# K-Nearest Neighbors Classification

Taught by: Bedoor AlShebli

Based on Kevyn Collins-Thompson's Applied Machine Learning Course

# K-NN Classifier

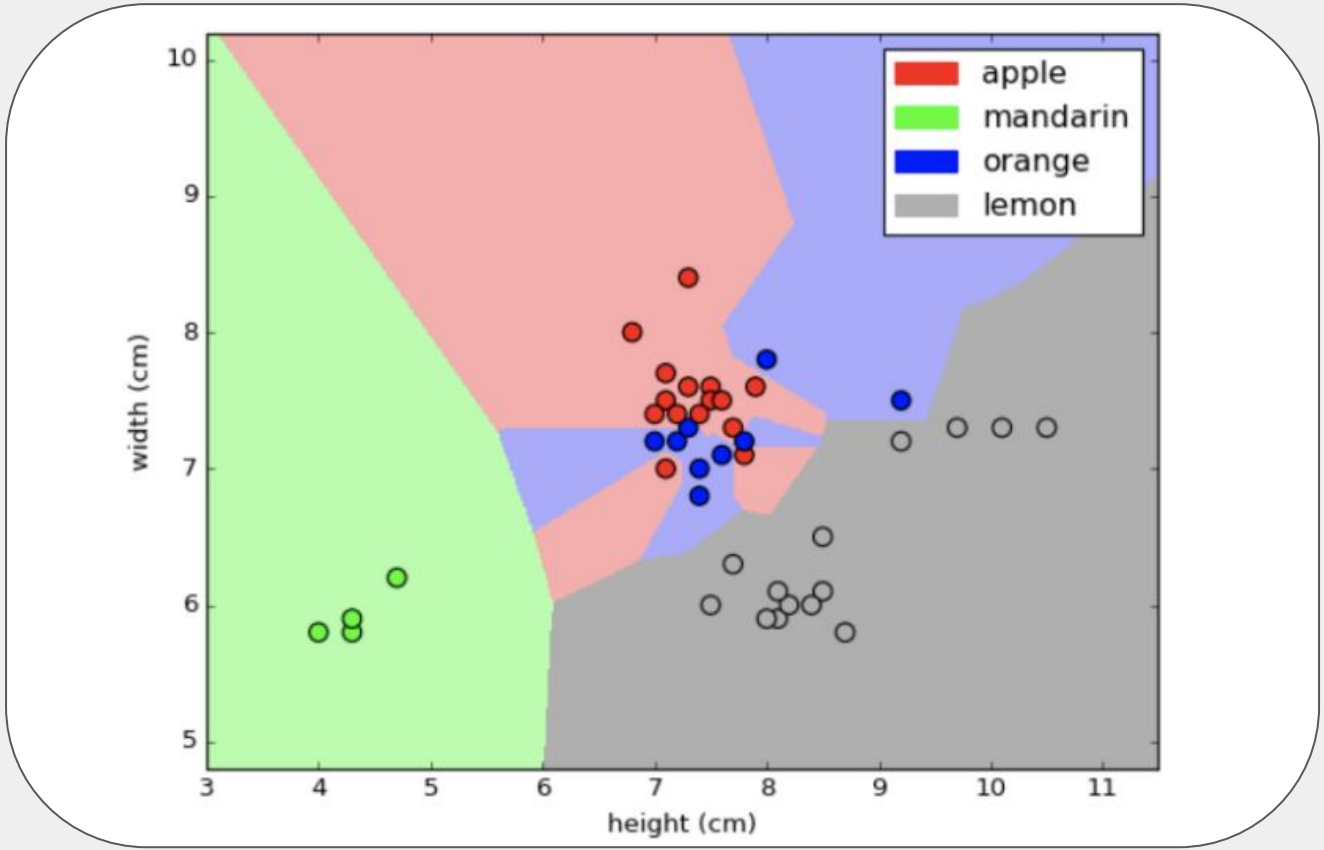
- An example of **instance or memory based supervised learning**
  - i.e. it works by memorizing the labeled examples in the training set
- The **k refers to the number of nearest neighbors** the classifier will retrieve and use in order to make its prediction

# K-NN Classifier: **The Algorithm**

Given a training set `X_train` with labels `y_train`, and given a new instance `x_test` to be classified:

1. Find the most similar instances (let's call them `X_NN`) to `x_test` that are in `X_train`.
2. Get the labels `y_NN` for the instances in `X_NN`
3. Predict the label for x_test by combining the labels `y_NN`. Eg) simple majority vote

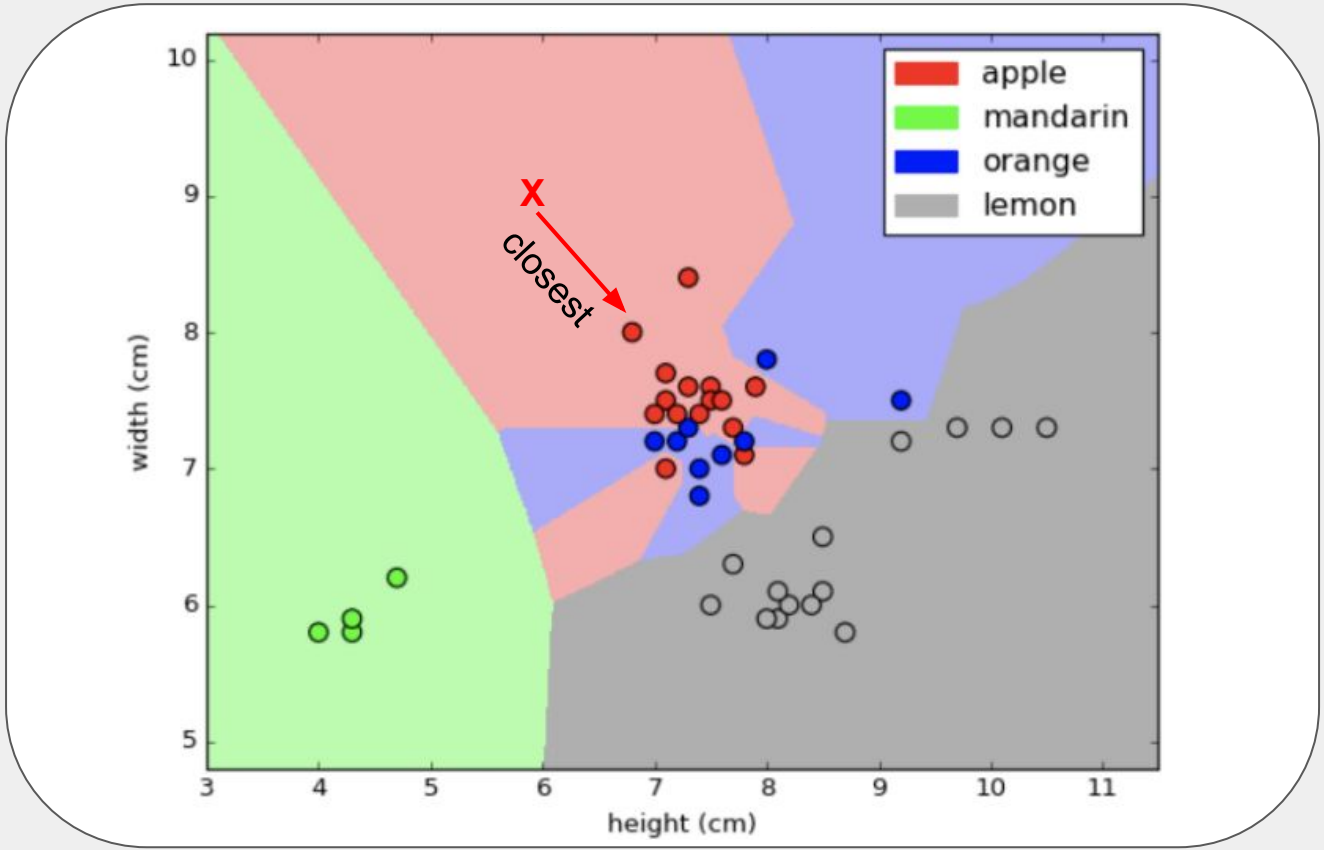
# A visual explanation of k-NN classifiers **(k=1)**




**The KNN algorithm assumed that similar things exist in close proximity.**

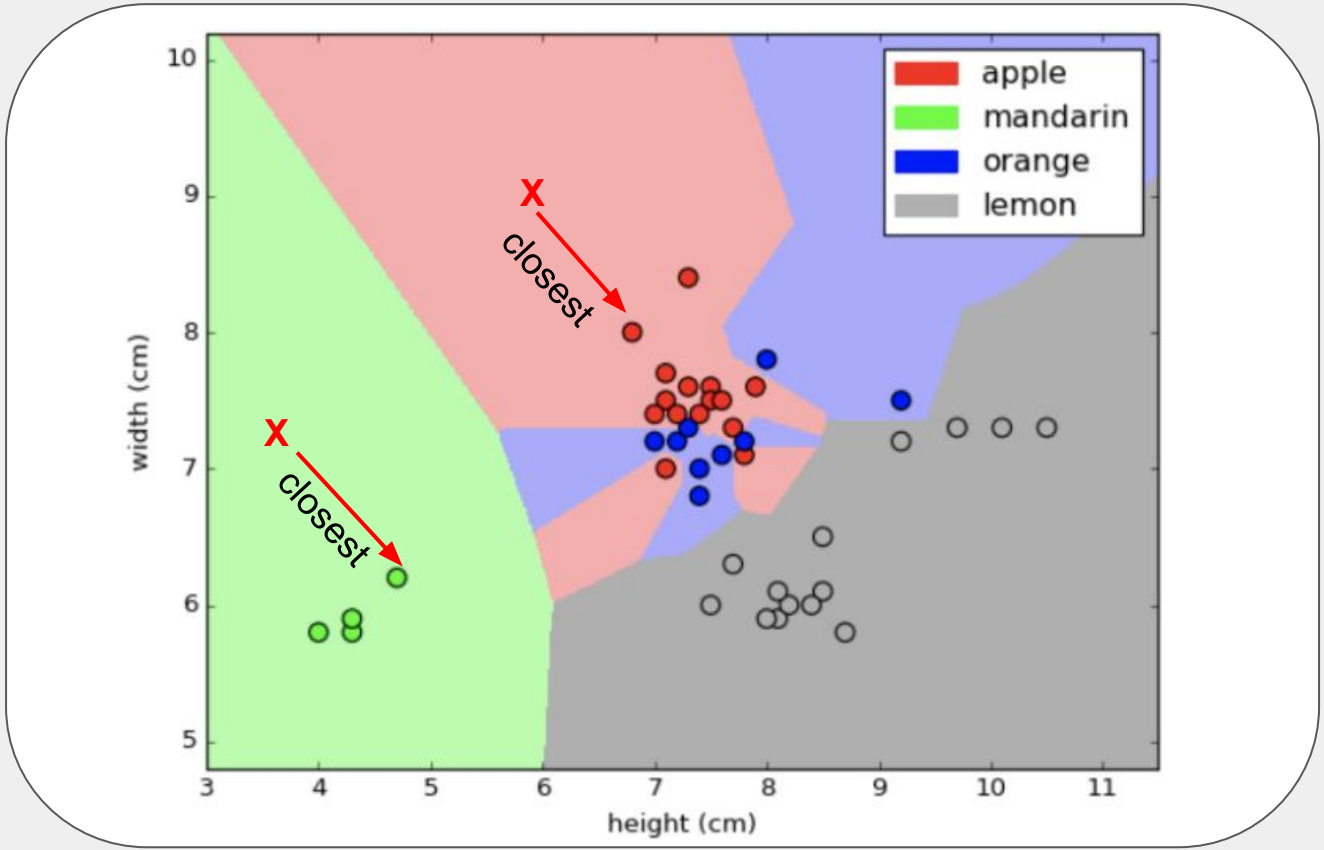
**“Birds of a feature, flock together”**

# A visual explanation of k-NN classifiers **(k=1)**



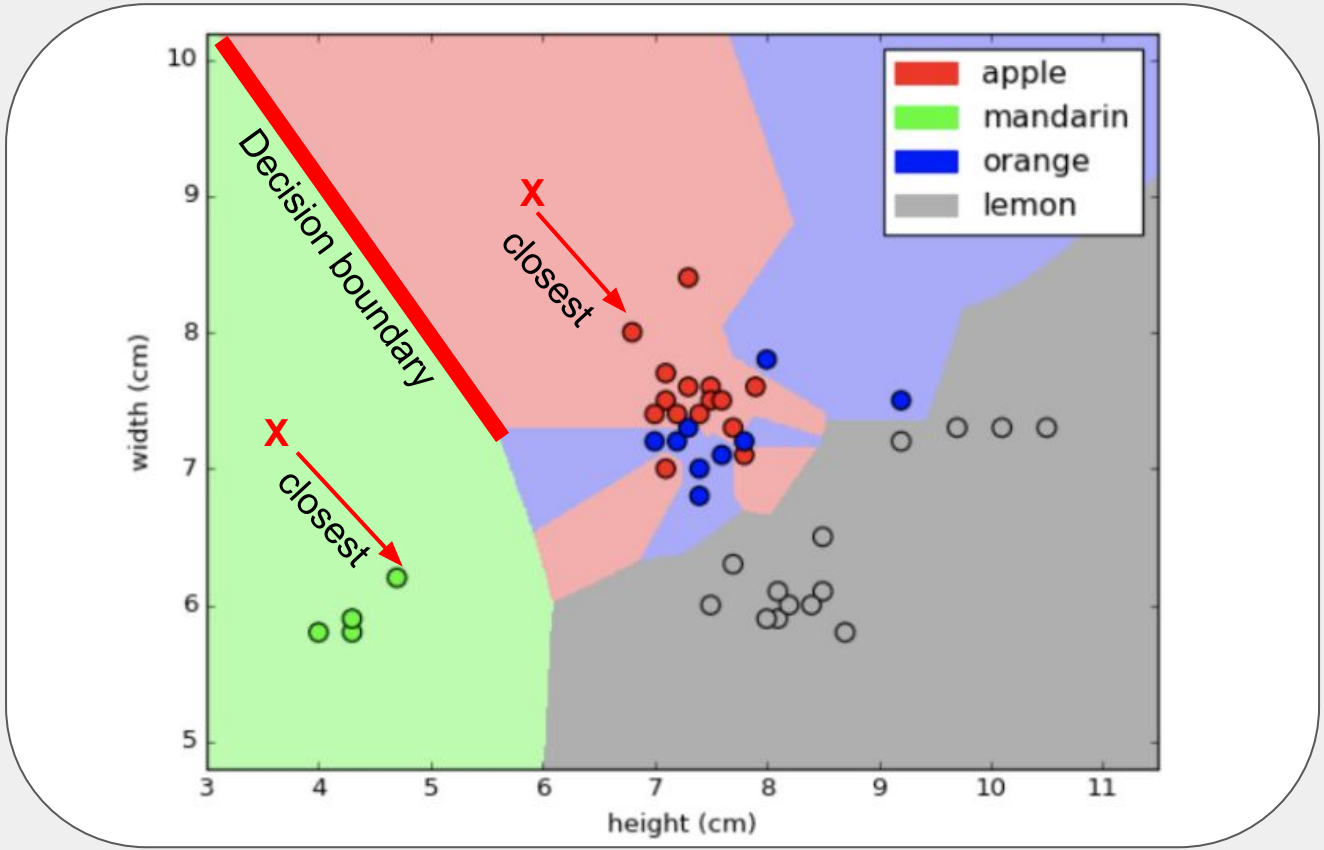

**What is x with height = 6 and width = 9?**

# A visual explanation of k-NN classifiers **(k=1)**



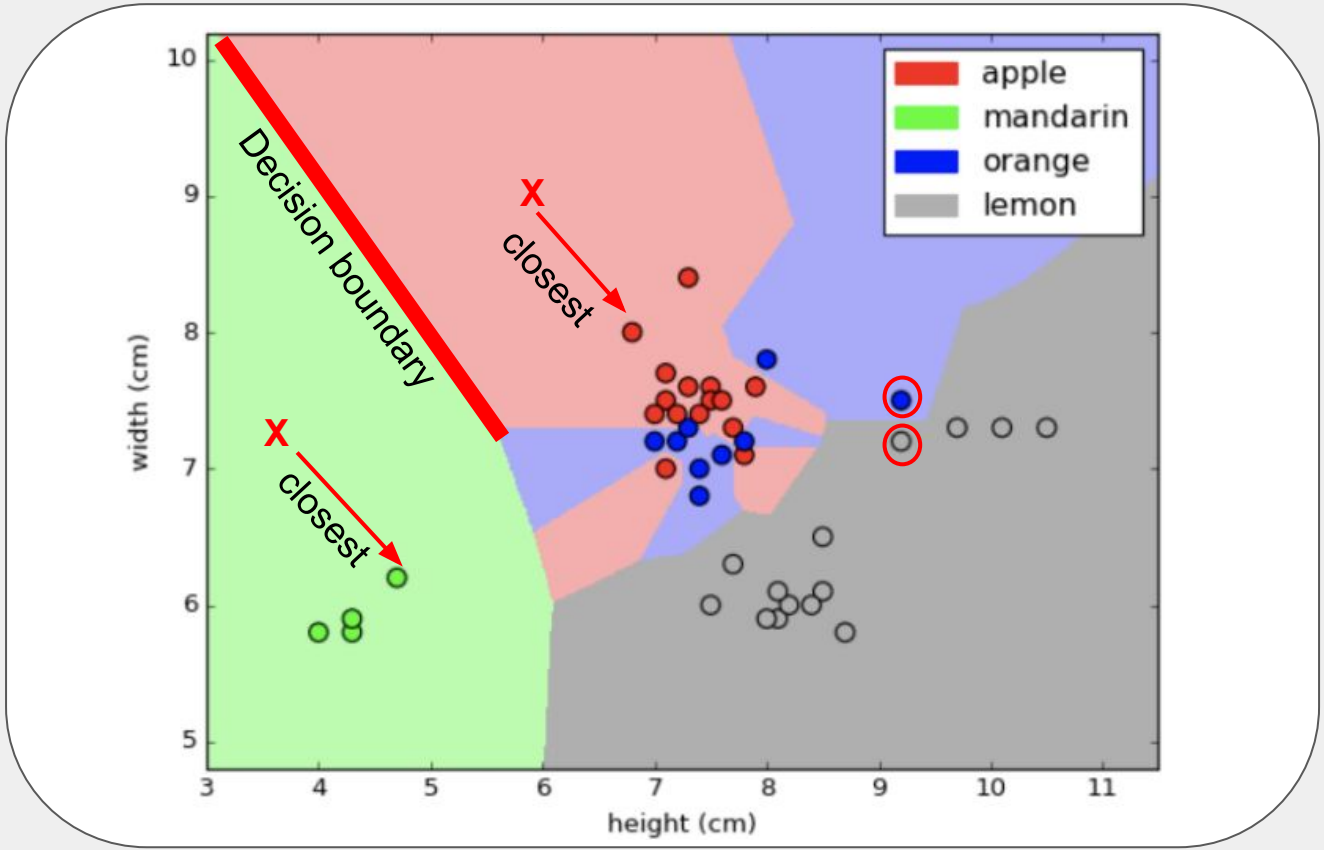

**What about this second x?**

# A visual explanation of k-NN classifiers **(k=1)**



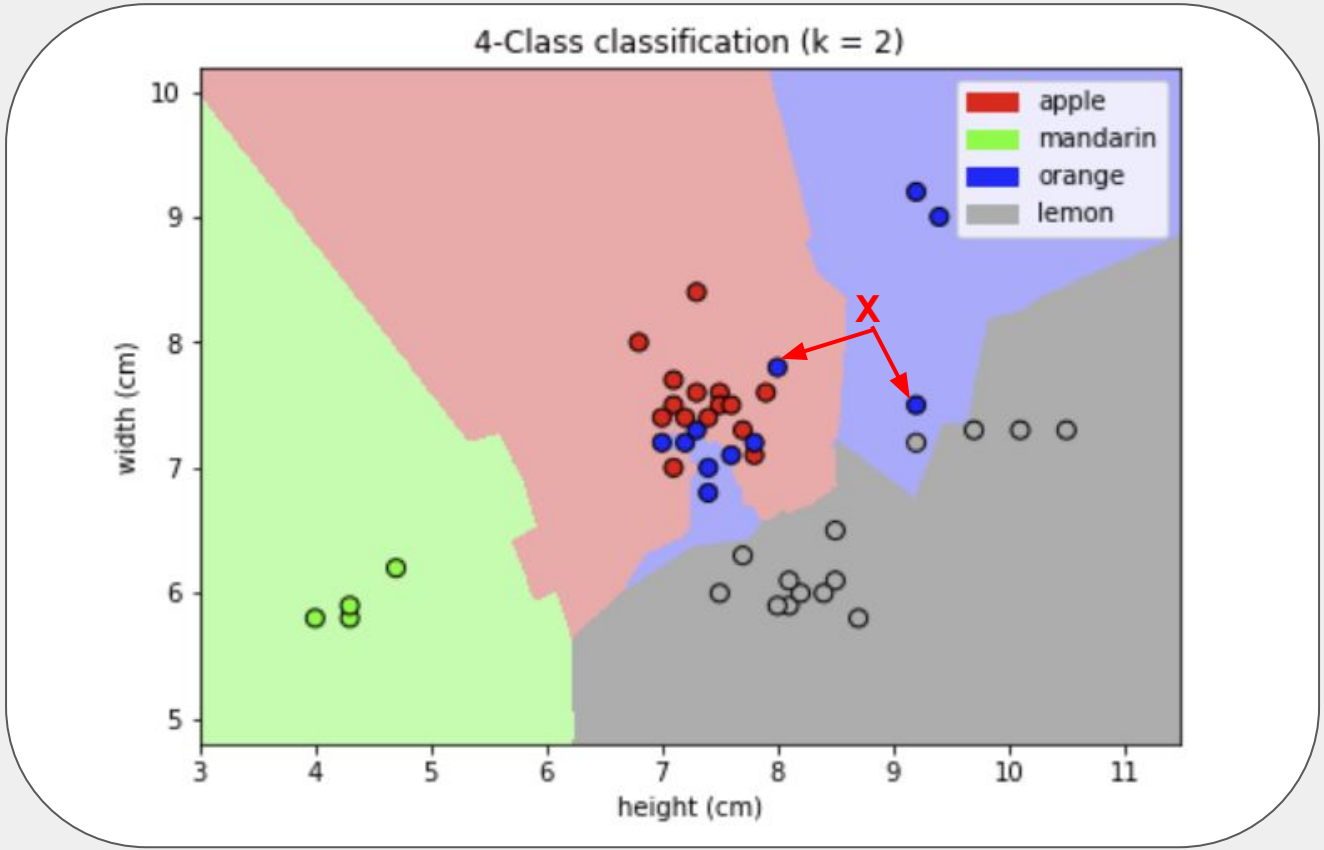

**Decision Boundaries with k=1**

# A visual explanation of k-NN classifiers **(k=1)**




**Decision boundaries are constructed to be equidistant based on the euclidean distance between two different labeled instances.**

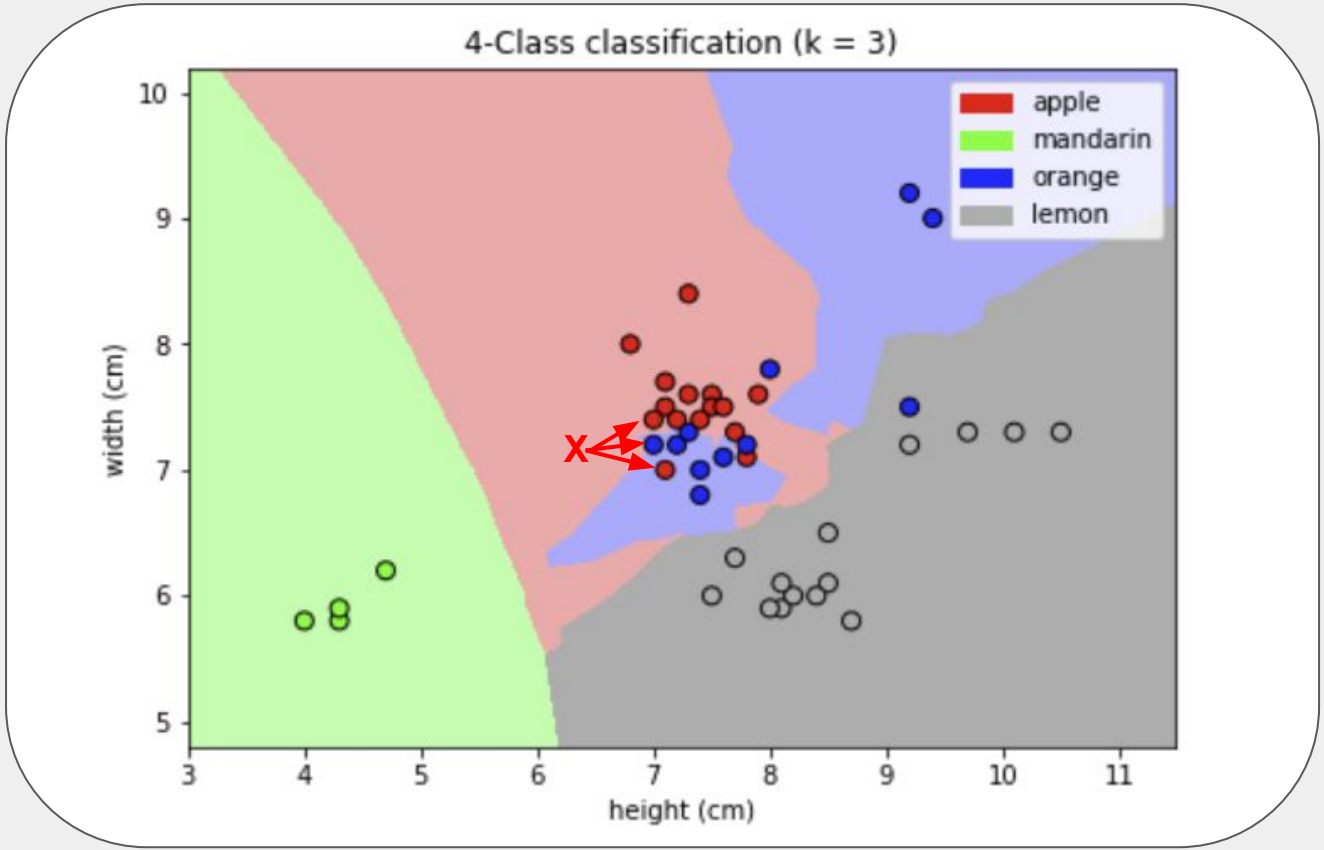
# A visual explanation of k-NN classifiers **(k=2,3,4?)**




**What if k = 2?**

**We look at the 2 closest points and use a simple majority vote to decide**

# A visual explanation of k-NN classifiers **(k=2,3,4?)**




**Typically k is odd, to avoid / resolve tie breaks.**

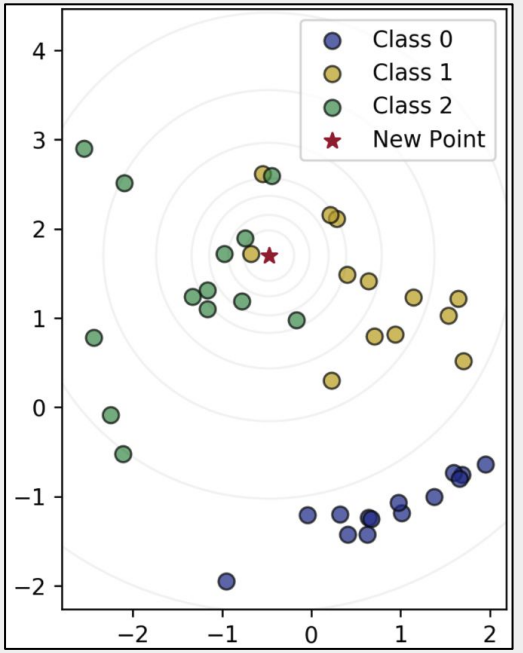
**X here would be an apple.**

# Nearest Neighbor Algorithm Requirements

Generally, to use a nearest neighbor algorithm, we need to specify the following:

1. **A distance metric**
    - In our example, we used a simple straight line or euclidean distance
2. **How many 'nearest' neighbors to look at, i.e. k?**
    - Must be at least 1
3. **Optional weighting function on the neighbor points.**
    - We may decide that neighbors closer to the new instance should have more influence or more votes. In our example, we chose to ignore this option, i.e. give “uniform” weighting.
4. **Method for aggregating the classes of neighbor points**
    - In our example, we used the default scikit-learn option, which is a simple majority vote.

# Questions



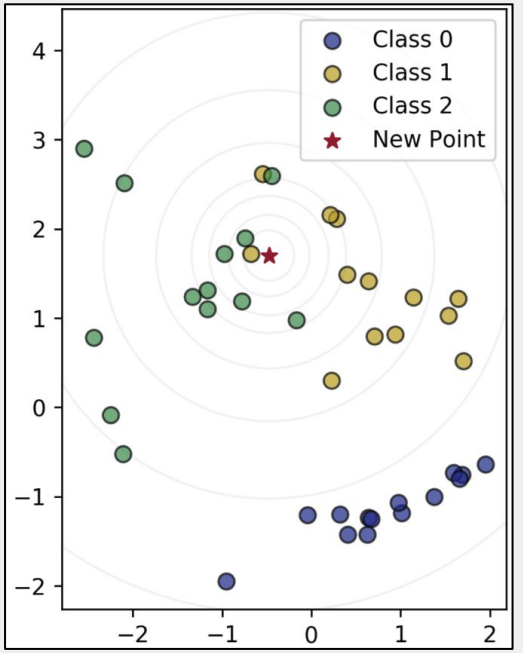

1- Given the following data for Classes 0, 1 and 2, what class would the KNN Classifier classify the new point if :

- k=1?
- k=3?

2- Which of the following is true for KNN classifiers? *Select all that apply.*

- A higher value of k leads to a more complex decision boundary.
- KNN partitions observations into k clusters, where each new observation belongs to the cluster with the nearest mean.
- KNN memorizes the entire training set.
- Given a data instance to classify, computes the probability of each possible class using a statistical model of the input features.

# Questions




1- Given the following data for Classes 0, 1 and 2, what class would the KNN Classifier classify the new point if :

- k=1? Class 1
- k=3? Class 2

2- Which of the following is true for KNN classifiers? *Select all that apply.*

- A higher value of k leads to a more complex decision boundary.
- KNN partitions observations into k clusters, where each new observation belongs to the cluster with the nearest mean.
- KNN memorizes the entire training set.
- Given a data instance to classify, computes the probability of each possible class using a statistical model of the input features.