AI 330: Machine Learning

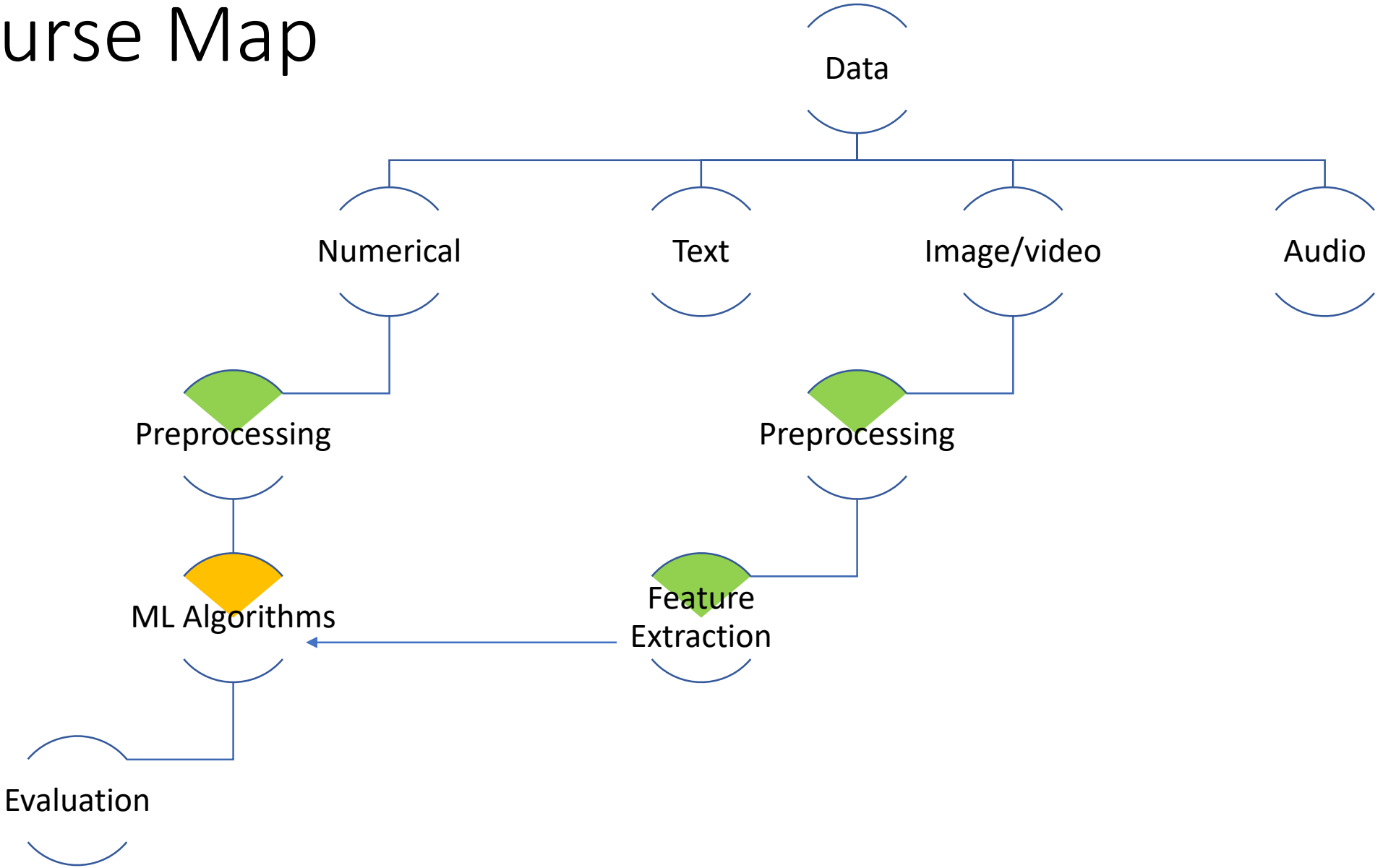
Fall 2023

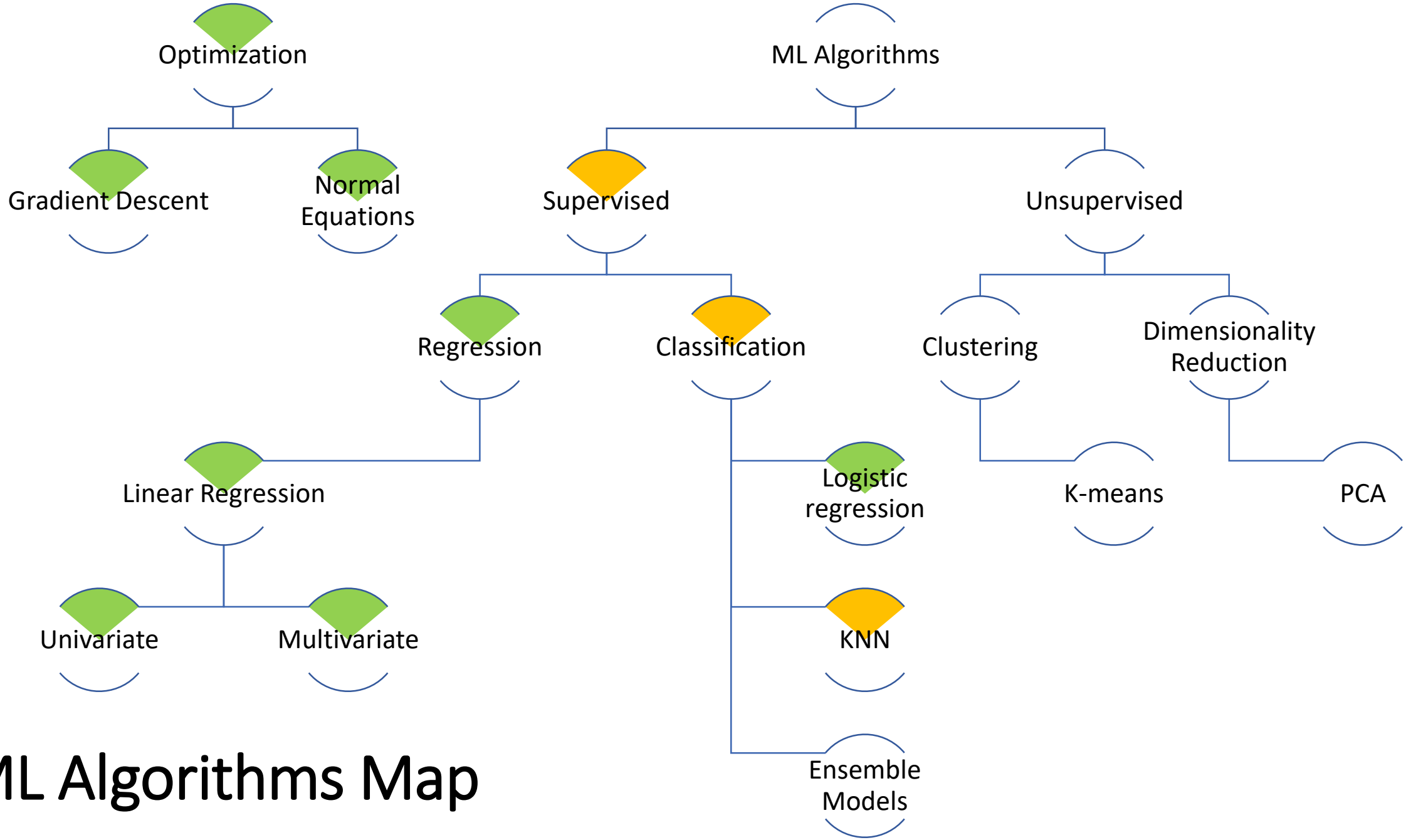
- **Dr. Wessam EL-Behaidy**
- Associate Professor, Computer Science Department,
- Faculty of Computers and Artificial Intelligence,
- Helwan University.

Dr. Ensaf Hussein

Associate Professor, Computer Science Department,
Faculty of Computers and Artificial Intelligence,
Helwan University.

Course Map


Data
Numerical
Text
Image/video
Audio
Preprocessing
Preprocessing
ML Algorithms
Feature Extraction
Evaluation



ML Algorithms Map

Lecture 7
k-Nearest Neighbor (kNN)

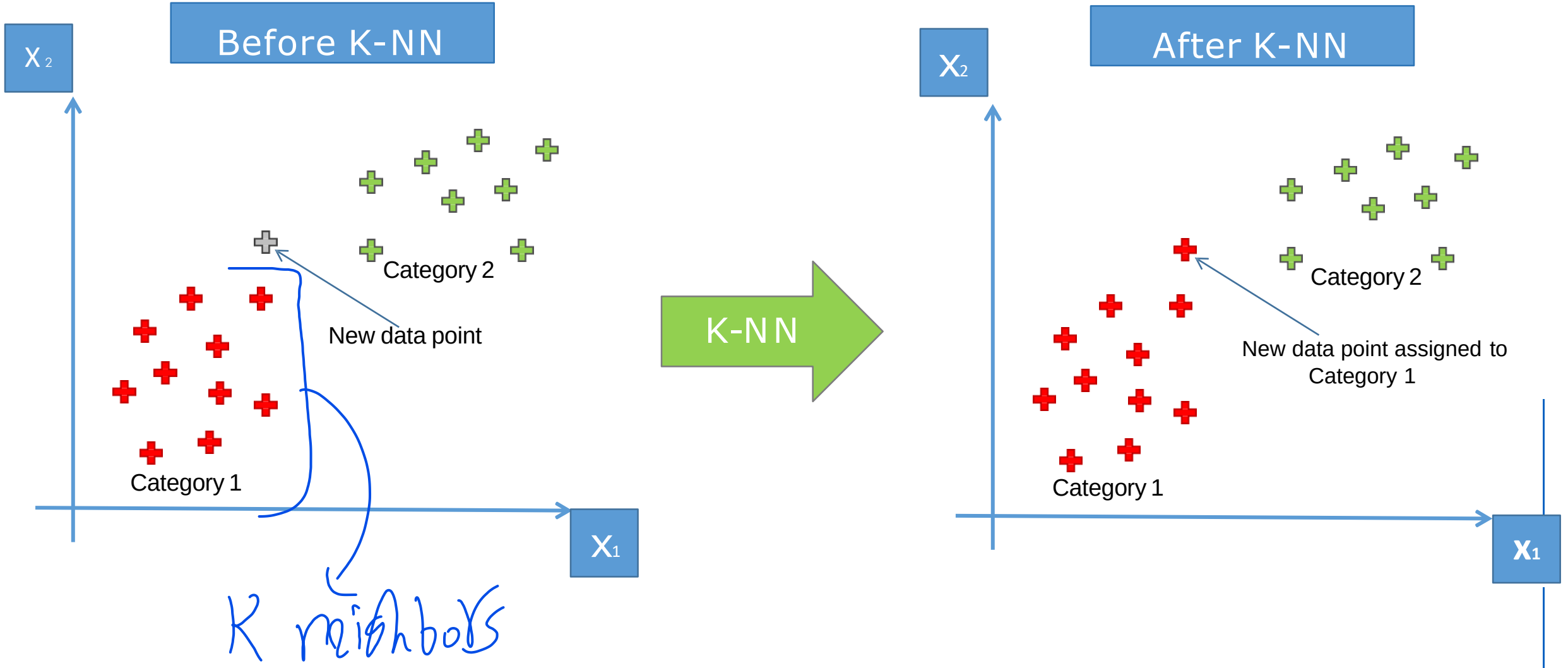
Slides of:
Machine Learning Specialization https://www.coursera.org/specializations/machine-learning-introduction
at Stanford University (**Andrew Ng**)

Grading Distribution (Update)

- Project: (20%)
  - Code “github”, pitching video, presentation.
- Programming Assignment (10%)- DataCamp track
- Midterm exam (20%)
- Final exam (50%)

k-NN Intuition

What k-NN does for you?



How did it do that?

STEP 1: Choose the number K of neighbors

STEP 2: Take the K nearest neighbors of the new data point, according to the Euclidean distance

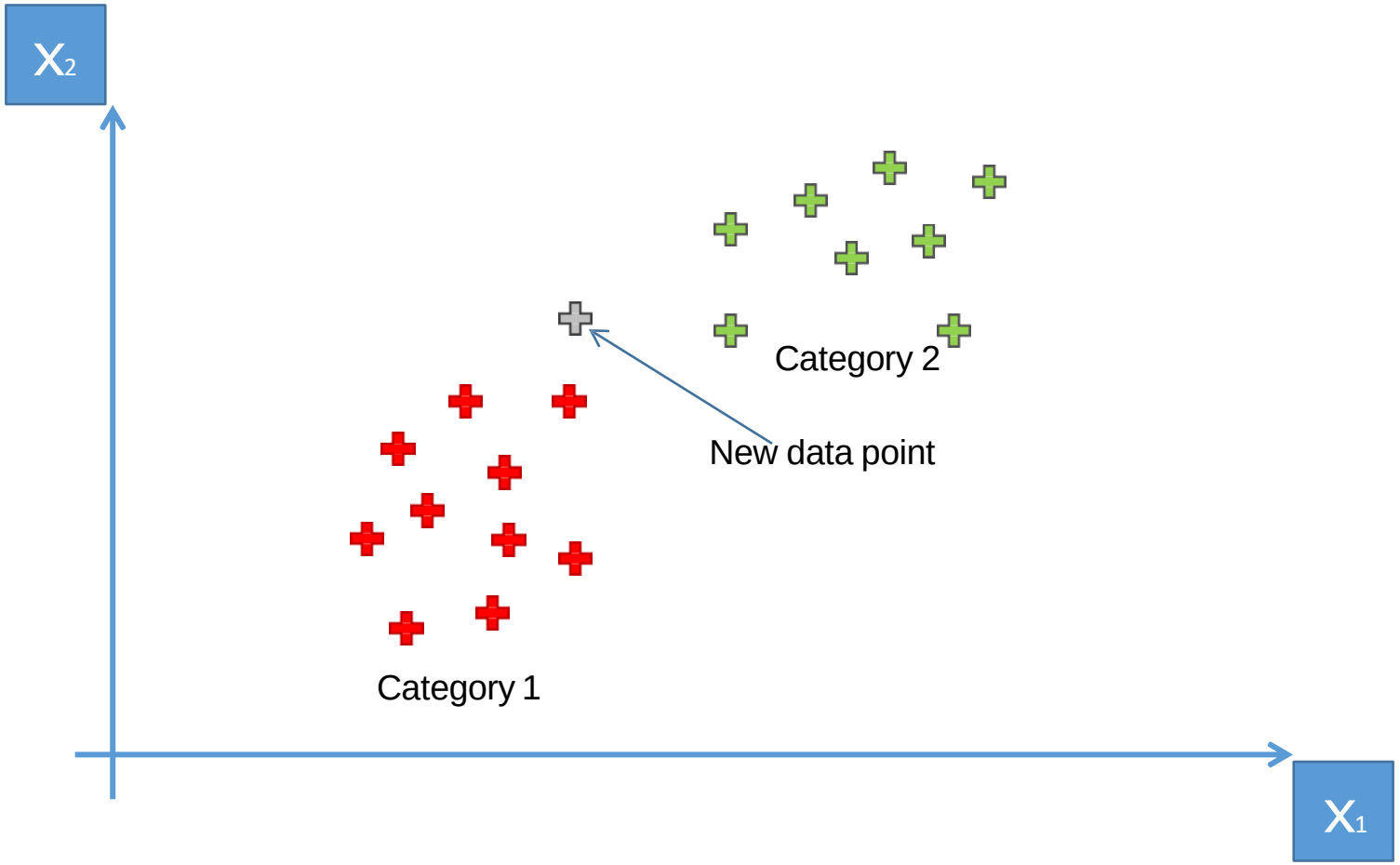
STEP 3: Among these K neighbors, count the number of data points in each category

STEP 4: Assign the new data point to the category where you counted the most neighbors

Your Model is Ready

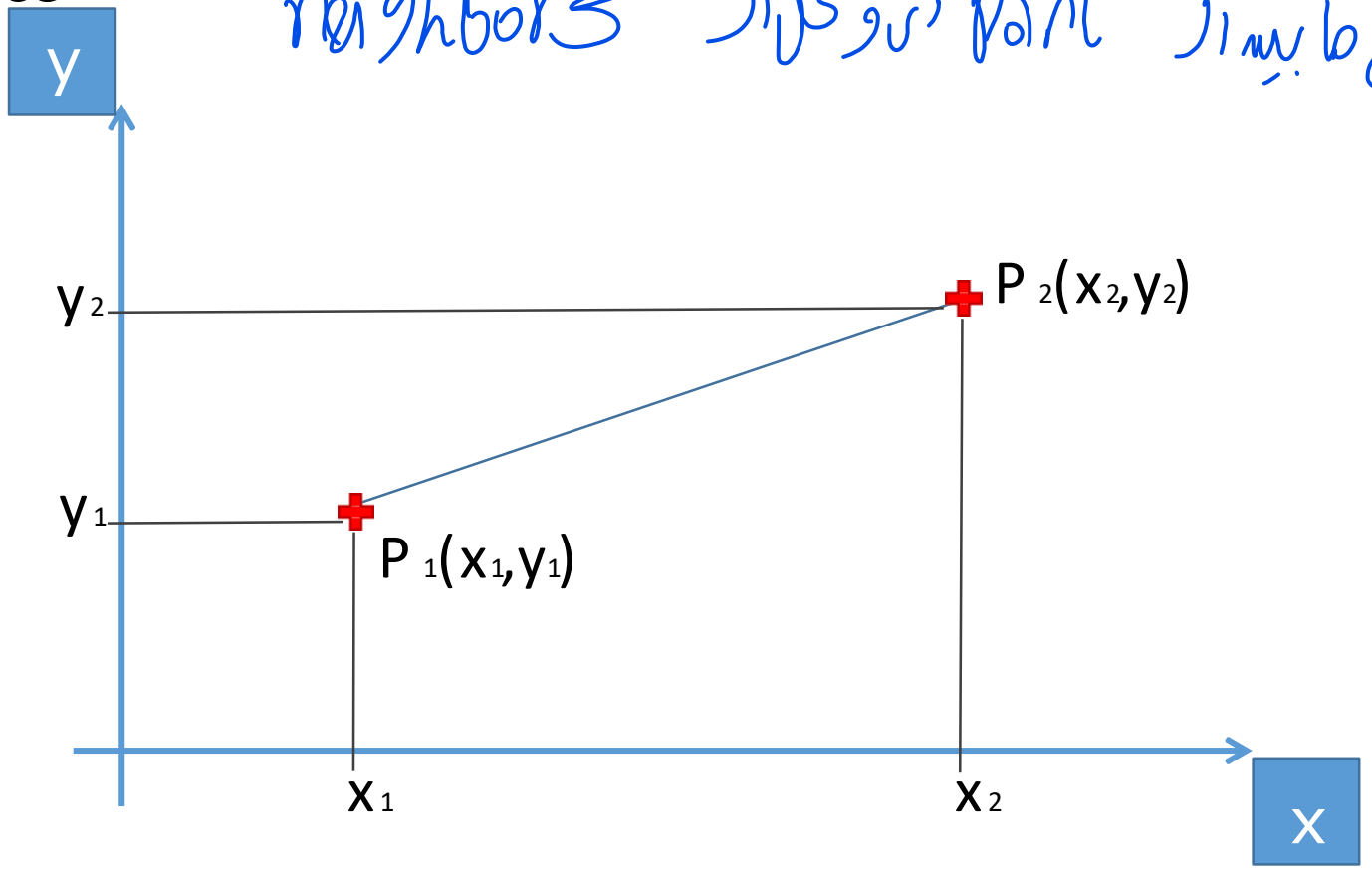
K-NN Algorithm

- STEP 1: Choose the number K of neighbors: K = 5



Euclidean Distance

STEP 2: Take the K nearest neighbors of the new data point, according to the Euclidean distance



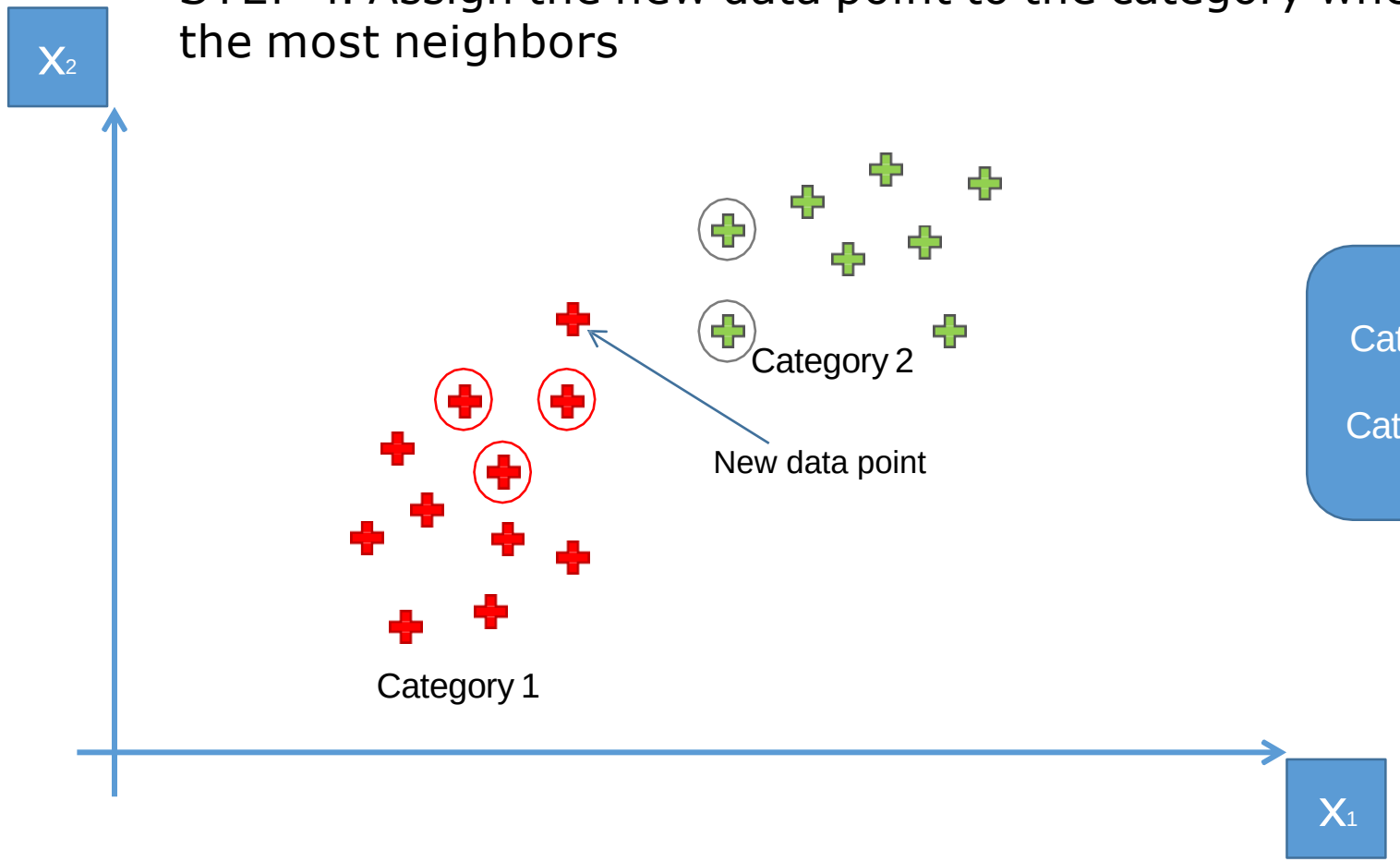
$$\text{Euclidean Distance between } P_1 \text{ and } P_2 = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

K-NN Algorithm

sorting from nearest

STEP 3: Among these K neighbors, count the number of data points in each category

STEP 4: Assign the new data point to the category where you counted the most neighbors

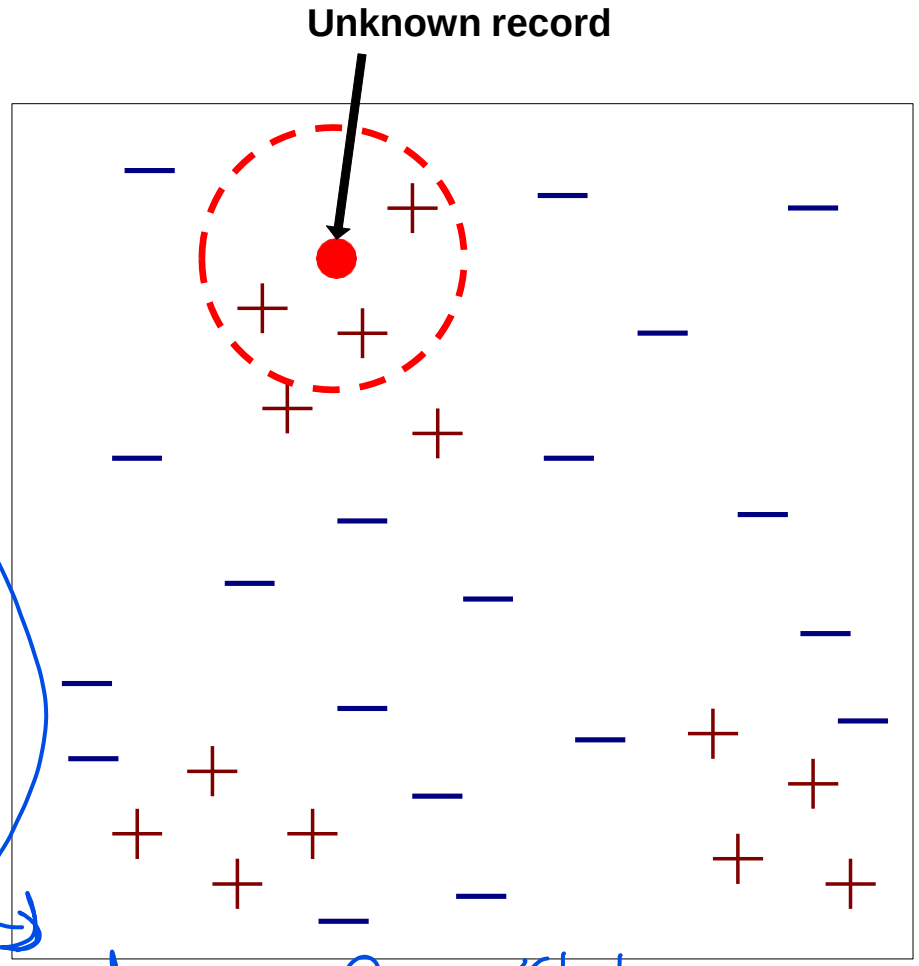


Category 1: 3 neighbors

Category 2: 2 neighbors

Simple Idea …

- Store all training examples
  - Each point is a “vector” of attributes
- Classify new examples based on most “similar” training examples
  - Similar means “closer” in vector space



What’s done in training?

Two Approaches for Learning

Eager learning
(e.g., logistic regression)

- Learn/Train
  - Induce an abstract model from data
- Test/Predict/Classify
  - Apply learned model to new data

Lazy learning
(e.g., kNN)

- Learn/Training
  - Just store data in memory
- Test/Predict/Classify
  - Compare new data to stored data
- Properties
  - Retains all information seen in training
  - Complex hypothesis space
  - Classification can be very slow

kNN Algorithm

Dataset, number of neighbors, example to predict its label

Algorithm 3 KNN-PREDICT($\mathbf{D}$, K, $\hat{x}$)

1: $S \leftarrow [\,]$
2: **for** $n = 1$ **to** N **do** // N number of examples that we have in dataset
3: $S \leftarrow S \oplus \langle \mathrm{d}(x_n, \hat{x}), n \rangle$ // store distance to training example n
4: **end for**
5: $S \leftarrow \textsc{sort}(S)$ // put lowest-distance objects first
6: $\hat{y} \leftarrow 0$
7: **for** $k = 1$ **to** K **do**
8: $\langle dist, n \rangle \leftarrow S_k$ // n this is the kth closest data point
9: $\hat{y} \leftarrow \hat{y} + y_n$ // vote according to the label for the nth training point
10: **end for**
11: **return** $\textsc{sign}(\hat{y})$ // return $+1$ if $\hat{y} > 0$ and -1 if $\hat{y} < 0$

Example

BRIGHTNESS	SATURATION	CLASS
20	35	?

Distance #1

For the first row, d1:

$d1 = \sqrt{(20 - 40)^2 + (35 - 20)^2}$

$= \sqrt{400 + 225}$

$= \sqrt{625}$

$= 25$

5 as the value of **K**, we'll only consider the first five rows. That is:

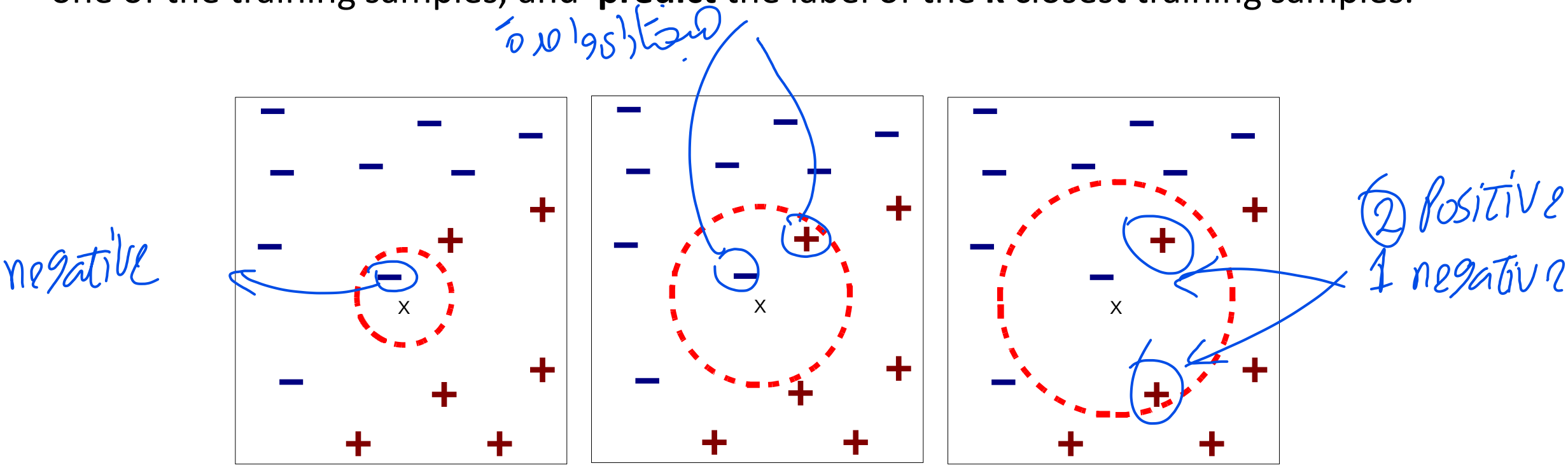
The majority class within the 5 nearest neighbors to the new entry is **Red**. Therefore, we'll classify the new entry as **Red**.

BRIGHTNESS	SATURATION	CLASS
40	20	Red
50	50	Blue
60	90	Blue
10	25	Red
70	70	Blue
60	10	Red
25	80	Blue

BRIGHTNESS	SATURATION	CLASS	DISTANCE
10	25	Red	10
40	20	Red	25
50	50	Blue	33.54
25	80	Blue	45
60	10	Red	47.17
70	70	Blue	61.03
60	90	Blue	68.01

k-Nearest Neighbors (kNN)

The k-nearest neighbor classifier will take a test sample, **compare** it to every single one of the training samples, and **predict** the label of the **k** closest training samples.



(a) 1-nearest neighbor (b) 2-nearest neighbor (c) 3-nearest neighbor

Variants of kNN

Weighted Voting

- Default: all neighbors have equal weight
- Extension: weight votes of neighbors by (inverse) distance
- The intuition behind weighted kNN, is to give more weight to the points which are nearby and less weight to the points which are farther away.

Epsilon-Ball Nearest Neighbors

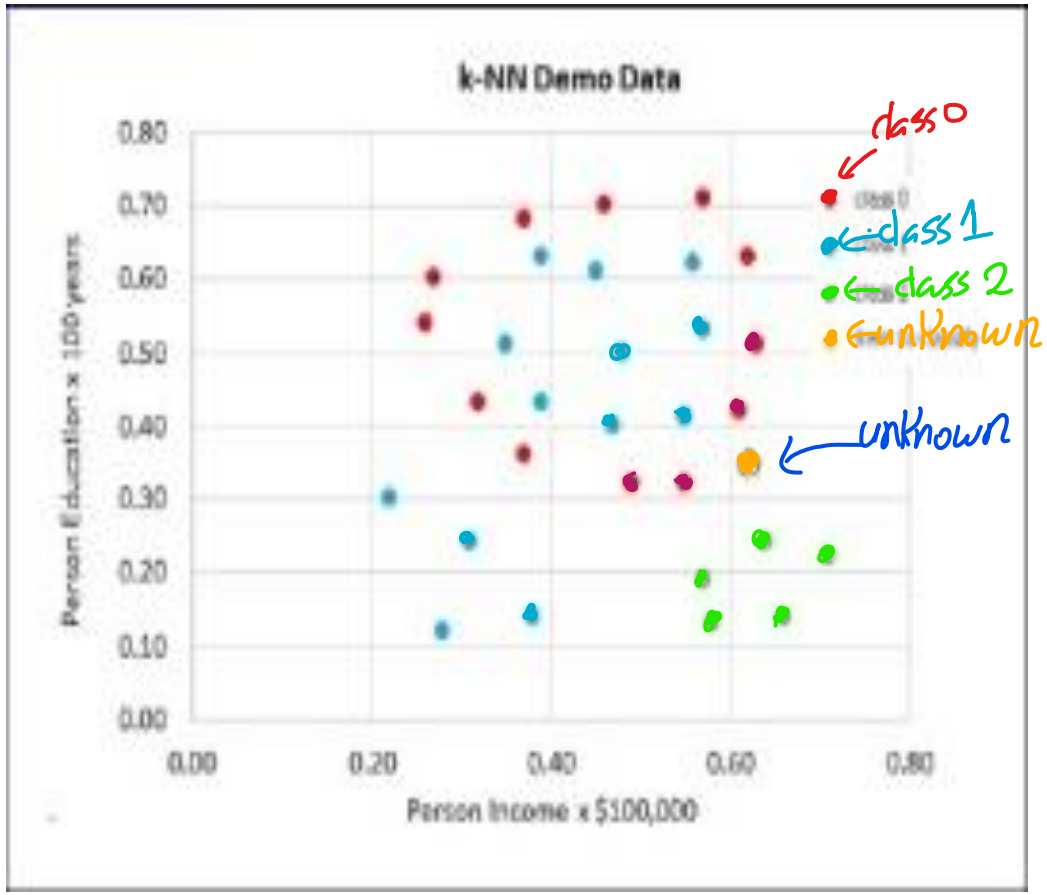
- Same general principle as K-NN, but change the method for selecting which training examples vote
- Instead of using K nearest neighbors, use all examples x such that

$$distance(\hat{x}, x) \leq \varepsilon$$

Weighted k-NN Example

- The item to predict is set as [0.62, 0.35] (colored yellow) .
- The labels can be 0, 1, or 2 (colored red, blue, green).
- The k = 6 closest data points and their distances are:

```
(0.61 0.42) class =  0  dist = 0.0707   1/dist =  14.1421
(0.55 0.32) class =  0  dist = 0.0762   1/dist =  13.1306
(0.55 0.41) class =  1  dist = 0.0922   1/dist =  10.8465
(0.64 0.24) class =  2  dist = 0.1118   1/dist =   8.9443
(0.49 0.32) class =  0  dist = 0.1334   1/dist =   7.4953
(0.71 0.22) class =  2  dist = 0.1581   1/dist =   6.3246
```

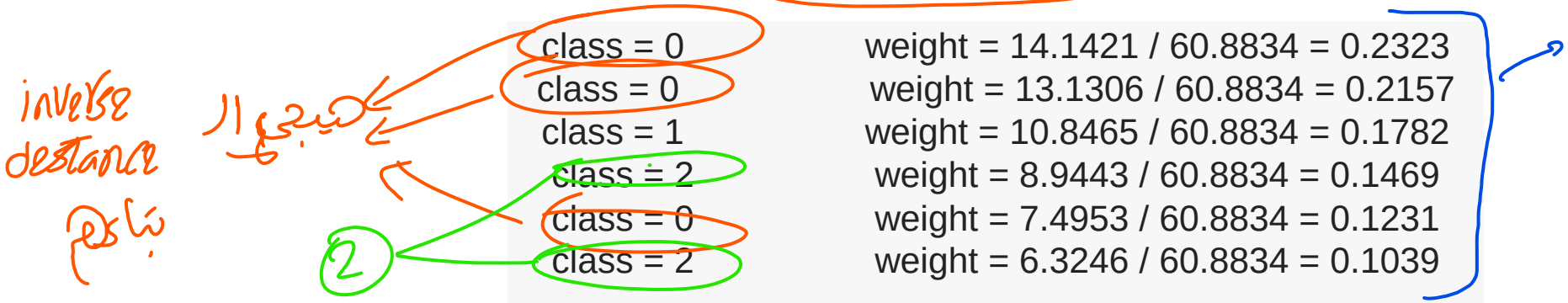


The sum of the inverse distances is 14.1421 + . . + 6.3246 = 60.8834.

https://jamesmccaffrey.wordpress.com/2022/05/03/weighted-k-nearest-neighbors-classification-example-using-python/

Weighted k-NN Example

- The voting weights are the inverse distances divided by their sum:



- The voting for each class is:

class 0 : 0.2323 + 0.2157 + 0.1231 = 0.5711
class 1 : = 0.1782
class 2 : 0.1469 + 0.1039 = 0.2508

- Because class 0 has the most voting weight, the prediction is that the unlabeled data item is class 0.

K-NN Components

Components of a KNN Classifier

- Distance metric
  - How do we measure distance between instances?
  - Determines the layout of the example space

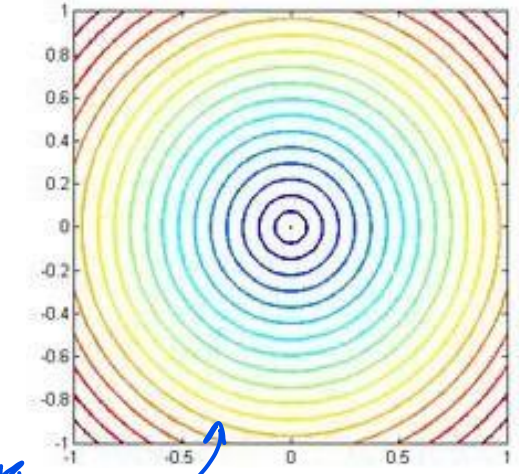
- The k hyperparameter
  - How large a neighborhood should we consider?
  - Determines the complexity of the hypothesis space

Very problem-dependent.
Must try them all out and see what works best.

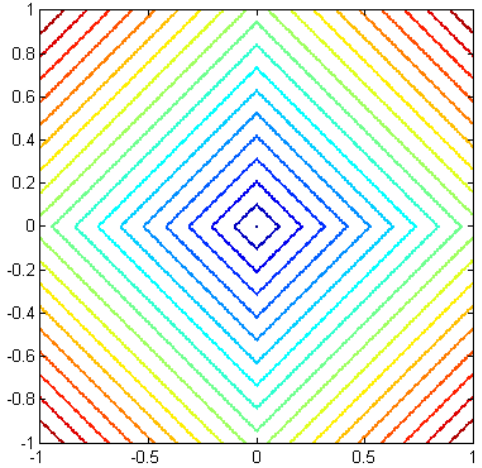
Distance Metrics

- We can use any distance function to select nearest neighbors.
- Different distances yield different neighborhoods

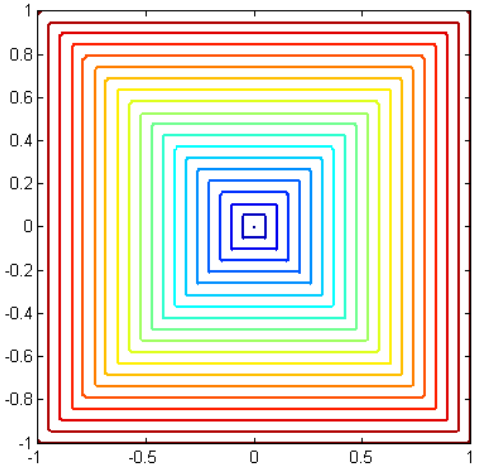
L2 distance
(= Euclidean distance)



L1 distance
(= Manhattan distance)



Max norm



Manhattan Distance (L1)

- The Manhattan distance in a 2-dimensional space is given as:

$$d_1(p, q) = |p_1 - q_1| + |p_2 - q_2|$$

- For instance:

  If we have two points A (3,6) and B (-5,4).

  ➔ L1(A,B)= |3-(-5)|+|6 – 4|= 8 + 2= 10.

- And the generalized formula for an n-dimensional space is given as:

$$d_1(p, q) = \sum_{i=1}^{n} |p_i - q_i|$$

- Where, n = number of dimensions and pi, qi = data points

Euclidean Distance (L2)

- The Euclidean distance in a 2-dimensional space is given as:

$$d_2(p,q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2}$$

- For instance:

  If we have two points A (3,6) and B (-5,4).

  ➔ L2 (A,B)= $[(3-(-5))^2 + (6-4)^2]^{1/2} = [16 + 4]^{1/2} = 4.472$.

- And the generalized formula for an n-dimensional space is given as:

$$d_2(p,q) = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2}$$

- Where, n = number of dimensions and pi, qi = data points

Distance Metric to compare images

- **Manhattan distance (L1):** Compare the images pixel by pixel and add up all the absolute differences.
- Given two images I_1 and I_2

$$d_1(I_1, I_2) = \sum_p |I_1^p - I_2^p|$$

test image					training image					pixel-wise absolute value differences				
56	32	10	18	−	10	20	24	17	=	46	12	14	1	
90	23	128	133		8	10	89	100		82	13	39	33	
24	26	178	200		12	16	178	170		12	10	0	30	
2	0	255	220		4	32	233	112		2	32	22	108	add → 456

Distance Metric to compare images

- **Euclidean distance (L2):** Computing the pixel-wise difference as before, but this time we square all of them, add them up and finally take the square root.

- Given two images I1 and I2: $$d_2(I_1, I_2) = \sqrt{\sum_p (I_1^p - I_2^p)^2}$$

نفس الكلام بس مربعهم وبجمعهم واحطهم تحت الجذر

The k hyper-parameter

how many neighbors do we use?

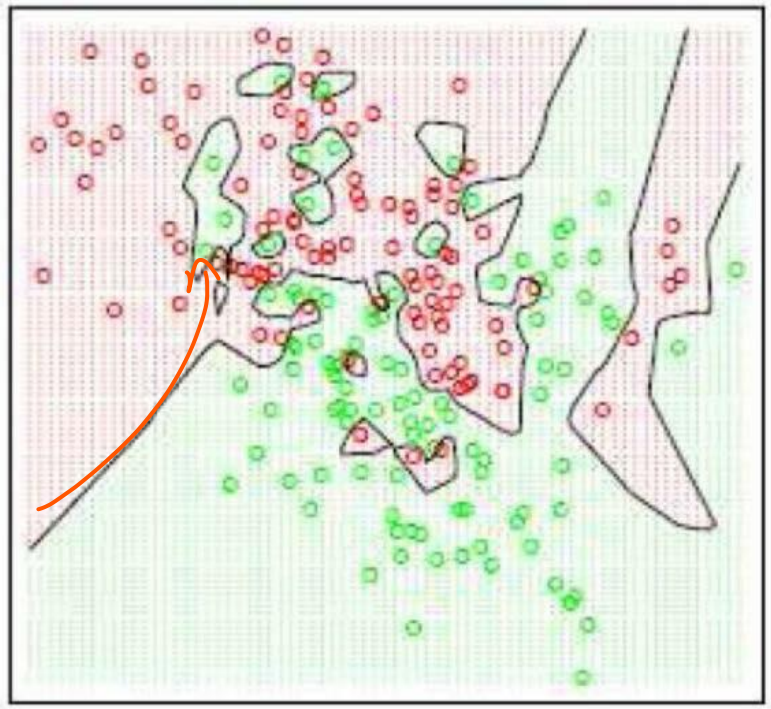
Tunes the complexity of the hypothesis space

– If k = 1, every training example has its own neighborhood

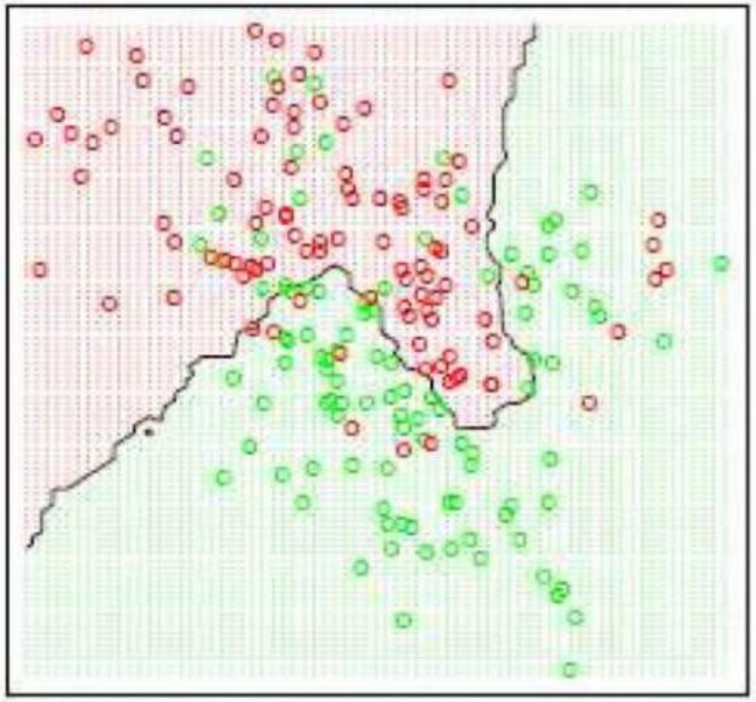
– If k = N, the entire feature space is one neighborhood!

• Higher k yields smoother decision boundaries

k=1 → more overfitting



k=15

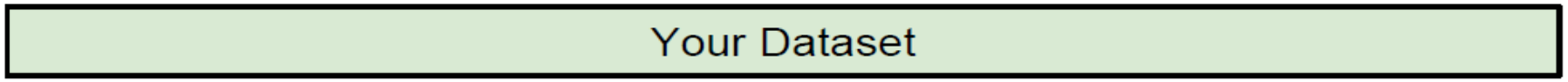


How would you set k in practice?

Setting Hyperparameters

Idea #1: Choose hyperparameters that work best on the data

BAD: K = 1 always works perfectly on training data



Idea #2: Split data into **train** and **test**, choose hyperparameters that work best on test data

BAD: No idea how algorithm will perform on new data

Idea #3: Split data into **train**, **val**, and **test**; choose hyperparameters on val and evaluate on test

Better!

60% 20% 20%

final accuracy

Validation Set

- The idea is to split the training set in two sets: a slightly smaller training set, and what we call a **validation set**.

- This validation set is essentially used as
  - a **fake test set** to **tune the hyper-parameters**.

- After we found the best hyper-parameters on validation set,
  - Stick with these values and
  - **Evaluate once on the actual test set (**for measuring the **generalization** of your classifier**)**.

Cross-Validation

- For small datasets, sometimes we use a more sophisticated technique for hyperparameter tuning called **cross-validation**.

  - **Instead of arbitrarily picking** the first data points to be the validation set and rest training set,
  - **Get a better and less noisy estimate** of how well hyperparameters work by iterating over different validation sets and averaging the performance across these.

Setting Hyperparameters: k fold Cross-Validation

- For example: 5-fold cross-validation

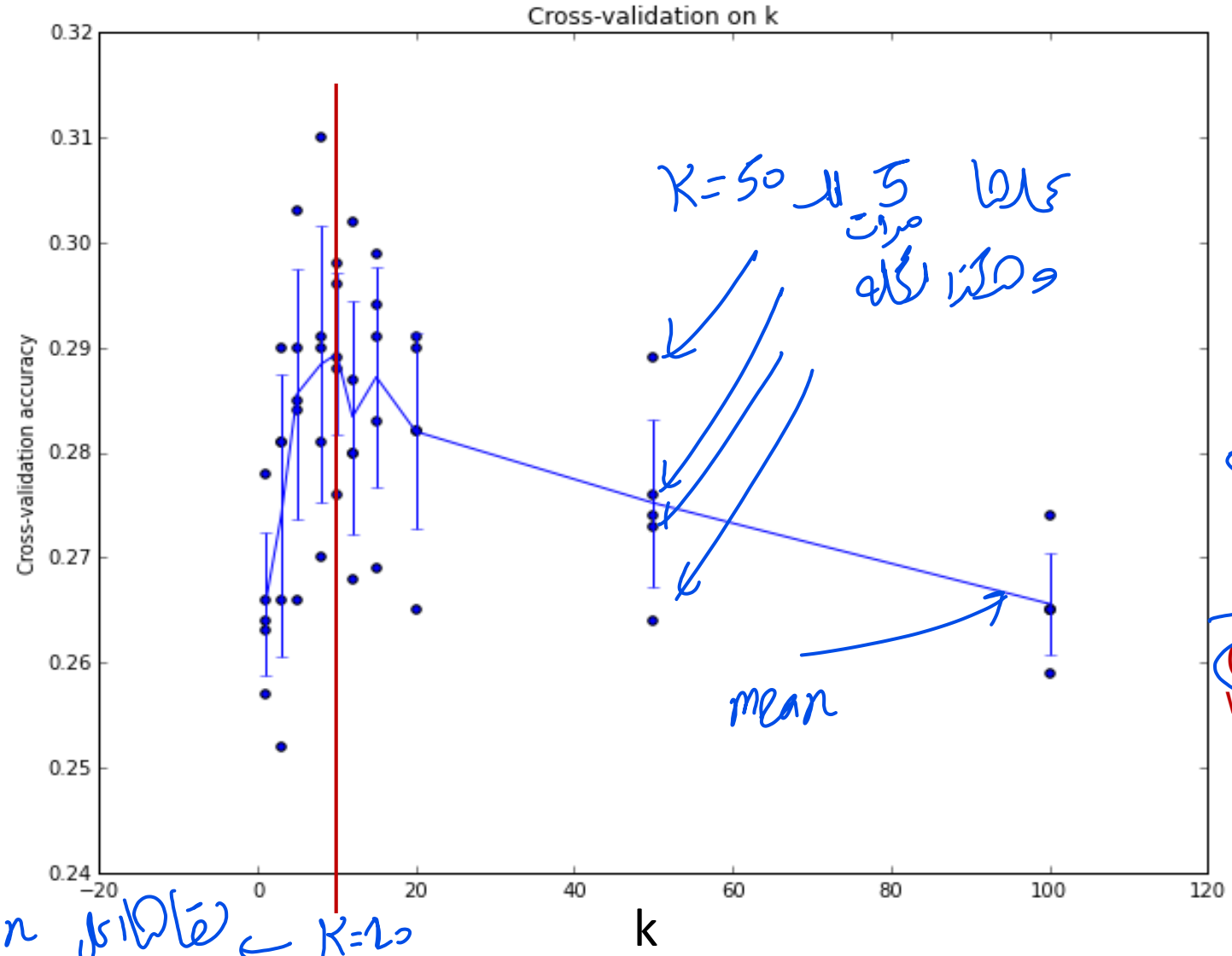
1. Split the training data into 5 equal folds (parts),
2. Use 4 of them for training, and 1 for validation.
3. Iterate over which fold is the validation fold, and evaluate the performance,
4. Finally average the performance across the different folds.

Idea #4: **Cross-Validation**: Split data into **folds**, try each fold as validation and average the results

fold 1	fold 2	fold 3	fold 4	fold 5	test
fold 1	fold 2	fold 3	fold 4	fold 5	test
fold 1	fold 2	fold 3	fold 4	fold 5	test

Useful for small datasets, but not used too frequently in deep learning

Example: Cross Validation on k



Example of
5-fold cross-validation
for the value of **k**.

Each point: single outcome.

The line goes through the mean,

Bars indicated standard deviation

(Seems that k ~= 10 works best for this data)

K-NN Pros and Cons

Feature Scale

- Example:
  - height of a person may vary from 1.5m to 1.8m
  - weight of a person may vary from 90lb to 300lb
  - income of a person may vary from $10K to $1M

What's the problem here?

- Attributes may have to be scaled to prevent distance measures from being dominated by one of the attributes.

Disadvantage of K-NN

- It is the computationally **expensive in testing phase** which is impractical in industry settings. Predict O(N)
  - This is bad: we want classifiers that are fast at prediction, slow in training is ok.
- KNN can suffer from **skewed class distributions**.
  - For example, if a certain class is very frequent in the training set, it will tend to dominate the majority voting of the new example (large number = more common).
- Finally, the accuracy of KNN can be severely degraded with high-dimension data. **For that, it is never used on images**.

for high dimensions the accuracy decrease

images

Advantage of K-NN

- Simple to understand and easy to implement.
- With zero to little training time, Train O(1)
- KNN works just as easily with **multiclass** data sets (more than 2 classes).

Thanks