Machine learning: prediction, classification and clustering

UBB Faculty of Sociology

Course Agenda

#1 Intro, Simple Linear Regression

#2 Python recap, Git, Handling data, EDA

#3 Regression, Decision Trees

#4 Bias, Variance, Overfitting, Classification

#5 Random Forest Classifier, Clustering

#6 Neural Networks

#7 Help Final Project

#8 Help Final Project

5. Random Forest, Clustering

5.1 Catch up

Share one thing that stood out to you:

one thing you found surprising / interesting / useful etc.

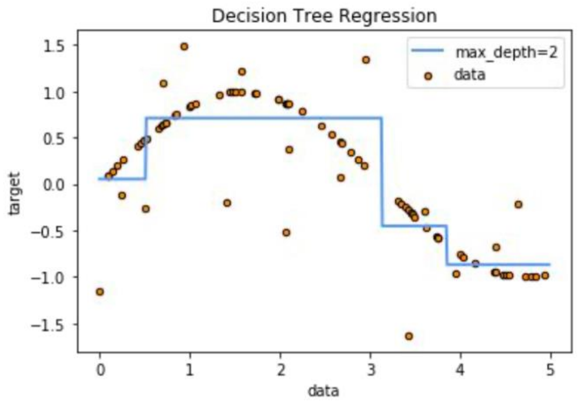
5.2 Decision Tree Recap

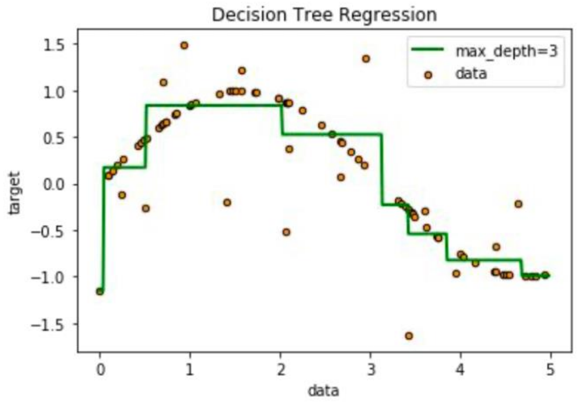
Decision Tree Regression

How it works: A two step repeating process

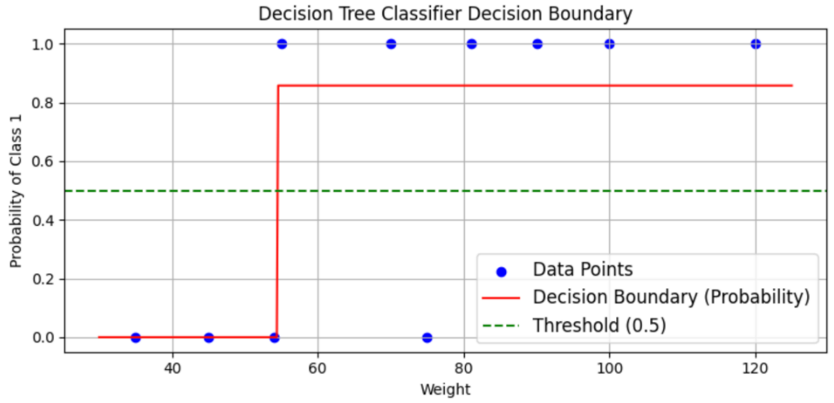
1. At each node, evaluate **all possible splits** of the dataset based on every feature.
2. Choose the split that **minimizes the regression error metric MSE**.

Then, repeat for the newly created subtrees





Decision Tree Classification

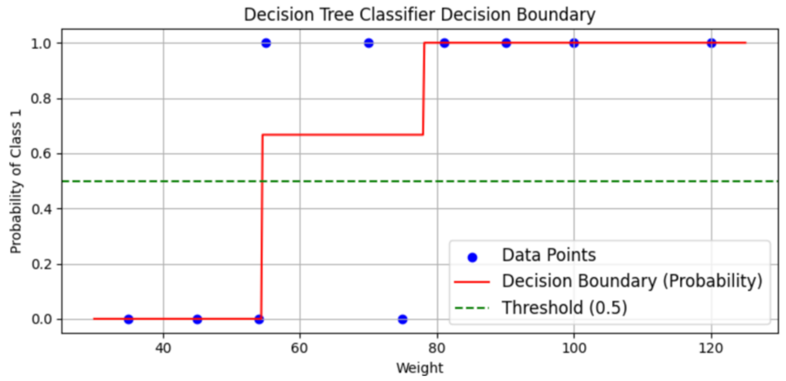


How it works: A two step repeating process

1. At each node, evaluate **all possible splits** of the dataset based on every feature.

2. Choose the split that **minimizes the classification error metric Gini impurity**

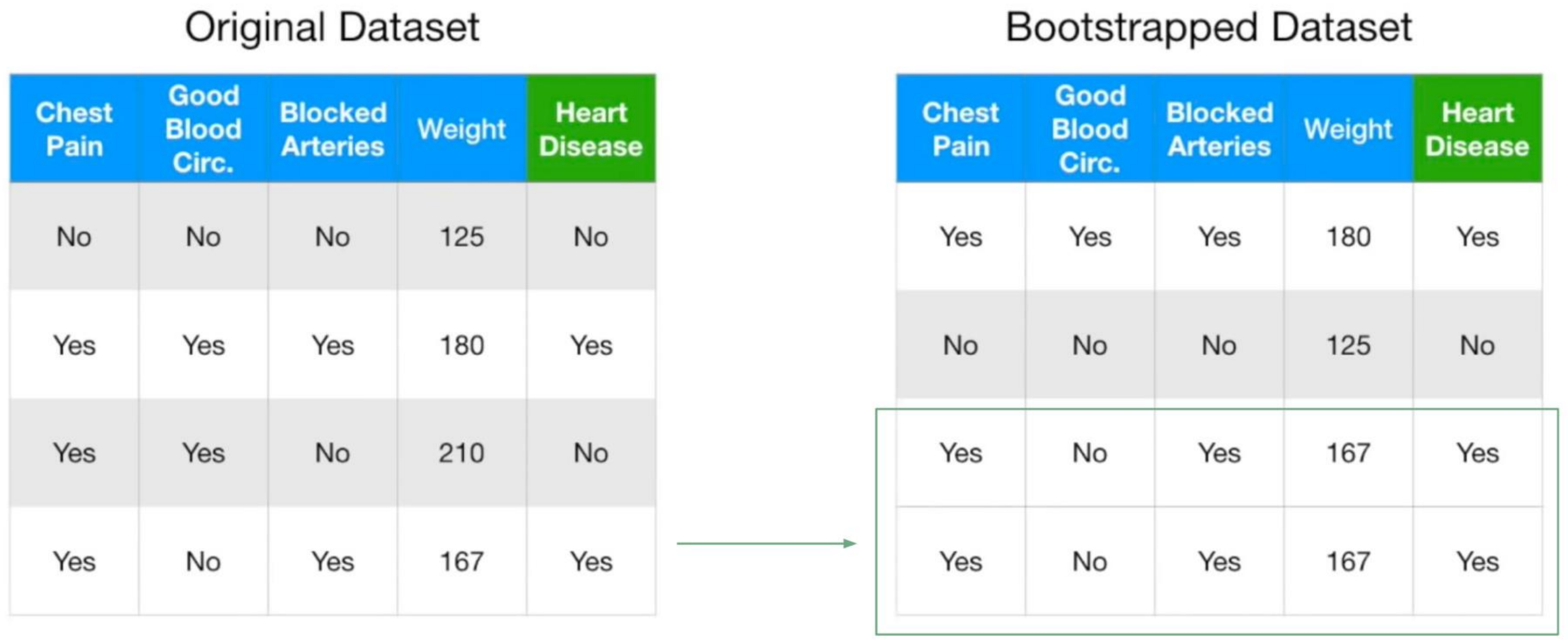
Then, repeat for the newly created subtrees



5.3 Random Forests

Are build from decision trees.

Step 1. Create a ‘bootstrapped’ dataset. Randomly select samples, with replacement.



Original Dataset

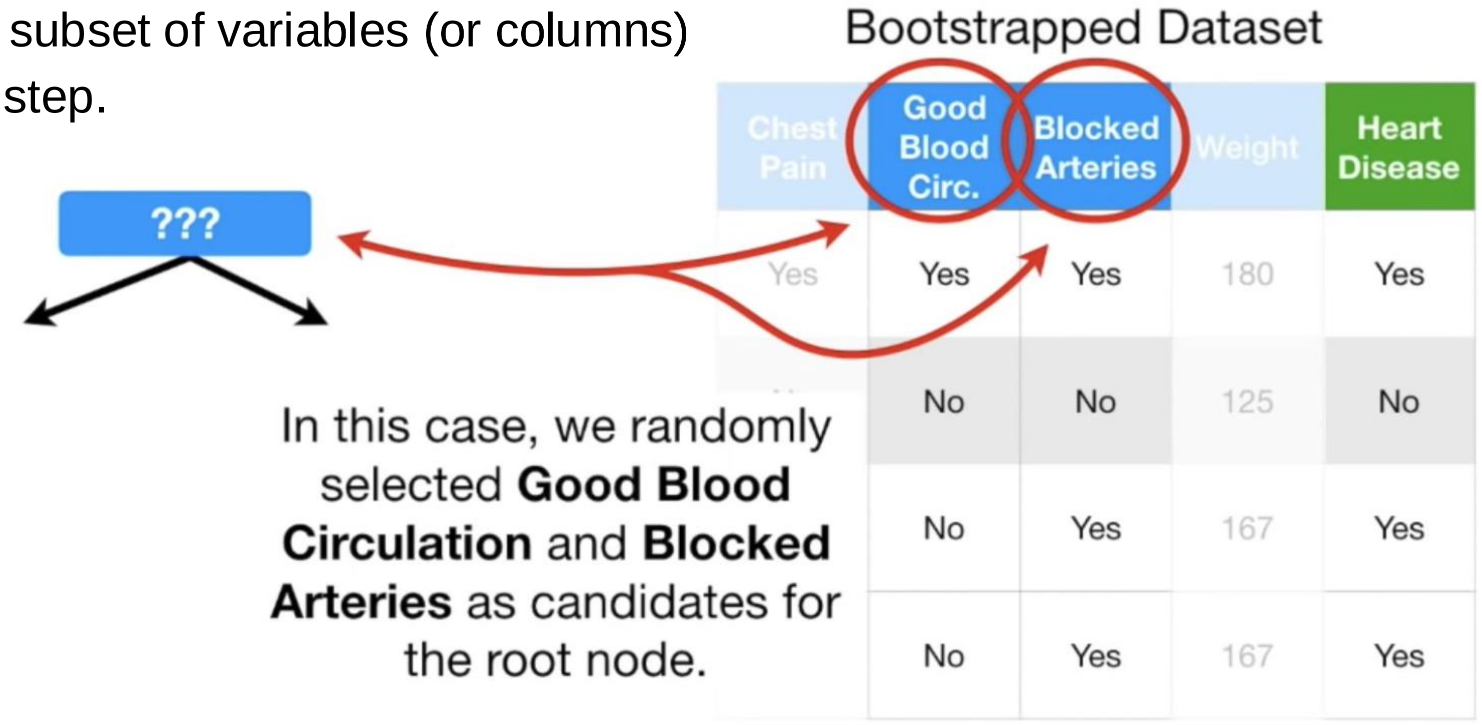
Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	No	Yes	167	Yes

Bootstrapped Dataset

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
Yes	Yes	Yes	180	Yes
No	No	No	125	No
Yes	No	Yes	167	Yes
Yes	No	Yes	167	Yes

Step 2. Create a decision tree using a bootstaped dataset, but only use a

random subset of variables (or columns) at each step.

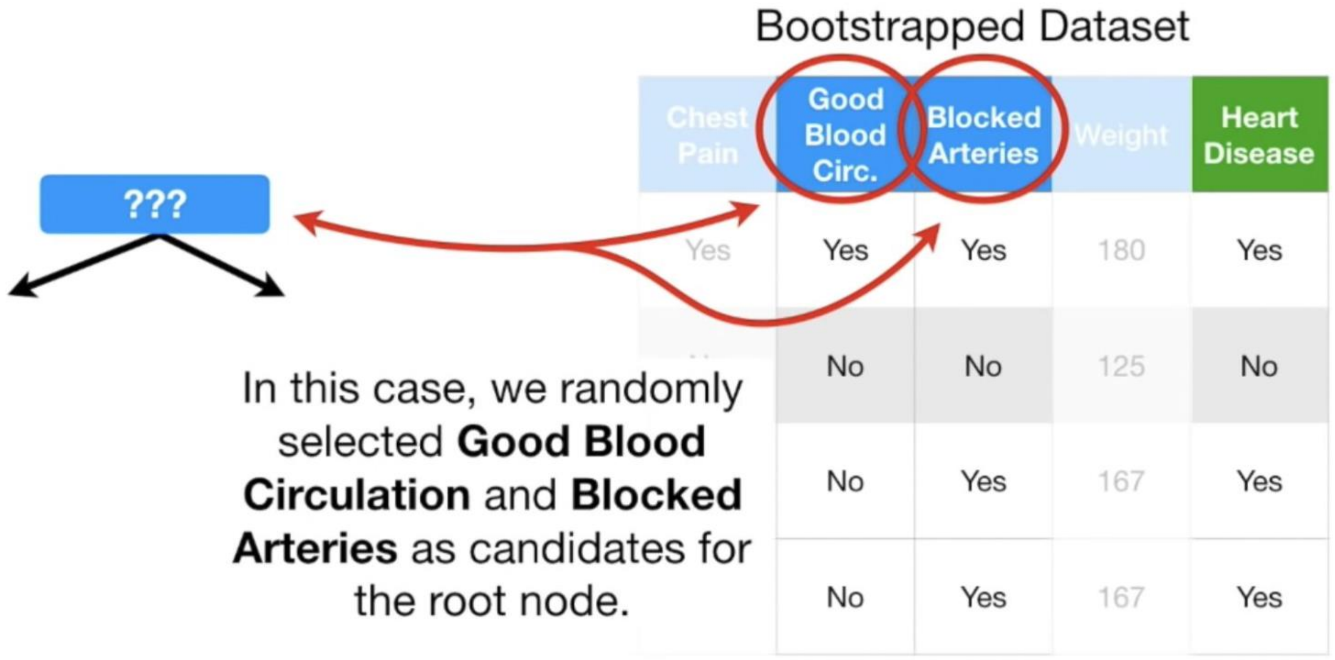


Bootstrapped Dataset

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
Yes	Yes	Yes	180	Yes
[illegible]	No	No	125	No
[illegible]	No	Yes	167	Yes
[illegible]	No	Yes	167	Yes

In this case, we randomly selected **Good Blood Circulation** and **Blocked Arteries** as candidates for the root node.

We find which variable out of two does the best job of separating the samples.

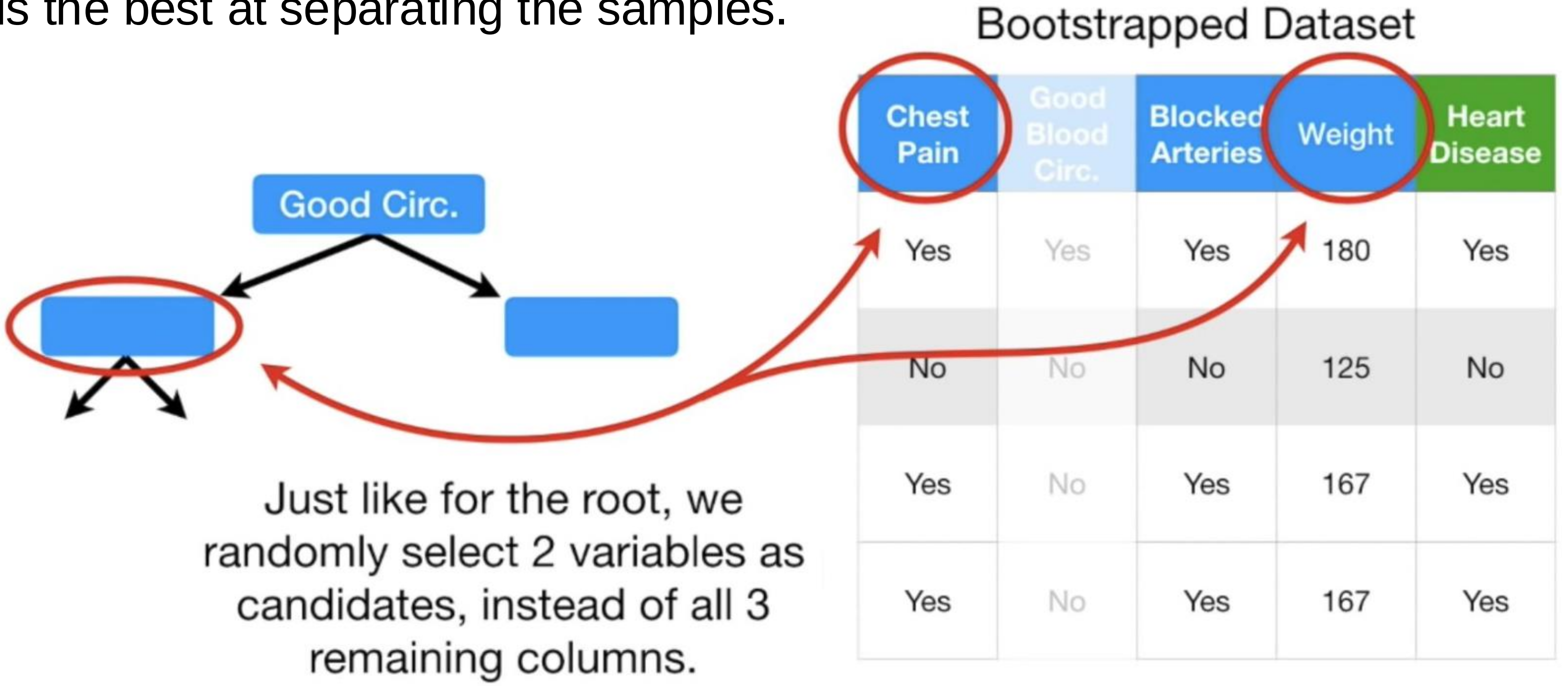


Bootstrapped Dataset

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
Yes	Yes	Yes	180	Yes
[illegible]	No	No	125	No
[illegible]	No	Yes	167	Yes
[illegible]	No	Yes	167	Yes

In this case, we randomly selected **Good Blood Circulation** and **Blocked Arteries** as candidates for the root node.

Let’s assume that is ‘Good Blood Circulation’. Now we have to figure out which variable to use food the next node. We choose other 2 variables and find out which one is the best at separating the samples.



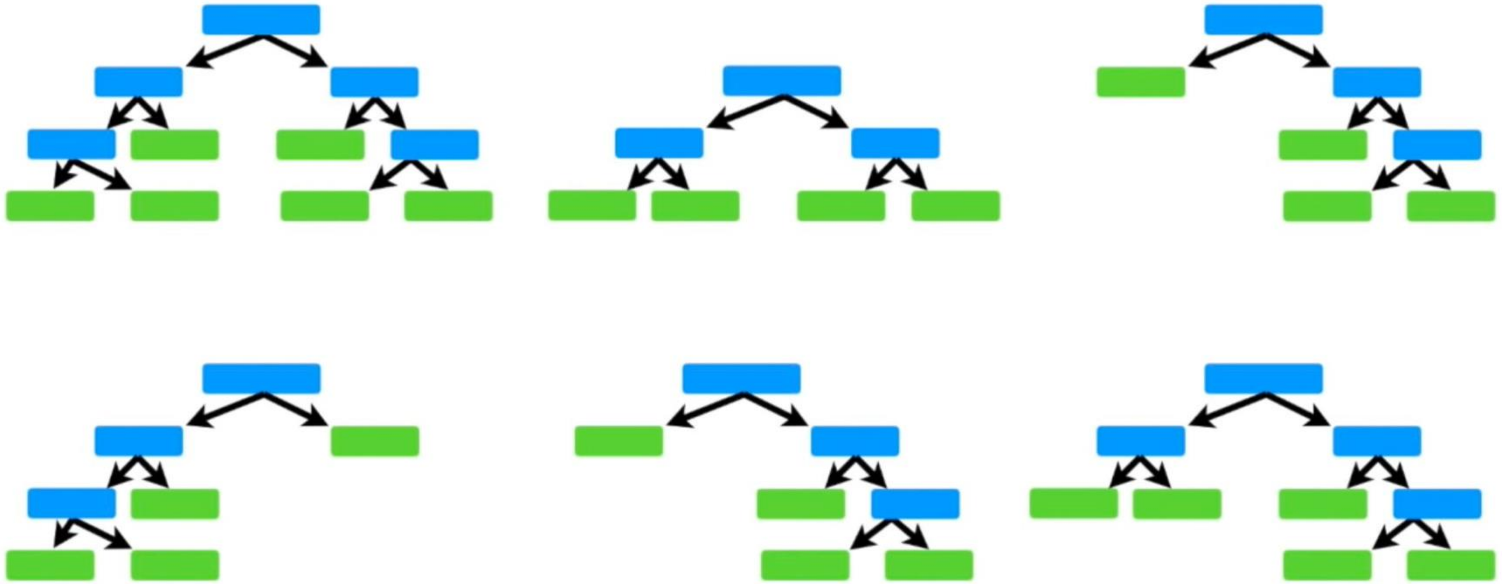
Bootstrapped Dataset

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
Yes	Yes	Yes	180	Yes
No	No	No	125	No
Yes	No	Yes	167	Yes
Yes	No	Yes	167	Yes

Just like for the root, we randomly select 2 variables as candidates, instead of all 3 remaining columns.

Go back to step 1 and repeat.

Create a decision tree using a bootstaped dataset, but only use a random subset of variables (or columns) at each step.



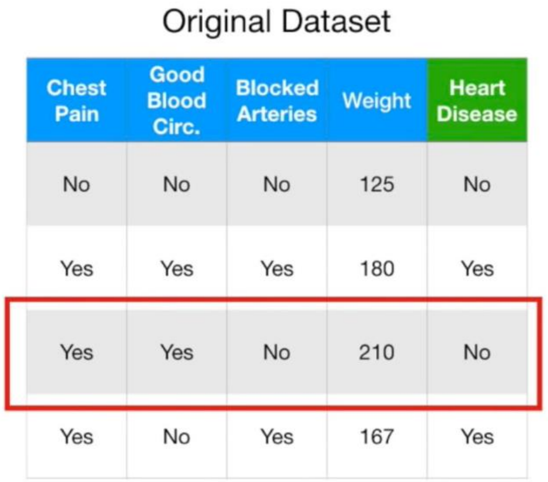
Bootstrapping the data plus using the **agg**regate to make a decision is called "**Bagging**"

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
Yes	No	No	168	**YES**

Heart Disease	
Yes	No
5	1

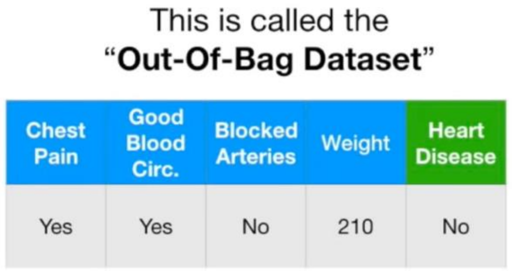
1. Evaluate on evaluation set.
2. Evaluate on OOB:

Estimate the accuracy of a random forest by evaluating Out-Of-Bag data on the trees that it was not used when created.



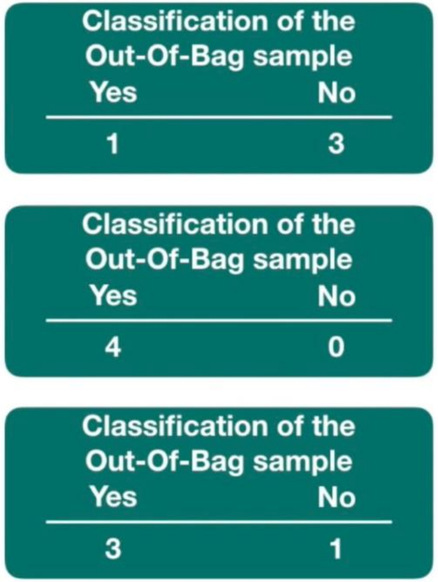
Original Dataset

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	No	Yes	167	Yes



This is called the "**Out-Of-Bag Dataset**"

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
Yes	Yes	No	210	No



Classification of the Out-Of-Bag sample

Yes	No
1	3

Classification of the Out-Of-Bag sample

Yes	No
4	0

Classification of the Out-Of-Bag sample

Yes	No
3	1

Random Forest:

- Each tree - bootstrapped sampling (random with replacement)
- Each node - random features (2 or more)
- Split: Gini impurity, Entropy
- Aggregates the result of trees, voting mechanism
- Bagging: bootstrapped + aggregate

Pros:

- High Accuracy: Combines multiple trees to reduce overfitting and improve prediction accuracy. Handles Missing Data: Can manage missing values and works with both categorical and numerical data.
- Feature Importance: Provides insights into which features significantly impact predictions.
- Resilient to Overfitting: The combination of multiple trees reduces the risk of overfitting, especially if hyperparameters are tuned properly.

Cons:

- Complexity: Difficult to interpret compared to single decision trees.
- Computationally Intensive, high memory usage: Training and prediction can be slow, especially with large datasets.
- Bias in Feature Importance: Can overestimate the importance of numerical or high-cardinality features.

5.4 Feature Permutation

- **Permutation Feature Importance**

Measures the increase in prediction error after permuting the feature.

- Feature is important if:
  - Shuffling its value increases model error.
- Feature is not important if:
  - Shuffling its value leaves error unchanged.

Permutation Feature Importance

Steps:

- Estimate the original error.
- For each feature:

  - Permute the feature values in the data to break its association with the true outcome

- Estimate error based on the prediction of the permuted feature,
- Calculate permutation feature importance,
- Sort features by descending feature importance.

5.5 Clustering, k-means

Unsupervised Learning

The data have no target attribute.

○ We want to explore the data to find some intrinsic structures in them.

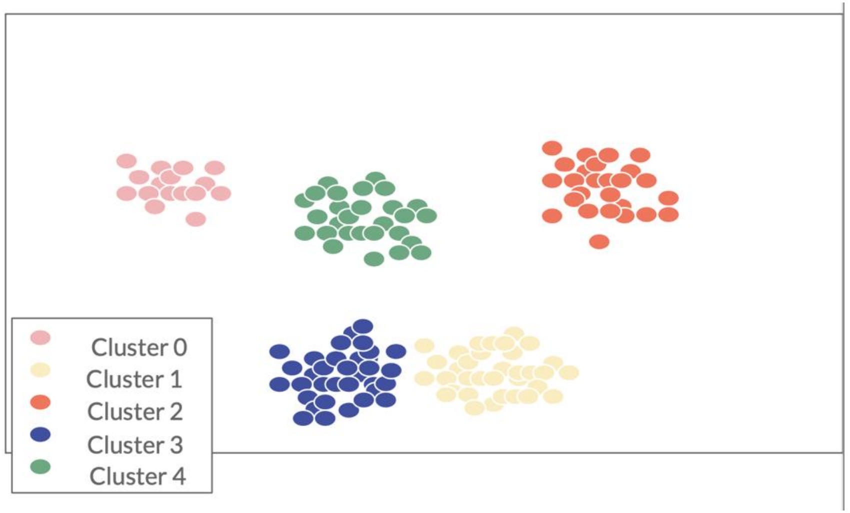
In real life, most of the data relates to each other in ways we don’t know about.

K-means

Unlabelled data: letting the ML find patterns for you
training set:
- attributes-features-predictors
- **NO class** (target feature).

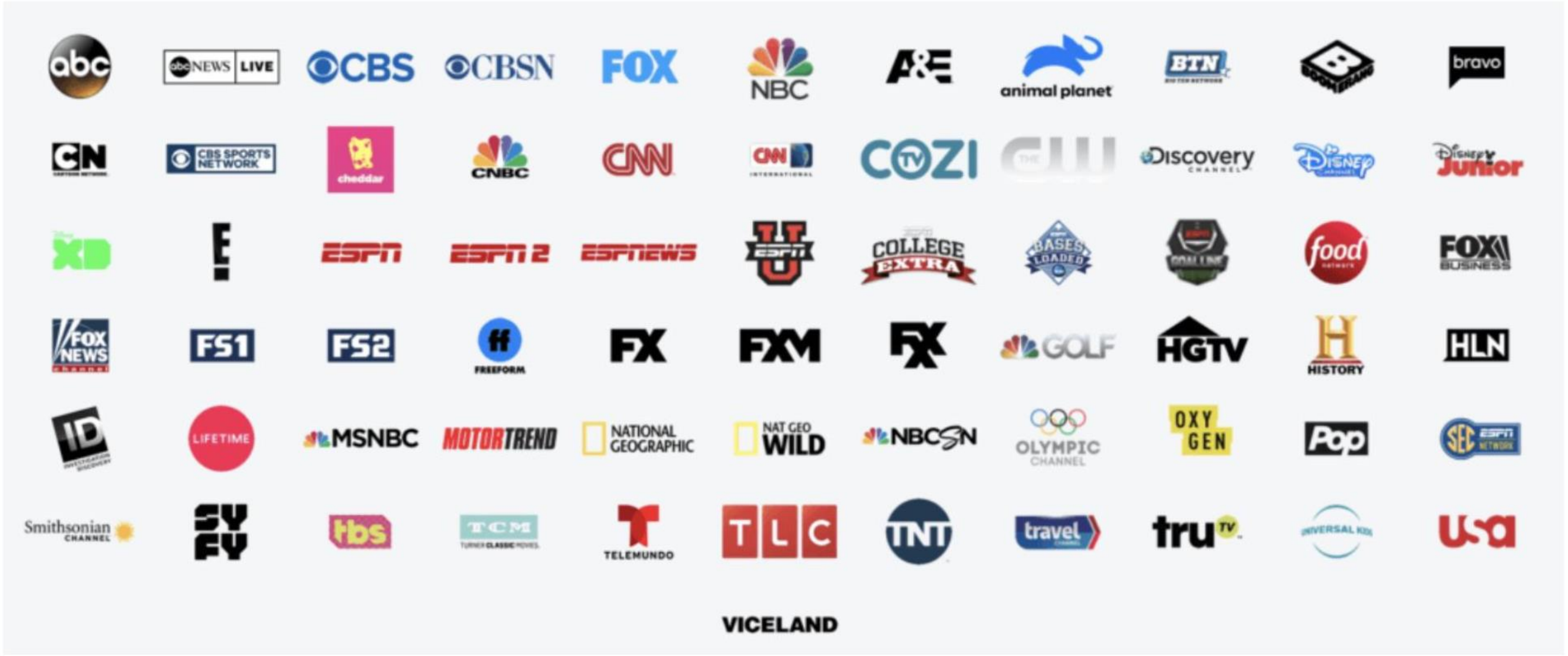
output: find groups of cases with

nr of clusters:
-deciding before-hand (difficult decision)
-More groups -> better fit, is it better?



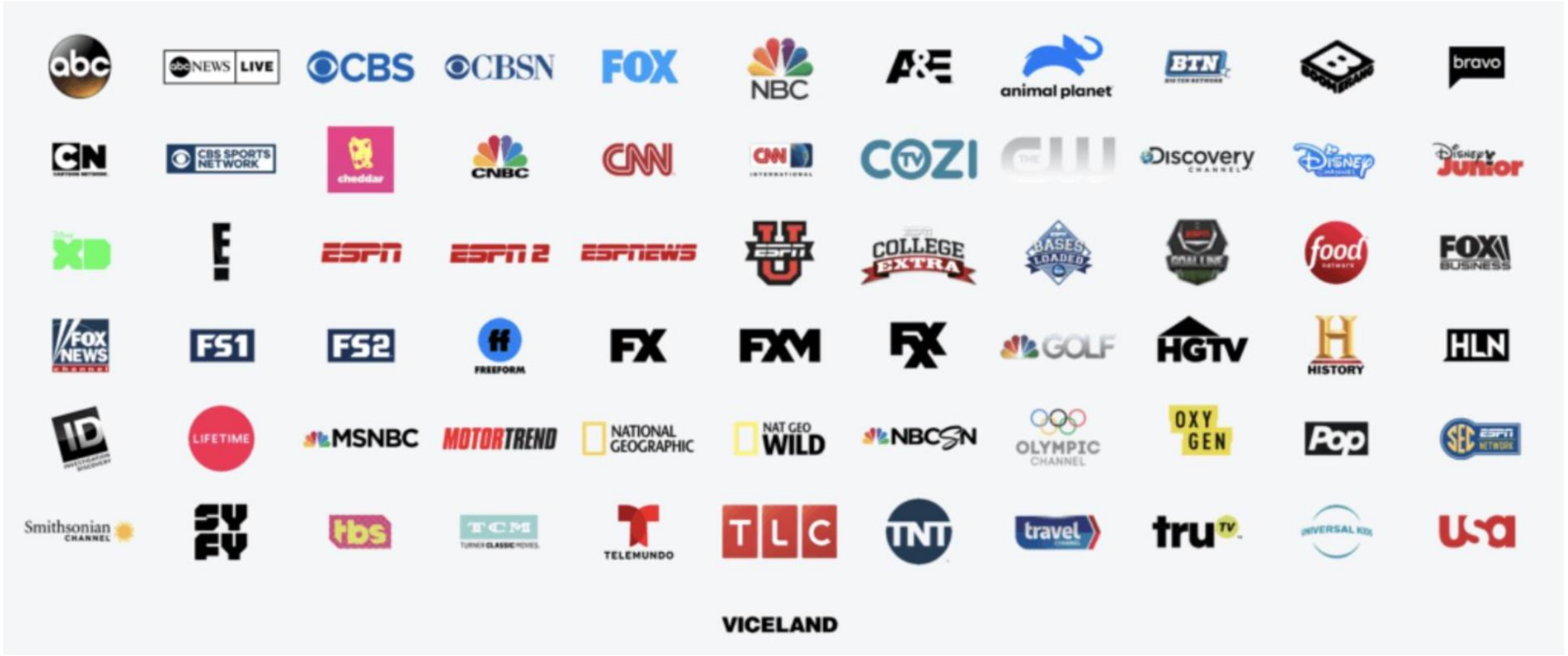
Distance: Media consumption

Suppose we record the number of hours each user watched each channel. Cluster viewers by media consumption.

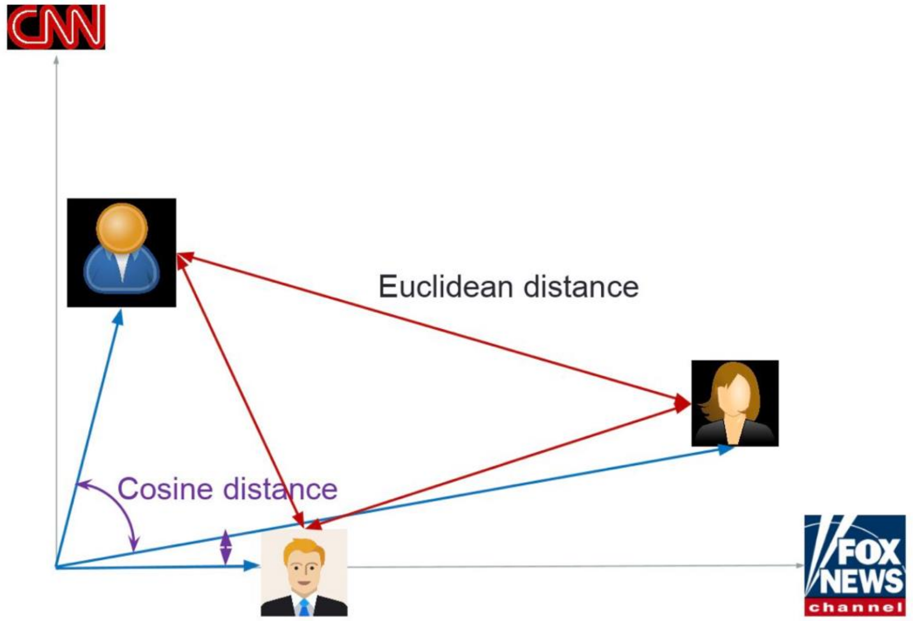


Use cases:

- Understand audience segmentation for better marketing strategies.
- Identify underperforming or highly popular channels among specific groups.
- Predict future viewing patterns based on cluster behavior.



Only focus on 2 channels:



K-means steps:

Code on Colab

1. **Choose the Number of Clusters (k)**:
   - Decide how many groups (k) you want to divide your data into.
2. **Initialize Centroids**:
   - Randomly select k centroids (points representing the center of each cluster.
3. **Assign Data Points to Nearest Centroid**:
   - For each data point, calculate its distance to each centroid.
   - Assign the data point to the cluster of the closest centroid.
4. **Update Centroids**:
   - Recalculate the position of each centroid by taking the **mean** of all data points assigned to that cluster.
5. **Repeat**:
   - Reassign data points to the nearest centroids based on the updated centroids.
   - Recalculate centroids for the new clusters.
6. **Stop When Converged**:
   - The algorithm stops when the centroids no longer change significantly or the maximum number of iterations is reached.

- **Intra-Cluster Distance (Compactness)**:
  - Measure how close the points within each cluster are to their cluster centroid.
  - Use metrics like **Sum of Squared Distances (SSD)** to ensure points are tightly packed within clusters. Lower values indicate better clustering.
- **Inter-Cluster Distance (Separation)**:
  - Measure how far apart the centroids of different clusters are.
  - Greater distances between centroids mean clusters are well-separated and distinct.

- **Silhouette Score:** measures both **Compactness and Separation:** how well data points fit within their clusters and how distinct the clusters are from each other.
- Elbow method: method for finding the optimal K value in a k-means clustering algorithm

5.6 Final Project Task 4: Census Data Clustering - Optional

5.5 Further Reading & Questions

#1 Random Forests: https://www.youtube.com/watch?v=J4Wdy0Wc_xQ

#2 Random Forests for missing data imputation and clustering:

https://www.youtube.com/watch?v=sQ870aTKqiM

#3 K-means clustering: https://www.youtube.com/watch?v=4b5d3muPQmA

#3 Feature Permutation Importance: https://www.youtube.com/watch?v=VUvShOEFdQo

#4 Anomaly detection example: https://www.datarobot.com/use-cases/credit-card-fraudulent-transactions/

Thank you !!

Machine Learning Engineer / Data Scientist
zahariesergiu@gmail.com
https://www.linkedin.com/in/zahariesergiu/
https://github.com/zahariesergiu/ubb-sociology-ml