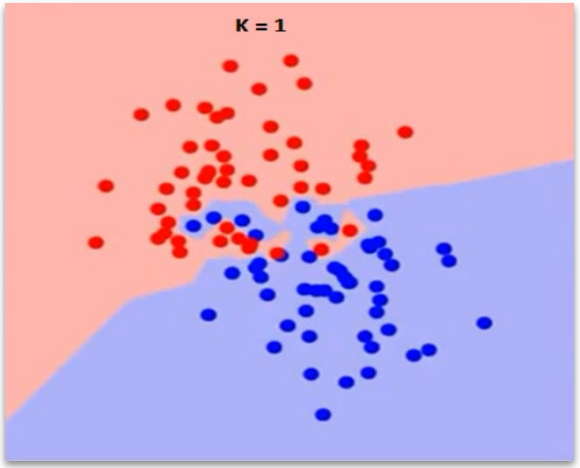
Test Sample ID	Correct?	Actual Label	Predicted Label	K-Nearest Neighbors (K=3)		
666942	✅	Benign	Benign	1059552	1156272	1164066
667204	✅	Malignant	Malignant	1223793	1228152	1296572
673637	✅	Benign	Benign	128059	1133136	1043999
684955	✅	Benign	Benign	1136142	1036172	1067444
688033	✅	Benign	Benign	1190485	1204242	1214092
691628	✅	Malignant	Malignant	314428	1102573	1096800
693702	✅	Benign	Benign	1190485	1204242	1214092
704097	✅	Benign	Benign	1320077	1344449	1035283
704168	❌	Benign	Malignant	1096800	1115282	1253955
706426	✅	Malignant	Malignant	1168736	1185609	1174428
709287	✅	Malignant	Malignant	1205138	1231387	1193544
718641	✅	Benign	Benign	1136142	1185610	1155546
721482	❌	Benign	Malignant	1091262	1148278	1185609
730881	✅	Malignant	Malignant	1176881	1189266	1171710
733639	✅	Benign	Benign	169356	1177027	1197270
733639	✅	Benign	Benign	1177027	1197270	1198641

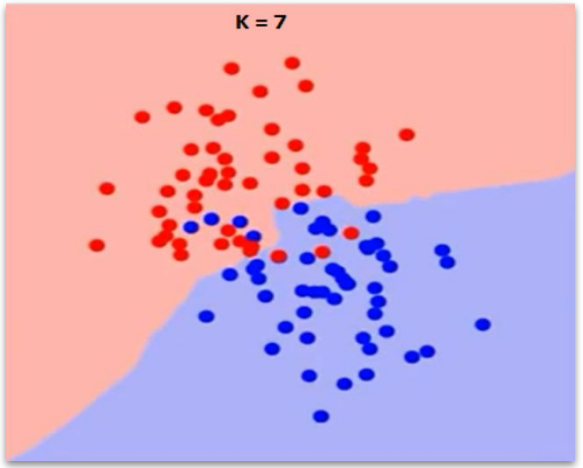
What value of K is best?

Case-Specific: Depends on dataset! Just have to try out different values

Low K – Very sensitive to noise, but advantage is selectivity (looking just at the closest possible / most similar points)

High K – Smoother decision boundaries, but at a trade-off of diminishing locality and what qualifies as a nearest neighbor





Further resources

- More on KNN and classification broadly from CS231N course notes
- Related algorithms and techniques:

K-means clustering

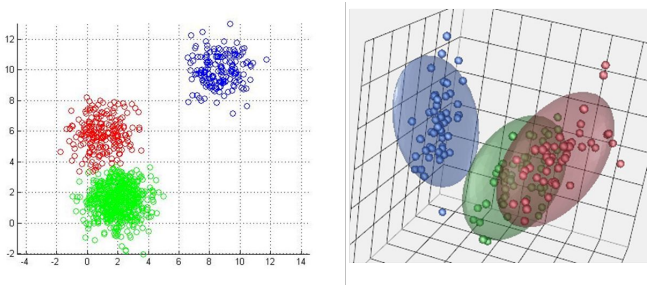
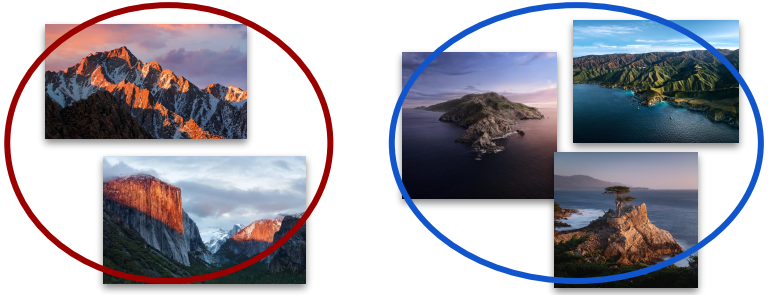
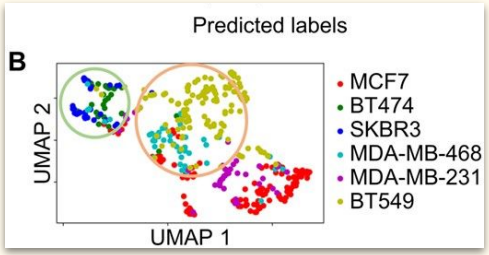


Image-based KNN



- Real world connection: “Deep learning-based classification of breast cancer cells using transmembrane receptor dynamics” (Kim et al. 2022) applies deep learning to assess the metastatic potential of breast cancer cells



Announcement from Haas Center to CS106S

Summer Undergraduate / Cardinal Quarter Fellowships

If you are interested in continuing your learning around your CS coursework consider a stipended Haas Center Undergraduate Fellowship this summer. There's a wide range of funding available, but some fellowships that might be of interest given your enrollment in this course are the Haas Center Undergraduate Fellowships, Public Service Projects Fellowship, CS+Social Good Fellowship, Social E Fellowship and the Roland Longevity Fellowship.

Each Cardinal Quarter Fellow receives a base stipend of $7000 to support travel and living expenses for a nine-week, full-time experience. Financial aid and supplemental funding, including a travel supplement and high cost of living supplement, is available to students who qualify. Cardinal Quarter programs are offered through 30+ campus partners, offering a wide range of service experiences.

The application deadline for most fellowships is February 4, 11:59pm, PST. Please reach out to a Cardinal Quarter Peer Advisor or email cardinalquarter@stanford.edu with any questions.

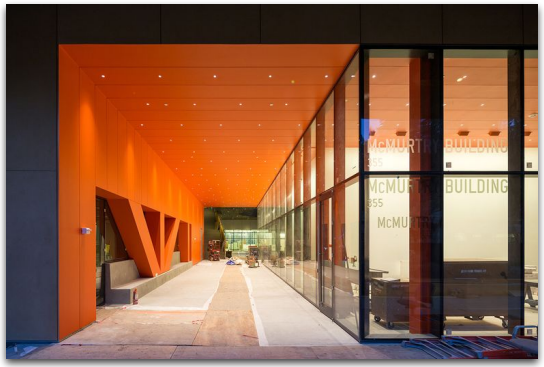
Check-Off Form!

Another **brief check-off form** (< 5 min to complete) for checking attendance!

For today, click the “Check-Off Form” link in the **Week 4** section of **cs106s.stanford.edu**.

Thank you!

Have an awesome week, and good luck on any midterms! 🍀



You're doing to do awesome