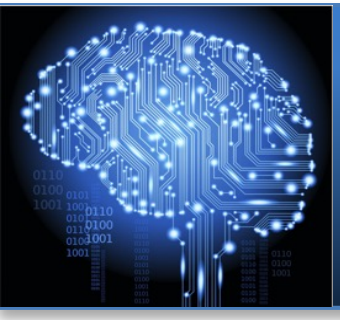


FIT1043 Lecture 7
Introduction to Data Science

Mahsa Salehi

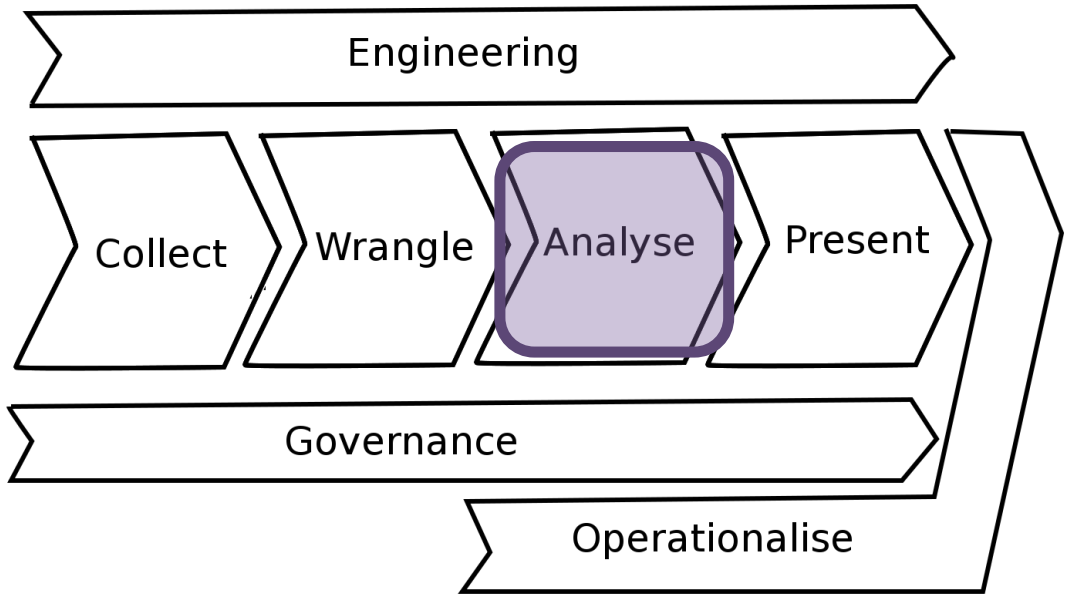
Faculty of Information Technology, Monash University

Semester 2, 2022

Unit Schedule

Week	Activities	Assignments
1	Overview of data science	Weekly Lecture/tutorial active participation assessment
2	Introduction to Python for data science	
3	Data visualisation and descriptive statistics	
4	Data sources and data wrangling	
5	Data analysis theory	Assignment 1
6	Regression analysis	
7	Classification and clustering	
8	Introduction to R for data science	
9	Characterising data and "big" data	Assignment 2
10	Big data processing	
11	Issues in data management	
12	Industry guest lecture	Assignment 3

Our Standard Value Chain



Weeks 5-7

Outline

- Classification
  - How to evaluate
  - Classification metrics
  - Decision trees
- Regression
  - Regression trees
- Ensemble learning
  - Random forest
- Clustering
  - K-means

Learning Outcomes (Week 7)

By the end of this week you should be able to:

- Differentiate between classification and regression models
- Explain how decision trees and regression trees work
- Explain how random forest works
- Explain how k-means clustering works
- Analyse confusion matrix and how to calculate prediction accuracy
- Differentiate between different classification metrics

Data Analysis Algorithms
Classification

From Data Mining: Concepts and Techniques

by J. Han et al, 2011

Classification

Cat **Dog**

?

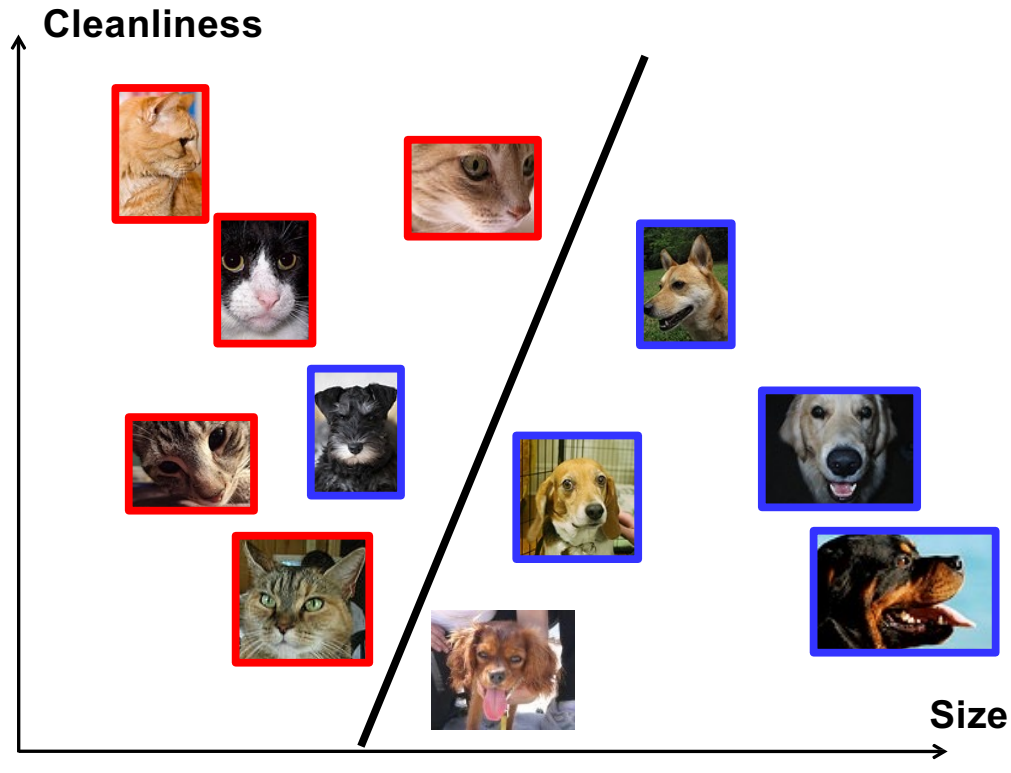
Classification

Cat

Dog

?

Classification (cont.)



Classification (Real World Example)

Question: Can we predict the diabetes status of a patient given their health measurements (i.e., 'pregnant', 'insulin', 'BMI', 'age')?

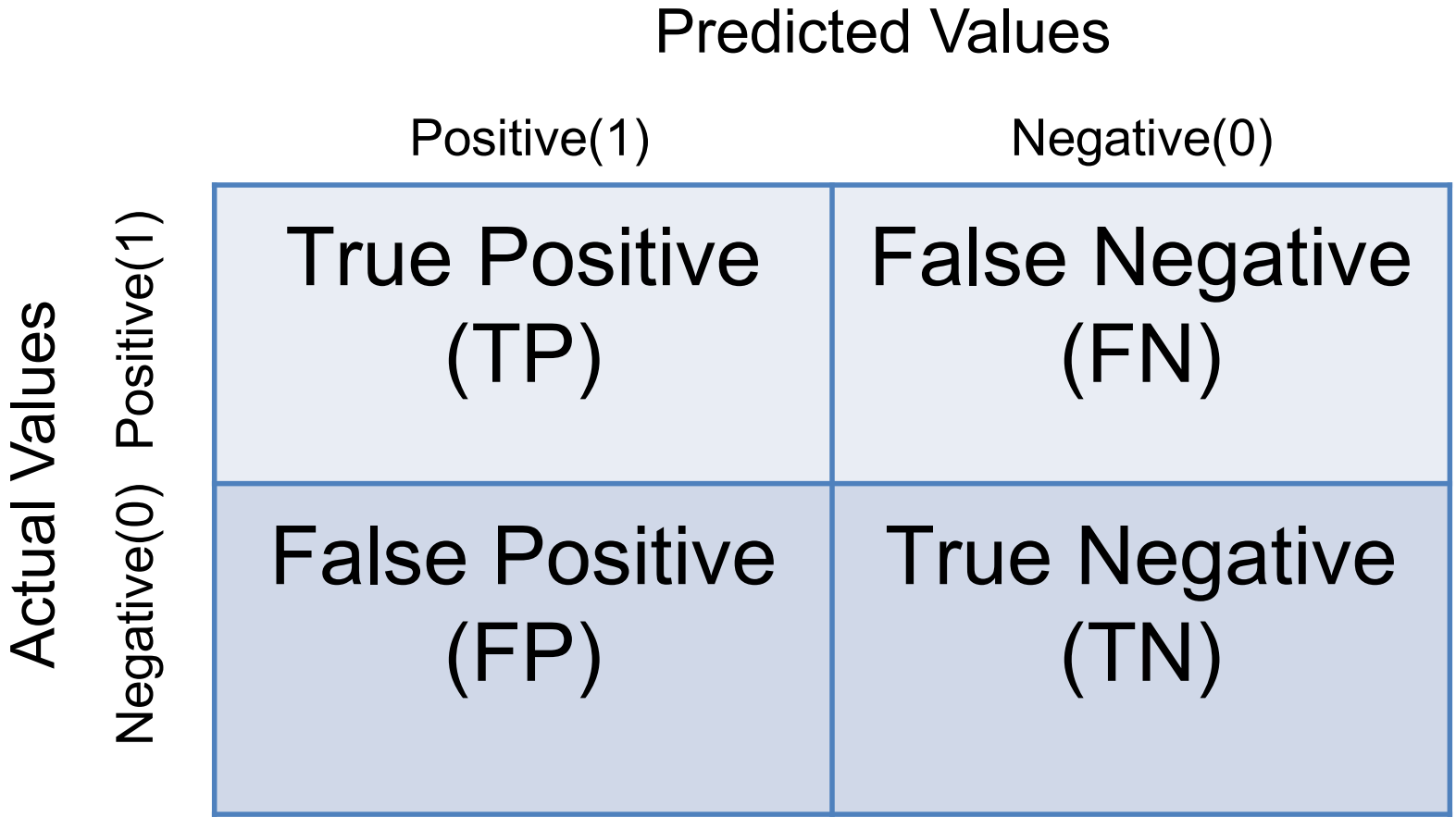
Dataset

How do we evaluate the prediction accuracy?

- Percentage of correct predictions by comparing the **actual** with the **predicted** response values

Confusion Matrix

- A tool to measure performance for classification



Classification Metrics

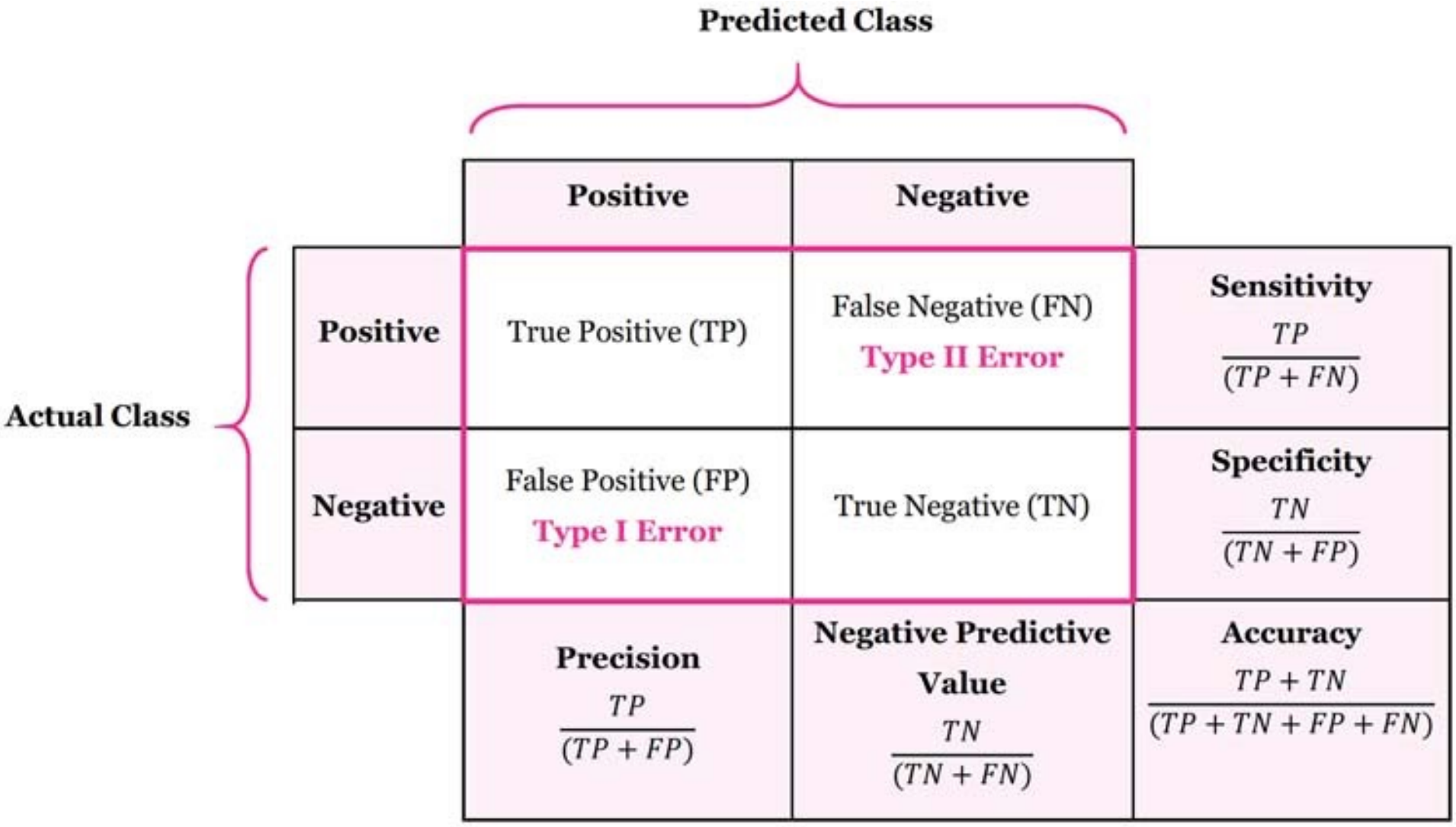


Image src: manisha-sirsat.blogspot.com

Classification Metrics

- **Accuracy:** Overall, how often is the prediction correct?
- **Sensitivity (Recall):** When the actual value is positive, how often is the prediction correct?
- **Specificity:** When the actual value is negative, how often is the prediction correct?
- **False Positive Rate:** When the actual value is negative, how often is the prediction incorrect?
- **Precision**: When a positive value is predicted, how often is the prediction correct?

FLUX Question

Determine classification accuracy for the following Confusion Matrix?

n=192	**Predicted: 0**	**Predicted: 1**	
Actual: 0	TN = 118	FP = 12	130
Actual: 1	FN = 47	TP = 15	62
	165	27	

Which Metrics Should be Used?

It depends ...

- **Spam filter:** Optimise **precision** or **specificity**
  - False negatives (spam goes to the inbox) are more acceptable than false positives (non-spam is caught by the spam filter)

- **Fraudulent transaction detector**: Optimise **sensitivity**
  - False positives (normal transactions that are flagged as possible fraud) are more acceptable than false negatives (fraudulent transactions that are not detected)

Data Analysis Algorithms
Decision Trees

Decision Trees and Regression Trees

What is Decision Trees?

- Predict binary (or categorical)outcomes

What is Regression Trees?

- Predict continuous (i.e. real) values

Tree

- Prediction model is a tree

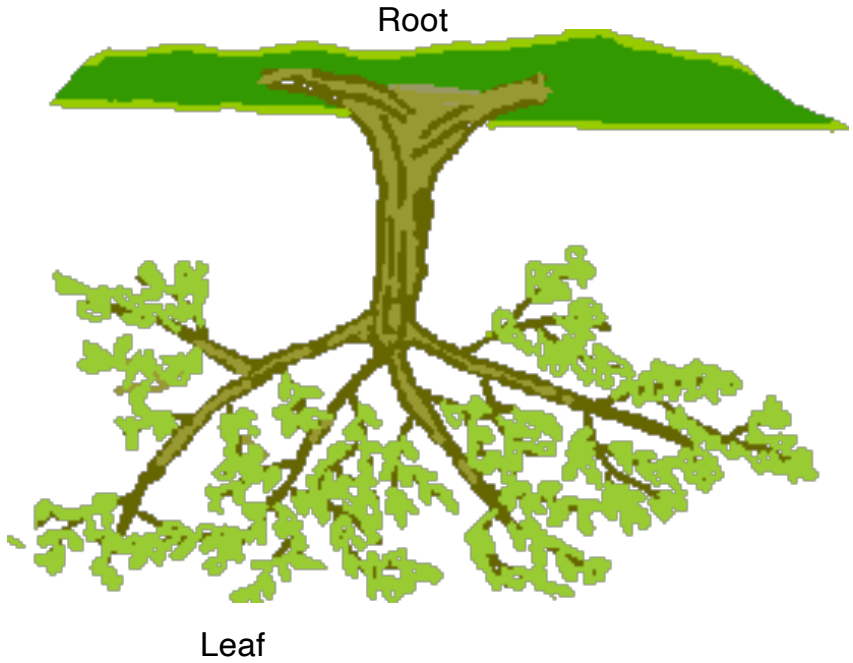
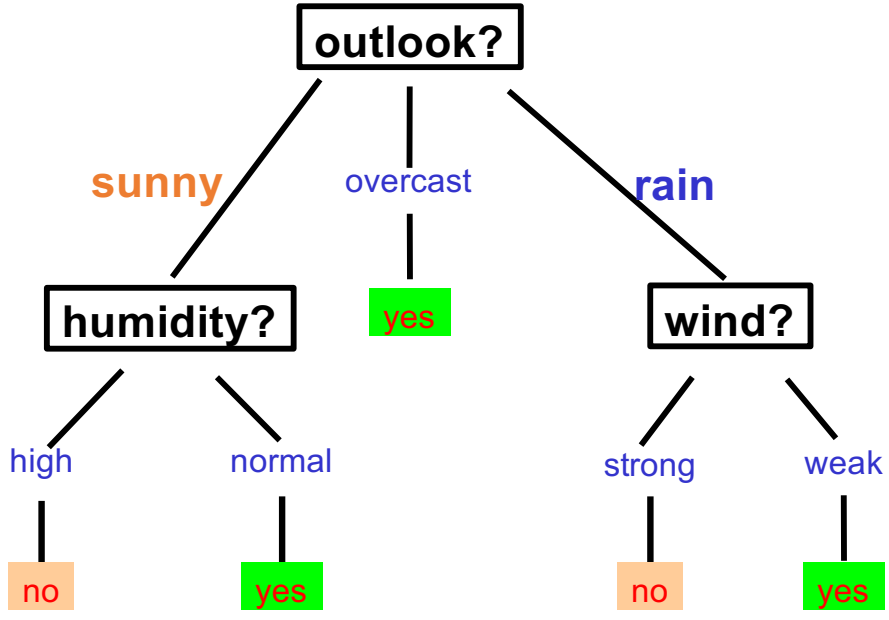
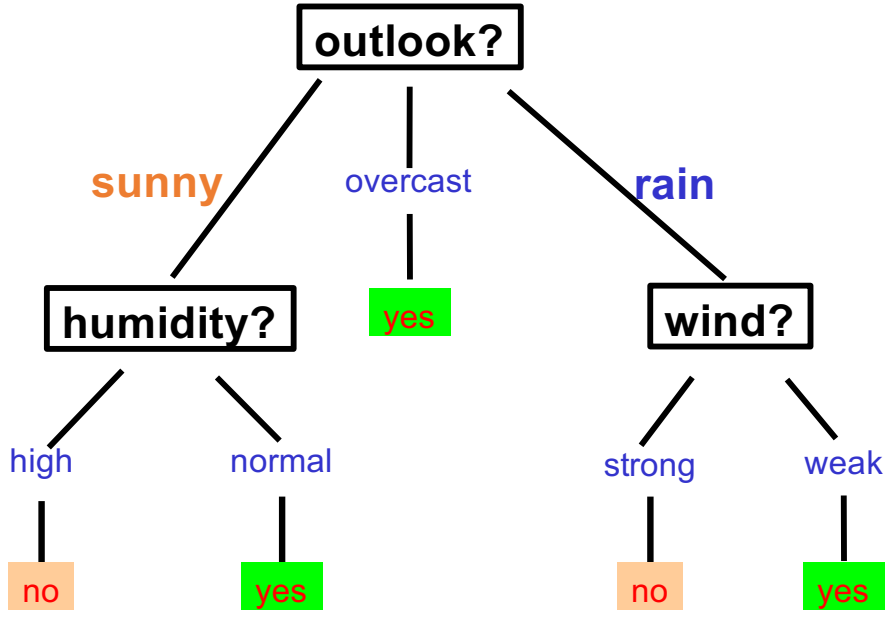


Image src: webdocs.cs.ualberta.ca

Decision Tree Example



Decision Tree Example



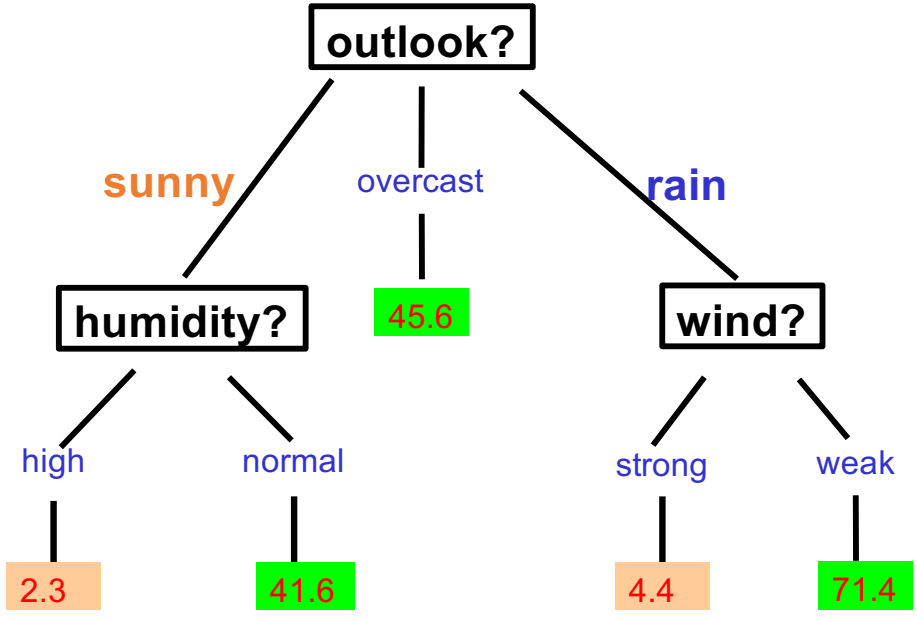
Set of rules:
G-Day to play tennis ⇔
(Sunny and Normal) or Overcast or (Rain and Weak)
B-Day to play tennis ⇔ ?

FLUX Question

According to the previous slide when is a bad day to play tennis?

A. When it's sunny and humidity is high
B. When it's rainy and wind is strong
C. Both the above options

Regression Tree Example



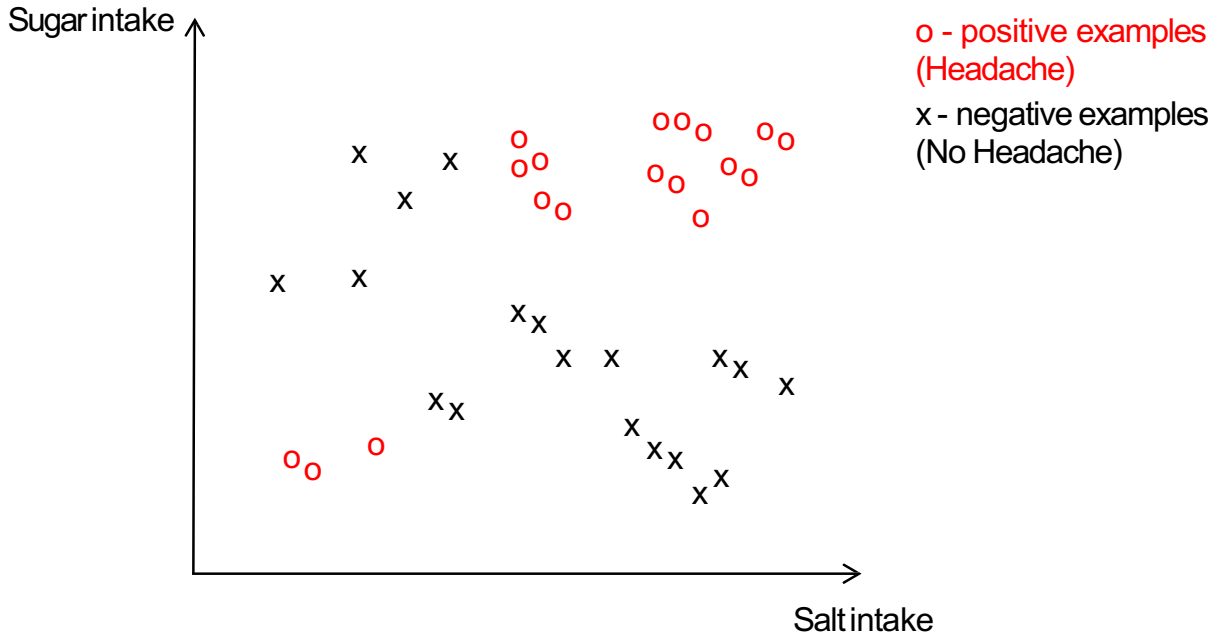
How to Build Regression and Decision Trees?

- Recursively partition (divide up) the feature space into regions

- While grouping similar instances together

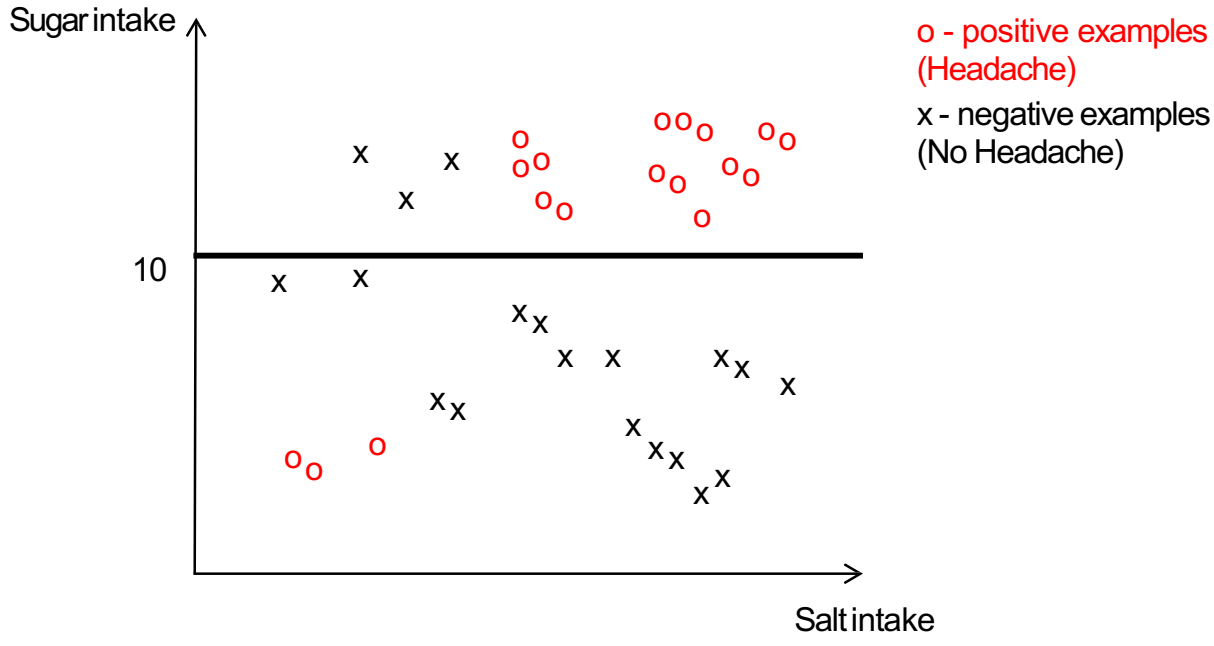
Recursive Partitioning

- At each iteration, we divide the data to group similar instances together



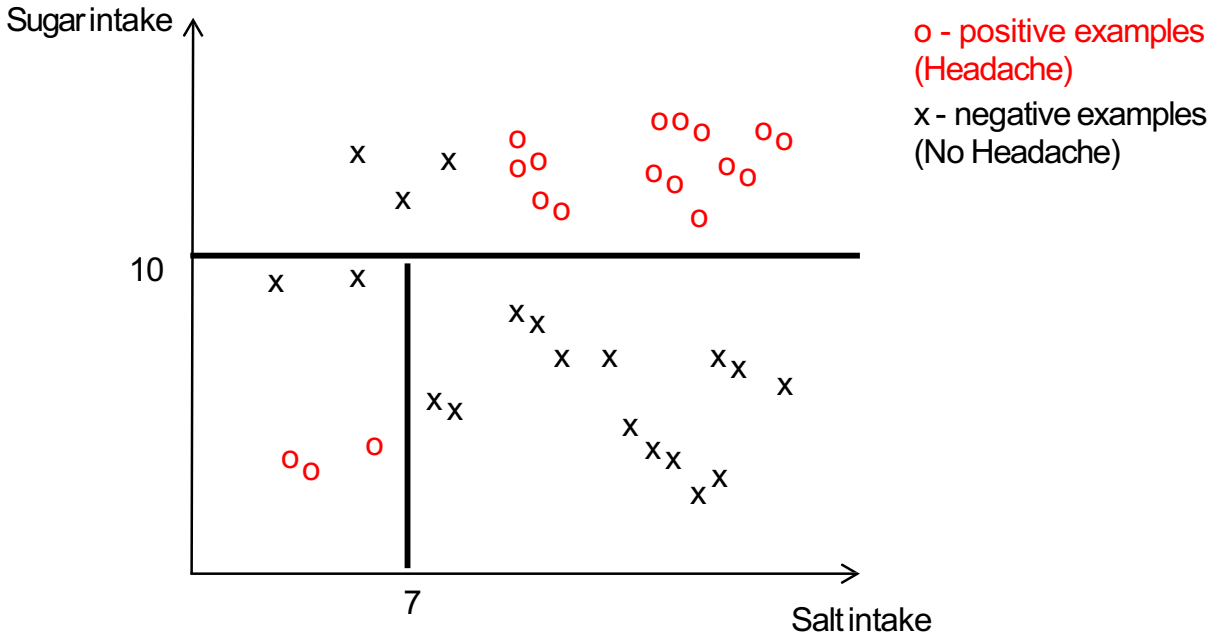
Recursive Partitioning

- At each iteration, we divide the data to group similar instances together



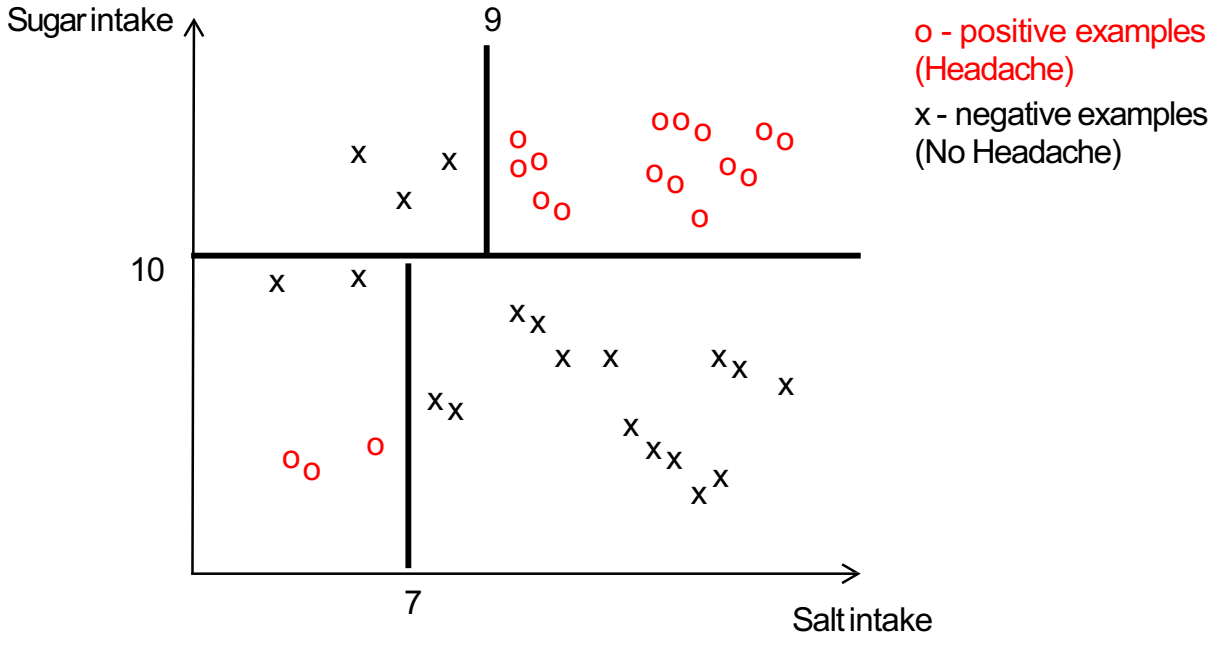
Recursive Partitioning

- At each iteration, we divide the data to group similar instances together



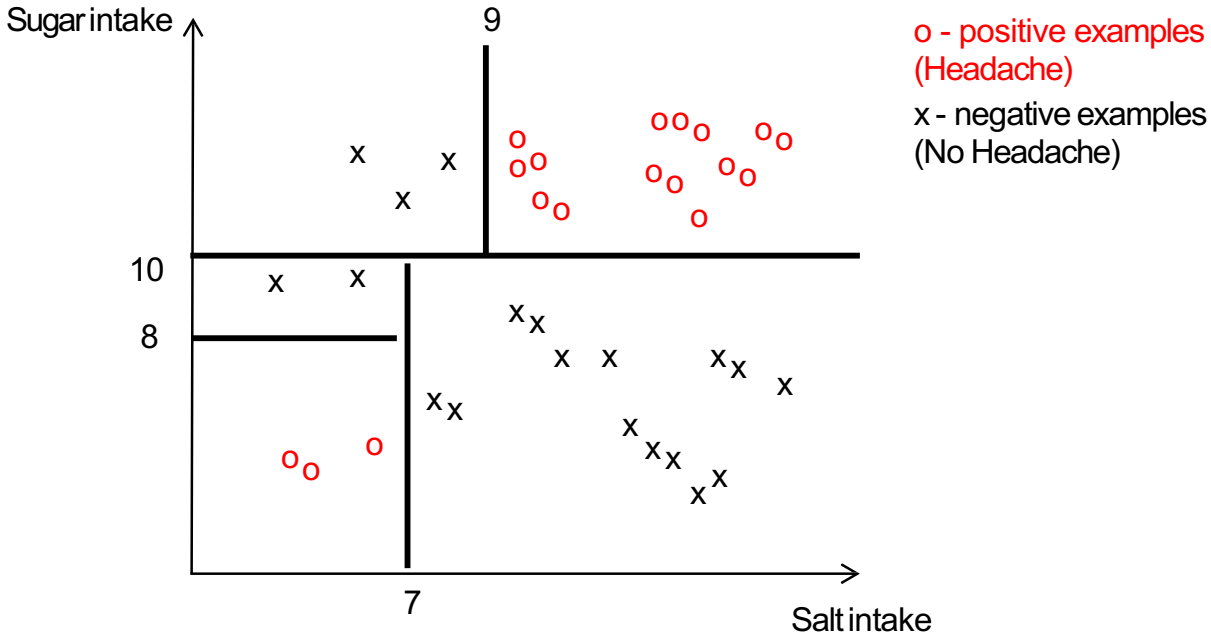
Recursive Partitioning

- At each iteration, we divide the data to group similar instances together



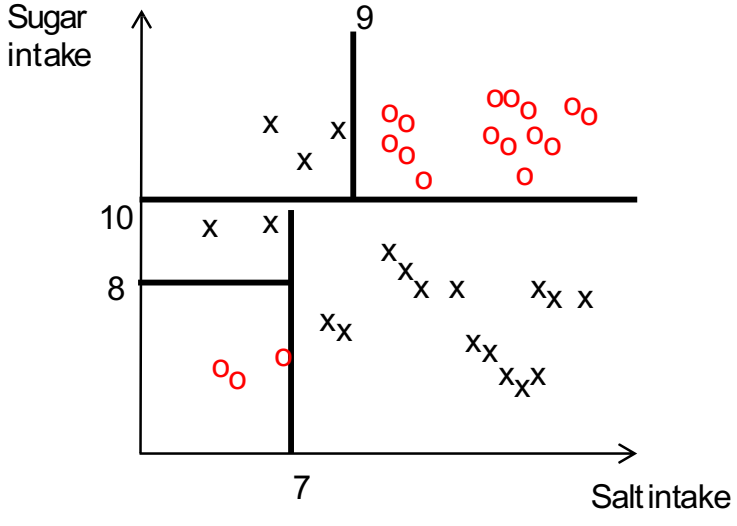
Recursive Partitioning

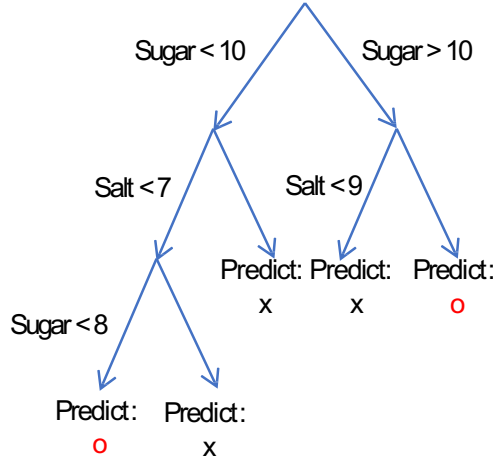
- At each iteration, we divide the data to group similar instances together



Prediction Model is a Tree

- This model learnt can be represented as a tree with predictions at the leaves:





Prediction in Decision and Regression Trees

Decision Trees:

- Prediction is the most common values in each region

Regression Trees:

- Prediction is usually the average value in each region

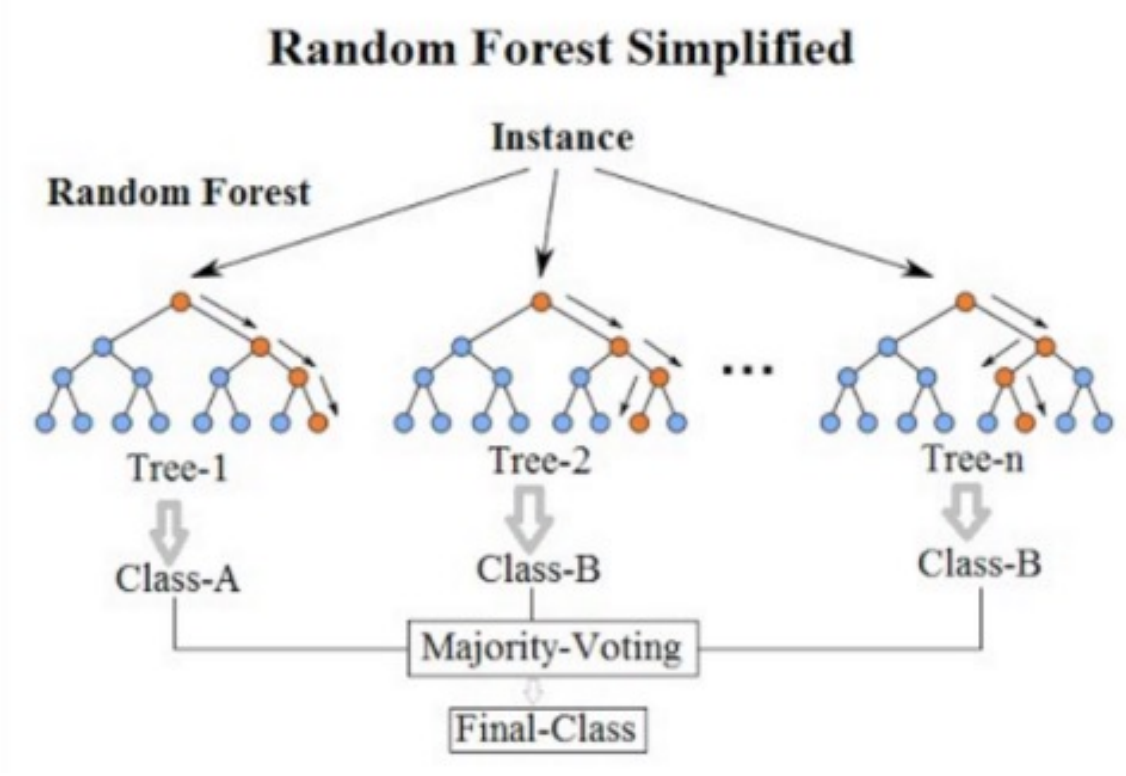
Decision/Regression Trees- More information

- Algorithms for building Decision & Regression trees differ on the criteria (e.g., Entropy) used to:
  - Decide on which feature to split on in each iteration
  - Decide when to stop splitting

Data Analysis Algorithms
Random Forest

What is Random Forest?

- Ensemble learning method that operate by constructing a number of decision trees



Data Analysis Algorithms
Clustering

What is Clustering?

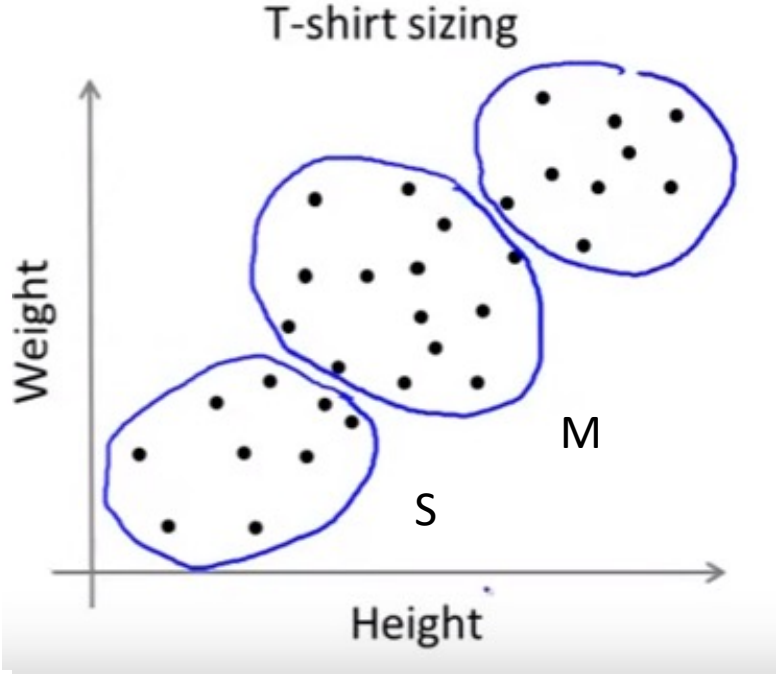
From lecture notes by [Andrew Ng]

- Grouping a set of data points into different subgroups based on their similarity

called clusters

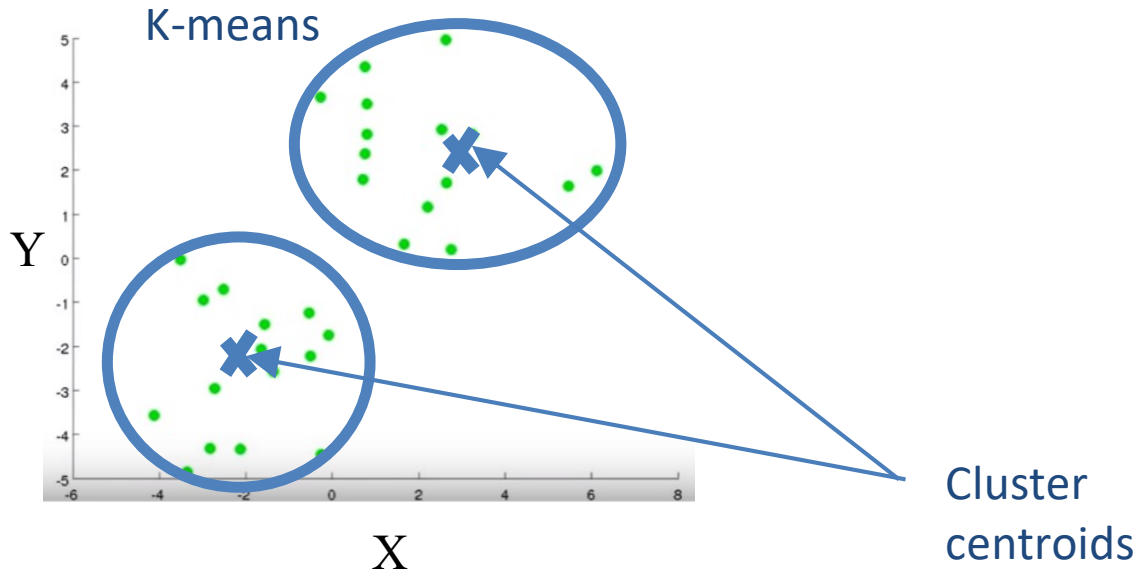
- K-means

- T-shirt manufacturer
- Group into 3 sizes: Small, Medium and Large



K-means Clustering

Example: Partition into two clusters based on similarity



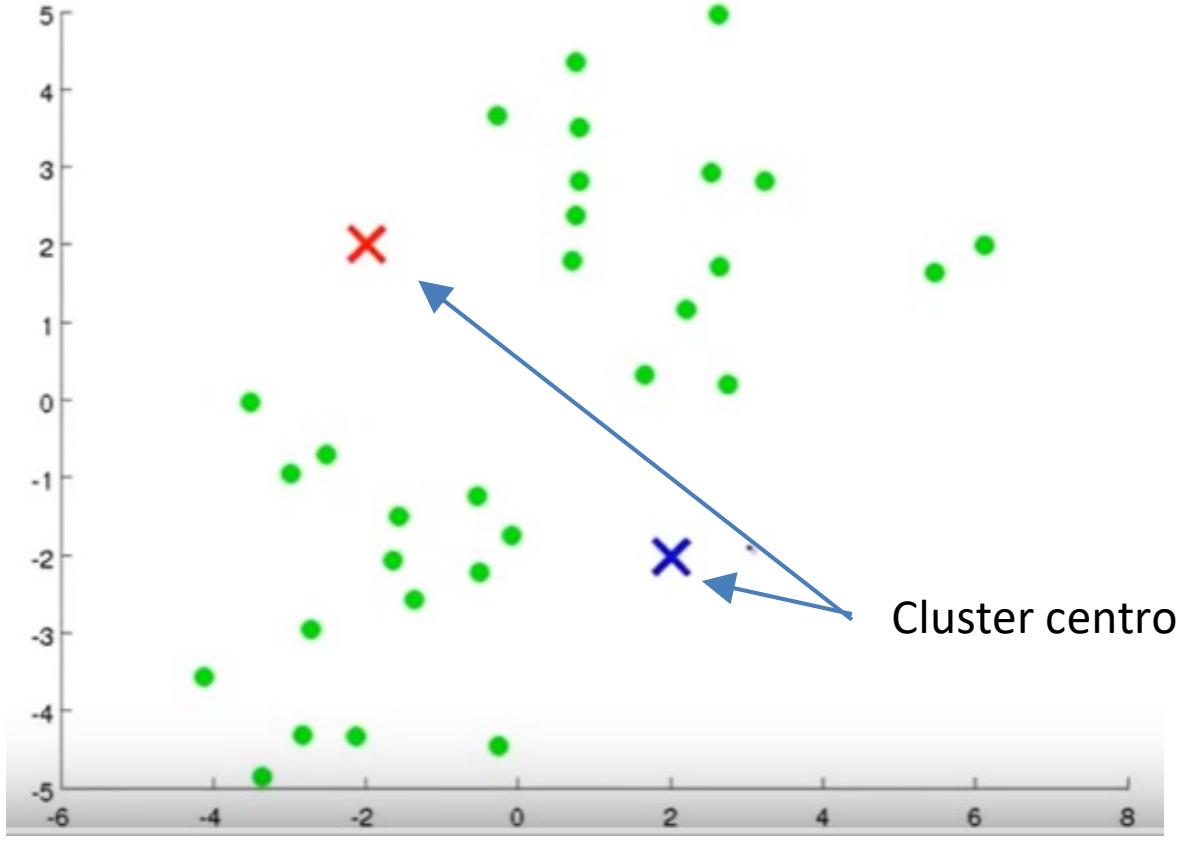
K-means

K=the number of clusters

Cluster centroid= The **mean** **(average)** of the location of all data points in a cluster

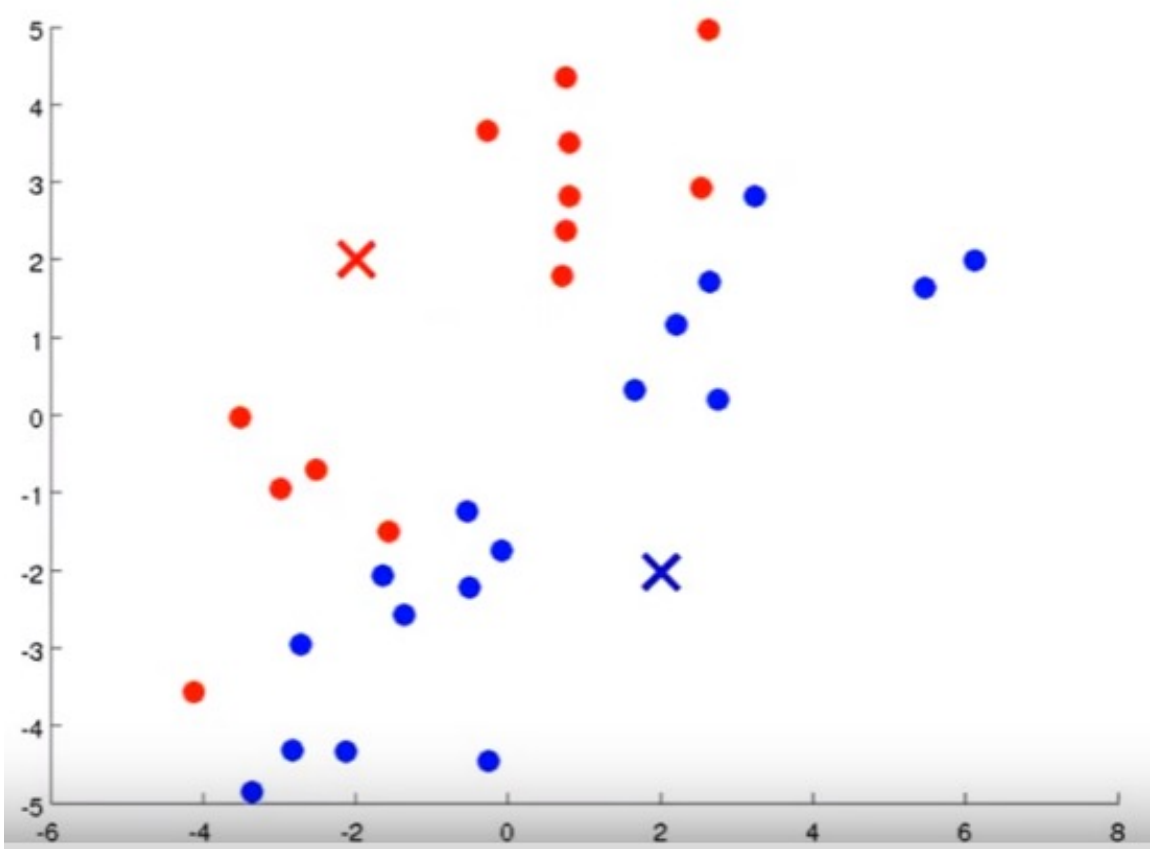
K-means Initial Step

- Randomly initialize two points



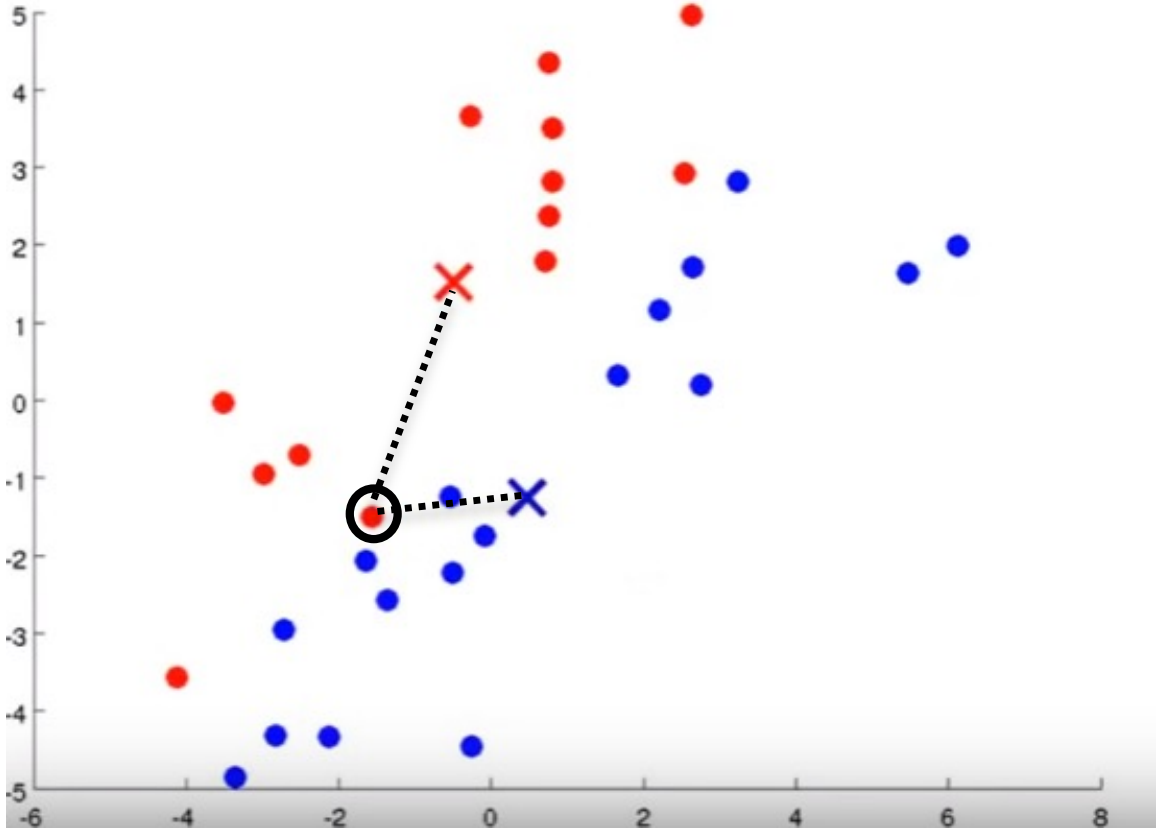
K-means Two Main Steps

1. Cluster assignment
2. Move centroid



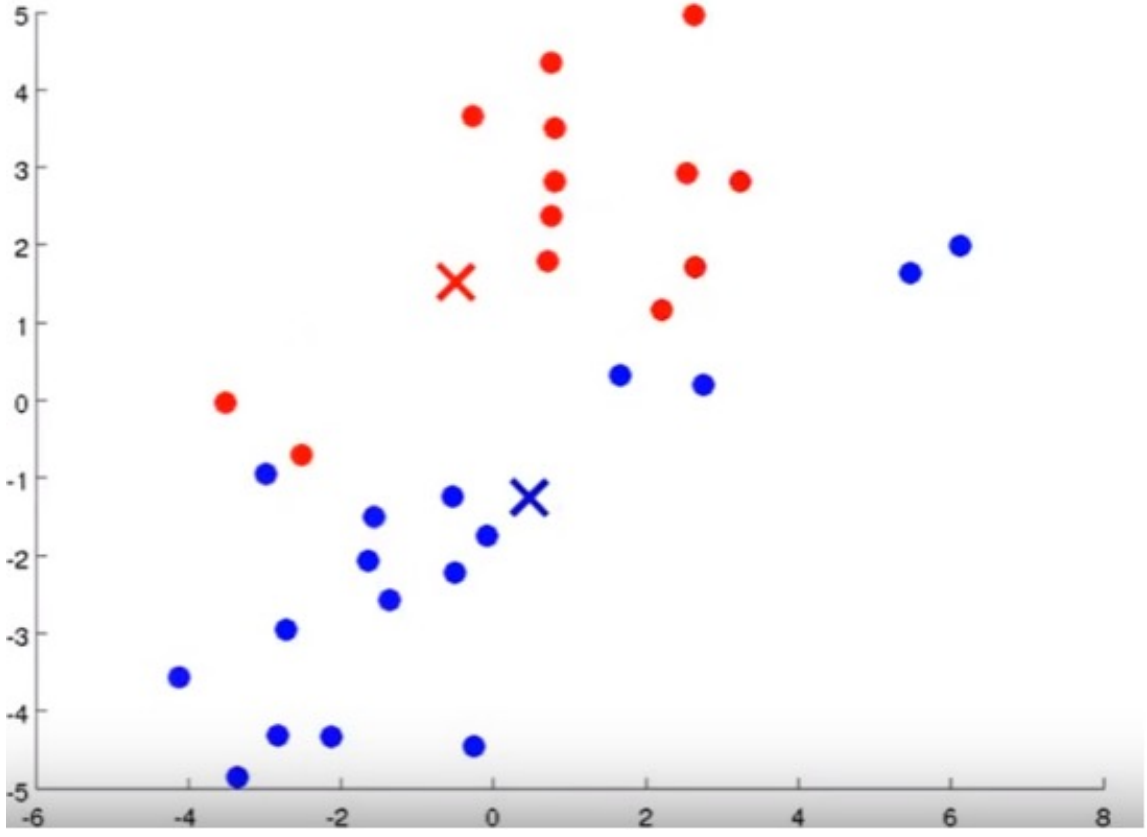
K-means Two Main Steps

1. Cluster assignment
2. Move centroid



K-means Two Main Steps

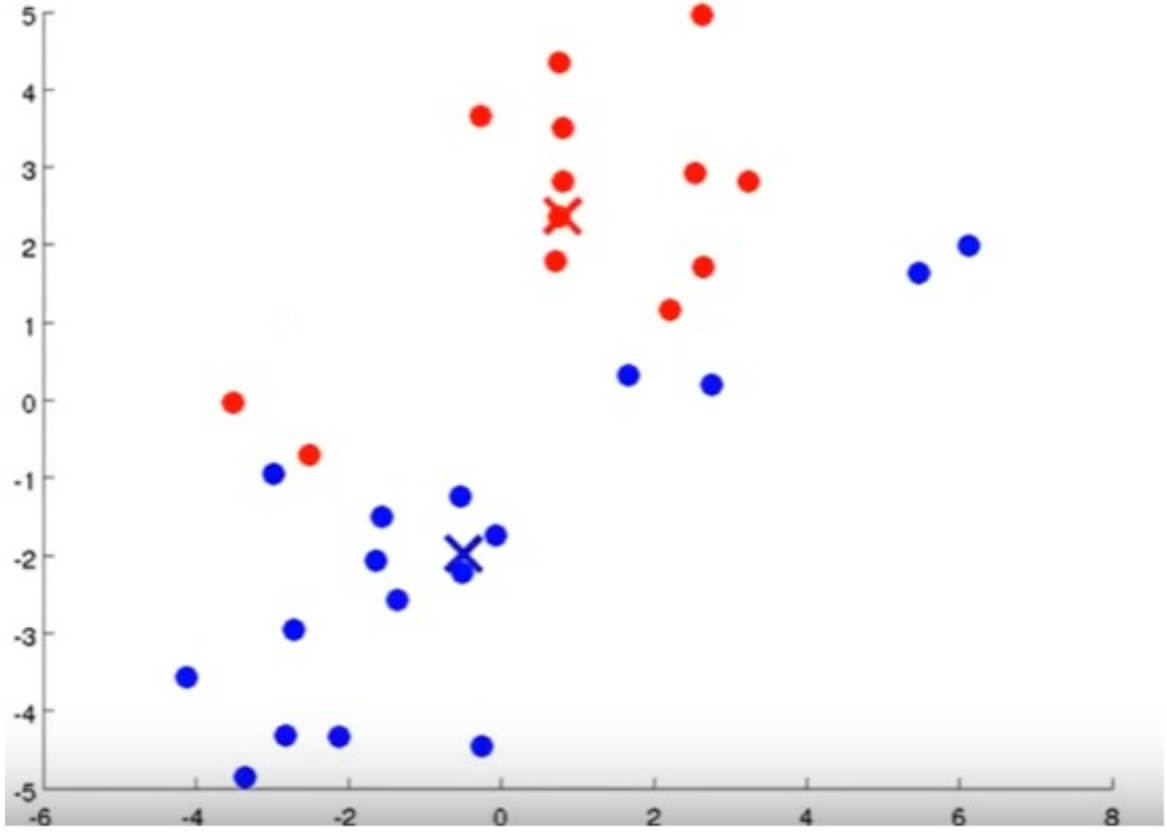
1. Cluster assignment
2. Move centroid



K-means Two Main Steps

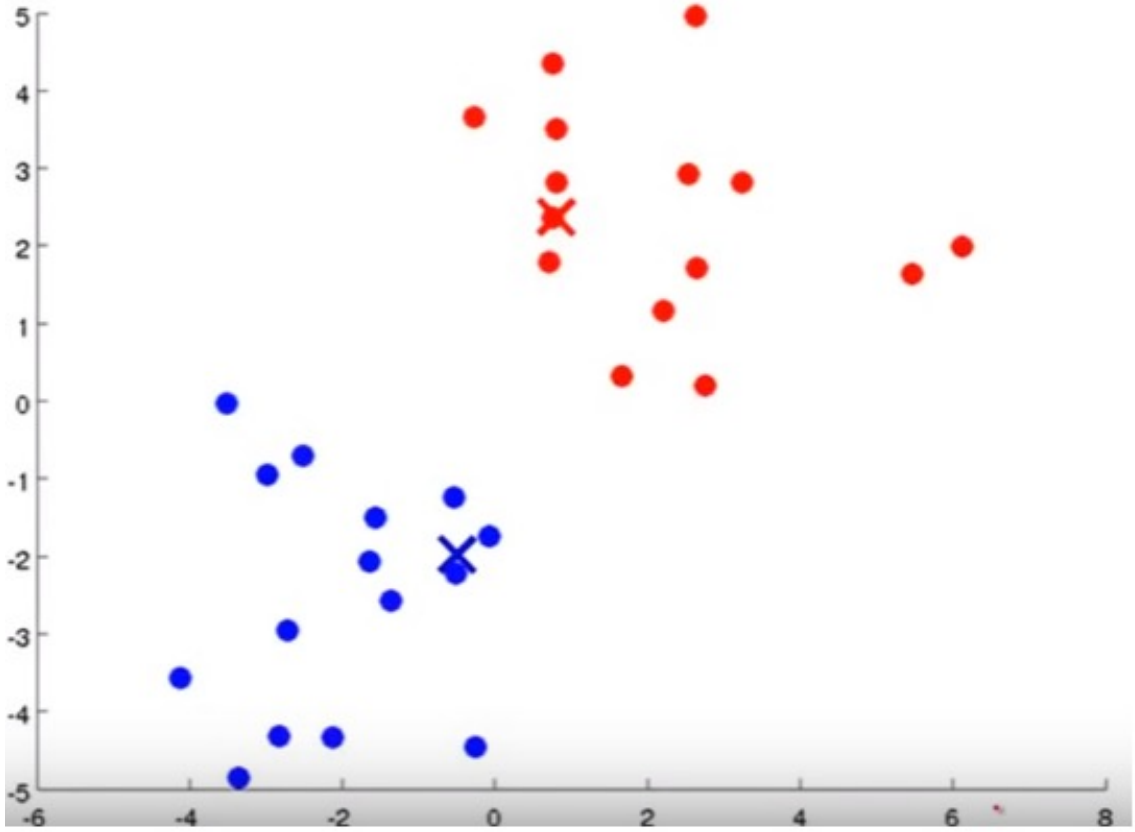
Iterate until there is no changes

1. Cluster assignment
2. Move centroid



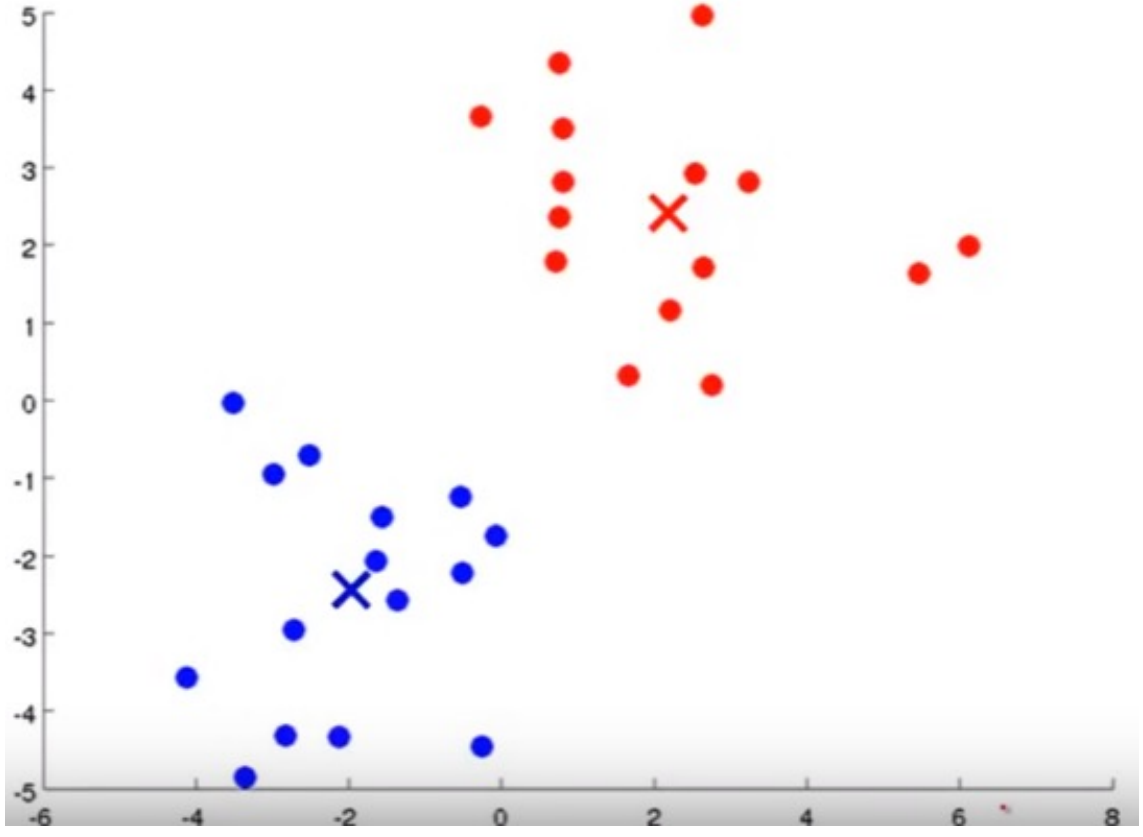
K-means Two Main Steps

1. Cluster assignment
2. Move centroid



K-means Two Main Steps

1. Cluster assignment
2. Move centroid

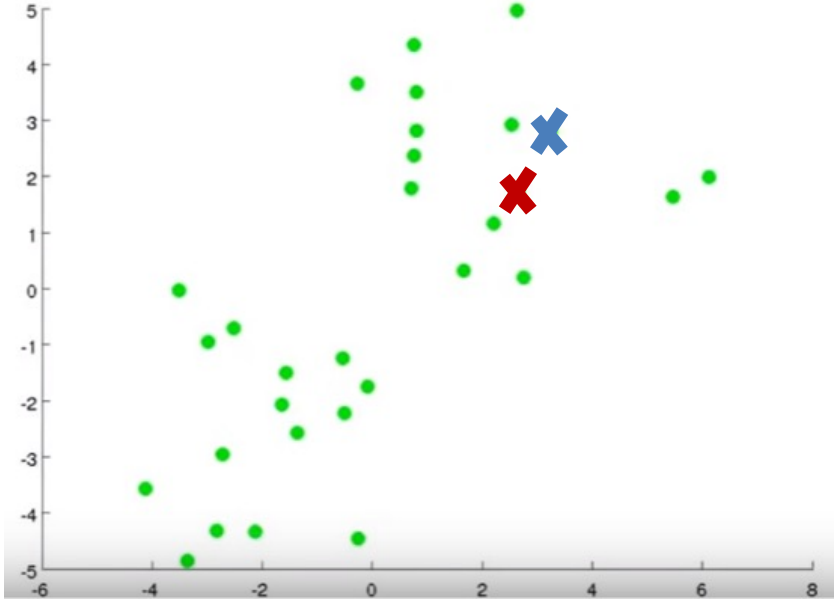


K-means Algorithm

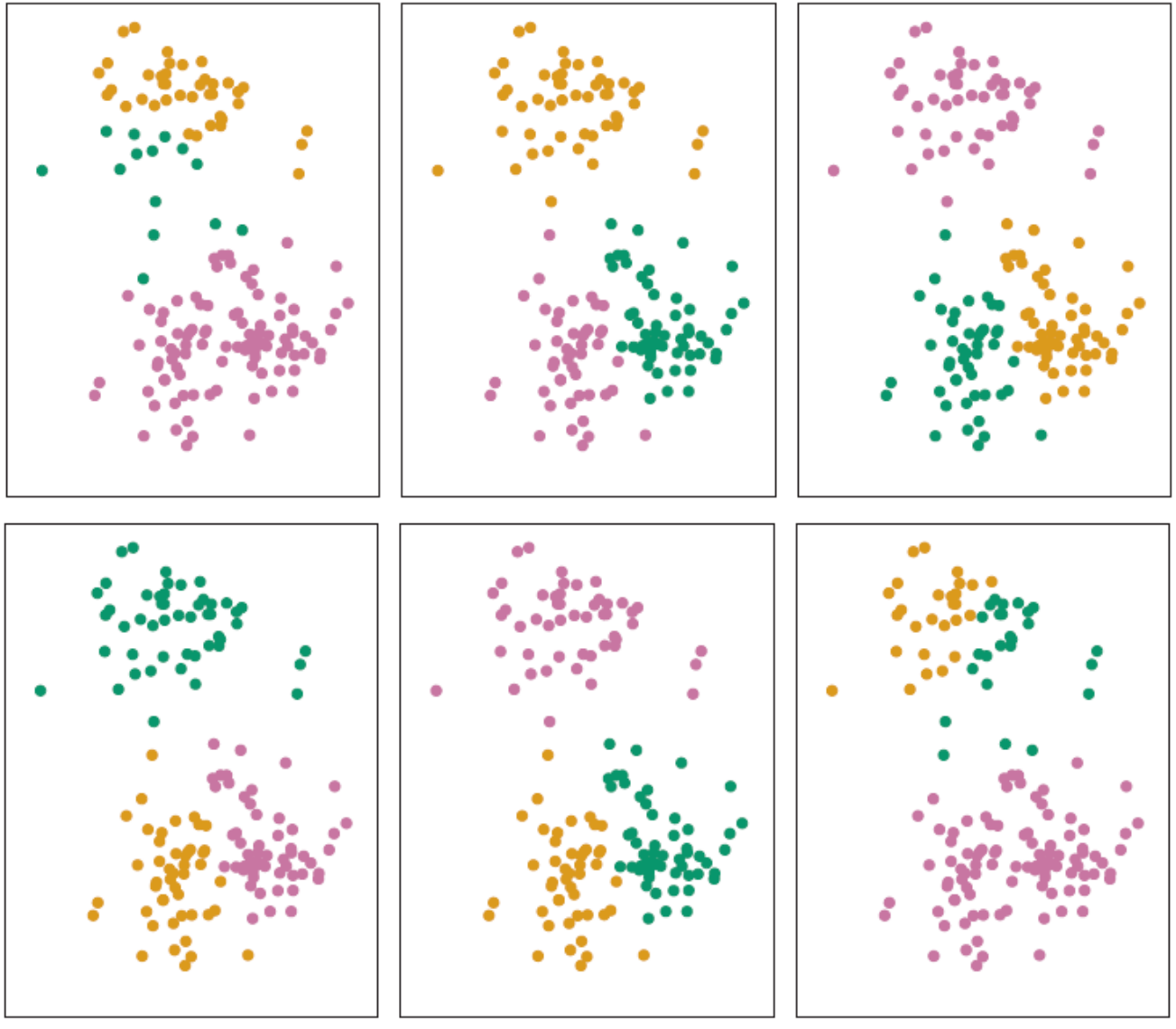
- **Input**:
  - A set of data points
  - The number of clusters (K)
- **Method**:
  - Select K initial random points
  - **Repeat**
    - Cluster assignment
    - Move the cluster centroids to the mean value of data points in the cluster
  - **Until** no change

Group Discussion

- Would the results change if we chose different initial points?



Impact of Random Initial Points

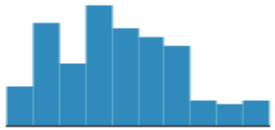


Two Key Messages We Learnt

- Steps of K-means clustering
- Importance of initial step in K-means

Decision Tree Implementation Python

Dataset

# sepal_length	# sepal_width	# petal_length	# petal_width	A species
 4.3 7.9	 2 4.4	 1 6.9	 0.1 2.5	Iris-setosa 33% Iris-versicolor 33% Other (1) 33%
5.1	3.5	1.4	0.2	Iris-setosa
4.9	3	1.4	0.2	Iris-setosa
4.7	3.2	1.3	0.2	Iris-setosa
4.6	3.1	1.5	0.2	Iris-setosa
5	3.6	1.4	0.2	Iris-setosa
5.4	3.9	1.7	0.4	Iris-setosa
4.6	3.4	1.4	0.3	Iris-setosa
5	3.4	1.5	0.2	Iris-setosa
4.4	2.9	1.4	0.2	Iris-setosa
4.9	3.1	1.5	0.1	Iris-setosa
5.4	3.7	1.5	0.2	Iris-setosa
4.8	3.4	1.6	0.2	Iris-setosa
4.8	3	1.4	0.1	Iris-setosa
4.3	3	1.1	0.1	Iris-setosa
5.8	4	1.2	0.2	Iris-setosa
5.7	4.4	1.5	0.4	Iris-setosa
5.4	3.9	1.3	0.4	Iris-setosa

Iris Data Set

Import Dataset

```
from sklearn.datasets import load_iris

#load dataset
iris=load_iris()

#display the names of features
print(iris.feature_names)
```

```
['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']
```

```
#display the names of different types of flowers
print(iris.target_names)
```

```
['setosa' 'versicolor' 'virginica']
```

```
#diplay the value of the features(first observation)
print(iris.data[0])
```

```
[5.1 3.5 1.4 0.2]
```

```
#display the index of flowers
print(iris.target[0])
```

```
0
```

```
#display the dataset
for i in range(len(iris.target)):
    print("Example %d: lable %s, features %s" %(i, iris.target[i],iris.data[i]))
```

```
Example 0: lable 0, features [5.1 3.5 1.4 0.2]
Example 1: lable 0, features [4.9 3.  1.4 0.2]
Example 2: lable 0, features [4.7 3.2 1.3 0.2]
Example 3: lable 0, features [4.6 3.1 1.5 0.2]
Example 4: lable 0, features [5.  3.6 1.4 0.2]
Example 5: lable 0, features [5.4 3.9 1.7 0.4]
Example 6: lable 0, features [4.6 3.4 1.4 0.3]
Example 7: lable 0, features [5.  3.4 1.5 0.2]
```

Splitting Dataset (Test/Training)

```
import numpy as np
from sklearn.datasets import load_iris

iris = load_iris()
test_idx = [0,50,100]

#training data
train_target = np.delete(iris.target, test_idx)
train_data = np.delete(iris.data, test_idx, axis = 0)

#testing data
test_target = iris.target[test_idx]
test_data = iris.data[test_idx]
```

Train a Classifier

```
from sklearn import tree

#create classifier decision tree and train it on the training data
clf = tree.DecisionTreeClassifier()
clf.fit(train_data, train_target)
```

```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, presort=False, random_state=None,
            splitter='best')
```

Predict Label For New Flower

```
# Making a Prediction On a New Sample
sample_one_pred = int(clf.predict([[5, 5, 1, 3]]))
sample_two_pred = int(clf.predict([[5, 5, 2.6, 1.5]]))
print(f"The first sample most likely is a {iris.target_names[sample_one_pred]} flower.")
print(f"The second sample most likely is a {iris.target_names[sample_two_pred]} flower.")
```

```
The first sample most likely is a setosa flower.
The second sample most likely is a versicolor flower.
```

Visualise The Tree

```
import matplotlib.pyplot as plt
fig = plt.figure(figsize=(25,20))
_ = tree.plot_tree(clf,
                   feature_names=iris.feature_names,
                   class_names=iris.target_names,
                   filled=True)
```

Visualise The Tree

Tutorial/Lab Week 7

- We use Python to
  - train Decision Trees and random forests
  - build clusters using k-means