



# **COMP9321:**

## **Data services engineering**

### **Week 7: Classification**

**Term1, 2020**

**By Mortada Al-Banna, CSE UNSW**

# Machine Learning for Data Analytics

1. **Define** and **Initialize** a Model
2. **Train** your Model (using your training dataset)
3. **Validate** the Model (by prediction using your test dataset)
4. Use it: **Explore** or **Deploy** as a web service
5. **Update** and **Revalidate**

# Supervised Learning

We are given input samples ( $X$ ) and output samples ( $y$ ) of a function  $y = f(X)$ .

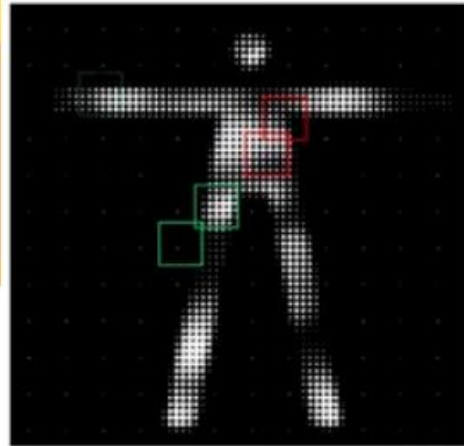
We would like to “learn”  $f$ , and evaluate it on new data.

- **Classification:**  $y$  is discrete (class labels).
- **Regression:**  $y$  is continuous, e.g. linear regression.

# Classification

- Supervised Learning
- You need the data labelled with the correct answer to train the algorithm
- Trained classifiers then can map input data to a category.

# Classification Examples



# k-Nearest Neighbour (k-NN)

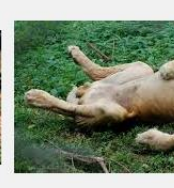
The KNN classifier is a **non parametric** and **instance-based** learning algorithm.

**Non-parametric** means it makes no explicit assumptions about the functional form of how the prediction is made, avoiding the dangers of mismodeling the underlying distribution of the data.

**Instance-based** learning means that our algorithm doesn't explicitly learn a model. Instead, it chooses to memorize the training instances which are subsequently used as “knowledge” for the prediction phase. Concretely, this means that only when a query to our database is made (i.e. when we ask it to predict a label given an input), will the algorithm use the training instances to spit out an answer.

# k-Nearest Neighbors

Given a query item:  
Find k closest matches  
in a labeled dataset ↓





# k-Nearest Neighbors

Given a query item:

Find k closest matches



Return the most

Frequent label





# k-Nearest Neighbors

$k = 3$  votes for “cat”





# k-Nearest Neighbors

2 votes for cat,

1 each for Buffalo,

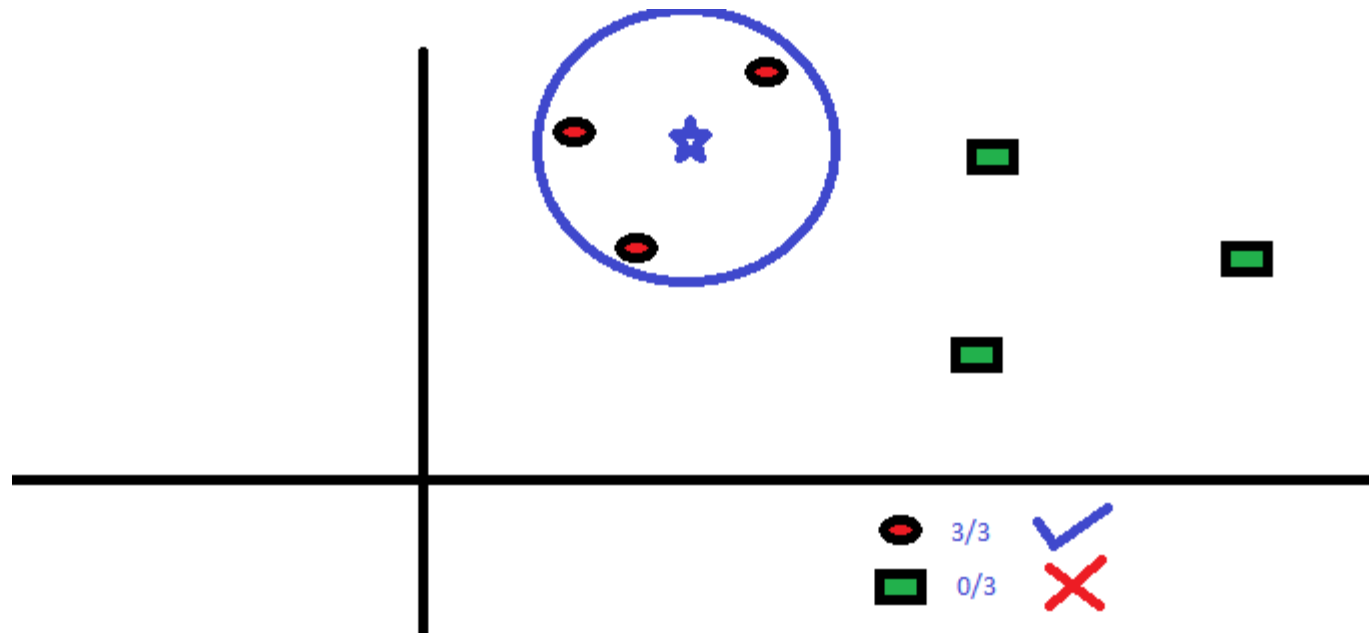
## Deer, Lion



Cat wins...

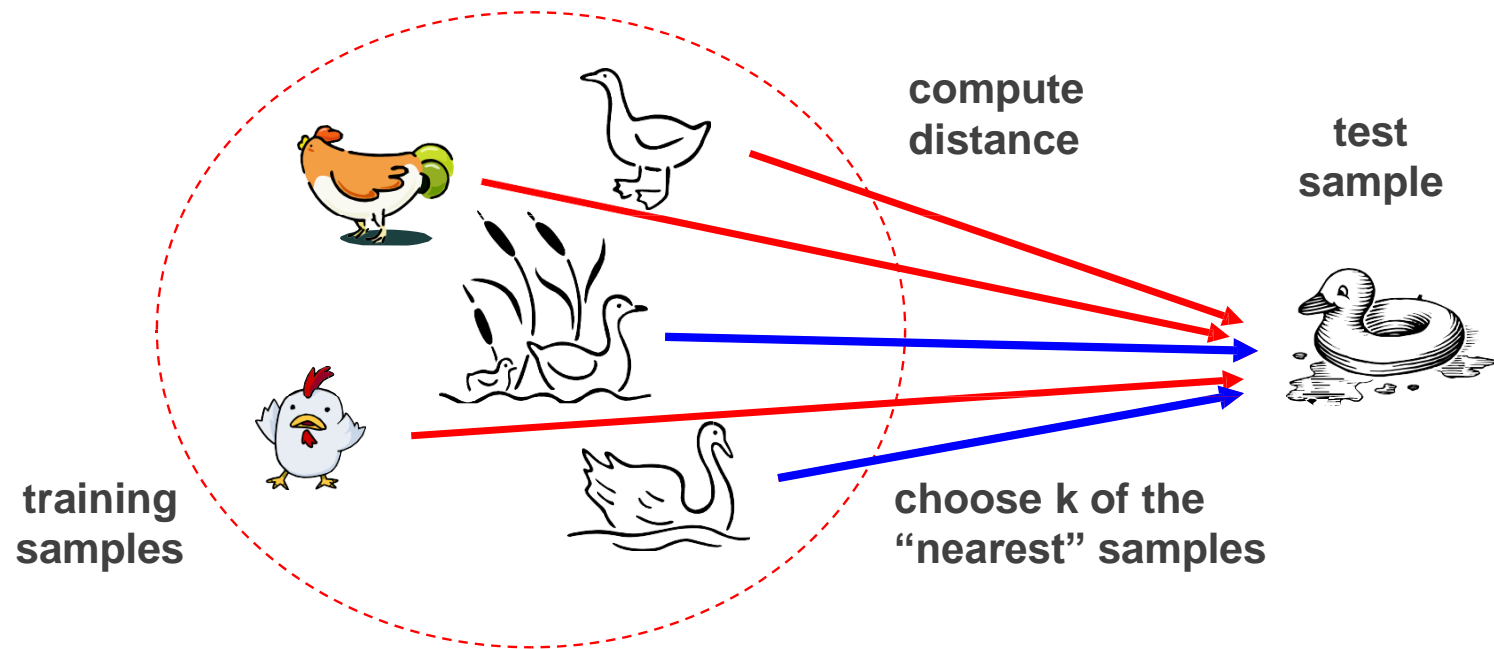


# k-Nearest Neighbour (k-NN)



# Nearest neighbor classifiers

- Basic idea:
  - If it walks like a duck, quacks like a duck, then it's probably a duck



# k- Nearest Neighbour Classifier Algorithm

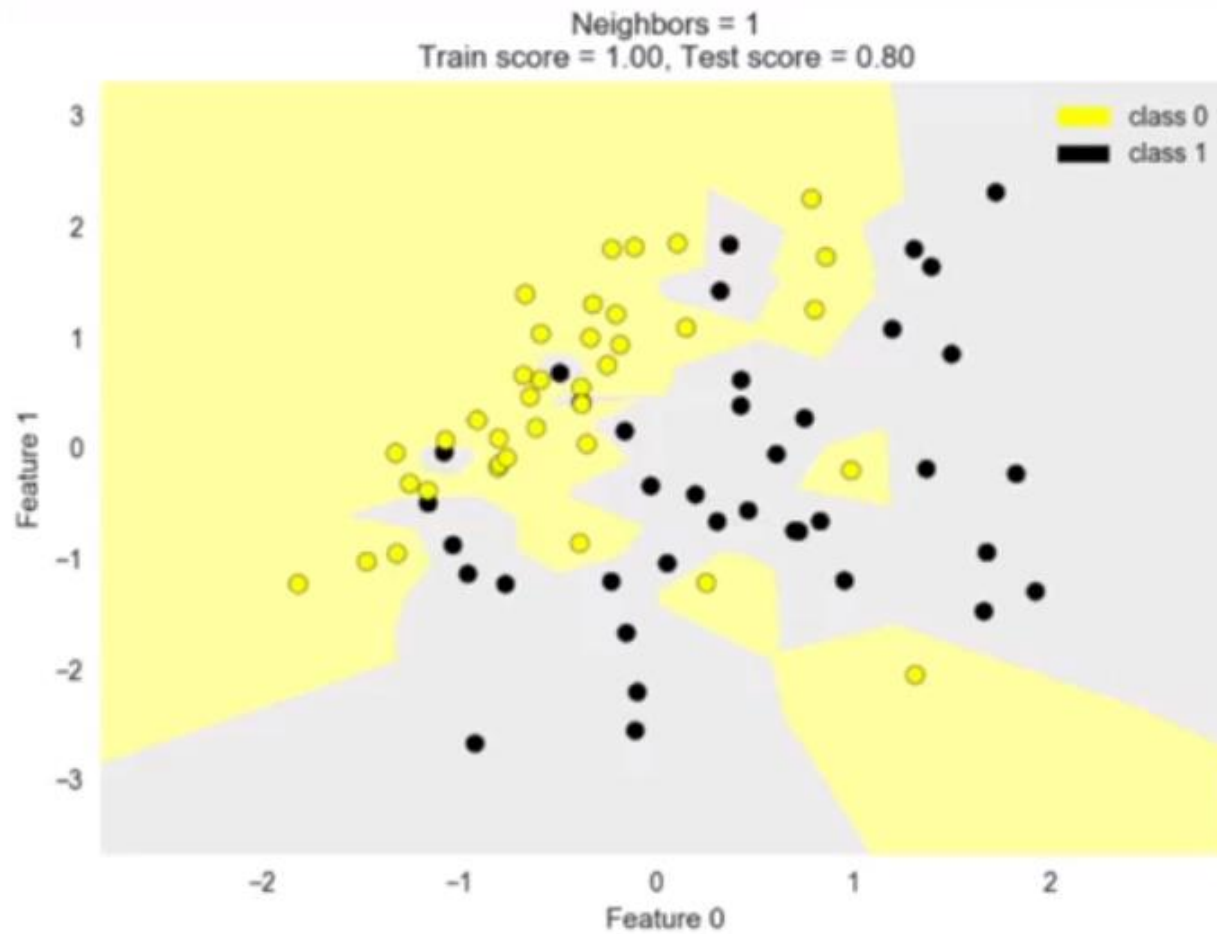
Give a training set  $X_{\text{train}}$  with labels  $y_{\text{train}}$  and given a new instance  $x_{\text{test}}$  to be classified:

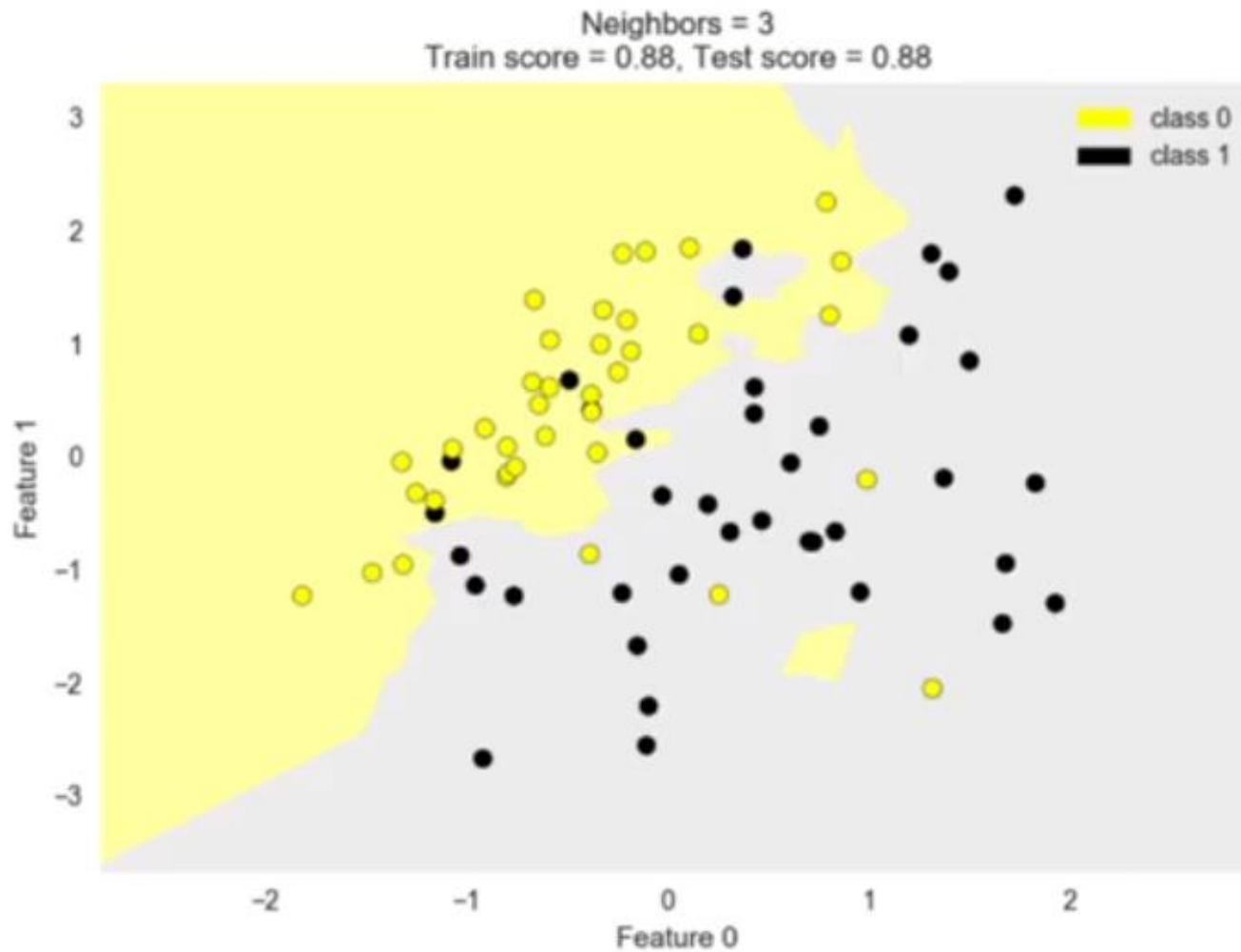
1. Find the most similar instances (let's call them  $X_{\text{NN}}$ ) to  $x_{\text{test}}$  that are in  $X_{\text{train}}$ .
2. Get the labels  $y_{\text{NN}}$  for the instances in  $X_{\text{NN}}$ .
3. Predict the label for  $x_{\text{test}}$  by combining the labels  $y_{\text{NN}}$  (e.g., using majority rule)

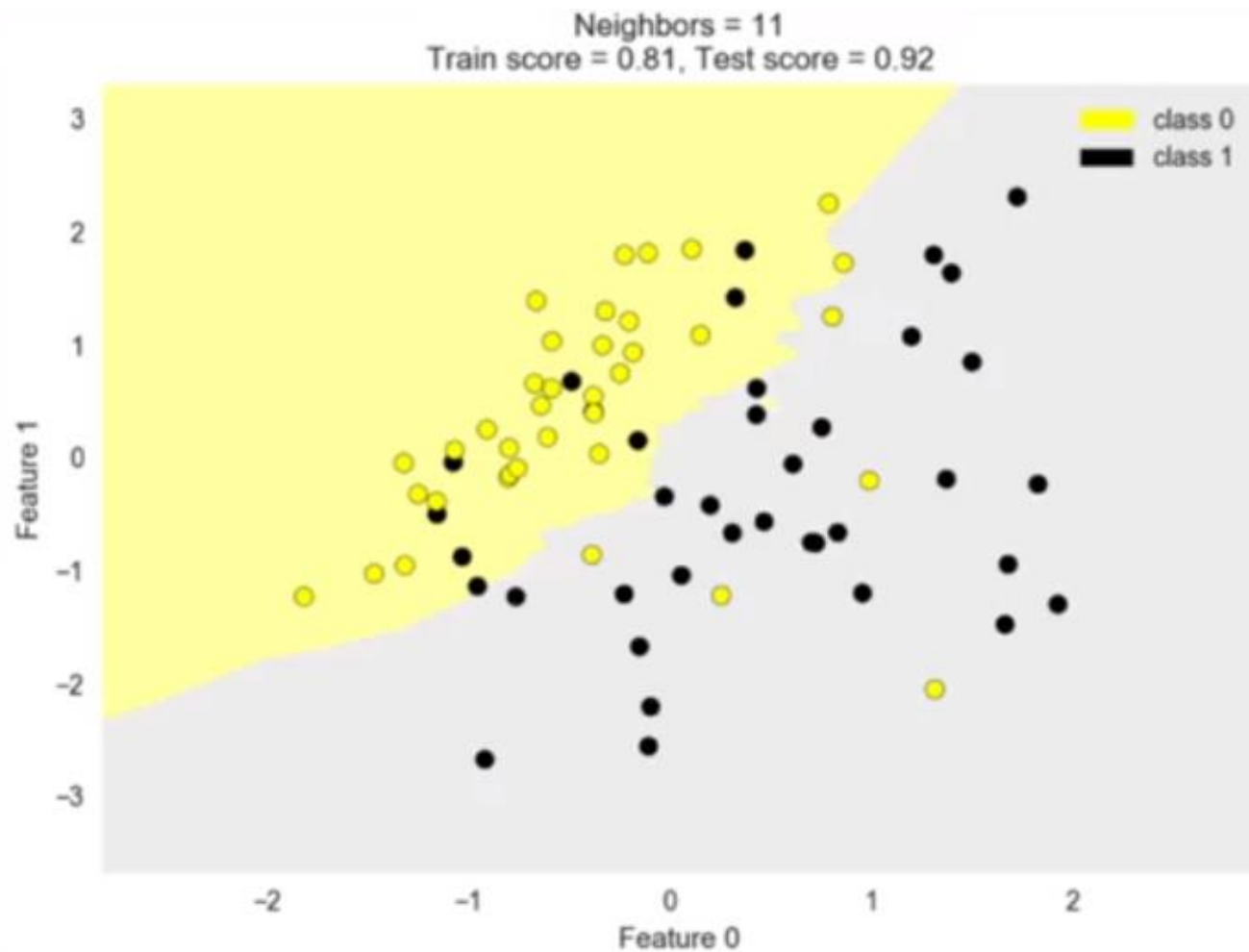


# Nearest Neighbour Need Four things Specified

1. A distance Metric (e.g., Euclidean)
2. How many nearest neighbours to look at (e.g., Five)
3. Optional Weighting function on the neighbours points (e.g., closer points are weighted higher than farther points)
4. How to aggregate the classes of neighbours points (e.g., simple majority voting)

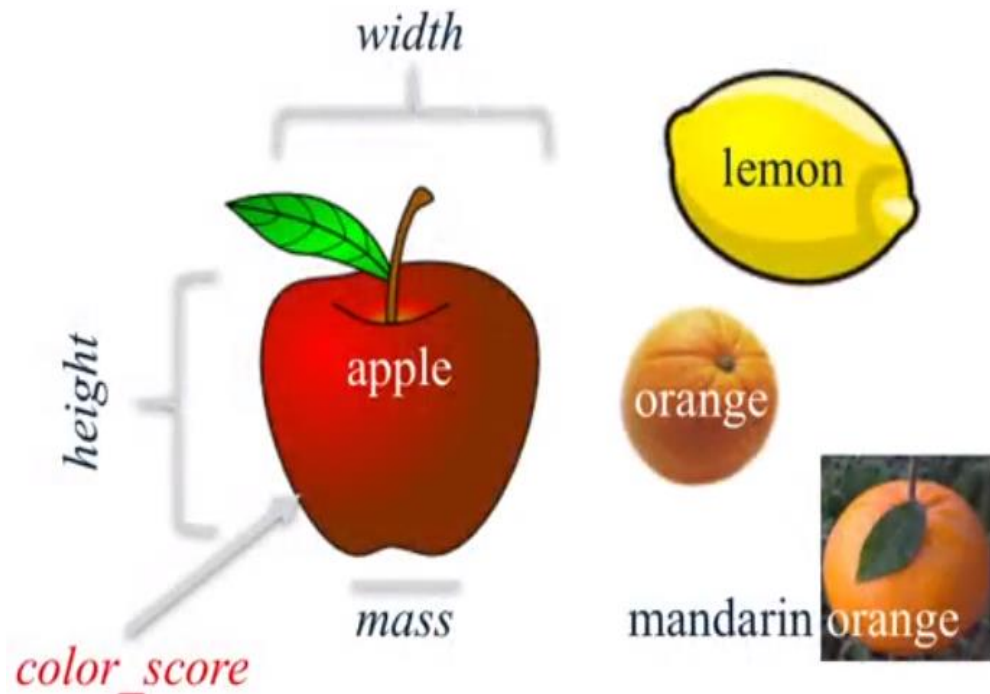






# Classification Data Set Example

## The Fruit Dataset



	fruit_label	fruit_name	fruit_subtype	mass	width	height	color_score
0	1	apple	granny_smith	192	8.4	7.3	0.55
1	1	apple	granny_smith	180	8.0	6.8	0.59
2	1	apple	granny_smith	176	7.4	7.2	0.60
3	2	mandarin	mandarin	86	6.2	4.7	0.80
4	2	mandarin	mandarin	84	6.0	4.6	0.79
5	2	mandarin	mandarin	80	5.8	4.3	0.77
6	2	mandarin	mandarin	80	5.9	4.3	0.81
7	2	mandarin	mandarin	76	5.8	4.0	0.81
8	1	apple	braeburn	178	7.1	7.8	0.92
9	1	apple	braeburn	172	7.4	7.0	0.89
10	1	apple	braeburn	166	6.9	7.3	0.93
11	1	apple	braeburn	172	7.1	7.6	0.92
12	1	apple	braeburn	154	7.0	7.1	0.88
13	1	apple	golden_delicious	164	7.3	7.7	0.70
14	1	apple	golden_delicious	152	7.6	7.3	0.69
15	1	apple	golden_delicious	156	7.7	7.1	0.69
16	1	apple	golden_delicious	156	7.6	7.5	0.67

fruit\_data\_with\_colors.txt

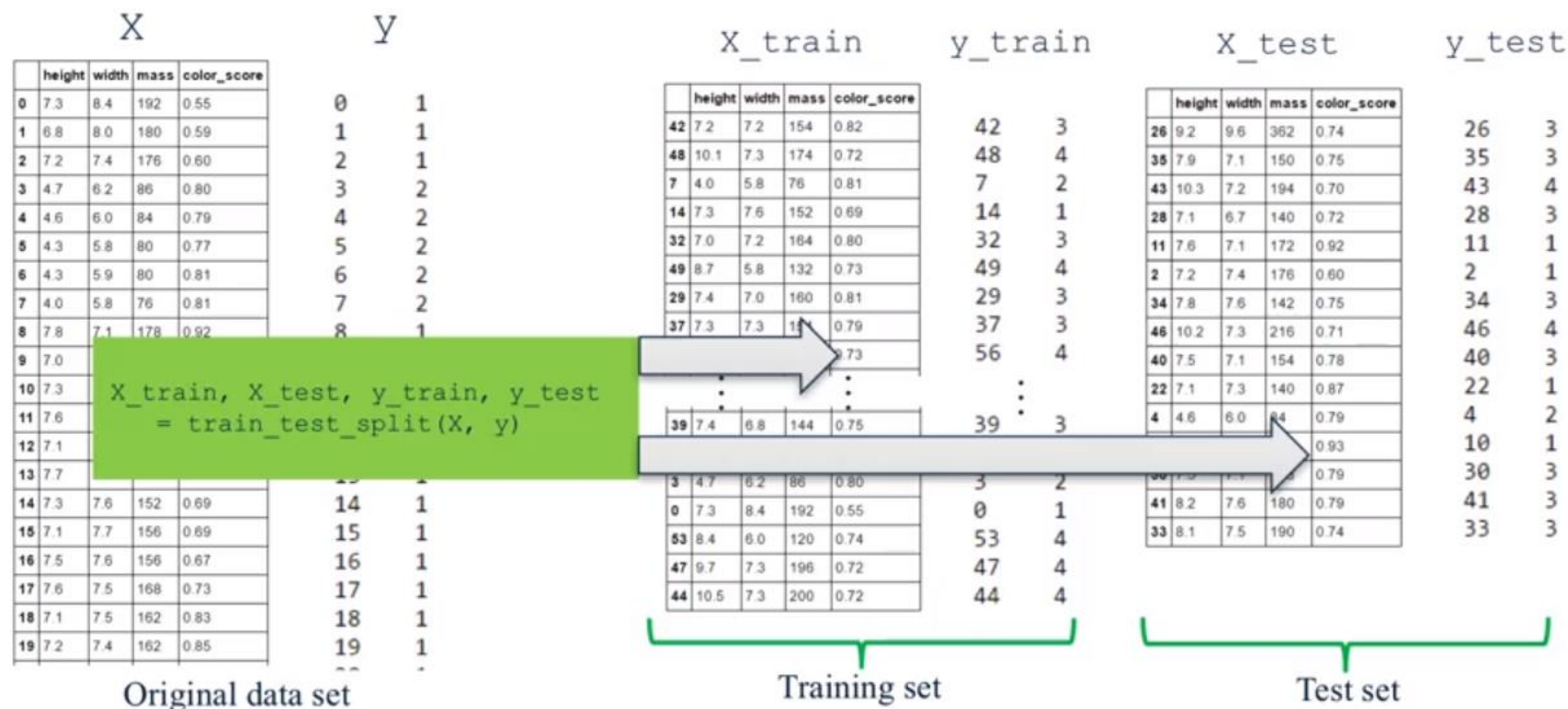
Credit: Original version of the fruit dataset created by Dr. Iain Murray, Univ. of Edinburgh



# Data As a Table

	fruit_label	fruit_name	fruit_subtype	mass	width	height	color_score
0	1	apple	granny_smith	192	8.4	7.3	0.55
1	1	apple	granny_smith	180	8.0	6.8	0.59
2	1	apple	granny_smith	176	7.4	7.2	0.60
3	2	mandarin	mandarin	86	6.2	4.7	0.80
4	2	mandarin	mandarin	84	6.0	4.6	0.79
5	2	mandarin	mandarin	80	5.8	4.3	0.77
6	2	mandarin	mandarin	80	5.9	4.3	0.81
7	2	mandarin	mandarin	76	5.8	4.0	0.81
8	1	apple	braeburn	178	7.1	7.8	0.92
9	1	apple	braeburn	172	7.4	7.0	0.89
10	1	apple	braeburn	166	6.9	7.3	0.93
11	1	apple	braeburn	172	7.1	7.6	0.92
12	1	apple	braeburn	154	7.0	7.1	0.88
13	1	apple	golden_delicious	164	7.3	7.7	0.70
14	1	apple	golden_delicious	152	7.6	7.3	0.69
15	1	apple	golden_delicious	156	7.7	7.1	0.69
16	1	apple	golden_delicious	156	7.6	7.5	0.67
17	1	apple	golden_delicious	168	7.5	7.6	0.73
18	1	apple	cripps_pink	162	7.5	7.1	0.83
19	1	apple	cripps_pink	162	7.4	7.2	0.85

# Training Set and Test Set



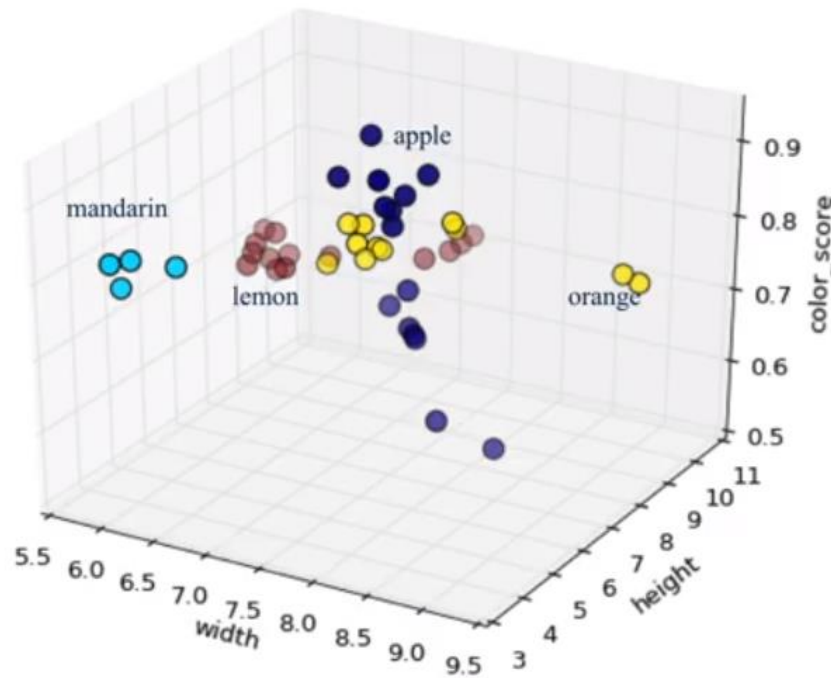
# Always Remember to inspect your Data

## Examples of incorrect or missing feature values

	fruit_label	fruit_name	fruit_subtype	mass	width	height	color_score
0	1	apple	granny_smith	192	8.4	7.3	0.55
1	1	apple	granny_smith	180	8.0	6.8	0.59
2	1	apple	granny_smith	176	7.4	7.2	192
3	2	mandarin	mandarin	86	6.2	4.7	0.80
4	2	mandarin	mandarin	84	6.0	4.6	0.79
5	2	mandarin	apple	80	5.8	4.3	0.77
6	2	mandarin	mandarin	80	5.9	4.3	0.81
7	2	mandarin	mandarin	76	5.8	4.0	0.81
8	1	apple	braeburn	78	7.1	7.8	0.92
9	1	apple	braeburn		7.4	7.0	0.89
10	1	apple	braeburn		6.9	7.3	0.93
11	1	apple	braeburn		7.1	7.6	0.92
12	1	apple	braeburn		7.0	7.1	0.88
13	1	apple	golden_delicious	161	7.3	7.7	0.70
14	1	apple	golden_delicious	152	7.6	7.3	0.69

# Plot your Data

## A three-dimensional feature scatterplot



# Choosing the $k$ in $k$ -Nearest Neighbors

How do we choose  $k$ ?

Larger  $k$  may lead to better performance

But if we set  $k$  too large we may end up looking at samples that are not neighbors (are far away from the query)

We can use cross-validation to find  $k$

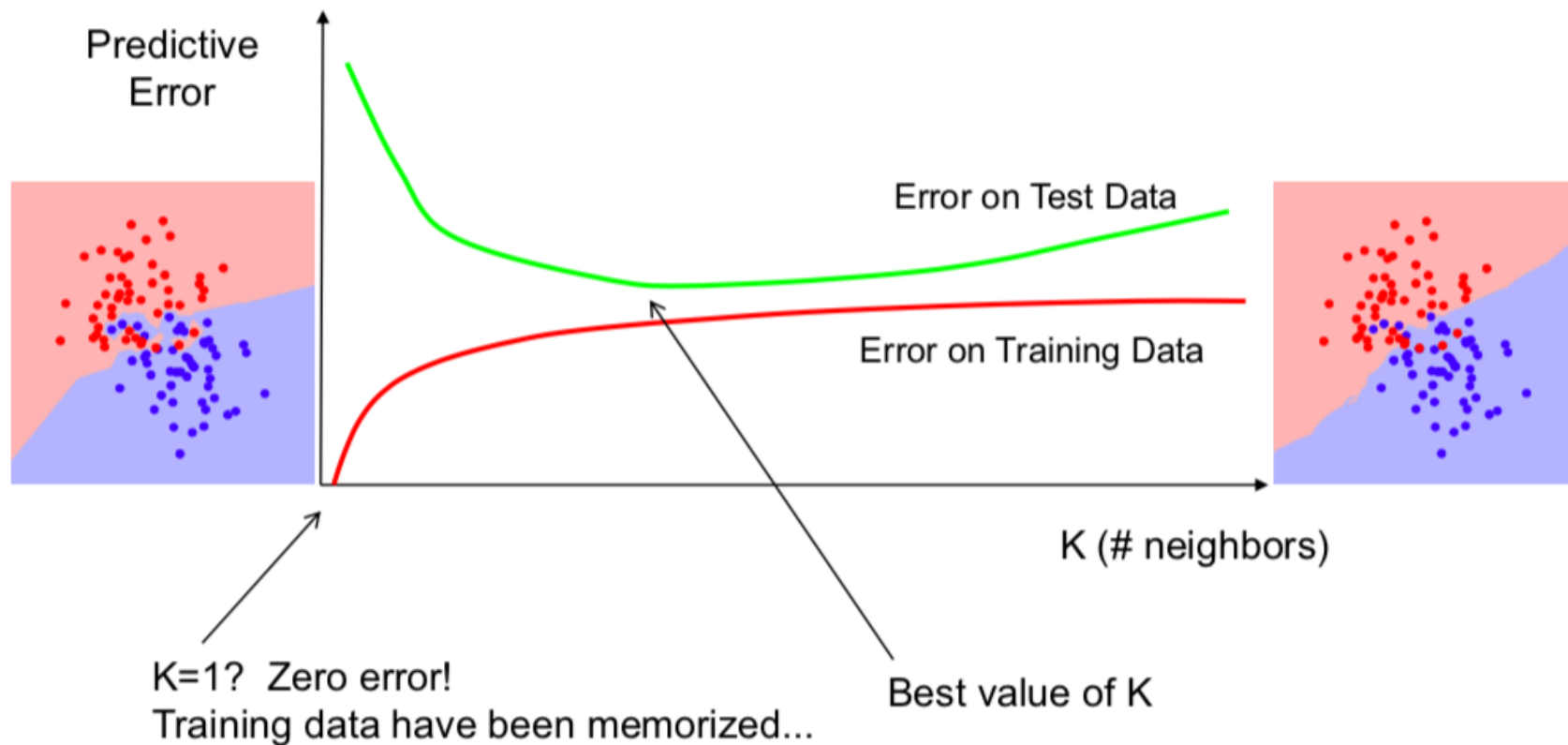
Rule of thumb is  $k < \sqrt{n}$ , where  $n$  is the number of training examples

[Slide credit: O. Veksler]



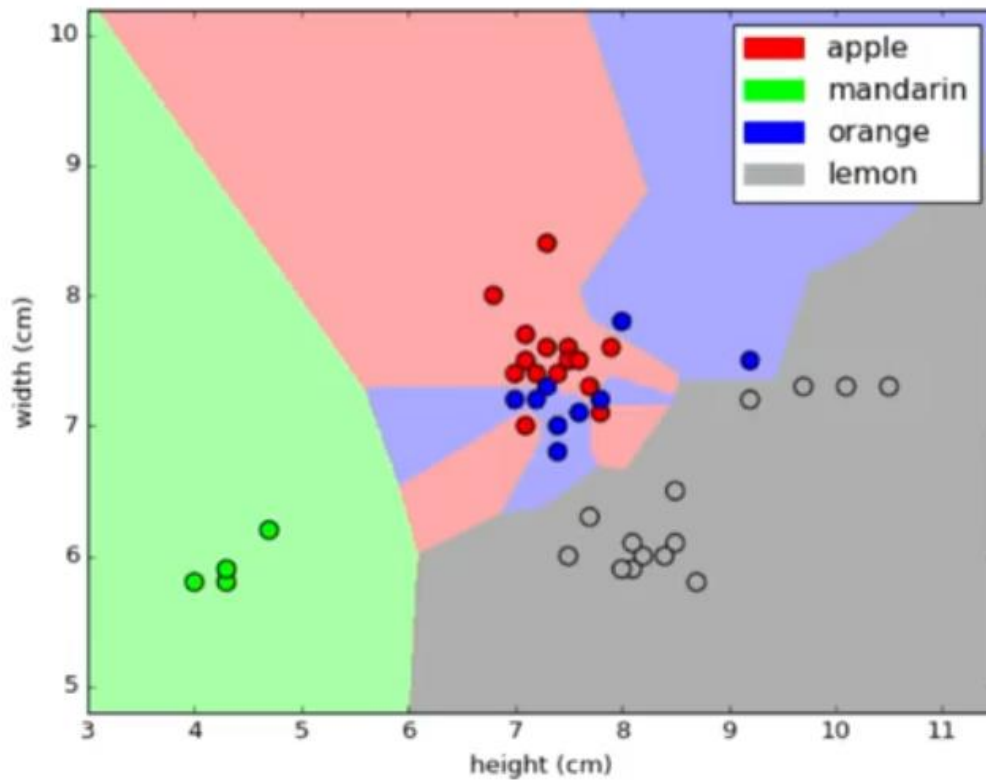
# Choosing k in k-Nearest Neighbors

- The training error rate and the validation error rate are two parameters we need to assess on different K-value



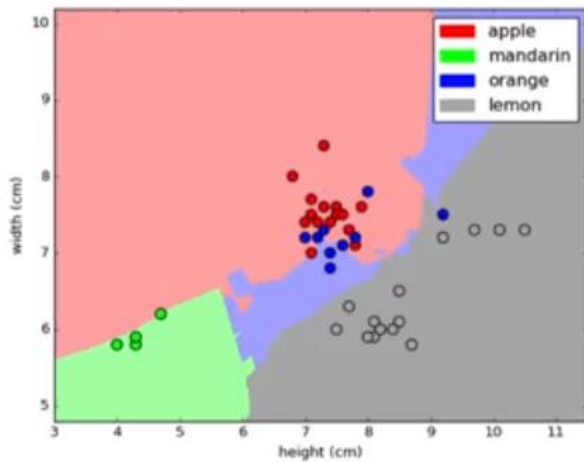
# Generalization, Overfitting and Underfitting

- Generalization ability refers to an algorithm's ability to give accurate predictions for new, previously unseen data.
- Assumptions:
  - Future unseen data (test set) will have the same properties as the current training set
  - This, models that are accurate on the training set are expected to be accurate on the test set
  - But that may not happen if the trained model is tuned too specifically to the training set.
- Models that are too complex for the amount of training data available are said to **overfit** and are not likely to generalize well to new data instances.
- Models that are too simple, that do not even do well on the training data, are said to **underfit** and also not likely to generalize well.

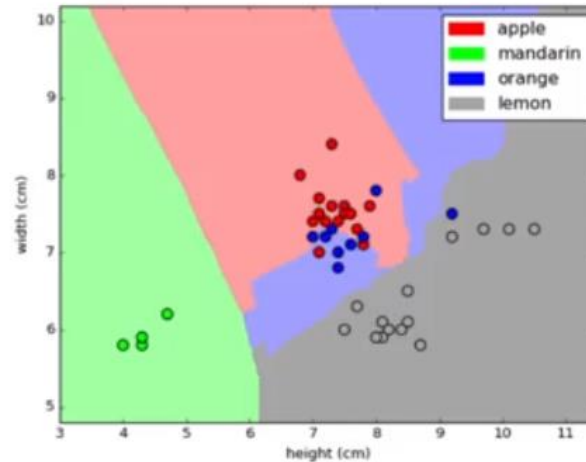


Fruit dataset  
Decision boundaries  
with  $k = 1$

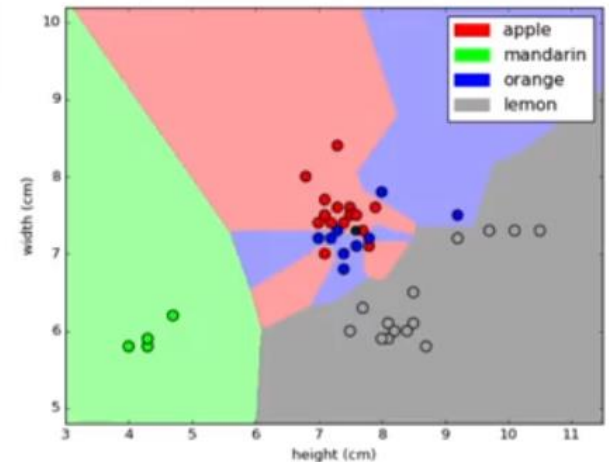
# Overfitting with k-NN classifiers



K=10



K=5



K=1

# Nearest Neighbor

## When to Consider

- Instance map to points in  $R^n$
- Less than 20 attributes per instance
- Lots of training data

## Advantages

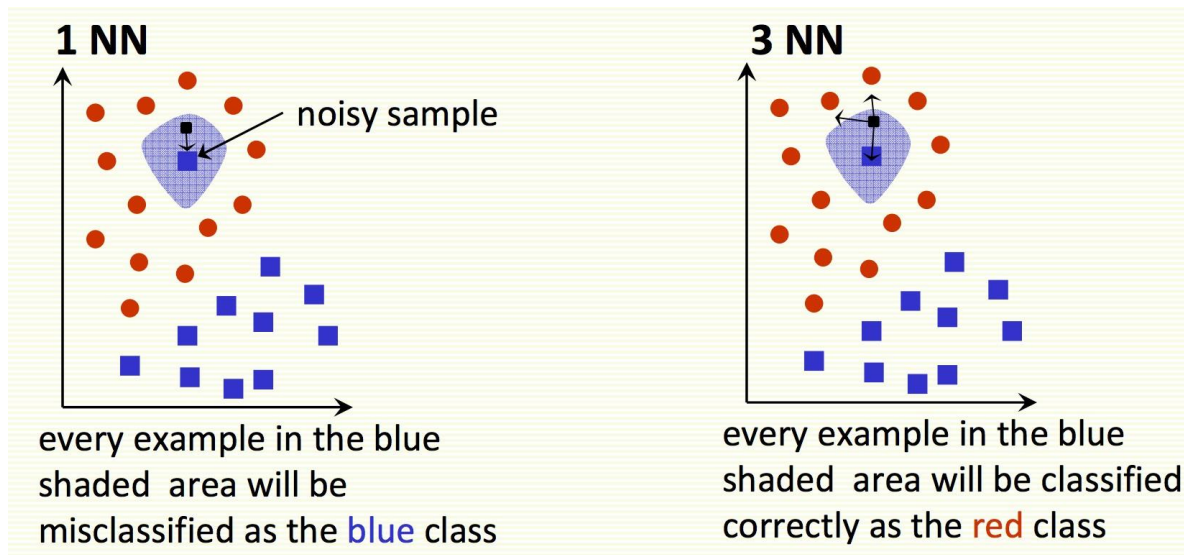
- Training is very fast
- Learn complex target functions
- Do not lose information

## Disadvantages

- Slow at query
- Easily fooled by irrelevant attributes
- The Bigger the dataset the more impact on performance



# k-Nearest Neighbors is Sensitive



Nearest neighbors **sensitive to mis-labeled data** ("class noise"). Solution?  
Smooth by having  $k$  nearest neighbors vote

# k-Nearest Neighbors: Complexity

**Expensive at test time:** To find one nearest neighbor of a query point  $\mathbf{x}$ , we must compute the distance to all  $N$  training examples. Complexity:  $O(kdN)$  for kNN

- Use subset of dimensions

- Compute only an approximate distance (e.g., LSH)

- Remove redundant data (e.g., condensing)

[Slide credit: David Claus]

# k-Nearest Neighbors: Complexity

**Storage Requirements:** Must store all training data

- Remove redundant data (e.g., condensing)

- Pre-sorting often increases the storage requirements

**High Dimensional Data:** “Curse of Dimensionality”

- Required amount of training data increases exponentially with dimension

- Computational cost also increases

# Fun Example: Where on Earth is this Photo From?

Problem: Where (e.g., which country or GPS location) was this picture taken?



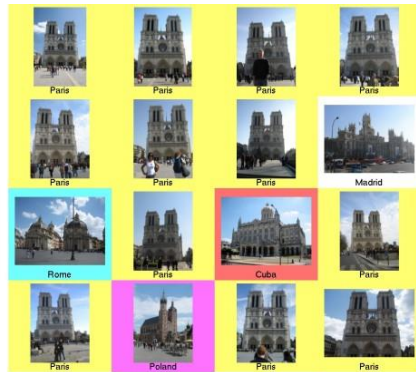
# Fun Example:

## Where on Earth is this Photo From?

Problem: Where (e.g., which country or GPS location) was this picture taken?

Get 6M images from Flickr with GPs info (dense sampling across world)  
Represent each image with meaningful features

Do kNN!

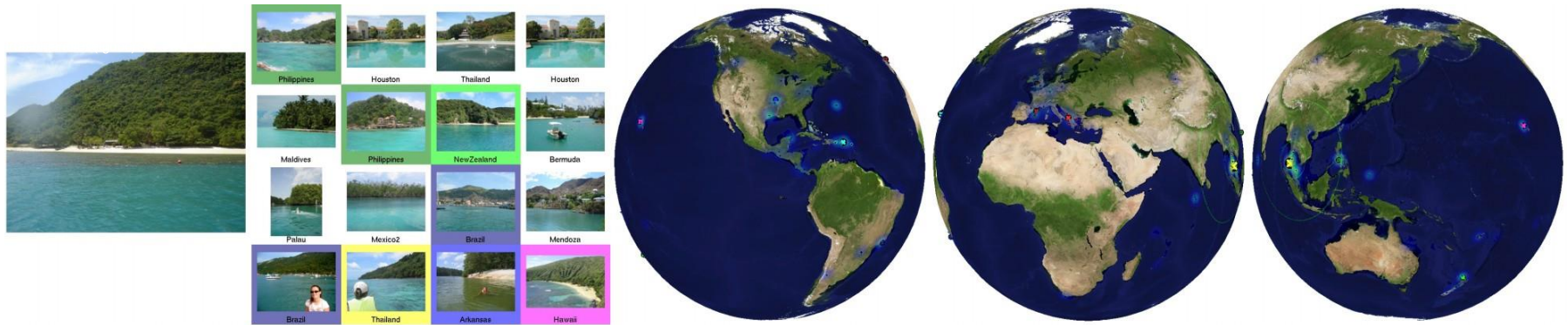




# Fun Example: Where on Earth is this Photo From?

Problem: Where (eg, which country or GPS location) was this picture taken?

Get 6M images from Flickr with gps info (dense sampling across world)  
Represent each image with meaningful features  
Do kNN (large  $k$  better, they use  $k = 120$ )!



# Useful Resources

- Mathematician hacking dating site <https://www.wired.com/2014/01/how-to-hack-okcupid/>
- <https://medium.com/@adi.bronshtein/a-quick-introduction-to-k-nearest-neighbors-algorithm-62214cea29c7>
- <https://towardsdatascience.com/building-improving-a-k-nearest-neighbors-algorithm-in-python-3b6b5320d2f8>
- <https://kevinzakka.github.io/2016/07/13/k-nearest-neighbor/>
- <https://www.analyticsvidhya.com/blog/2018/03/introduction-k-neighbours-algorithm-clustering/>