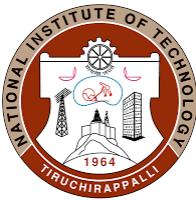


CSPE65: Machine Learning Techniques and Practices

Ph: 999 470 4853

Dr. R. Bala Krishnan
Asst. Prof.
Dept. of CSE
NIT, Tiruchirappalli – 620 015
E-Mail: balakrishnan@nitt.edu



ML Algorithms



Classification Algorithm

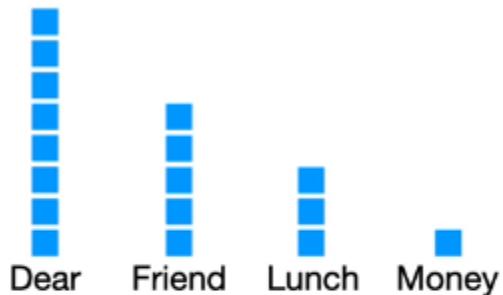
Supervised Learning Algorithms

- Linear Classifier -> Logistic Regression, **Naive Bayes Classifier**
- **Nearest Neighbor**
- Support Vector Machine
- **Decision Trees**
- Boosted Trees
- **Random Forest**
- Neural Network

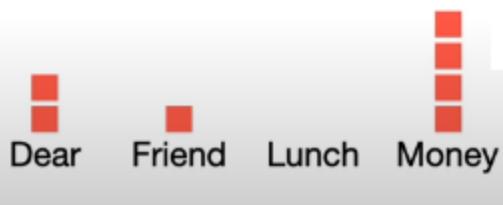
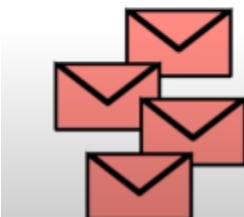
Naive Bayes Classifier

- Classify “Spam” and “Normal” messages

Normal Message = 8

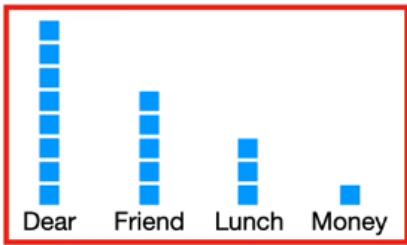


Spam Message = 4



Naive Bayes Classifier

- Classify “Spam” and “Normal” messages

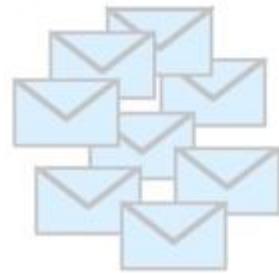


$$p(\text{ Friend } | \text{ Normal }) = \frac{5}{17} = 0.29$$

$$p(\text{ Lunch } | \text{ Normal }) = \frac{3}{17} = 0.18$$

$$p(\text{ Dear } | \text{ Normal }) = \frac{8}{17} = 0.47$$

$$p(\text{ Money } | \text{ Normal }) = \frac{1}{17} = 0.06$$



$$\begin{aligned} p(\text{ Dear } | \text{ N }) &= 0.47 \\ p(\text{ Friend } | \text{ N }) &= 0.29 \\ p(\text{ Lunch } | \text{ N }) &= 0.18 \\ p(\text{ Money } | \text{ N }) &= 0.06 \end{aligned}$$

$$p(\text{ N }) = \frac{8}{8+4} = 0.67$$

$$p(\text{N}) = 0.67$$



$$\begin{aligned} p(\text{ Dear } | \text{ S }) &= 0.29 \\ p(\text{ Friend } | \text{ S }) &= 0.14 \\ p(\text{ Lunch } | \text{ S }) &= 0.00 \\ p(\text{ Money } | \text{ S }) &= 0.57 \end{aligned}$$

$$p(\text{ S }) = \frac{4}{4+8} = 0.33$$

$$p(\text{S}) = 0.33$$

- Because we have calculated the probabilities of discrete, individual words, and not the probability of something continuous, like weight or height, these probabilities are also called as likelihoods
- Initial guess that we observe a *Normal* messages is called a Prior Probability

New Message = Dear Friend

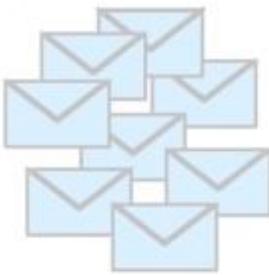
$$p(\text{ N }) \times p(\text{ Dear } | \text{ N }) \times p(\text{ Friend } | \text{ N })$$

$$0.67 \times 0.47 \times 0.29 = 0.09 \propto p(\text{ N } | \text{ Dear Friend })$$

$$p(\text{ S }) \times p(\text{ Dear } | \text{ S }) \times p(\text{ Friend } | \text{ S })$$

$$0.33 \times 0.29 \times 0.14 = 0.01 \propto p(\text{ S } | \text{ Dear Friend })$$

Naive Bayes Classifier



$$\begin{aligned} p(\text{Dear} | \text{N}) &= 0.47 \\ p(\text{Friend} | \text{N}) &= 0.29 \\ p(\text{Lunch} | \text{N}) &= 0.18 \\ p(\text{Money} | \text{N}) &= 0.06 \end{aligned}$$

$$p(\text{N}) = 0.67$$



$$\begin{aligned} p(\text{Dear} | \text{S}) &= 0.29 \\ p(\text{Friend} | \text{S}) &= 0.14 \\ p(\text{Lunch} | \text{S}) &= 0.00 \\ p(\text{Money} | \text{S}) &= 0.57 \end{aligned}$$

$$p(\text{S}) = 0.33$$

$$p(\text{N}) \times p(\text{Lunch} | \text{N}) \times p(\text{Money} | \text{N})^4 = 0.000001$$

$$p(\text{S}) \times p(\text{Lunch} | \text{S}) \times p(\text{Money} | \text{S})^4 = 0.00122$$

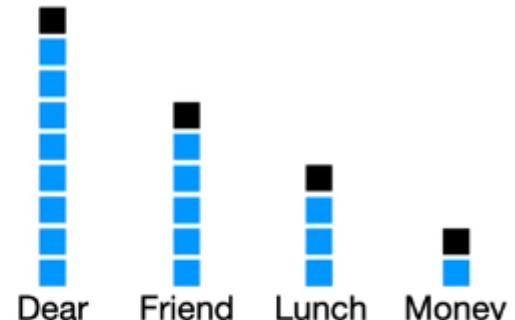
New Message = Lunch Money Money Money

$$p(\text{N}) \times p(\text{Lunch} | \text{N}) \times p(\text{Money} | \text{N})^4 = 0.000001$$

$$p(\text{S}) \times p(\text{Lunch} | \text{S}) \times p(\text{Money} | \text{S})^4 = 0$$



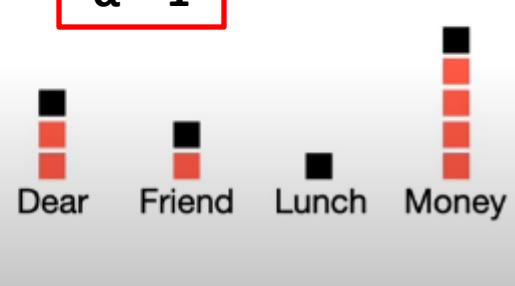
$$p(\text{N}) = 0.67$$



$$\alpha = 1$$



$$p(\text{S}) = 0.33$$



Naive Bayes Classifier



$$p(N) = 0.67$$

$$\begin{aligned} p(\text{Dear} | N) &= 0.43 \\ p(\text{Friend} | N) &= 0.29 \\ p(\text{Lunch} | N) &= 0.19 \\ p(\text{Money} | N) &= 0.10 \end{aligned}$$

Treats all word orders the same

By ignoring relationships among words, Naive Bayes has high bias. But, because it works well in practise, it has low variance.

Score for **Dear Friend** =

$$p(N) \times p(\text{Dear} | N) \times p(\text{Friend} | N) = 0.08$$

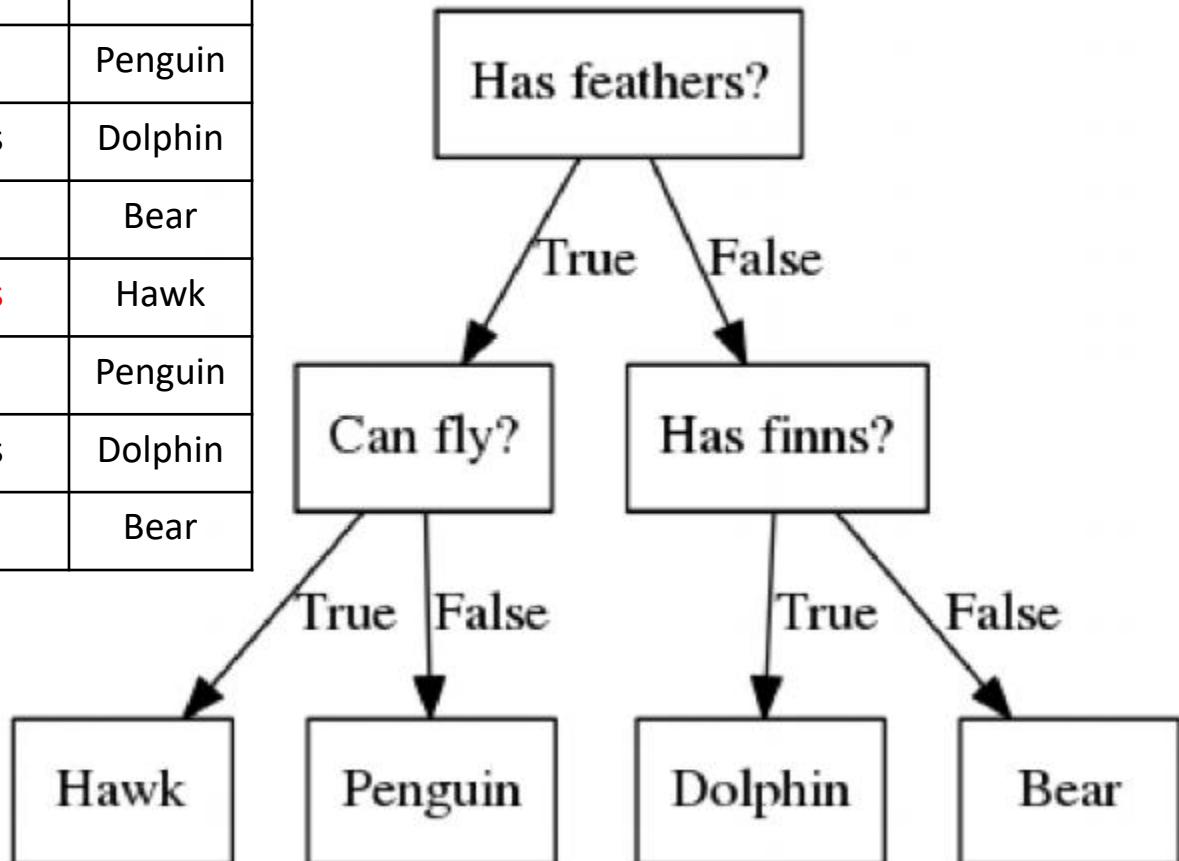
Score for **Friend Dear** =

$$p(N) \times p(\text{Friend} | N) \times p(\text{Dear} | N) = 0.08$$

Decision Trees

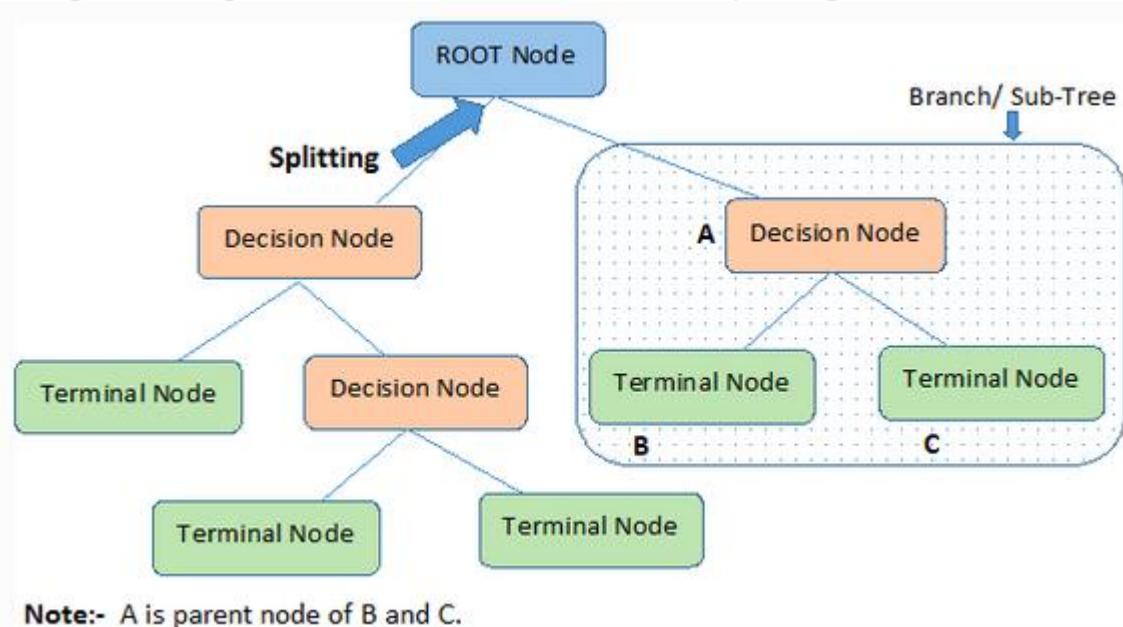
Sl. No.	Feather	Fly	Finns	Target
1.	Yes	Yes	Yes	Hawk
2.	Yes	No	No	Penguin
3.	No	No	Yes	Dolphin
4.	No	Yes	No	Bear
5.	Yes	Yes	Yes	Hawk
6.	Yes	No	No	Penguin
7.	No	No	Yes	Dolphin
8.	No	Yes	No	Bear

animal_tree



Decision Trees (Terminologies)

1. **Root Node:** This attribute is used for dividing the data into two or more sets. The feature attribute in this node is selected based on Attribute Selection Techniques.
2. **Branch or Sub-Tree:** A part of the entire decision tree is called a branch or sub-tree.
3. **Splitting:** Dividing a node into two or more sub-nodes based on if-else conditions.
4. **Decision Node:** After splitting the sub-nodes into further sub-nodes, then it is called the decision node.
5. **Leaf or Terminal Node:** This is the end of the decision tree where it cannot be split into further sub-nodes.
6. **Pruning:** Removing a sub-node from the tree is called pruning.



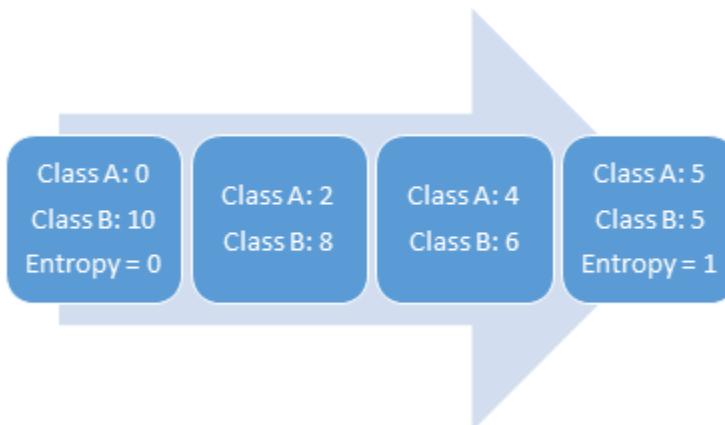


Decision Trees (Terminologies)

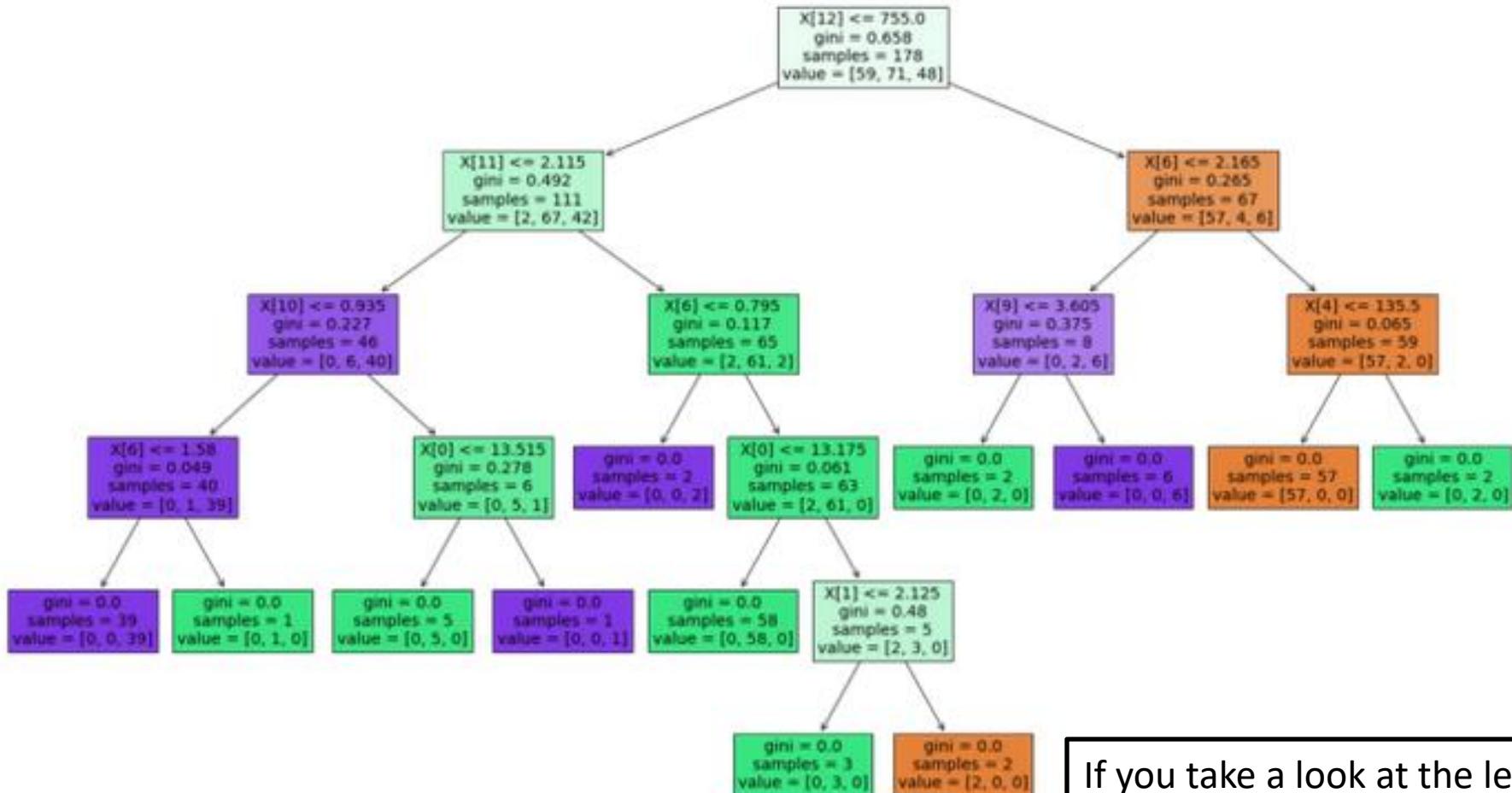
- **Attribute Selective Measures:**
 - Information Gain or ID3
 - Entropy
 - Gini Impurity or Gini Index

Decision Trees (Terminologies)

- **Gini** is a measure of impurity. “Gini impurity is a measure of how often a randomly chosen element from the set would be incorrectly labeled if it was randomly labeled according to the distribution of labels in the subset”.
 - It basically means that impurity increases with randomness. For instance, let’s say we have a box with ten balls in it. If all the balls are same color, we have no randomness and impurity is zero. However, if we have 5 blue balls and 5 red balls, impurity is 1.
- The other function to evaluate the quality of a split is **entropy** which is a measure of uncertainty or randomness. The more randomness a variable has, the higher the entropy is.



Decision Trees (Terminologies)



```

DecisionTreeClassifier(ccp_alpha=0.0, 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='deprecated',
                      random_state=None, splitter='best')
  
```

If you take a look at the leaf nodes (the nodes at the end of tree), you will see that gini is equal to zero.



Decision Trees (Terminologies)

Calculation 1: Find the Entropy of the total dataset

$$\text{Entropy}(S) = \frac{-p}{p+n} \log_2\left(\frac{p}{p+n}\right) - \frac{n}{p+n} \log_2\left(\frac{n}{p+n}\right)$$

p = no of positive cases(Loan_Status accepted)

n = number of negative cases(Loan_Status not accepted)

Calculation 2: Find the Entropy and Average Information for every column

Average Information in every column is

$$I(\text{Attribute}) = \sum \frac{p_i + n_i}{p+n} \text{Entropy}(A)$$

Calculation 3: Find the Gain for every column

$$\text{Gain} = \text{Entropy}(S) - I(\text{Attribute})$$



Decision Trees (Terminologies)

Information Gain(ID3)

Entropy is the main concept of this algorithm, which helps determine a feature or attribute that gives maximum information about a class is called Information gain or ID3 algorithm. By using this method, we can reduce the level of entropy from the root node to the leaf node.

Mathematical Formula :

$$E(S) = \sum_{i=1}^c - p_i \log_2 p_i$$

'p', denotes the probability of $E(S)$, which denotes the entropy. The feature or attribute with the highest ID3 gain is used as the root for the splitting.



Decision Trees (Terminologies)

Gini index

The measure of the degree of probability of a particular variable being wrongly classified when it is randomly chosen is called the Gini index or Gini impurity. The data is equally distributed based on the Gini index.

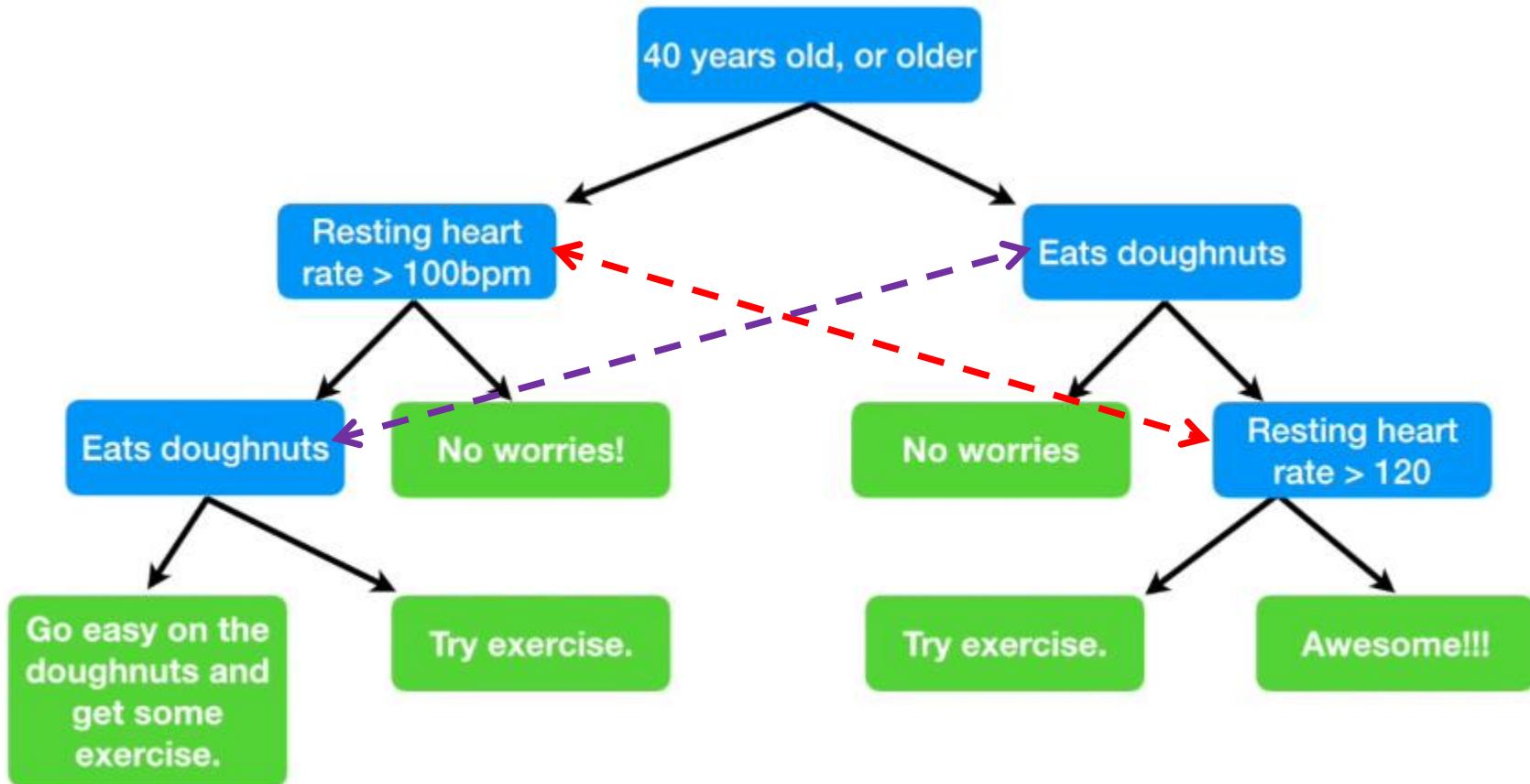
Mathematical Formula :

$$\text{Gini} = 1 - \sum_{i=1}^n (p_i)^2$$

P_i = probability of an object being classified into a particular class.

When you use the Gini index as the criterion for the algorithm to select the feature for the root node.,The feature with the least Gini index is selected.

Decision Trees



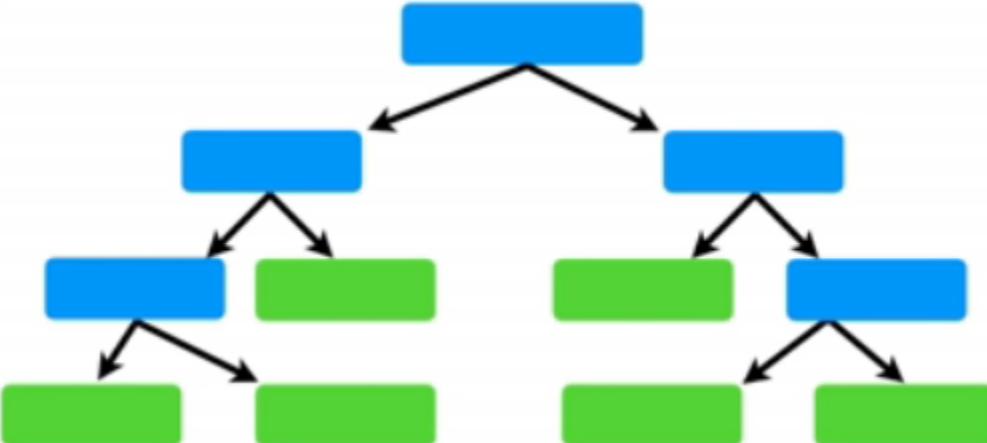
Questions:

- How to identify the best parameter to split?
- When to stop?

- Root Node or Root
- Internal Node or Node
- Leaf Node or Leaf

Decision Trees (for “Yes / No” Data)

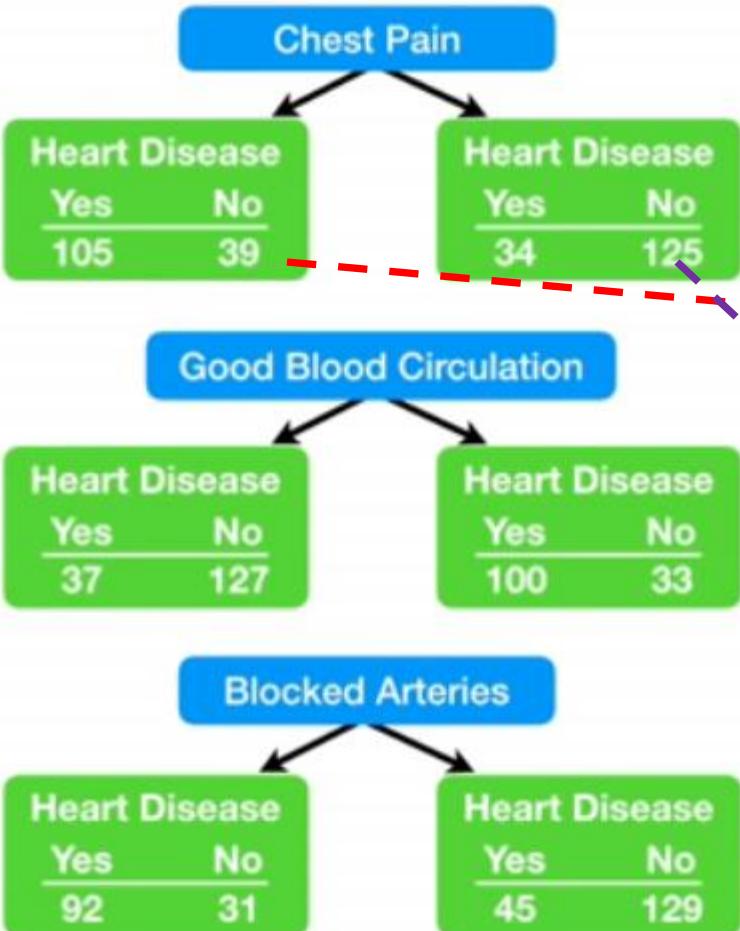
Chest Pain	Good Blood Circulation	Blocked Arteries	Heart Disease
No	No	No	No
Yes	Yes	Yes	Yes
Yes	Yes	No	No
Yes	No	No	Yes
etc...	etc...	etc...	etc...



Chest Pain	Good Blood Circulation	Blocked Arteries	Heart Disease
No	No	No	No
Yes	Yes	Yes	Yes
Yes	Yes	No	No
Yes	No	No	Yes
etc...	etc...	etc...	etc...

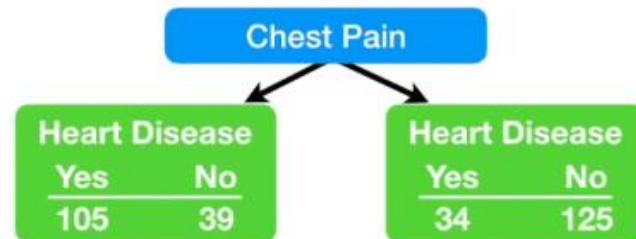


Decision Trees



Gini impurity = $1 - (\text{the probability of "yes"})^2 - (\text{the probability of "no"})^2$
 $= 1 - \left(\frac{105}{105 + 39}\right)^2 - \left(\frac{39}{105 + 39}\right)^2 = 0.395$

Gini impurity = $1 - (\text{the probability of "yes"})^2 - (\text{the probability of "no"})^2$
 $= 1 - \left(\frac{34}{34 + 125}\right)^2 - \left(\frac{125}{34 + 125}\right)^2 = 0.336$



Gini impurity = 0.395

0.336

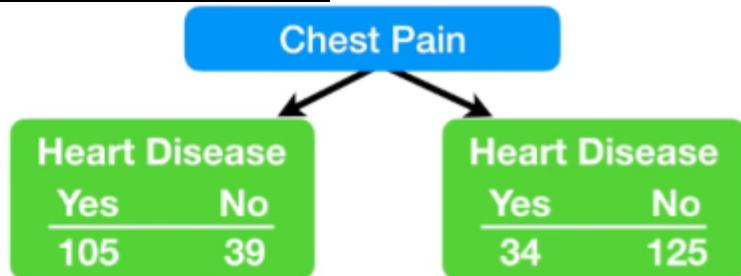
Gini impurity for Chest Pain = weighted average of Gini impurities for the leaf nodes

$$= \left(\frac{144}{144 + 159}\right) 0.395 + \left(\frac{159}{144 + 159}\right) 0.336 = 0.364$$

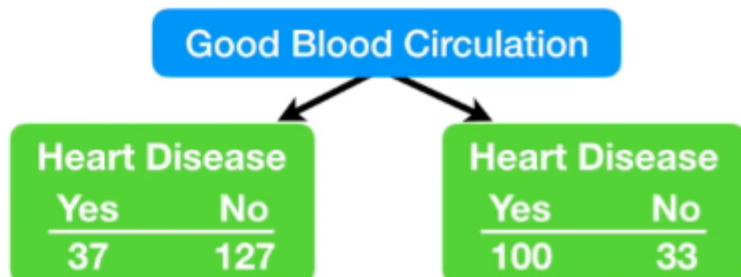
Impure

Decision Trees

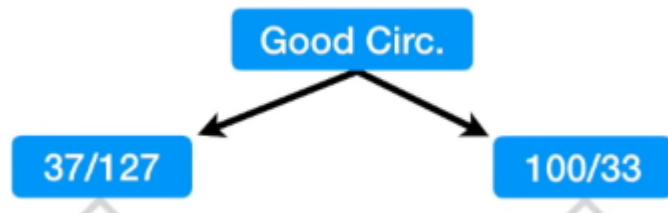
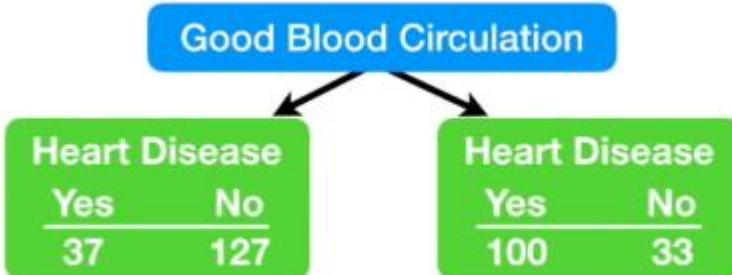
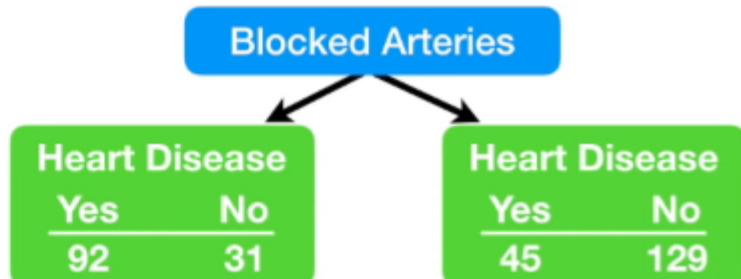
Gini impurity for Chest Pain = 0.364



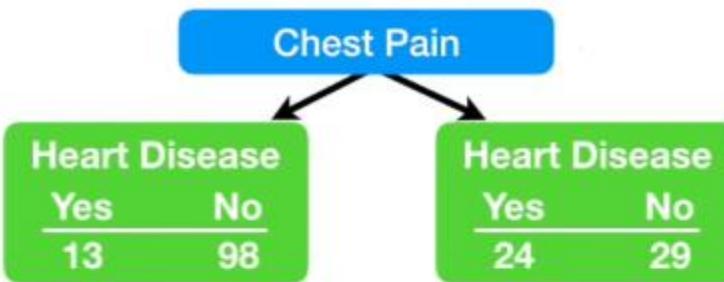
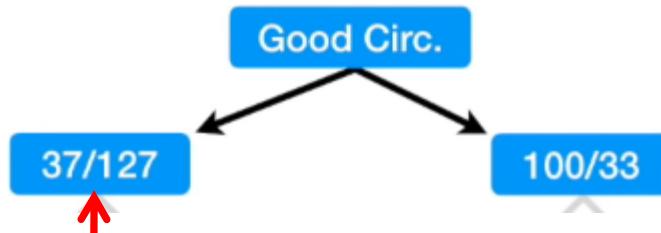
Gini impurity for Good Blood Circulation = 0.360



Gini impurity for Blocked Arteries = 0.381

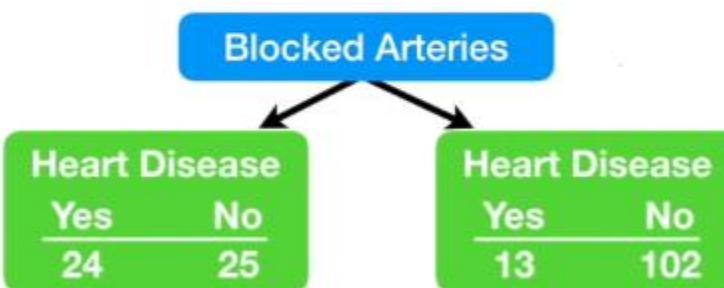


Decision Trees

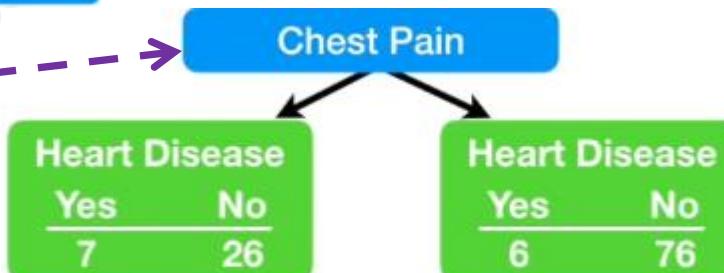
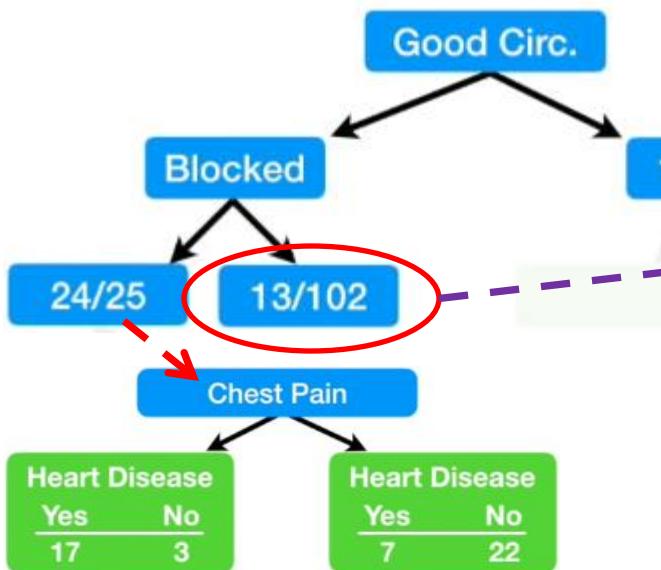


Now we need to figure how well **chest pain** and **blocked arteries** separate these 164 patients (37 with heart disease and 127 without heart disease).

Gini impurity for Chest Pain = 0.3

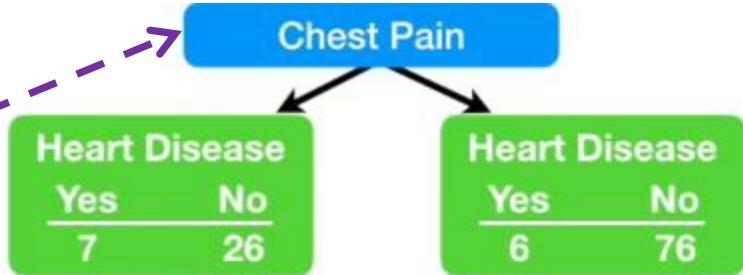
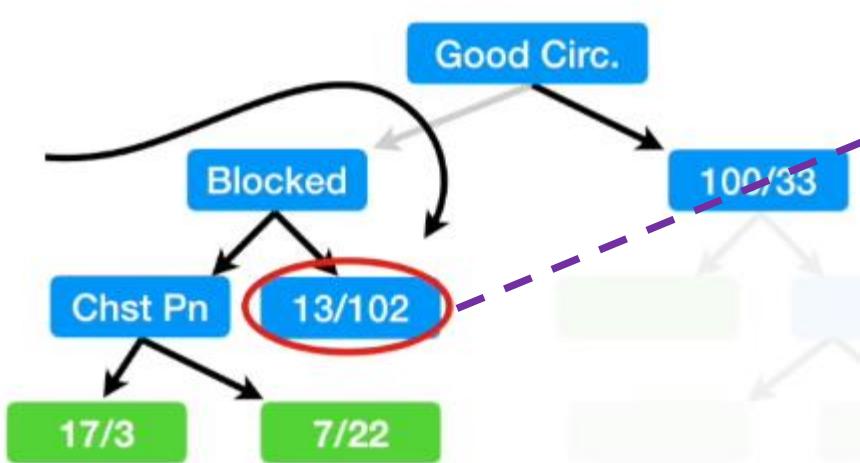


Gini impurity for Blocked Arteries = 0.290



Gini impurity for Chest Pain = 0.29

Decision Trees



Gini impurity for Chest Pain = 0.29

The Gini impurity for this node, before using chest pain to separate patients is...

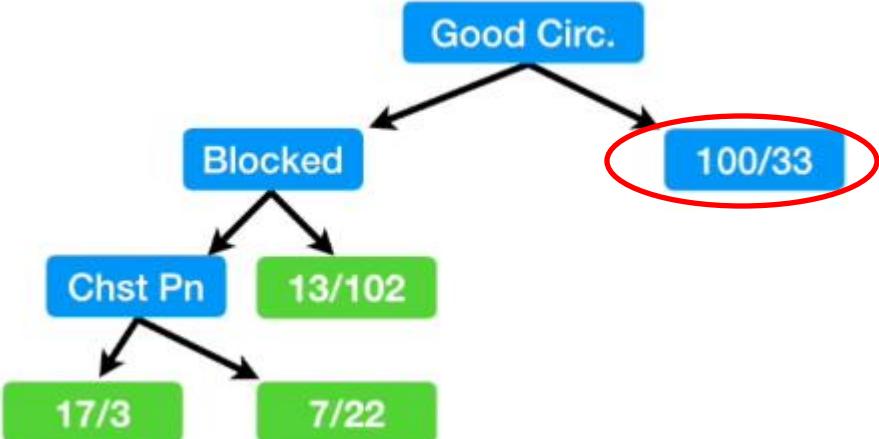
$$= 1 - (\text{the probability of "yes"})^2 - (\text{the probability of "no"})^2$$

$$= 1 - \left(\frac{13}{13+102} \right)^2 - \left(\frac{102}{13+102} \right)^2$$

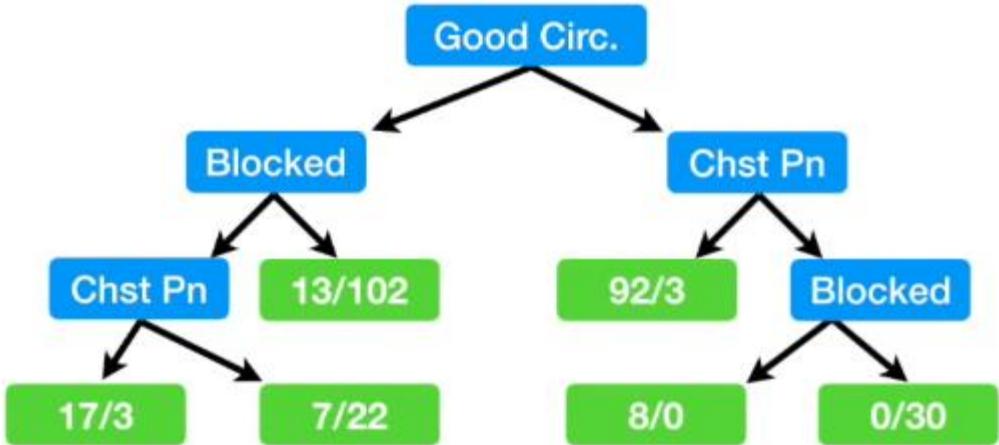
$$= 0.2$$

Steps:

- 1) Calculate all of the Gini impurity scores.
- 2) If the node itself has the lowest score, than there is no point in separating the patients any more and it becomes a leaf node.
- 3) If separating the data results in an improvement, than pick the separation with the lowest impurity value.



Decision Trees (for Numerical Data)



- Unlike other supervised learning algorithms, the decision tree algorithm can be used for solving regression and classification problems too

Splitted the dataset using “Yes / No” questions

How do we determine what's the best weight to use to divide the data?

Weight	Heart Disease
220	Yes
180	Yes
225	Yes
190	No
155	No

Step 1) Sort the patients by weight, lowest to highest.

Weight	Heart Disease
155	No
180	Yes
190	No
220	Yes
225	Yes

Step 2) Calculate the average weight for all adjacent patients.

Weight	Heart Disease
155	No
167.5	?
180	Yes
185	?
190	No
205	?
220	Yes
222.5	?
225	Yes

Step 3) Calculate the impurity values for each average weight.

Weight	Heart Disease
155	No
167.5	?
180	Yes
185	?
190	No
205	?
220	Yes
222.5	?
225	Yes

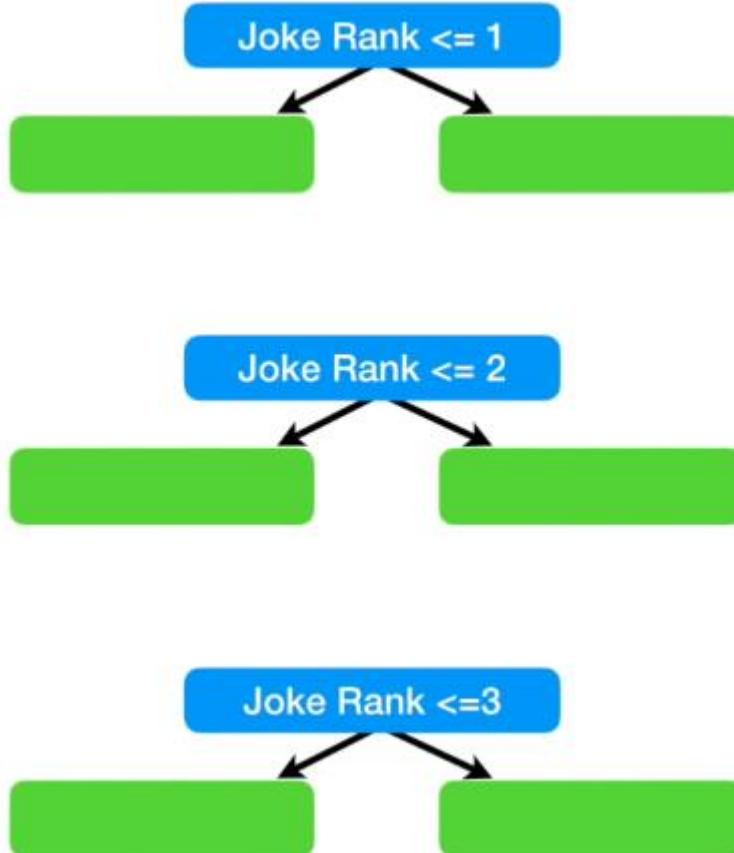
Weight	Heart Disease	Gini Impurity
155	No	
167.5	?	Gini impurity = ?
180	Yes	
185	?	Gini impurity = ?
190	No	
205	?	Gini impurity = ?
220	Yes	
222.5	?	Gini impurity = ?
225	Yes	

Decision Trees (for Rank Data)

Ranked data is similar to numeric data, except instead now we calculate impurity scores for all of the possible ranks.

Possible Rank Values = [1, 2, 3, 4]

Rank my jokes...	Likes StatQuest
1	Yes
1	No
3	Yes
1	Yes
etc...	etc...

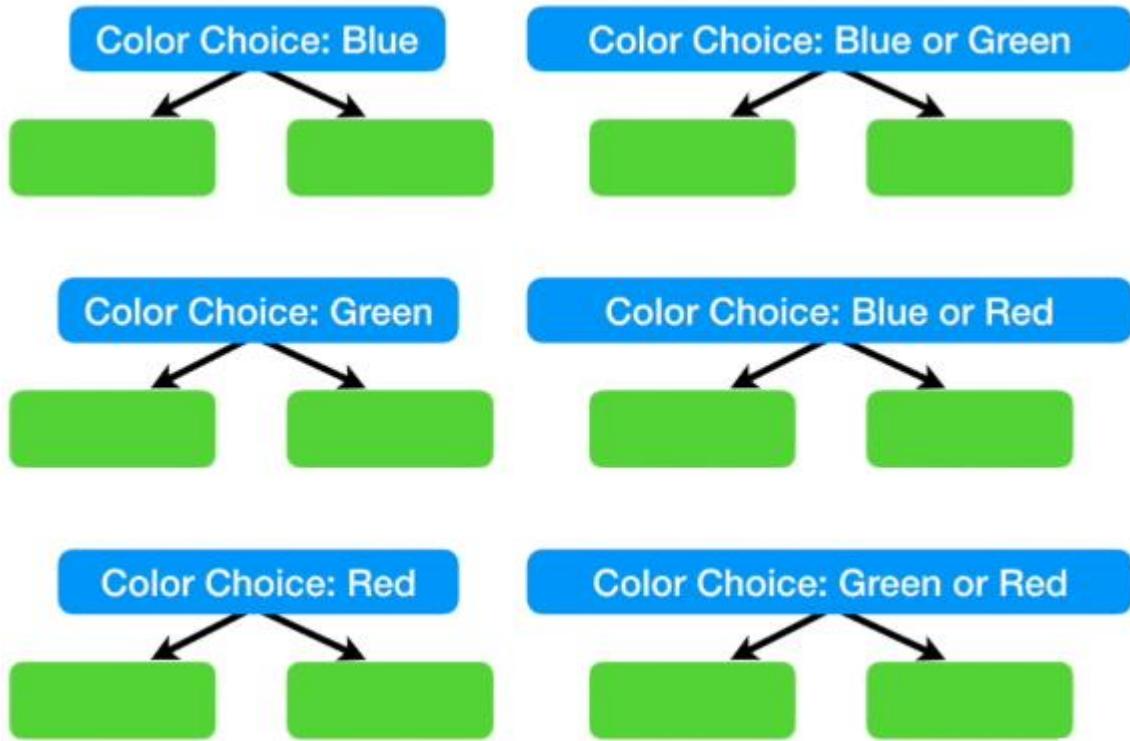


Note: Do not calculate for Rank ≤ 4 .
 Otherwise, all samples will fall in that category.

Decision Trees (for Multiple Choice Data)

Possible Choices = [Red, Green, Blue]

Color Choice	Likes StatQuest
Green	Yes
Blue	No
Red	Yes
Green	Yes
etc...	etc...



Note: Do not calculate "Red or Green or Blue". Otherwise, all samples will fall in that category.

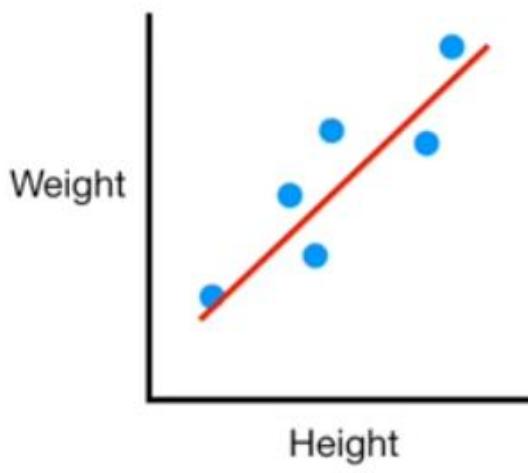
- Accuracy will be low
- Overfitting -> They work great with the data used to create them, but they are not flexible when it comes to classifying new samples.

- <https://www.kdnuggets.com/2020/01/decision-tree-algorithm-explained.html>
- <https://towardsai.net/p/programming/decision-trees-explained-with-a-practical-example-fe47872d3b53>
- <https://towardsdatascience.com/hyperparameters-of-decision-trees-explained-with-visualizations-1a6ef2f67edf>

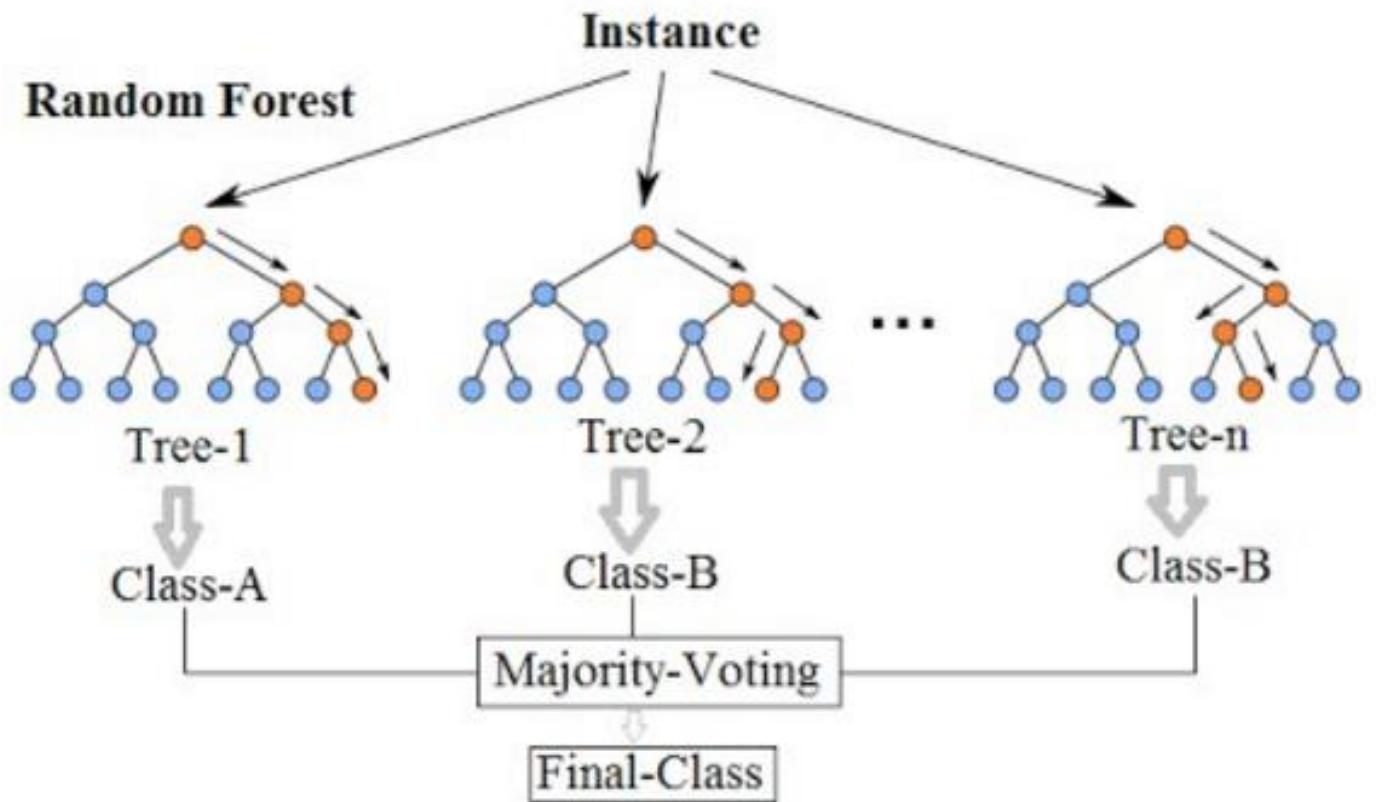
Handling Missing Data

Chest Pain	Good Blood Circulation	Blocked Arteries	Heart Disease			Blocked Arteries	Heart Disease
No	No	No	No			No	No
Yes	Yes	Yes	Yes			Yes	Yes
Yes	Yes	No	No			Yes	No
Yes	No	???	Yes			Yes	Yes
etc...	etc...	etc...	etc...			etc...	etc...

Height	Good Blood Circulation	Weight	Heart Disease			Height	Good Blood Circulation	Weight	Heart Disease
5'7"	No	155	No			5'7"	No	155	No
6'	Yes	180	Yes			6'	Yes	180	Yes
5'4"	Yes	120	No			5'4"	Yes	120	No
5'8"	No	???	Yes			5'8"	No	???	Yes
etc...	etc...	etc...	etc...			etc...	etc...	etc...	etc...



Random Forest



Applications:

- Classification
- Regression

Concerns:

- # of trees generated
- **Maximum # of features to be considered**
- Minimum # of samples in each class that are required to split a node

Random Forest

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

Step 1) Create a bootstrapped dataset -> Size is similar to original

Random Sampling with Replacement

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 the bootstrapped dataset, but use only 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
No	No	No	125	No
Yes	No	Yes	167	Yes
Yes	No	Yes	167	Yes

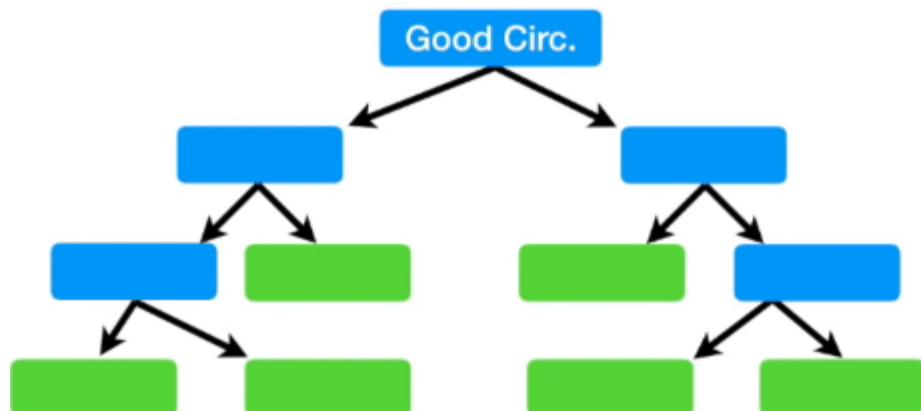
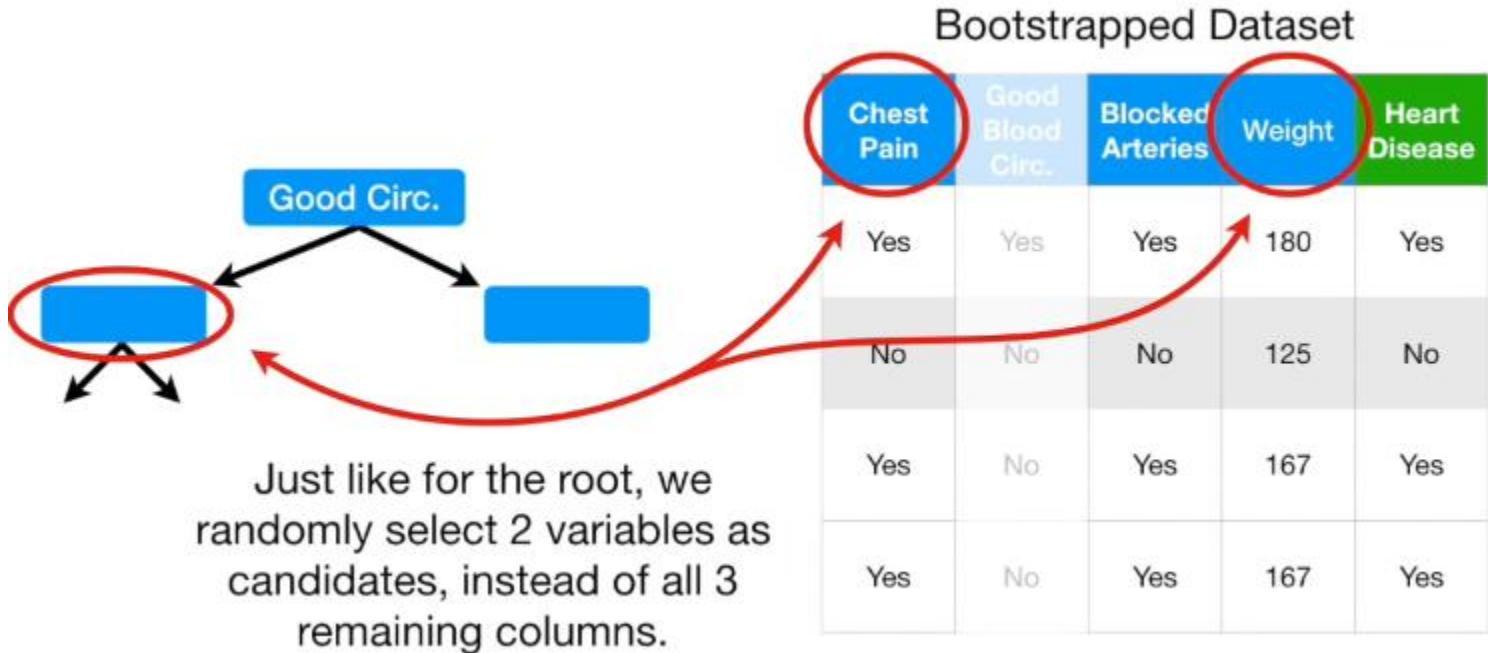
...we randomly select 2.



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

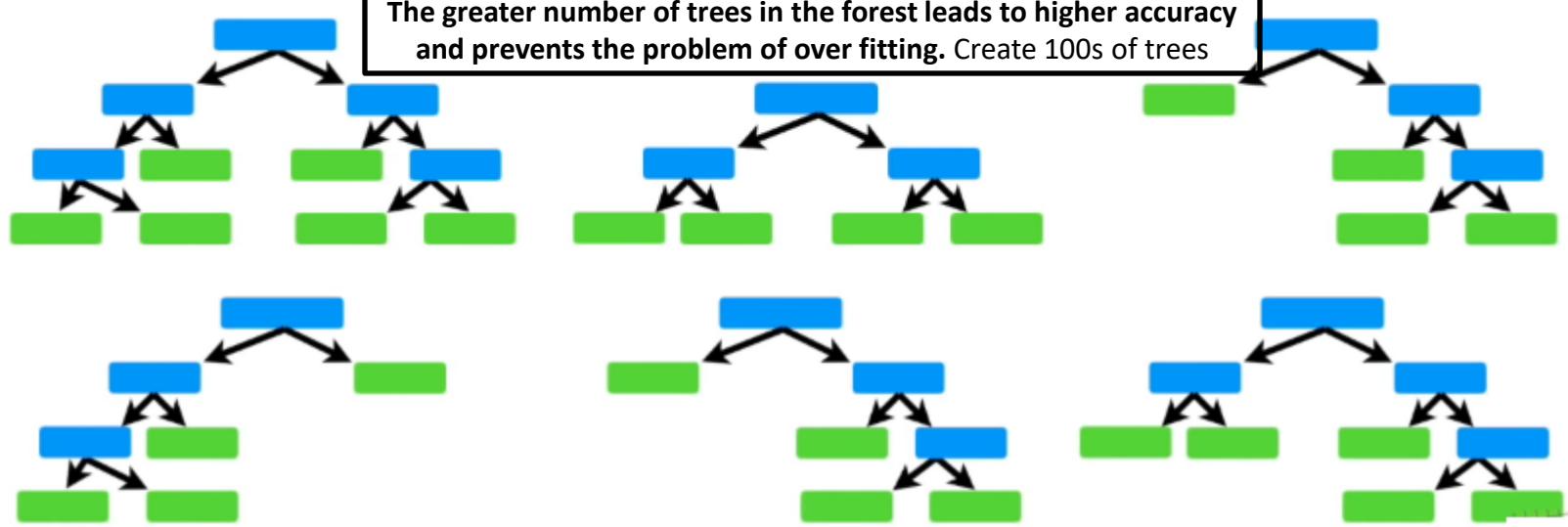
Random Forest



We built a tree...

- 1) Using a bootstrapped dataset
- 2) Only considering a random a subset of variables at each step.

Random Forest



Using a bootstrapped sample and considering only a subset of the variables at each step results in a wide variety of trees.

The variety is what makes random forests more effective than individual decision trees.

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

5 trees say Yes and 1 tree say No.

Heart Disease	
Yes	No
5	1

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

Boot Strapping + Aggregating (to make decision) = Bagging

Random Forest

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

Unused Sample



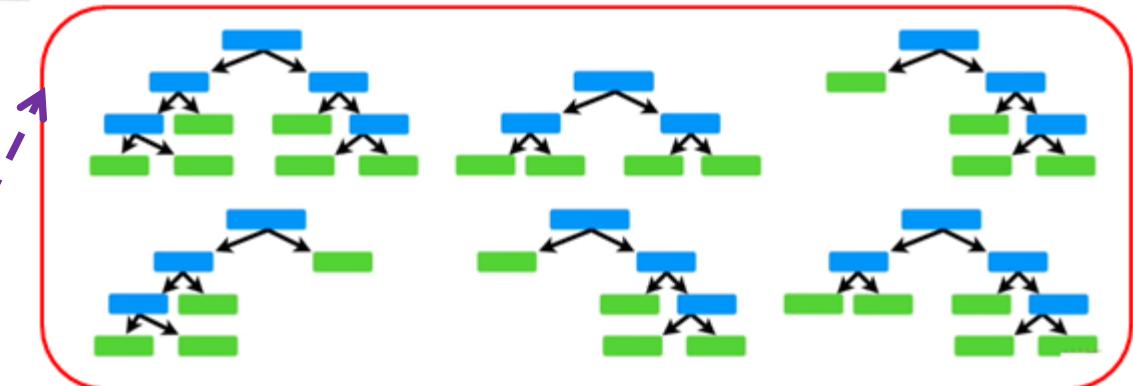
Bootstrapped Dataset

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

Typically, 1/3 of the original data does not end up in bootstrapped dataset

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

Random Forest

Out-of-Bag Sample 1

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

Correctly Labelled

Classification of the Out-Of-Bag sample

Yes

1

No

3

Out-of-Bag Sample 2

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
Yes	Yes	Yes	180	Yes

Correctly Labelled

Classification of the Out-Of-Bag sample

Yes

4

No

0

Out-of-Bag Sample 3

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

Incorrectly Labelled

Classification of the Out-Of-Bag sample

Yes

3

No

1

OK, we now know how to:

- 1) Build a Random Forest
- 2) Use a Random Forest
- 3) Estimate the accuracy of a Random Forest.

The proportion of Out-Of-Bag samples that were *incorrectly* classified is the “Out-Of-Bag Error”

Random Forest

How to find the maximum # of features to be considered?

Now we can compare the Out-Of-Bag error for a random forest built using only 2 variables per step...



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

<https://www.javatpoint.com/machine-learning-random-forest-algorithm>

...and we test a bunch of different settings and choose the most accurate random forest.

...to random forest built using 3 variables per step...



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

Compare and choose which is best

Although random forest can be used for both classification and regression tasks, it is not more suitable for Regression tasks

1) Build a Random Forest

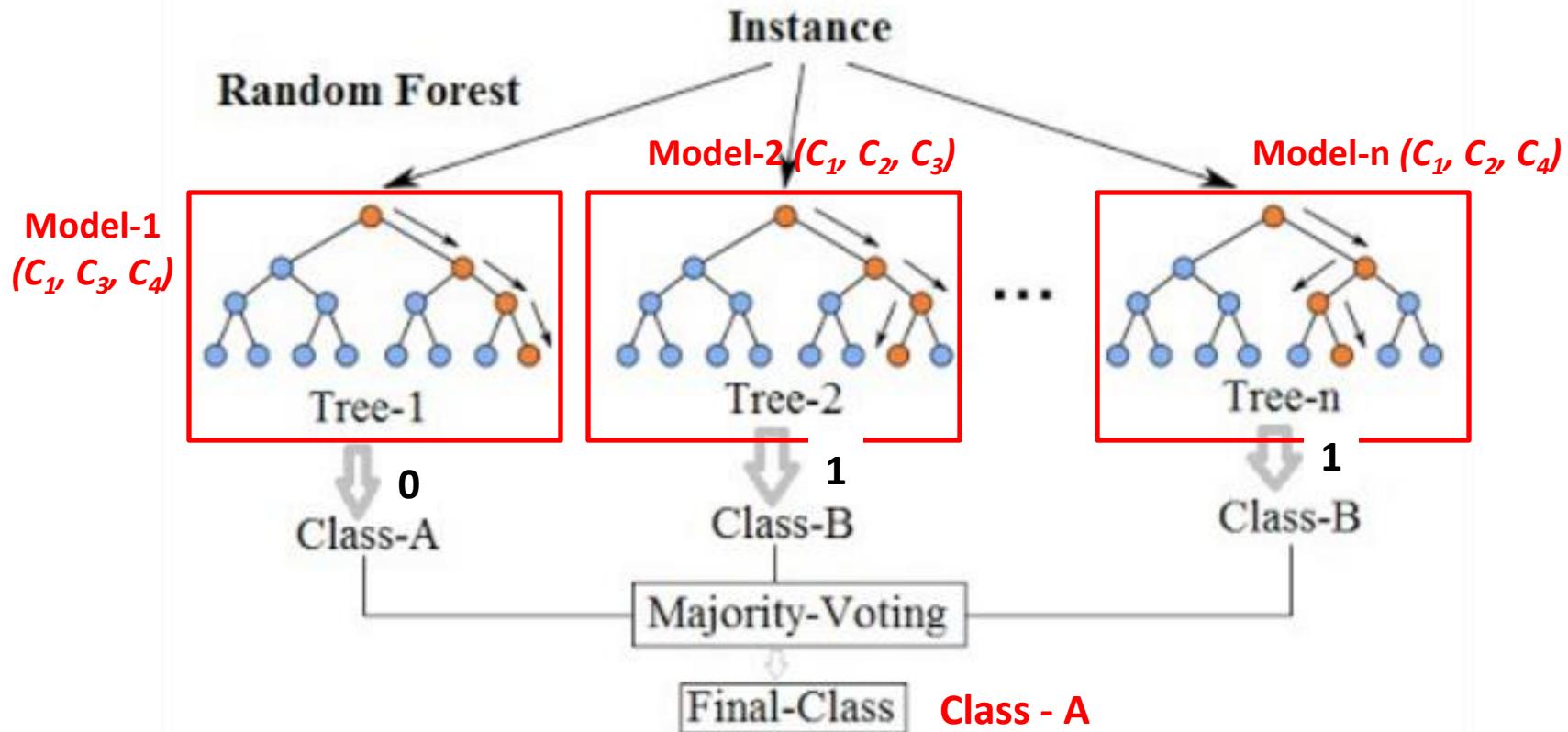
2) Estimate the accuracy of a Random Forest.

...change the number of variables used per step...

Typically, we start by using the square of the number of variables and then try a few settings above and below that value.

Random Forest

Row + Feature Sampling with Replacement



- <https://towardsdatascience.com/understanding-random-forest-58381e0602d2>
- <https://www.javatpoint.com/machine-learning-random-forest-algorithm>



Techniques to Consolidate

- Hard Voting or Majority Voting or Maximum voting

Colleague 1	Colleague 2	Colleague 3	Colleague 4	Colleague 5	Final rating
5	4	5	4	4	4

- Averaging

Colleague 1	Colleague 2	Colleague 3	Colleague 4	Colleague 5	Final rating
5	4	5	4	4	4.4



Techniques to Consolidate

- Weighted Average

	Colleague 1	Colleague 2	Colleague 3	Colleague 4	Colleague 5	Final rating
weight	0.23	0.23	0.18	0.18	0.18	
rating	5	4	5	4	4	4.41



Building an Ensemble Model

- Three key steps
 - Select Data, Train Classifiers, Combine Classifiers
- Data Selection
 - Maximize diversity of the models
 - Better the performance of your final classifier
 - Smaller the variance of its predictions
- Train
 - Bagging algorithm uses different subsets of the data to train
 - ✓ Eg: Random Forest Algorithm
 - Boosting algorithm focuses more on the misclassified examples during training
 - ✓ Each additional model focuses on the misclassified data
 - ✓ Eg: Boosted Decision Tree



Building an Ensemble Model

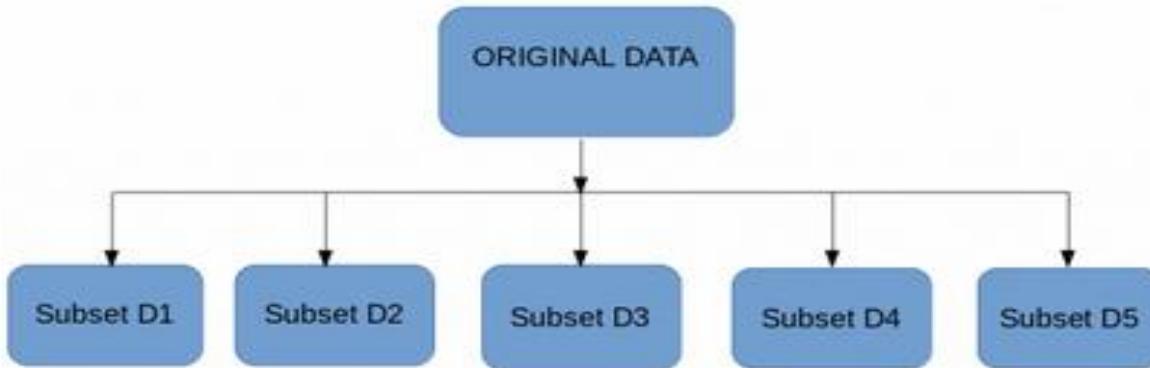
- Combining Classifiers
 - Combine the results to make a final prediction
 - Simple majority to Weighted majority voting
- Ensemble models are really exciting
- Offers potential for breakthroughs in classification problems



Bagging

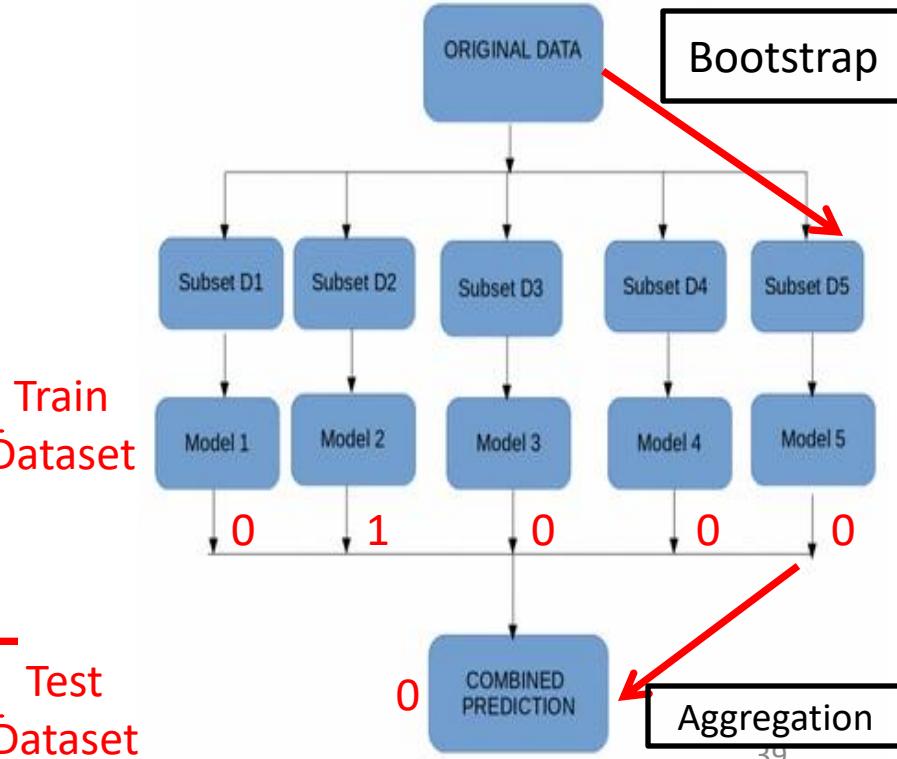
- Bootstrap Aggregating
- Ensemble learning method that is commonly used to reduce variance within a noisy dataset
- Combine the results of multiple models to get a generalized result
- If you create all the models on the same set of data and combine it, will it be useful?
- High chance that these models will give the same result since they are getting the same input
- Techniques
 - Bagging Meta-Estimator
 - Random Forest

Bagging Meta-Estimator



Row Sampling with Replacement

Shape	Diag	# of Diag	Length (in cm)	Height (in cm)	Class
Rectangle	Yes	2	100	10	1
Rectangle	Yes	2	1000	100	1
Square	Yes	2	100	100	0
Triangle	No	0	100	1000	0
Circle	No	0	100	100	0
Diamond	Yes	2	100	100	0
Rectangle	Yes	2	100	5	1





Bagging

- As they provide a way to reduce overfitting
 - Bagging methods work best with strong and complex models (e.g., fully developed decision trees)
 - Boosting methods which usually work best with weak models (e.g., shallow decision trees)
- Bagging methods come in many flavours but mostly differ from each other by the way they draw random subsets of the training set:
 - When random subsets of the dataset are drawn as random subsets of the samples, then this algorithm is known as Pasting
 - When samples are drawn with replacement, then the method is known as Bagging
 - When random subsets of the dataset are drawn as random subsets of the features, then the method is known as Random Subspaces
 - Finally, when base estimators are built on subsets of both samples and features, then the method is known as Random Patches



Bagging

```
>>> from sklearn.ensemble import BaggingClassifier  
>>> from sklearn.neighbors import KNeighborsClassifier  
>>> bagging = BaggingClassifier(KNeighborsClassifier(),  
...                                max_samples=0.5, max_features=0.5)
```

- In scikit-learn, bagging methods are offered as a unified `BaggingClassifier` meta-estimator, taking as input a user-specified base estimator along with parameters specifying the strategy to draw random subsets
- In particular, `max_samples` and `max_features` control the size of the subsets (in terms of samples and features), while `bootstrap` and `bootstrap_features` control whether samples and features are drawn with or without replacement
- When using a subset of the available samples the generalization accuracy can be estimated with the out-of-bag samples by setting `oob_score=True`
- As an example, the snippet above illustrates how to instantiate a bagging ensemble of `KNeighborsClassifier` base estimators, each built on random subsets of 50% of the samples and 50% of the features

<https://scikit-learn.org/stable/modules/ensemble.html>



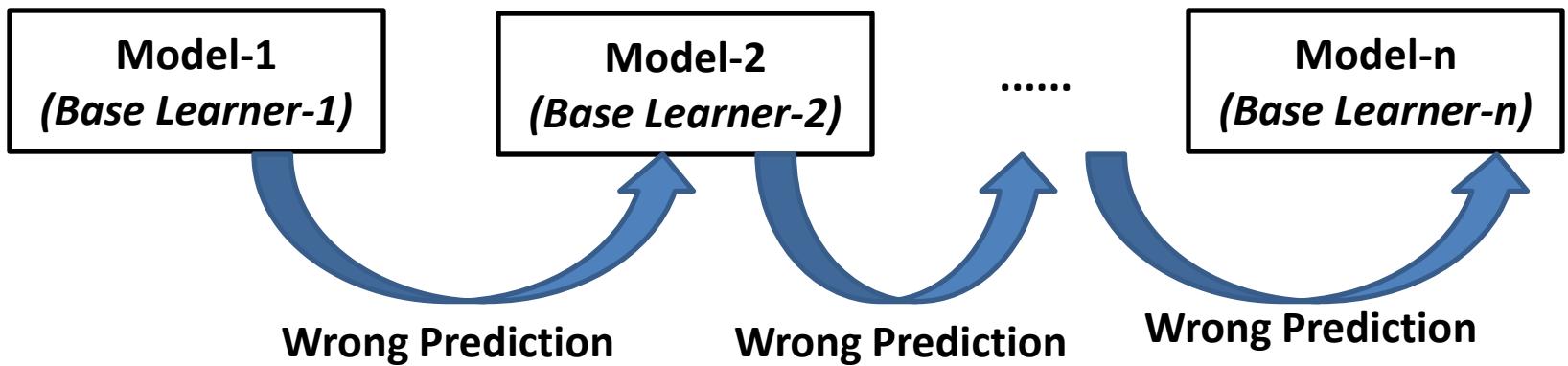
Boosting

- Combines a number of weak learners to form a strong learner
- Individual models will not perform well on the entire dataset
- Work well for some part of the dataset
- Each model actually boosts the performance of the ensemble
- Techniques
 - AdaBoost
 - GBM
 - XGBM
 - Light GBM
 - CatBoost



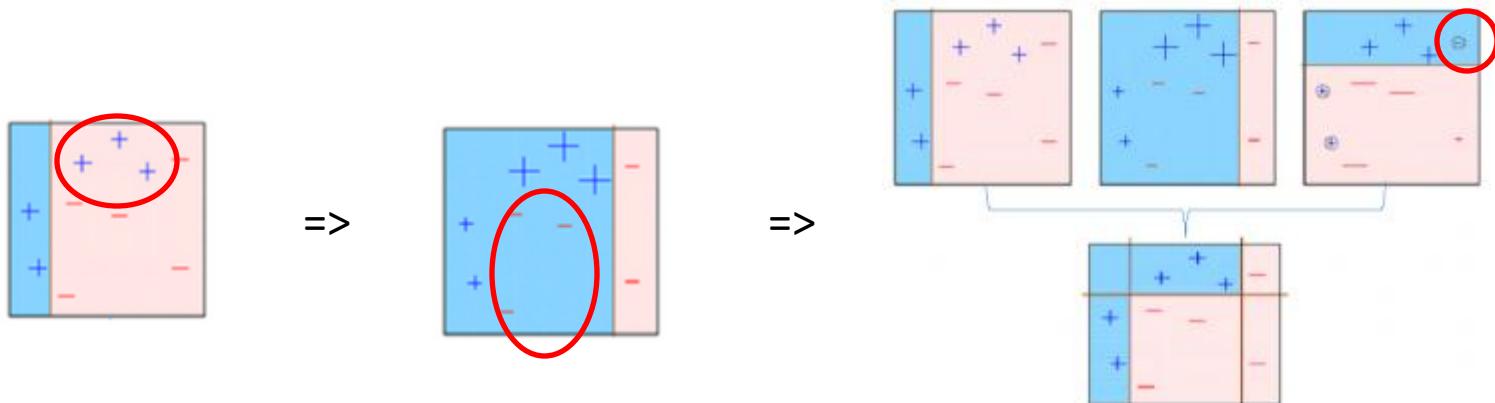
Boosting

Shape	Diag	# of Diag	Length (in cm)	Height (in cm)	Class
Rectangle	Yes	2	100	10	1
Rectangle	Yes	2	1000	100	1
Square	Yes	2	100	100	0
Triangle	No	0	100	1000	0
Circle	No	0	100	100	0
Diamond	Yes	2	100	100	0
Rectangle	Yes	2	100	5	1



Boosting

- If a data point is incorrectly predicted by the first model, probably all models, will combining the predictions provide better results?



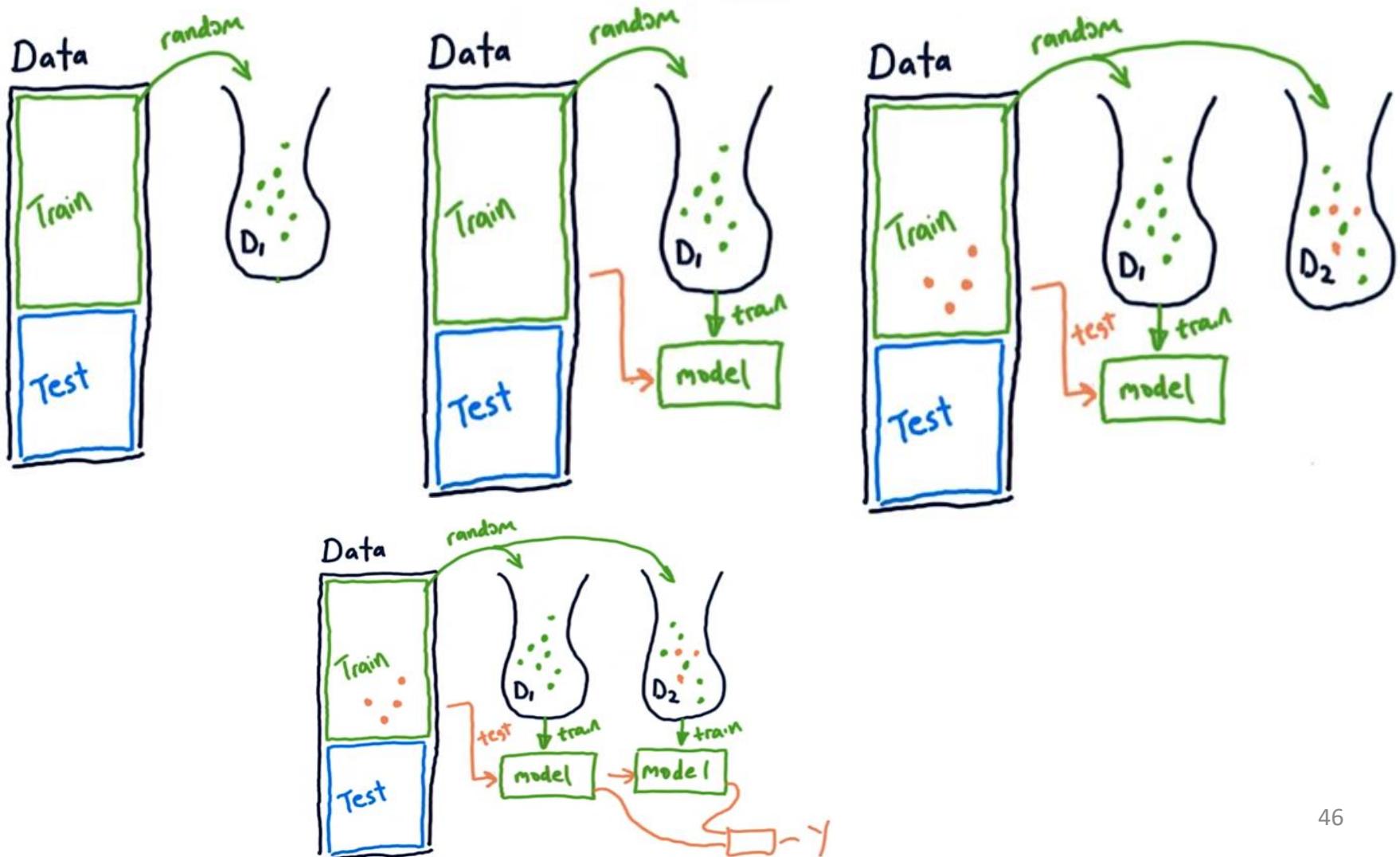


Boosting vs Bagging

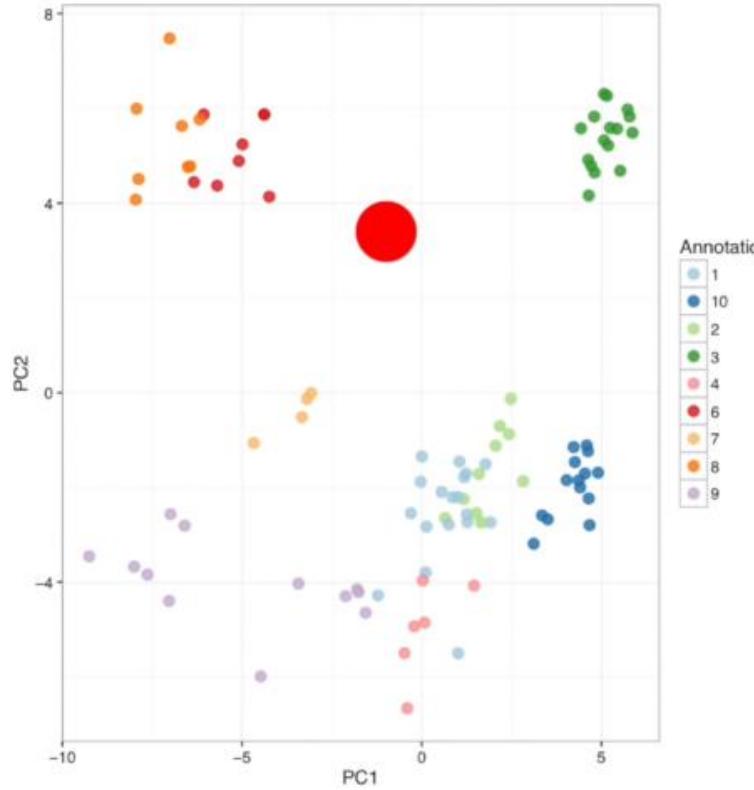
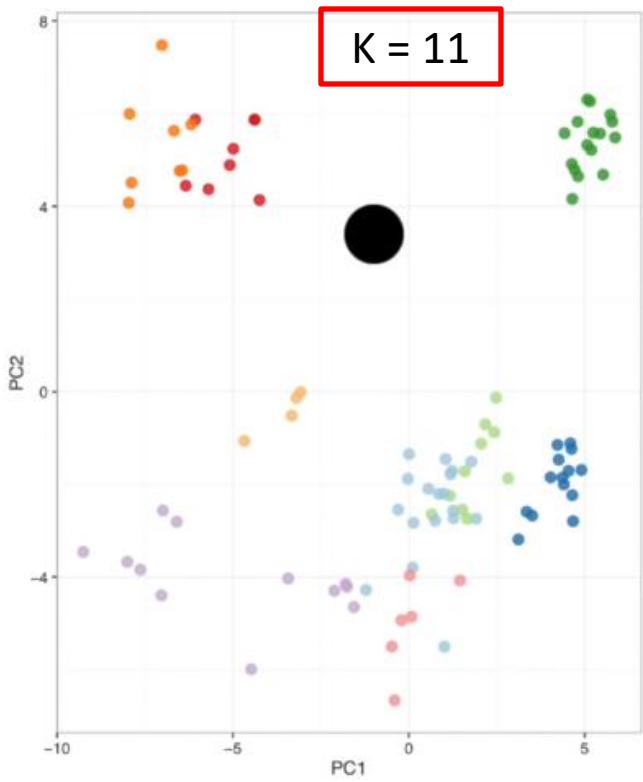
Sl. No.	Boosting	Bagging
1.	Boosting methods are leveraged when low variance and high bias is observed	Bagging methods are typically used on weak learners which exhibit high variance and low bias
2.	<ul style="list-style-type: none">• Learn sequentially• Means that a series of models are constructed and with each new model iteration, the weights of the misclassified data in the previous model are increased• This redistribution of weights helps the algorithm identify the parameters that it needs to focus on to improve its performance	Weak learners are trained in parallel

AdaBoost

Adaptive Boosting



K-Nearest Neighbour



7 nearest neighbors are **RED**.
 3 nearest neighbors are **ORANGE**.
 1 nearest neighbor is **GREEN**.

Since **RED** got the most votes, the final assignment is **RED**.

- Low values for K (like K = 1, 2) can be noisy and subject to the effect of outliers
- Large values for K smooth over things. But, don't keep too large, otherwise a category with only a few samples in it will always be outvoted by other categories.



Implementing K-Nearest Neighbour?

```
from sklearn.neighbors import KNeighborsClassifier
```

```
knn = KNeighborsClassifier(n_neighbors=1)
```

```
knn.fit(X_train,y_train)
```

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
metric_params=None, n_jobs=1, n_neighbors=1, p=2,
weights='uniform')
```

```
pred = knn.predict(X_test)
```

```
from sklearn.metrics import classification_report,confusion_matrix
```

```
print(confusion_matrix(y_test,pred))
```

```
[[125  18]
 [ 13 144]]
```

```
print(classification_report(y_test,pred))
```

	precision	recall	f1-score	support
0	0.91	0.87	0.89	143
1	0.89	0.92	0.90	157
avg / total	0.90	0.90	0.90	300

Predicted Class	Actual Class	
	Cat	Non-cat
	5 TP	2 FP
Non-cat	3 FN	3 TN

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{TP}{TP + FN}$$

$$F1 Score = 2 * \frac{\text{Recall} * \text{Precision}}{\text{Recall} + \text{Precision}}$$

Minkowski distance or Minkowski metric is a metric in a normed vector space which can be considered as a generalization of both the Euclidean distance and the Manhattan distance



Implementing K-Nearest Neighbour?

Error Rate = Mean(Predicted_Class != Y_Actual_Class)

```
error_rate = []

# Will take some time
for i in range(1,40):

    knn = KNeighborsClassifier(n_neighbors=i)
    knn.fit(X_train,y_train)
    pred_i = knn.predict(X_test)
    error_rate.append(np.mean(pred_i != y_test))

plt.figure(figsize=(10,6))
plt.plot(range(1,40),error_rate,color='blue', linestyle='dashed', marker='o',
         markerfacecolor='red', markersize=10)
plt.title('Error Rate vs. K Value')
plt.xlabel('K')
plt.ylabel('Error Rate')
```

WITH K=23

```
[[132 11]
 [ 5 152]]
```

	precision	recall	f1-score	support
0	0.96	0.92	0.94	143
1	0.93	0.97	0.95	157
avg / total	0.95	0.95	0.95	300





Support Vector Machine

- Can be used for both Classification and Regression problems
 - SVM -> Output is mainly logical / discrete (Classification)
 - Linear separable SVM
 - Non-Linear Separable SVM
 - SVR -> Output is a real value (Regression/Prediction)
 - SVC
- Terminologies
 - Hyperplane
 - Marginal Distance
 - Support Vector
 - Kernel
 - Polynomial Kernel
 - RDF Kernel
 - Sigmoid Kernel

- <https://www.javatpoint.com/machine-learning-support-vector-machine-algorithm>
- https://en.wikipedia.org/wiki/Support-vector_machine
- <https://monkeylearn.com/blog/introduction-to-support-vector-machines-svm/>

Support Vector Machine



Both classification and Regression problems

Not Obese

Obese

Margin → Distance between the observation and threshold



Not Obese

Obese

*Maximal Margin
Classifier use
Maximal Margin*



Not Obese

Obese

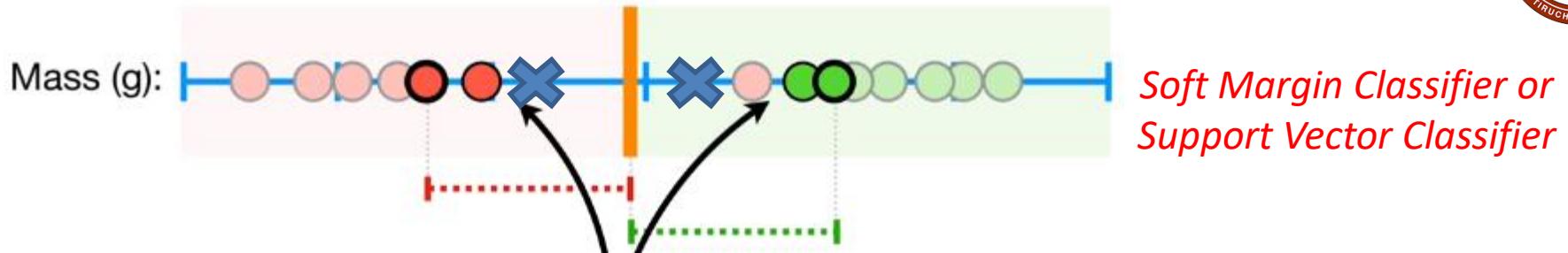
*Maximal Margin
Classifier →
Sensitive to outliers*



Bias / Variance Trade-off

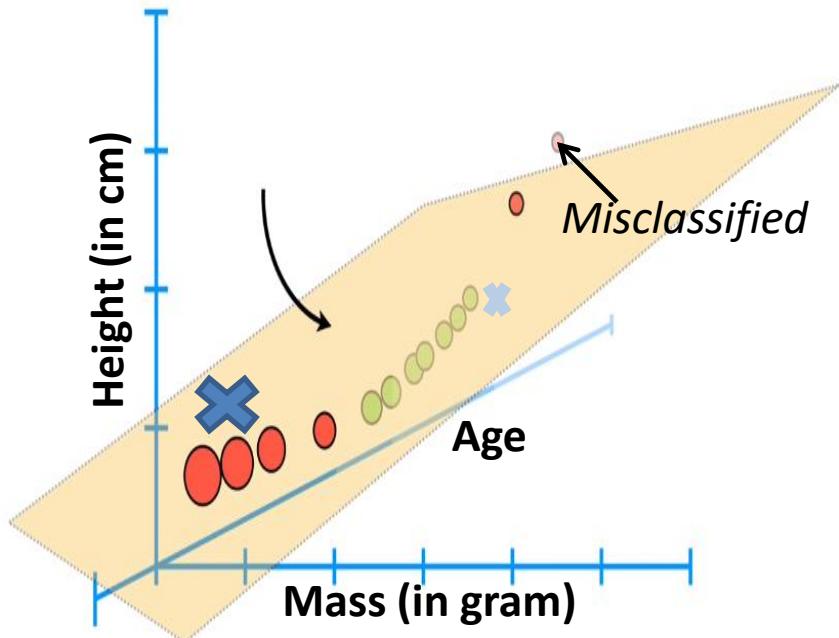
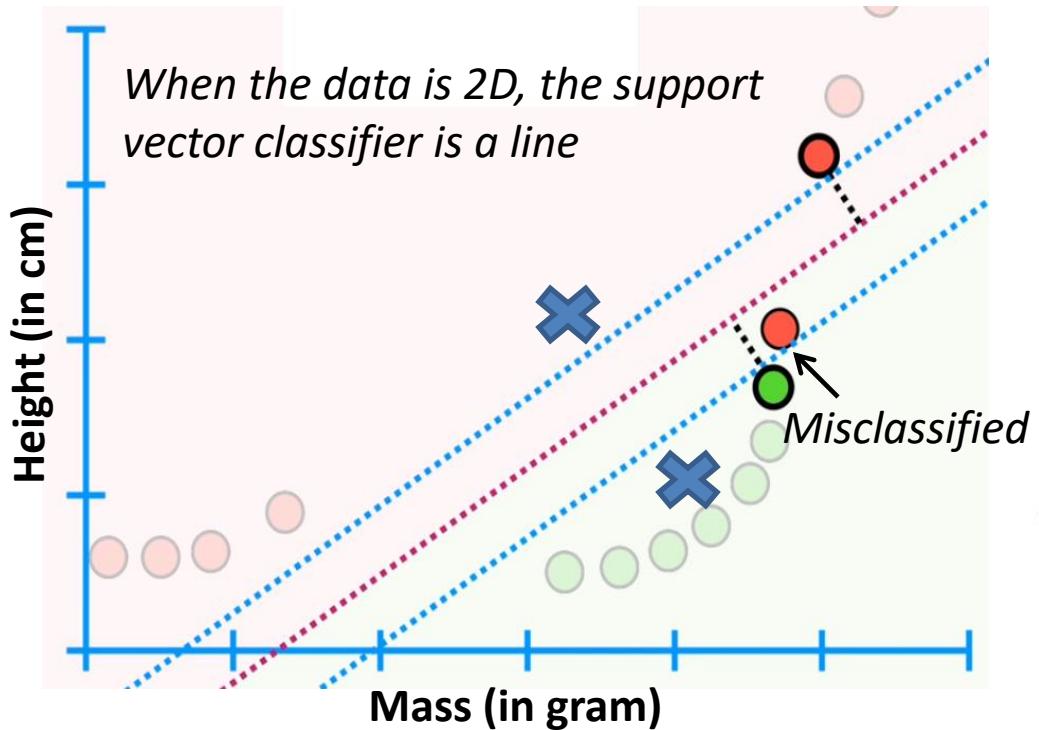
*Soft Margin ->
Allow some
miscalculations*

Support Vector Machine

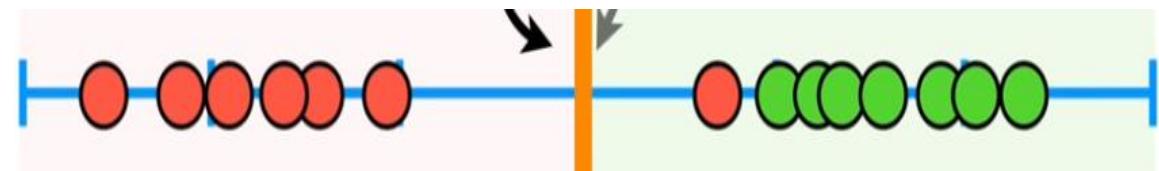


Suppose if we use cross-validation to find the soft margin, then we allow 1 miscalculation and 2 correct observations to be correctly classified within the soft margin

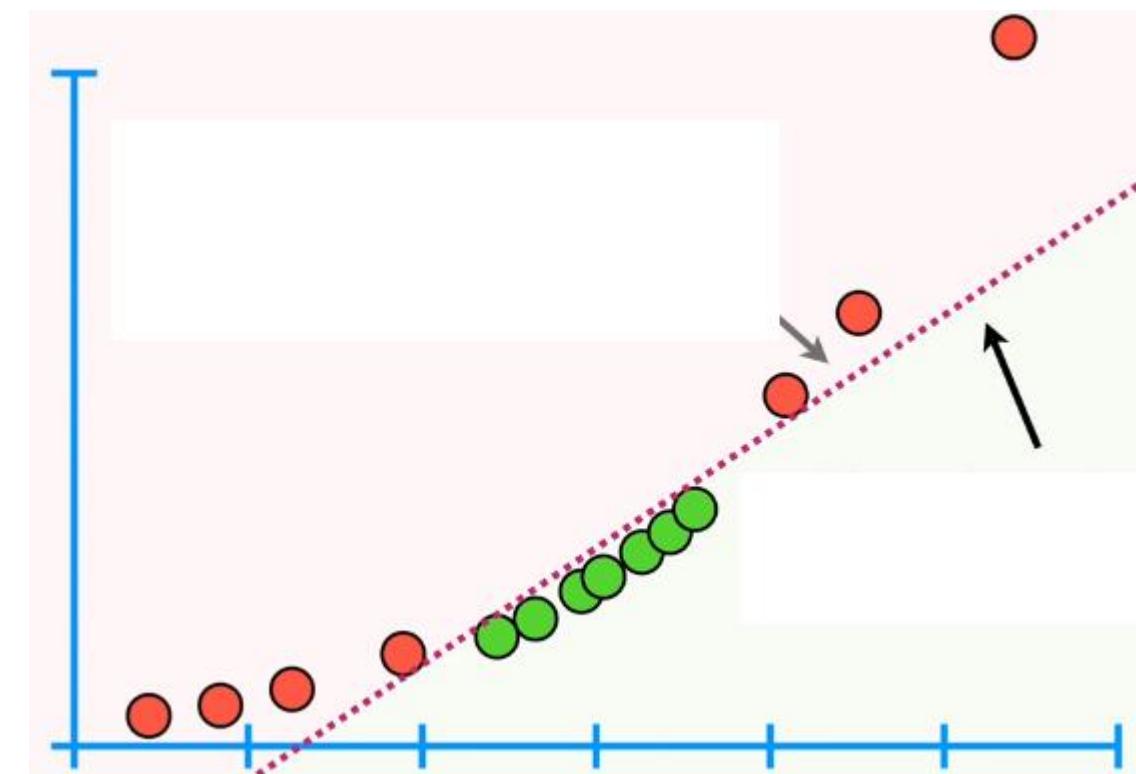
Observations on edge and within the soft margin are called Support Vectors



Support Vector Machine

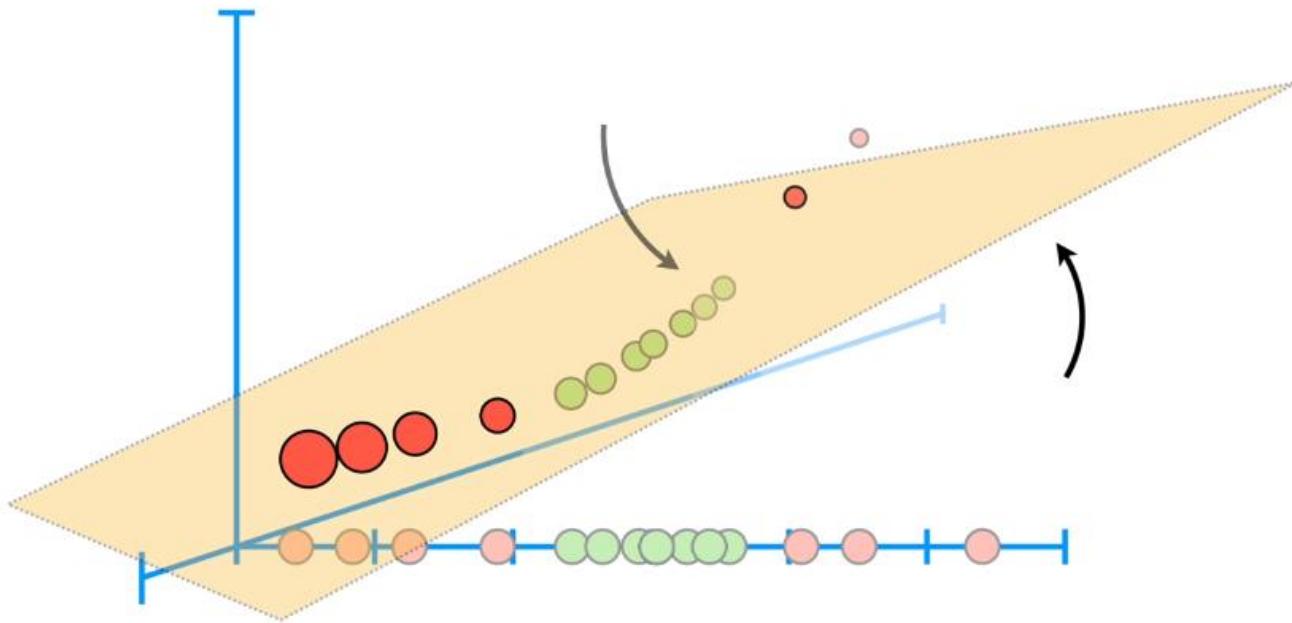


- *If the data is 1D, then the Support Vector Classifier is a single point on a 1D number line*

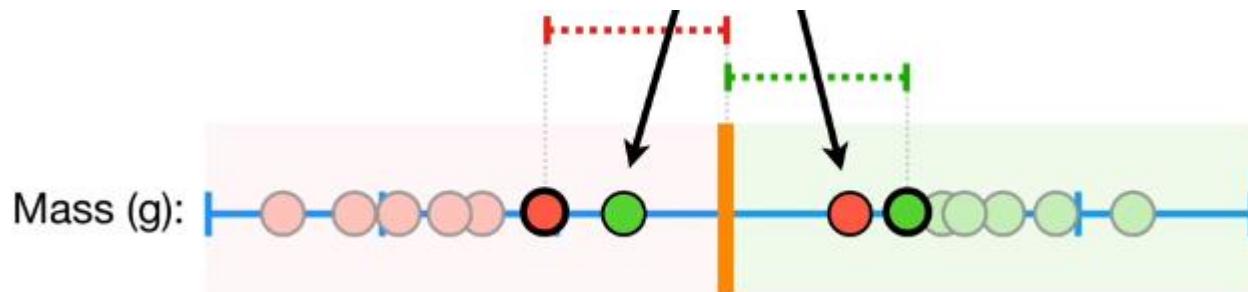


- *If the data is 2D, then the Support Vector Classifier is a line on a 2D space*

Support Vector Machine

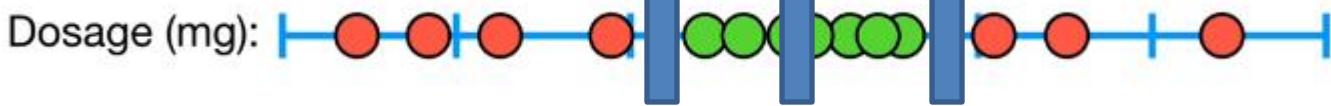
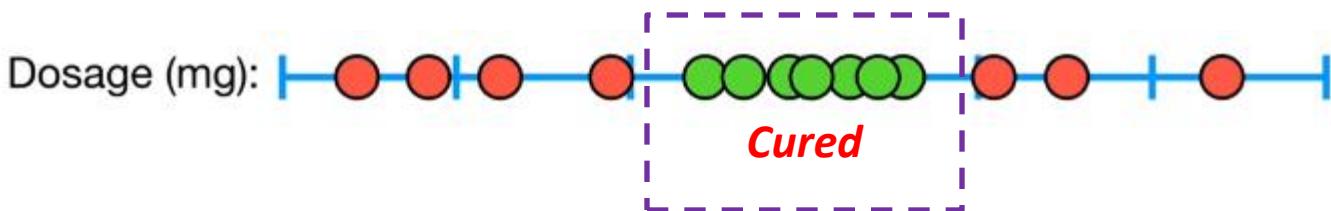


- If the data is 3D, then the Support Vector Classifier is a plane
- If the data is 4D or more, then the Support Vector Classifier is a hyperplane

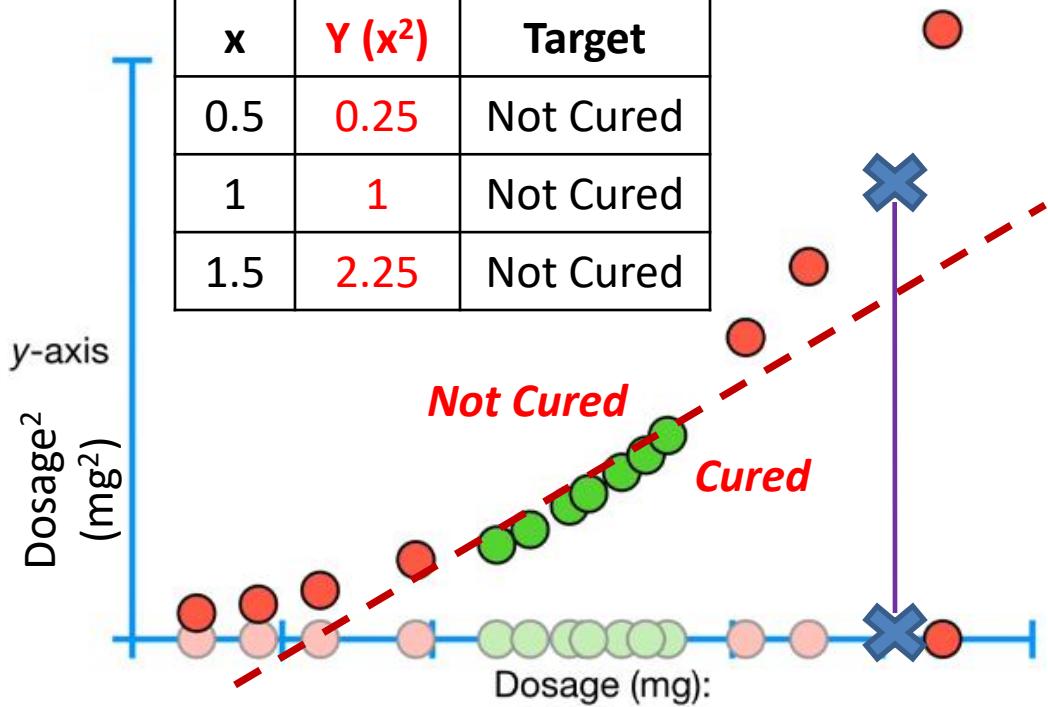


- Handle Outliers
- Since they allow misclassification, they can handle overlapped classifications

Support Vector Machine



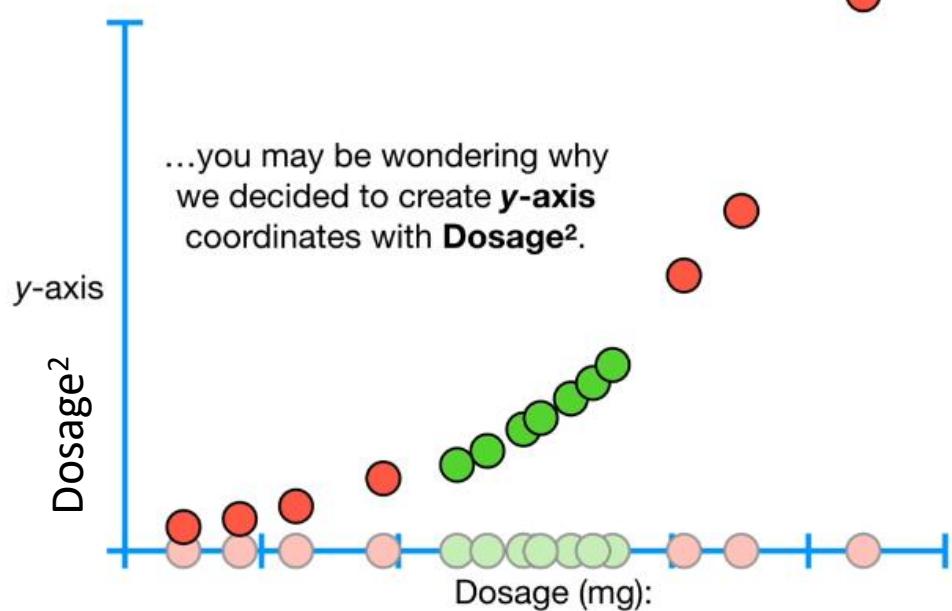
x	$Y(x^2)$	Target
0.5	0.25	Not Cured
1	1	Not Cured
1.5	2.25	Not Cured



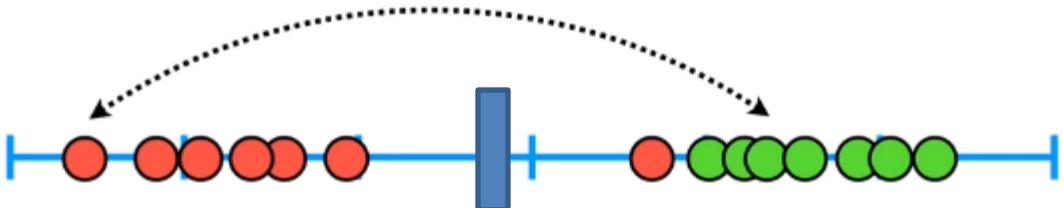
1. Start with data in low dimension
2. Move the data into a higher dimension
3. Find a support vector classifier that separates the data into 2 groups

Support Vector Machine

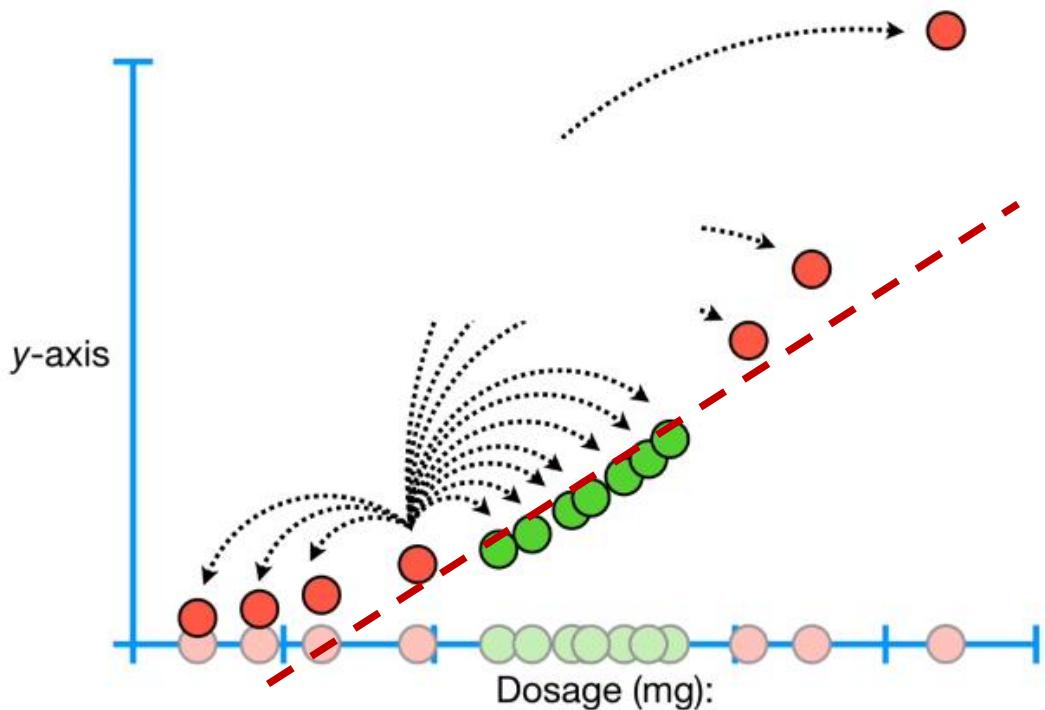
...you may be wondering why we decided to create **y-axis** coordinates with **Dosage²**.



Support Vector Machine

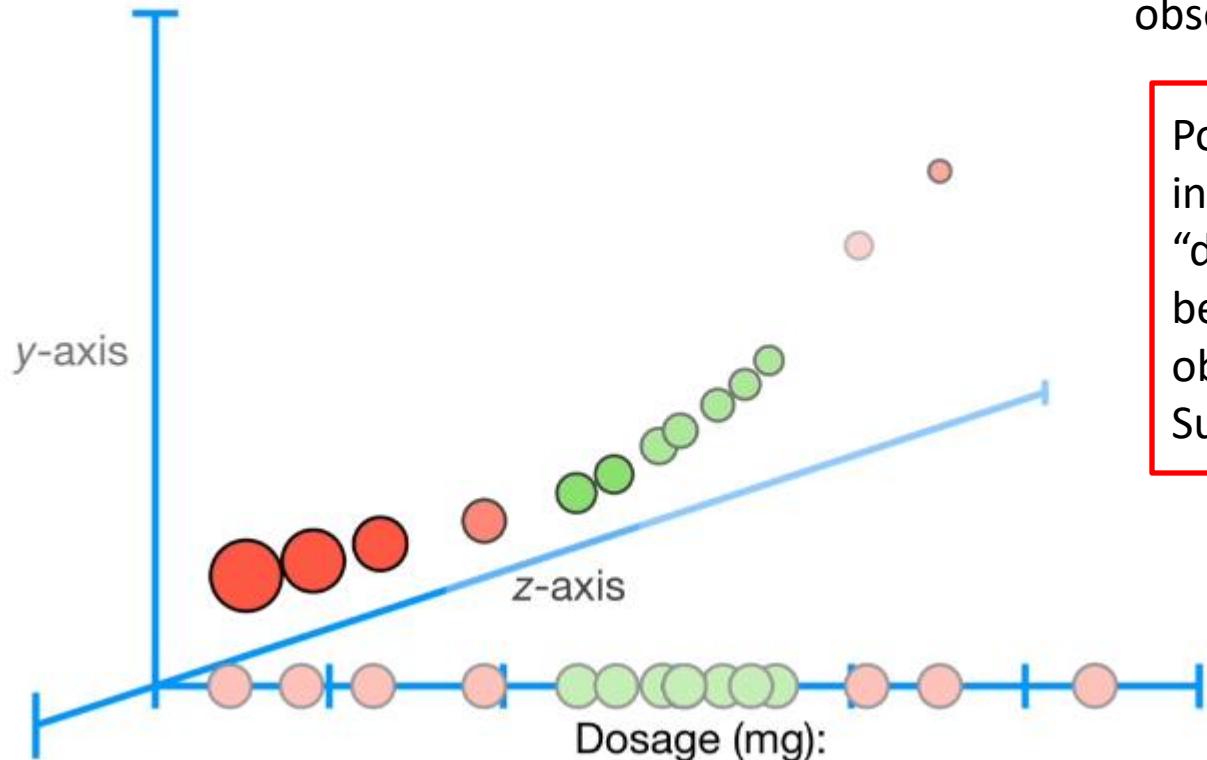


- Polynomial Kernel \rightarrow Parameter “d” which stands for the degree of the polynomial
- When $d = 1$, it computes the relationship between each pair of observations in 1D



- When $d = 2$, it computes the relationship between each pair of observations in 2D

Support Vector Machine



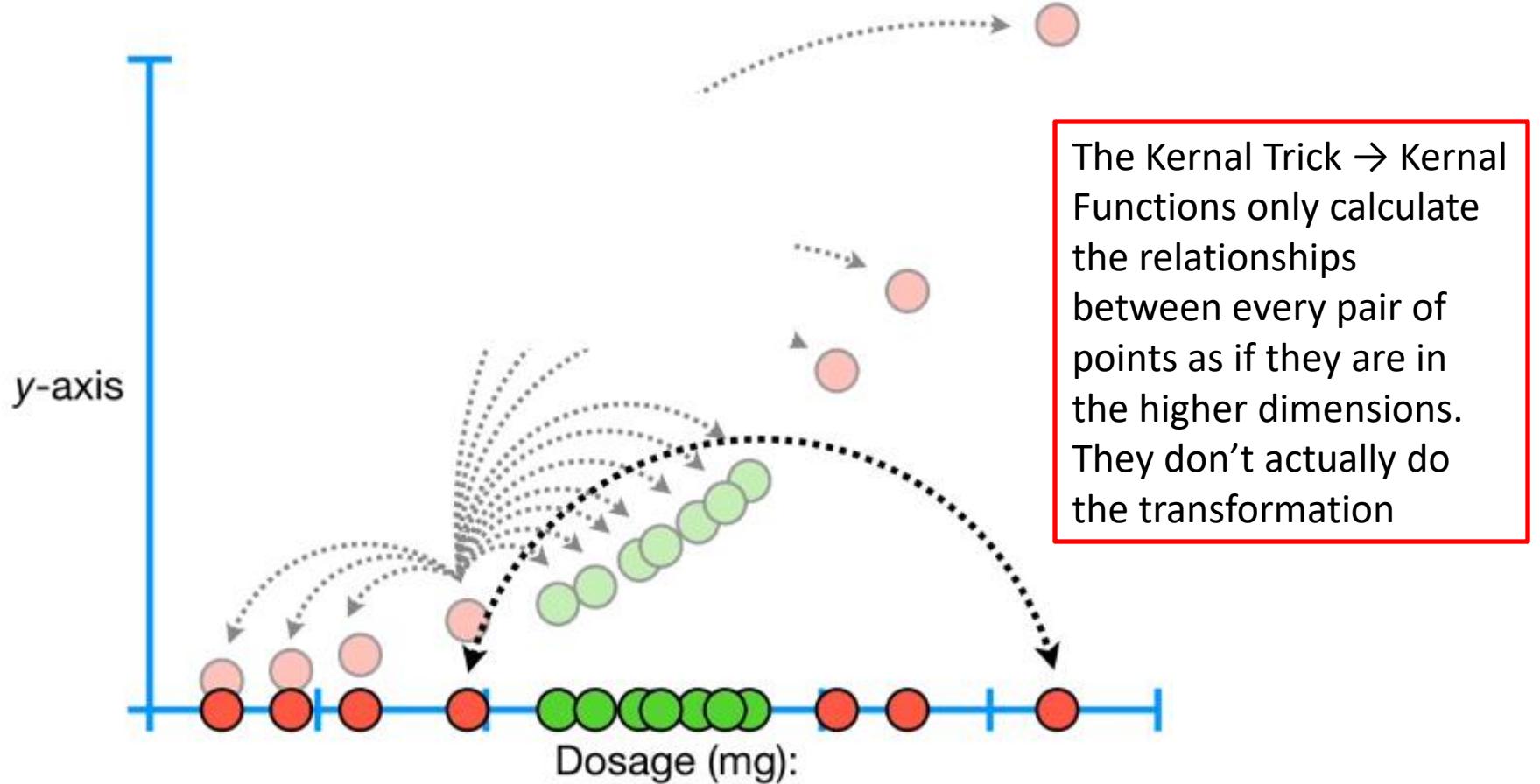
- When $d = 3$, it computes the relationship between each pair of observations in 3D

Polynomial Kernel systematically increases dimensions by setting “ d ” and the relationships between each pair of observations are used to find a Support Vector Classifier

Use cross-validation to find the best value for “ d ”

Radial Kernel → Infinite Dimensions

Support Vector Machine





Clustering Algorithm

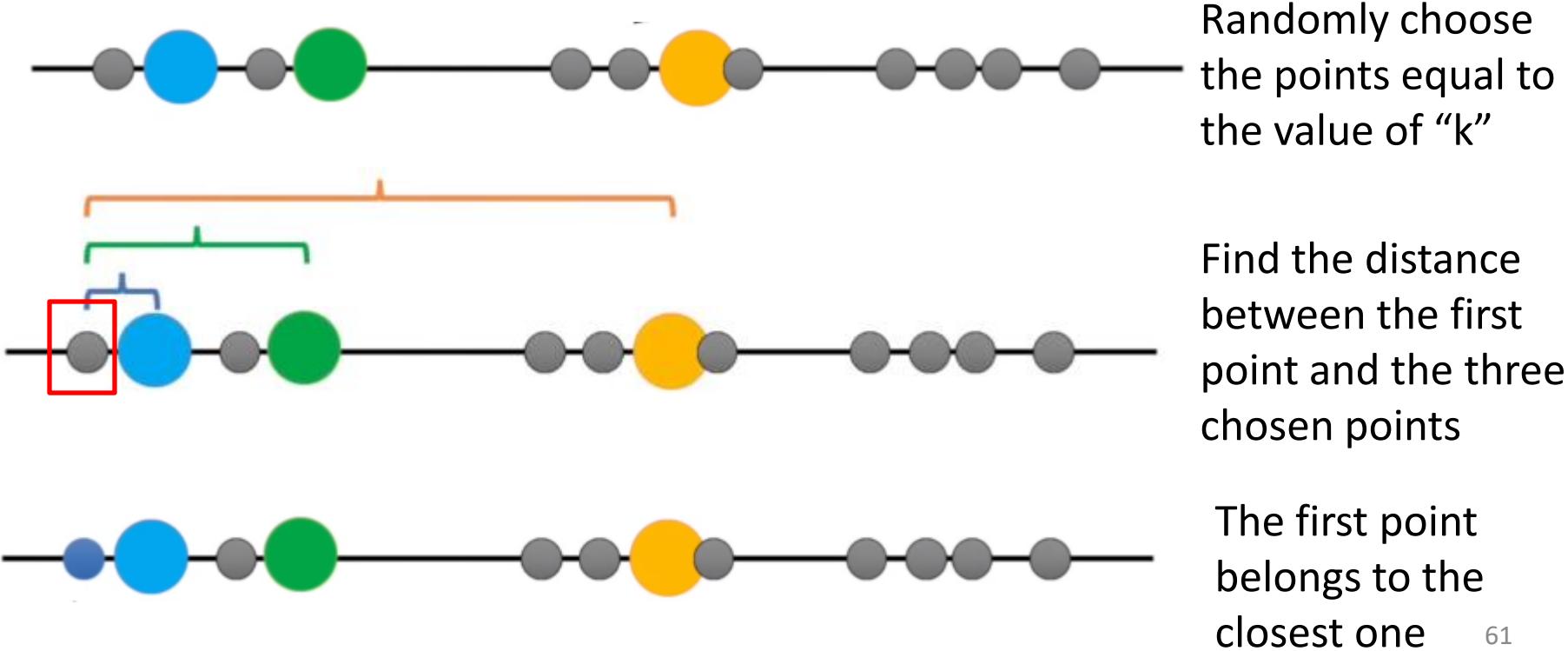
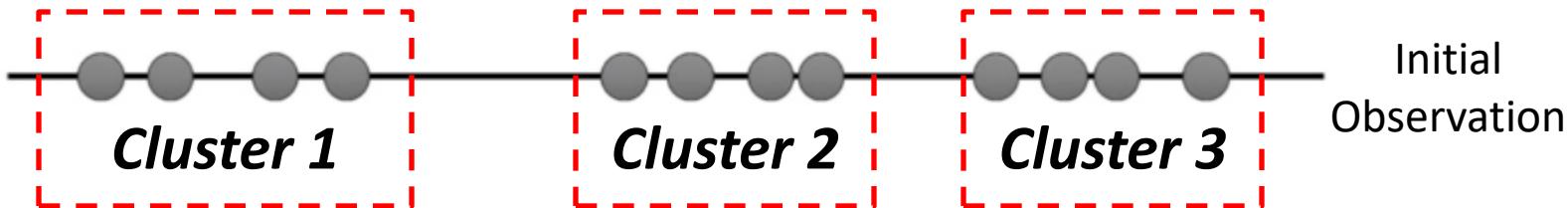
Unsupervised Learning Algorithms

- K-Means Clustering
- K-Medians Clustering
- Agglomerative Hierarchical Clustering

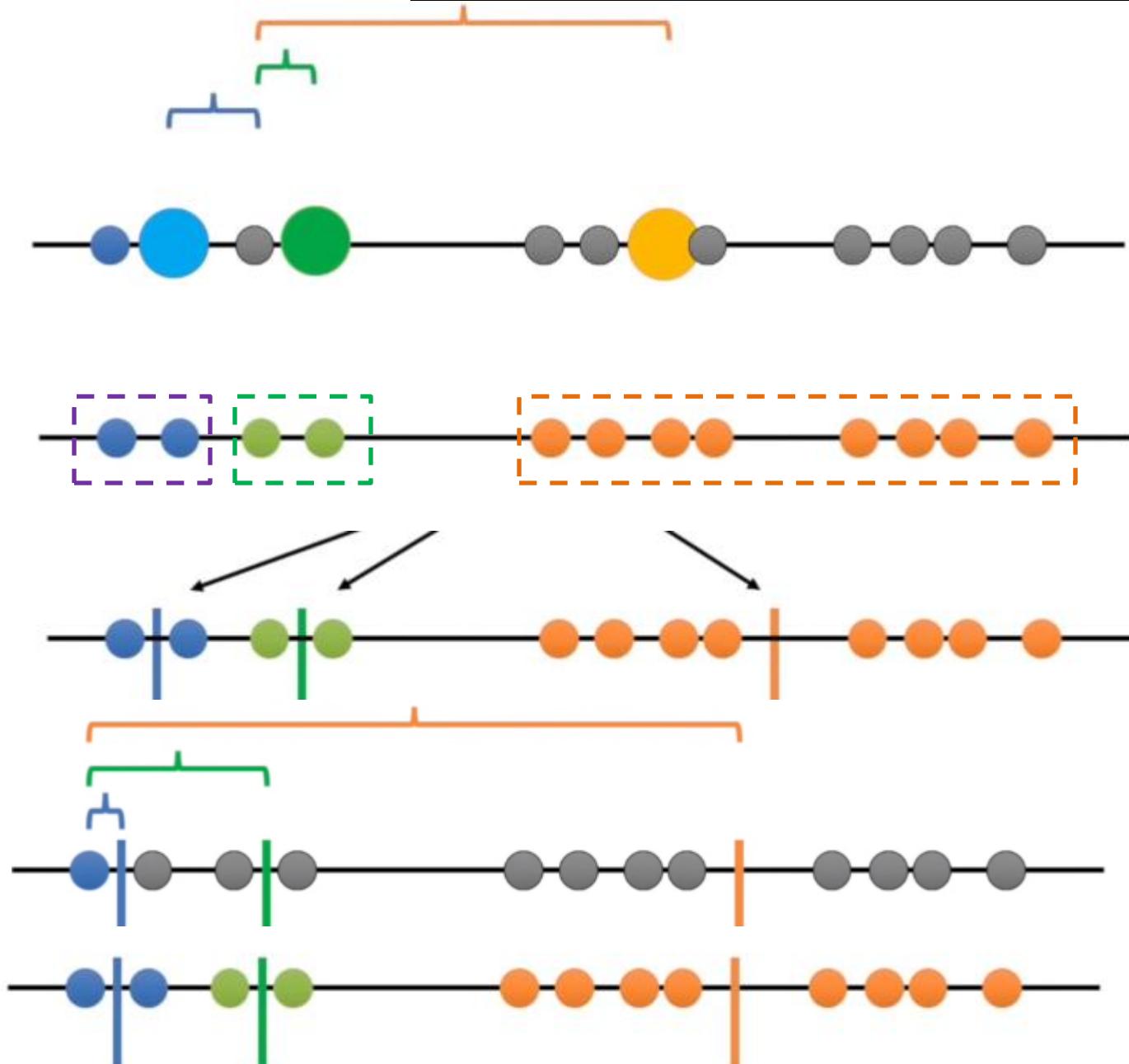
- <https://www.simplilearn.com/tutorials/machine-learning-tutorial/k-means-clustering-algorithm>
- <https://www.analytixlabs.co.in/blog/types-of-clustering-algorithms/>
- <https://towardsdatascience.com/the-5-clustering-algorithms-data-scientists-need-to-know-a36d136ef68>
- <https://towardsdatascience.com/feature-engineering-deep-dive-into-encoding-and-binning-techniques-5618d55a6b38>

K-Means Clustering

K -> No. of Clusters in your data



K-Means Clustering



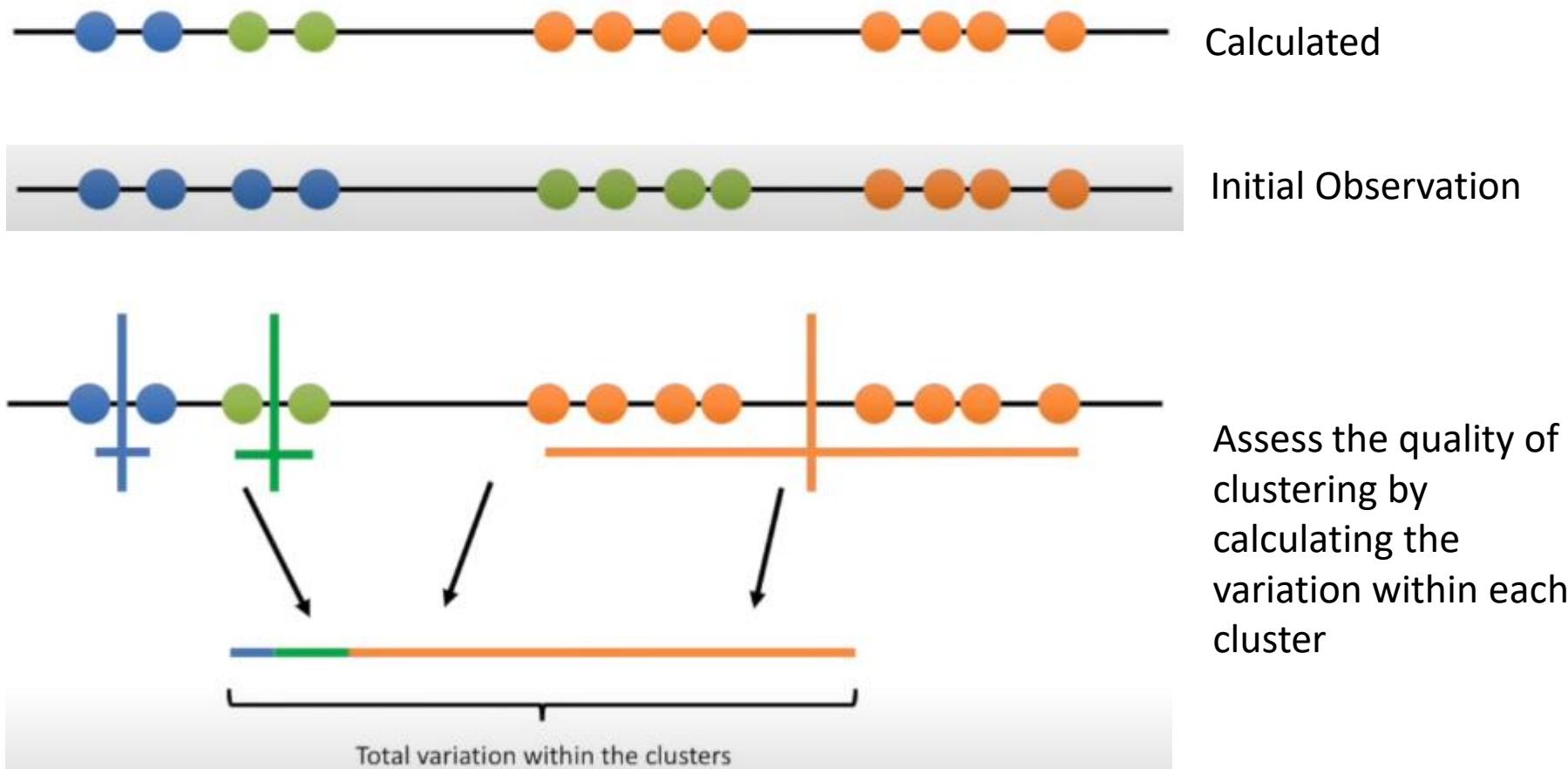
Find the distance
between the second
point and the three
chosen points

Calculate the mean
for each cluster

Repeat the process
with the identified
mean

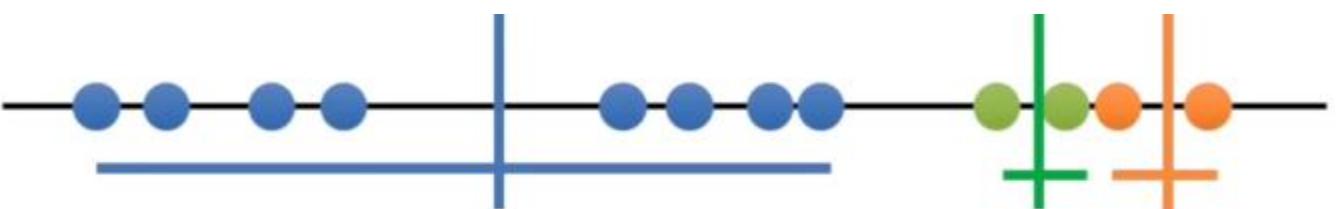
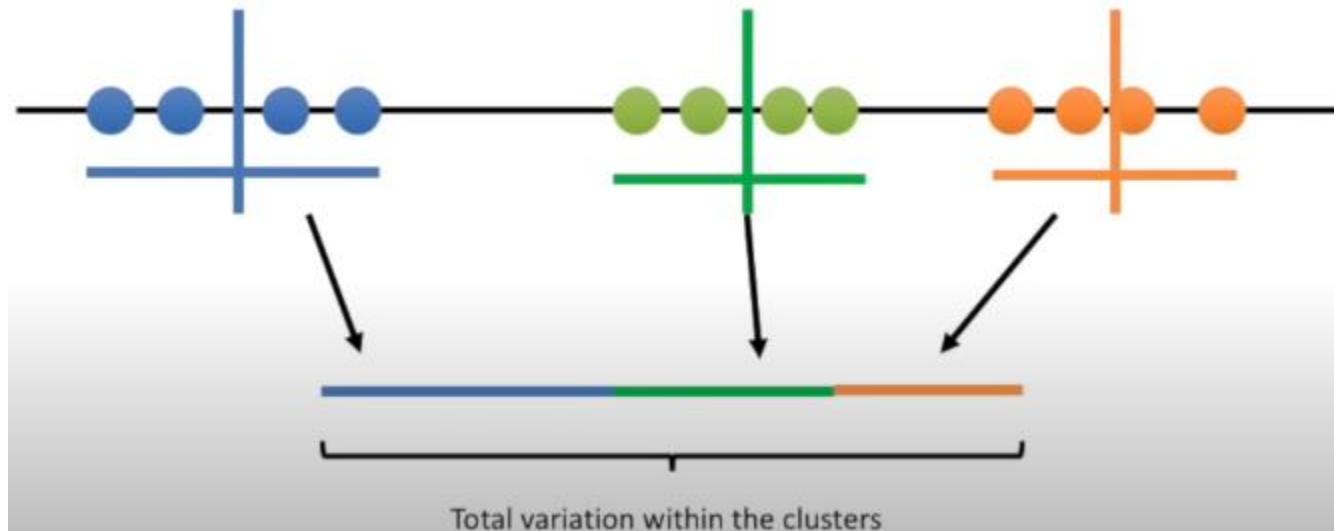
Since there is no
change, we are done ⁶²

K-Means Clustering



Since, k-means clustering can't see the best clustering, it's only option is to keep track of these clusters, and their total variance, and do the whole thing again with different starting points

K-Means Clustering

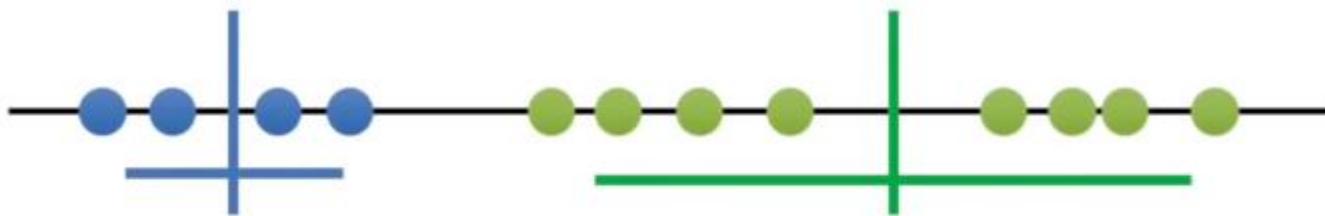


1st cluster attempt:

2nd cluster attempt:

3rd cluster attempt:

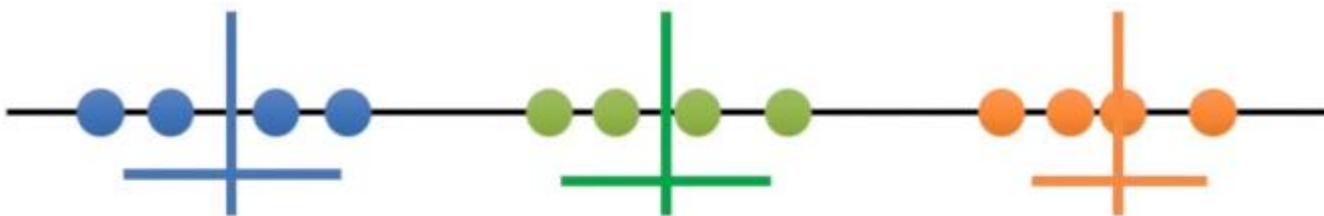
K-Means Clustering (Choosing k)



$K = 2$ is better, and we can quantify how much better by comparing the total variation within the 2 clusters to $K = 1$

$K = 1$ 

$K = 2$ 



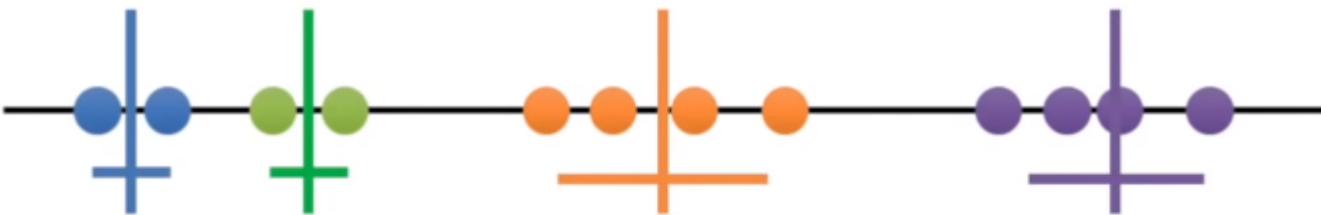
$K = 3$ is even better! We can quantify how much better by comparing the total variation within the 3 clusters to $K = 2$

$K = 1$ 

$K = 2$ 

$K = 3$ 

K-Means Clustering (*choosing k*)

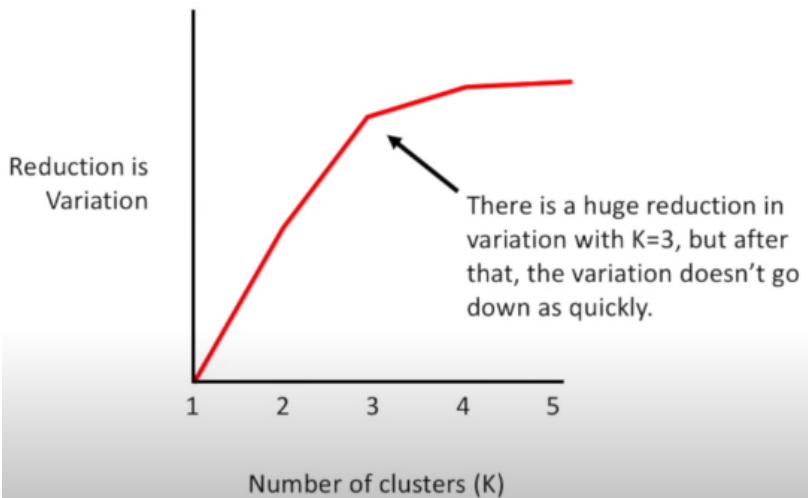


The total variation within each cluster is less than when K=3



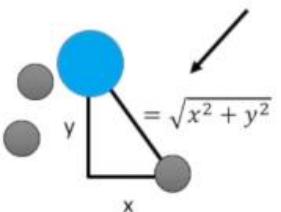
Total variance of $k = 4$ is lesser when compared with the total variance when $k = 3$

Each time we add a new cluster, the total variation within each cluster is smaller than before. And, when there is only one point per cluster, the variation = 0



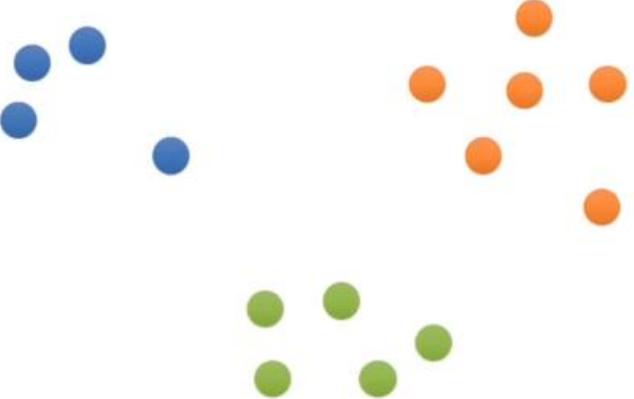
This is called an “Elbow Plot”. You can pick “K” by finding the “elbow” in the plot

K-Means Clustering



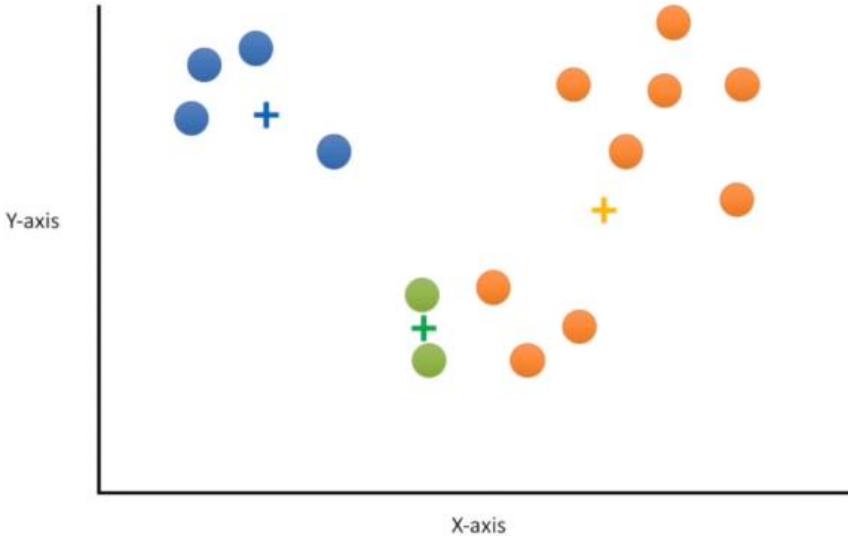
Y-axis

X-axis



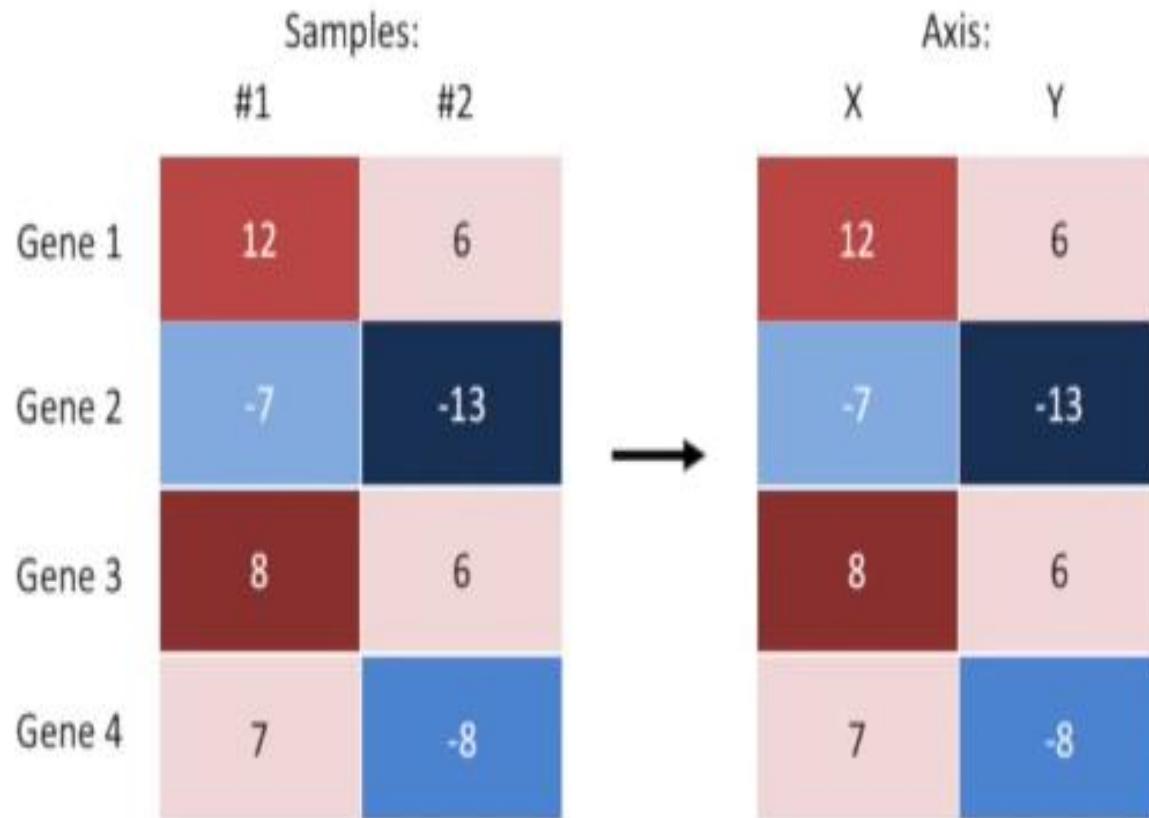
T-axis

X-axis



X-axis

K-Means Clustering



When we have 2 samples, or 2 axes, the Euclidean distance is:

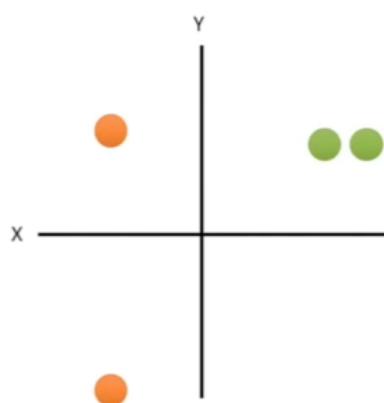
$$\sqrt{x^2 + y^2}$$

When we have 3 samples, or 3 axes, the Euclidean distance is:

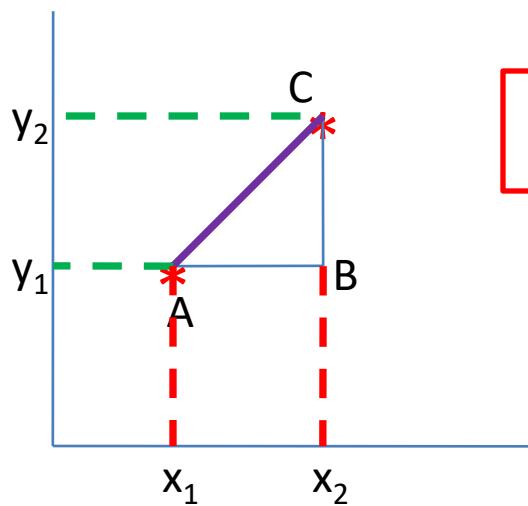
$$\sqrt{x^2 + y^2 + z^2}$$

When we have 4 samples, or 4 axes, the Euclidean distance is:

$$\sqrt{x^2 + y^2 + z^2 + a^2}$$



Euclidean Distance vs Manhattan Distance



Euclidean Distance

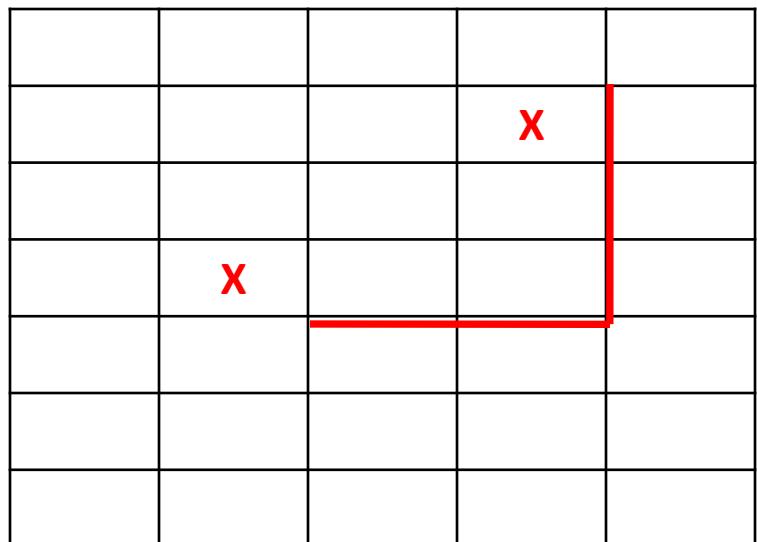
$$\begin{aligned} AC^2 &= AB^2 + BC^2 \\ &= \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \end{aligned}$$

Bangalore

Chennai

Manhattan Distance

$$= |(x_2 - x_1)| + |(y_2 - y_1)|$$





THANK YOU