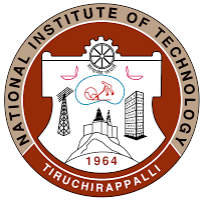
CSPE65: Machine Learning Techniques and Practices

Dr. R. Bala Krishnan
Asst. Prof.
Dept. of CSE
NIT, Tiruchirappalli – 620 015
E-Mail: balakrishnan@nitt.edu
Ph: 999 470 4853



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

Naïve Bayes Classifier

- Gaussian Naive Bayes
- Multinomial Naive Bayes
- Bernoulli Naive Bayes

Naïve Bayes Classifier

- **Gaussian Naive Bayes :** Because of the assumption of the **normal distribution**, Gaussian Naive Bayes is used in cases when all our features are **continuous**. For example in **Iris dataset** features are sepal width, petal width, sepal length, petal length. So its features can have different values in data set as width and length can vary. We cannot represent features in terms of their occurrences. This means data is continuous. Hence we use Gaussian Naive Bayes here
- **Multinomial Naive Bayes** : It is used when we have **discrete data** (e.g. movie ratings ranging 1 and 5 as each rating will have certain **frequency** to represent). In text learning we have the count of each word to predict the class or label
    - We notice that the more dollar signs ($) there are in an email, the more likely that email is spam. We can do this for many kinds of words, say (CASH or Lottery), but instead of labeling them 0 or 1, we actually count how many times each word appears in the email. This helps the model by giving it information, not just on whether the word was there, but also how many times the word appeared because we know that this is a signal to help our classifier. The algorithm assumes that the features are drawn from a multinomial distribution
- **Bernoulli Naive Bayes :** It assumes that all our features are binary such that they take only two values. Means **0s** can represent "word does not occur in the document" and **1s** as "word occurs in the document"

https://www.quora.com/What-is-the-difference-between-the-the-Gaussian-Bernoulli-Multinomial-and-the-regular-Naive-Bayes-algorithms

Naïve Bayes Classifier

Naive Bayes methods are a set of supervised learning algorithms based on applying Bayes' theorem with the "naive" assumption of conditional independence between every pair of features given the value of the class variable. Bayes' theorem states the following relationship, given class variable y and dependent feature vector x_1 through $x_{n'}$:

$$P(y \mid x_1, \dots, x_n) = \frac{P(y)P(x_1, \dots, x_n \mid y)}{P(x_1, \dots, x_n)}$$

Using the naive conditional independence assumption that

$$P(x_i|y, x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n) = P(x_i|y),$$

for all i, this relationship is simplified to

$$P(y \mid x_1, \dots, x_n) = \frac{P(y) \prod_{i=1}^{n} P(x_i \mid y)}{P(x_1, \dots, x_n)}$$

Since $P(x_1, \dots, x_n)$ is constant given the input, we can use the following classification rule:

$$P(y \mid x_1, \dots, x_n) \propto P(y) \prod_{i=1}^{n} P(x_i \mid y)$$

$$\Downarrow$$

$$\hat{y} = \arg\max_y P(y) \prod_{i=1}^{n} P(x_i \mid y),$$

and we can use Maximum A Posteriori (MAP) estimation to estimate $P(y)$ and $P(x_i \mid y)$; the former is then the relative frequency of class y in the training set.

The different naive Bayes classifiers differ mainly by the assumptions they make regarding the distribution of $P(x_i \mid y)$.

Gaussian Naïve Bayes Classifier

GaussianNB implements the Gaussian Naive Bayes algorithm for classification. The likelihood of the features is assumed to be Gaussian:

$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right)$$

The parameters σ_y and μ_y are estimated using maximum likelihood.

```
>>> from sklearn.datasets import load_iris
>>> from sklearn.model_selection import train_test_split
>>> from sklearn.naive_bayes import GaussianNB
>>> X, y = load_iris(return_X_y=True)
>>> X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5, random_state=0)
>>> gnb = GaussianNB()
>>> y_pred = gnb.fit(X_train, y_train).predict(X_test)
>>> print("Number of mislabeled points out of a total %d points : %d"
...       % (X_test.shape[0], (y_test != y_pred).sum()))
Number of mislabeled points out of a total 75 points : 4
```

https://scikit-learn.org/stable/modules/naive_bayes.html

Multinomial Naïve Bayes Classifier

`MultinomialNB` implements the naive Bayes algorithm for multinomially distributed data, and is one of the two classic naive Bayes variants used in text classification (where the data are typically represented as word vector counts, although tf-idf vectors are also known to work well in practice). The distribution is parametrized by vectors $\theta_y = (\theta_{y1}, \ldots, \theta_{ym})$ for each class y, where n is the number of features (in text classification, the size of the vocabulary) and θ_{yi} is the probability $P(x_i \mid y)$ of feature i appearing in a sample belonging to class y.

The parameters θ_y is estimated by a smoothed version of maximum likelihood, i.e. relative frequency counting:

$$\hat{\theta}_{yi} = \frac{N_{yi}}{N_y}$$

where $N_{yi} = \sum_{x \in T} x_i$ is the number of times feature i appears in a sample of class y in the training set T, and $N_y = \sum_{i=1}^{n} N_{yi}$ is the total count of all features for class y.

https://scikit-learn.org/stable/modules/naive_bayes.html

Word Count Vector

Features

DOCUMENT	REVIEW	CLASS
1	The rooms were good and I liked the location since it was good	+
2	The hotel was very bad and the stay was unpleasant	-
3	Liked the huge play area and the food was nice	+
4	The stay was good and pleasant	+
5	The location was good but was bad overall because The staff were rude	-

"+" Class:

Word	Count
Good	3
Liked	2
Bad	0
Unpleasant	0
Nice	1
Pleasant	1
Rude	0

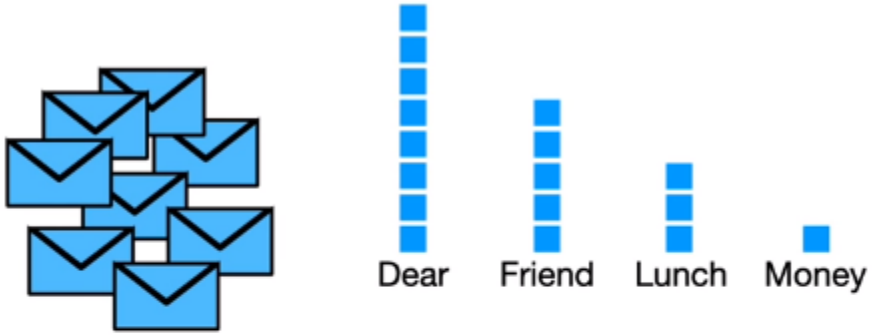
"-" Class:

Word	Count
Good	1
Liked	0
Bad	2
Unpleasant	1
Nice	0
Pleasant	0
Rude	1

Multinomial Naive Bayes Classifier

- Classify “Spam” and “Normal” messages

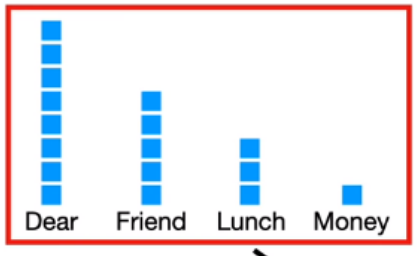
Normal Message = 8



Spam Message = 4

Multinomial Naive Bayes Classifier

- Classify “Spam” and “Normal” messages



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

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

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

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

$$p(\text{Dear} \mid \text{N}) = 0.47$$
$$p(\text{Friend} \mid \text{N}) = 0.29$$
$$p(\text{Lunch} \mid \text{N}) = 0.18$$
$$p(\text{Money} \mid \text{N}) = 0.06$$

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

p(N) = 0.67

- 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

$$p(\text{Dear} \mid \text{S}) = 0.29$$
$$p(\text{Friend} \mid \text{S}) = 0.14$$
$$p(\text{Lunch} \mid \text{S}) = 0.00$$
$$p(\text{Money} \mid \text{S}) = 0.57$$

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

p(S) = 0.33

New Message = *Dear Friend*

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

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

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

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

Multinomial Naïve Bayes Classifier

`MultinomialNB` implements the naive Bayes algorithm for multinomially distributed data, and is one of the two classic naive Bayes variants used in text classification (where the data are typically represented as word vector counts, although tf-idf vectors are also known to work well in practice). The distribution is parametrized by vectors $\theta_y = (\theta_{y1}, \ldots, \theta_{yn})$ for each class y, where n is the number of features (in text classification, the size of the vocabulary) and θ_{yi} is the probability $P(x_i \mid y)$ of feature i appearing in a sample belonging to class y.

The parameters θ_y is estimated by a smoothed version of maximum likelihood, i.e. relative frequency counting:

$$\hat{\theta}_{yi} = \frac{N_{yi} + \alpha}{N_y + \alpha n}$$

where $N_{yi} = \sum_{x \in T} x_i$ is the number of times feature i appears in a sample of class y in the training set T, and $N_y = \sum_{i=1}^{n} N_{yi}$ is the total count of all features for class y.

The smoothing priors $\alpha \geq 0$ accounts for features not present in the learning samples and prevents zero probabilities in further computations. Setting $\alpha = 1$ is called Laplace smoothing, while $\alpha < 1$ is called Lidstone smoothing.

https://scikit-learn.org/stable/modules/naive_bayes.html

Multinomial Naive Bayes Classifier

p(**Dear** | **N**) = 0.47
p(**Friend** | **N**) = 0.29
p(**Lunch** | **N**) = 0.18
p(**Money** | **N**) = 0.06

p(N) = 0.67

p(**Dear** | **S**) = 0.29
p(**Friend** | **S**) = 0.14
p(**Lunch** | **S**) = 0.00
p(**Money** | **S**) = 0.57

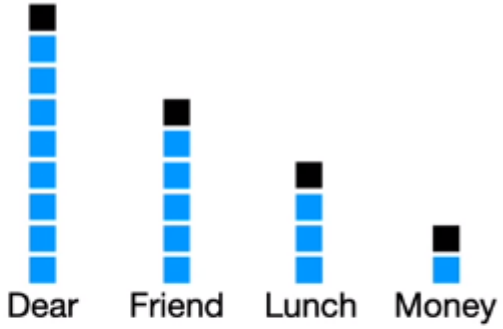
p(S) = 0.33

New Message = Lunch Money Money Money Money

$p(N) \times p(\text{Lunch} \mid N) \times p(\text{Money} \mid N)^4 = 0.000002$

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

p(N) = 0.67

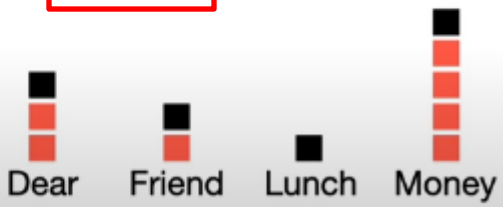


α = 1

$p(N) \times p(\text{Lunch} \mid N) \times p(\text{Money} \mid N)^4 = 0.00001$

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

p(S) = 0.33



Multinomial Naive Bayes Classifier

p(**Dear** | **N**) = 0.43
p(**Friend** | **N**) = 0.29
p(**Lunch** | **N**) = 0.19
p(**Money** | **N**) = 0.10

p(**N**) = 0.67

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**) × p(**Dear** | **N**) × p(**Friend** | **N**) = 0.08

Score for **Friend Dear** =

p(**N**) × p(**Friend** | **N**) × p(**Dear** | **N**) = 0.08

https://www.geeksforgeeks.org/naive-bayes-classifiers/#:~:text=Naive%20Bayes%20classifiers%20are%20a,is%20independent%20of%20each%20other.

Bernoulli Naïve Bayes Classifier

`BernoulliNB` implements the naive Bayes training and classification algorithms for data that is distributed according to multivariate Bernoulli distributions; i.e., there may be multiple features but each one is assumed to be a binary-valued (Bernoulli, boolean) variable. Therefore, this class requires samples to be represented as binary-valued feature vectors; if handed any other kind of data, a `BernoulliNB` instance may binarize its input (depending on the `binarize` parameter).

Confident	Studied	Sick	Result
Yes	No	No	Fail
Yes (1)	No	Yes	Pass
No	Yes	Yes	Fail
No (0)	Yes	No	Pass
Yes (1)	Yes	Yes	Pass

The decision rule for Bernoulli naive Bayes is based on

$$P(x_i \mid y) = P(i \mid y)x_i + (1 - P(i \mid y))(1 - x_i)$$

which differs from multinomial NB's rule in that it explicitly penalizes the non-occurrence of a feature i that is an indicator for class y, where the multinomial variant would simply ignore a non-occurring feature.

In the case of text classification, word occurrence vectors (rather than word count vectors) may be used to train and use this classifier. `BernoulliNB` might perform better on some datasets, especially those with shorter documents. It is advisable to evaluate both models, if time permits.

https://scikit-learn.org/stable/modules/naive_bayes.html

Word Occurrence Vector

Features

DOCUMENT	REVIEW	CLASS
1	The rooms were good and I liked the location since it was good	+
2	The hotel was very bad and the stay was unpleasant	-
3	Liked the huge play area and the food was nice	+
4	The stay was good and pleasant	+
5	The location was good but was bad overall because The staff were rude	-

"+" Class:

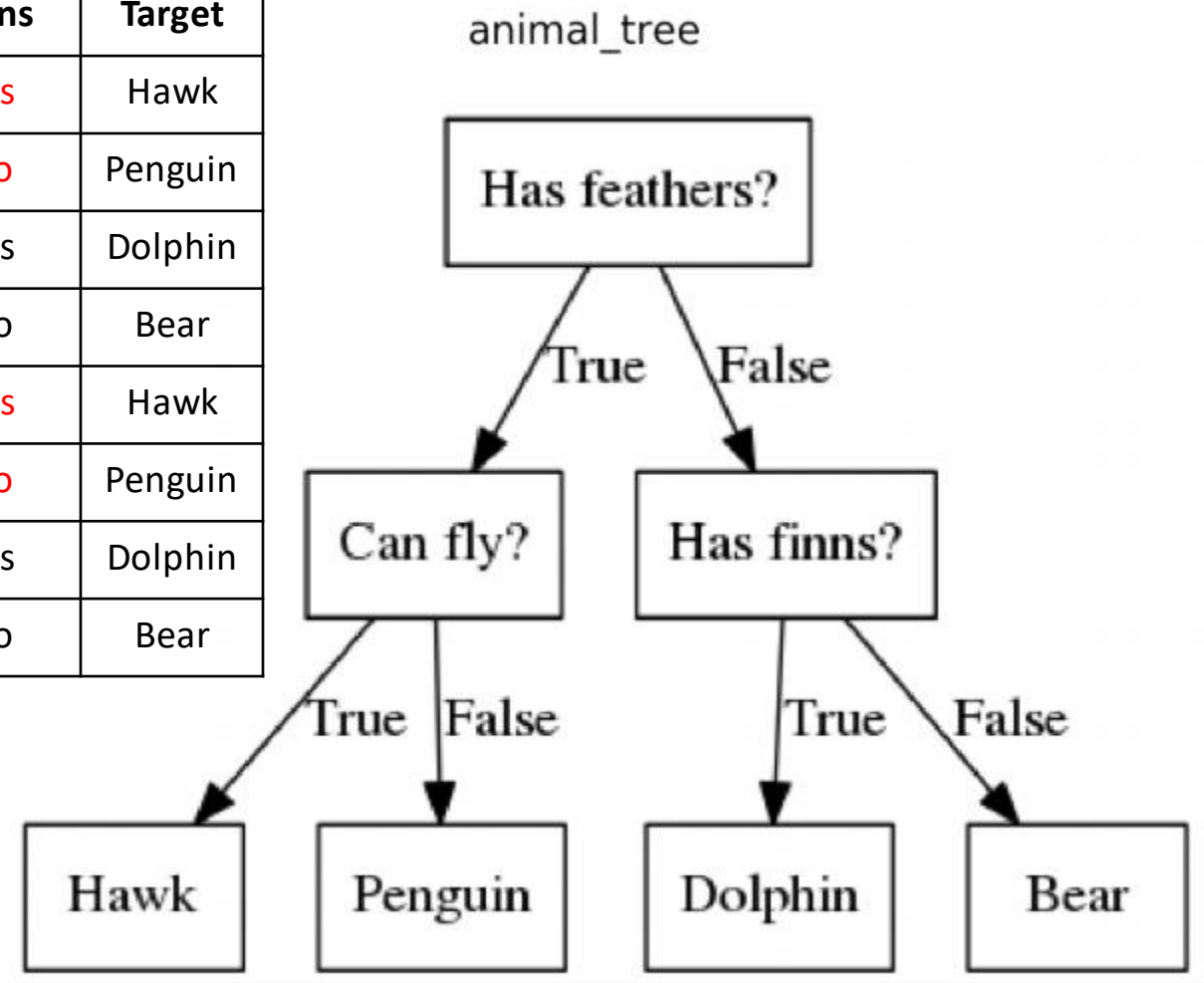
Features \ Documents	D1	D3	D4
good	1	0	1
liked	1	1	0
bad	0	0	0
unpleasant	0	0	0
nice	0	1	0
pleasant	0	0	1
rude	0	0	0

"-" Class:

Features \ Documents	D2	D5
good	0	1
liked	0	0
bad	1	1
unpleasant	1	0
nice	0	0
pleasant	0	0
rude	0	1

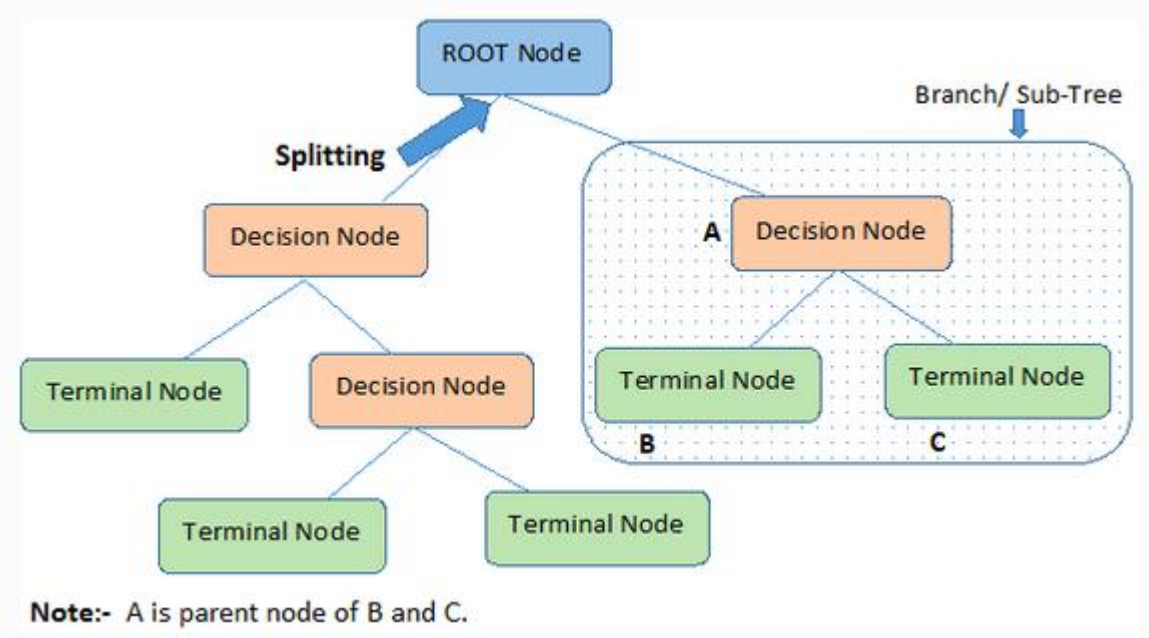
Decision Trees

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



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.

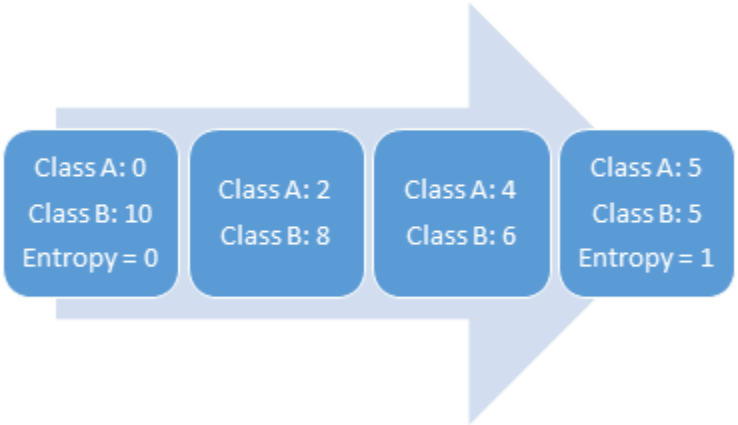


Decision Trees (Terminologies)

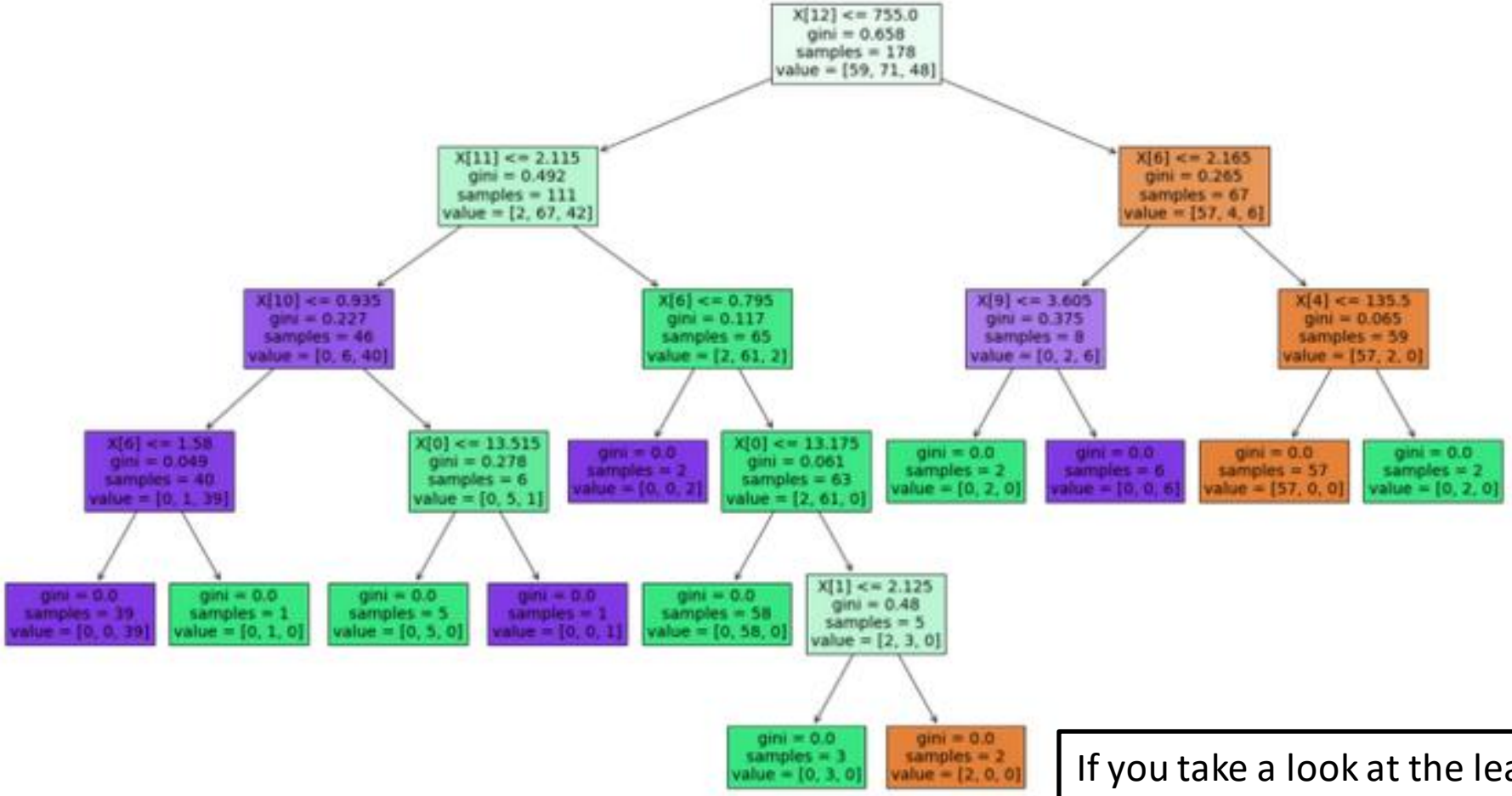
- **Attribute Selective Measures:**
  - Information Gain or ID3
    - Entropy
  - Gini Impurity or Gini Index

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

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

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

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

Calculation 3: Find the Gain for every column

$$Gain = Entropy(S) - I(Attribute)$$

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.

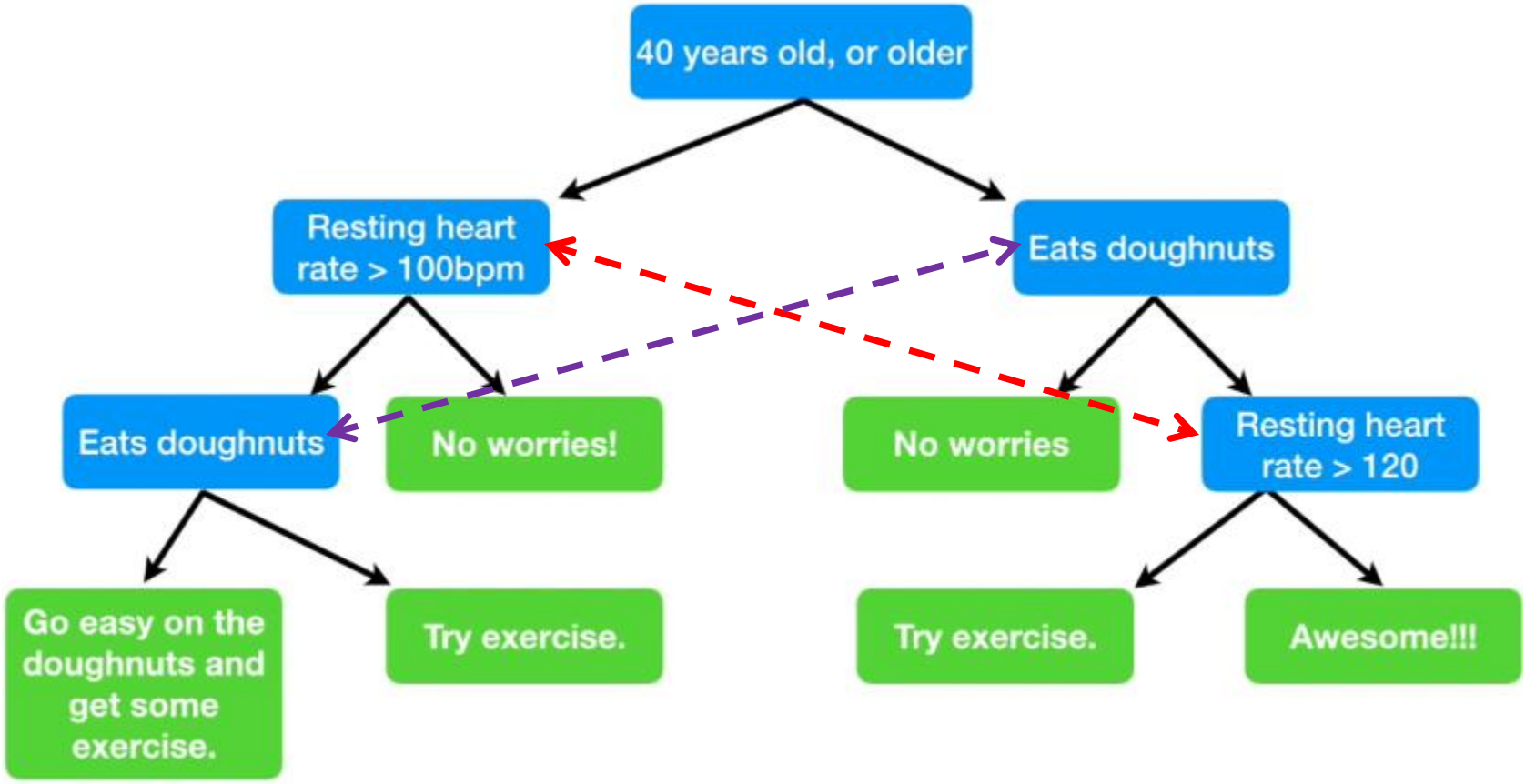
Mathematical Formula :

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

Pi= 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



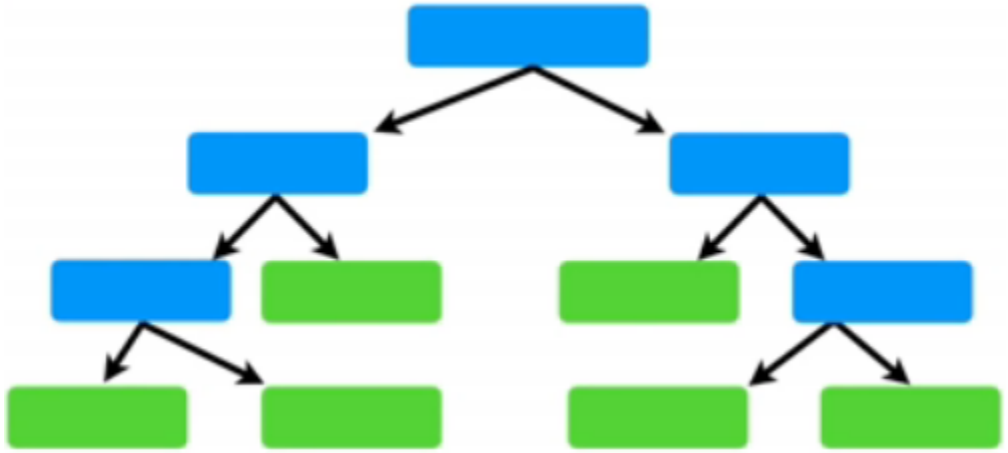
Questions:

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

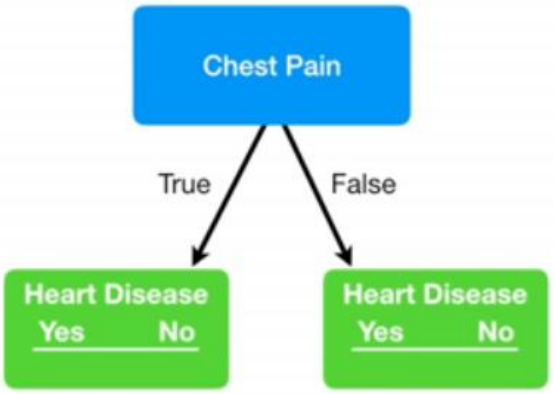
- Root Node or Root
- Internal Node or Node
- Leaf Node or Leaf

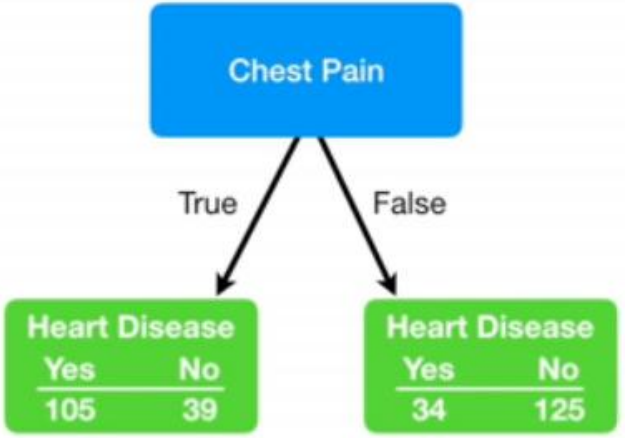
Decision Trees *(for "Yes / No" Data)*

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…

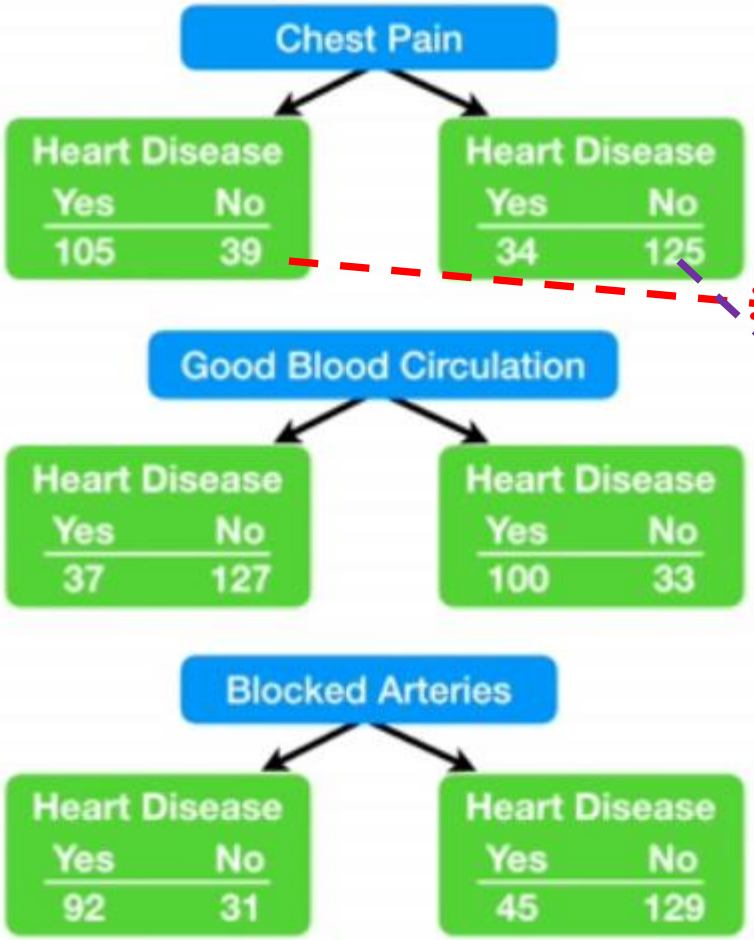


Chest Pain	Good Blood Circulation	Blocked Arteries	Heart Disease
No	No	No	No
Yes	Yes	Yes	Yes
Yes	Yes	No	No
Yes	No	No	Yes
etc…	etc…	etc…	etc…





Decision Trees



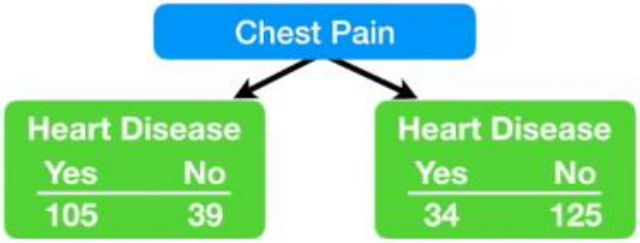
Impure

Gini impurity = 1 - (the probability of "yes")^2 - (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 - (the probability of "yes")^2 - (the probability of "no")^2

$$= 1 - \left(\frac{34}{34 + 125}\right)^2 - \left(\frac{125}{34 + 125}\right)^2 = 0.336$$



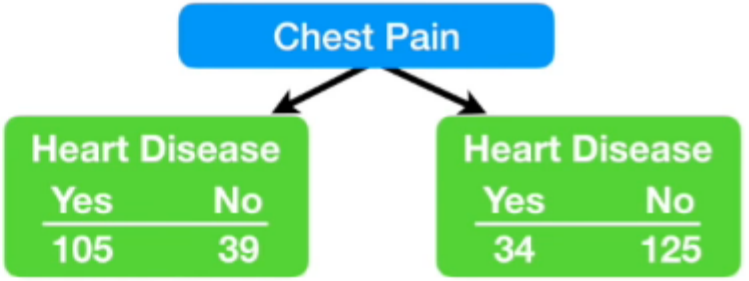
Gini impurity = 0.395 0.336

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

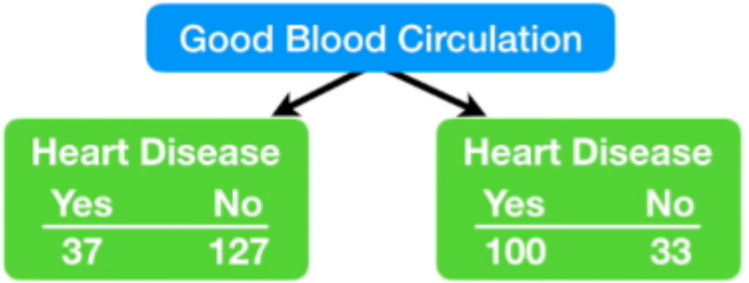
$$= \left(\frac{144}{144 + 159}\right) 0.395 + \left(\frac{159}{144 + 159}\right) 0.336 = 0.364$$

Decision Trees

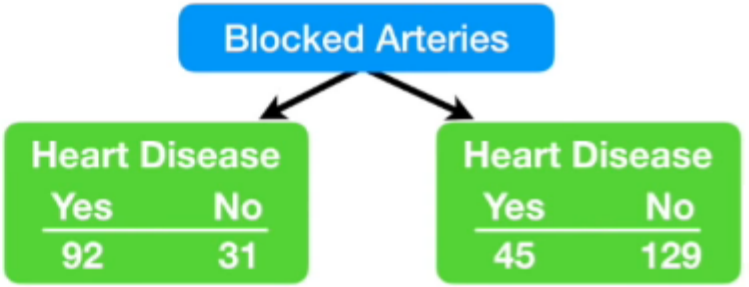
Gini impurity for Chest Pain = 0.364

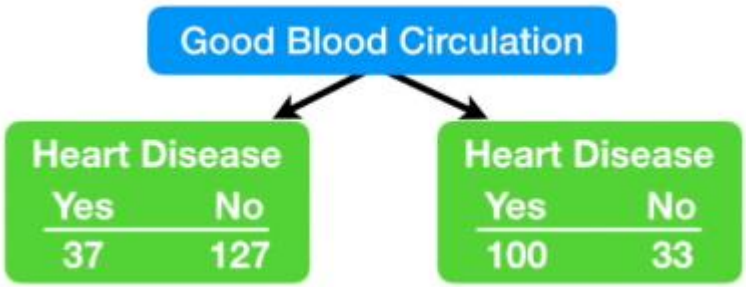


Gini impurity for Good Blood Circulation = 0.360

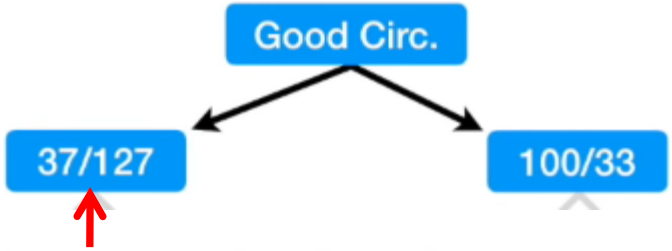


Gini impurity for Blocked Arteries = 0.381

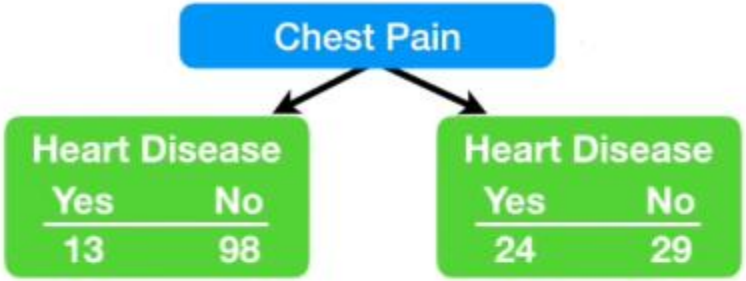




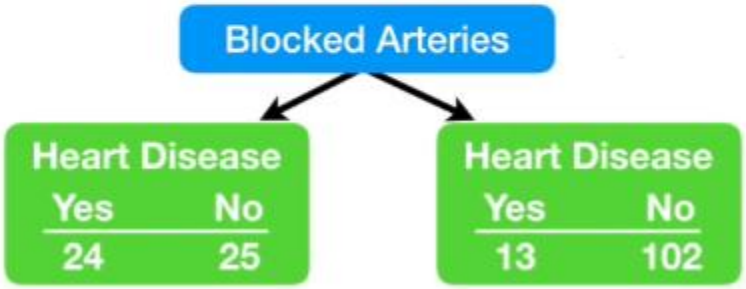
Decision Trees



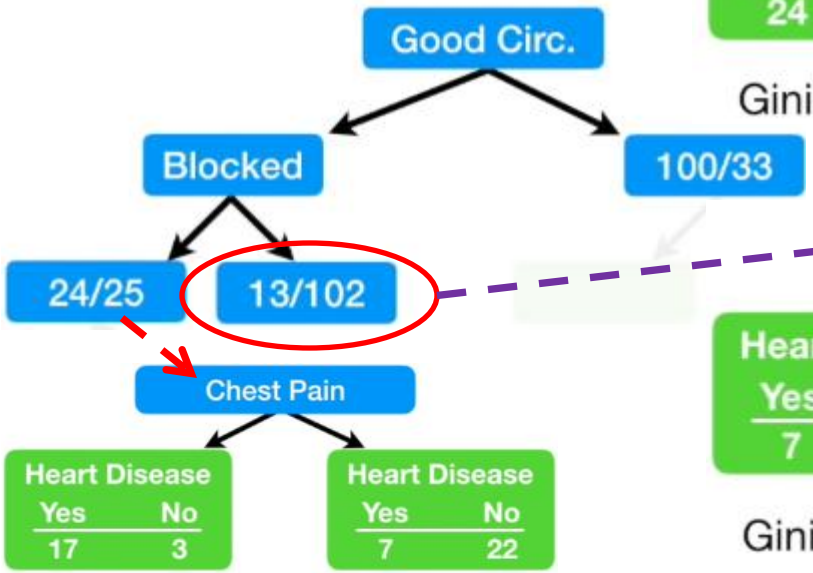
Now we need to figure how well **chest pain** and **blocked arteries** separate these 164 patients (37 with heart disease and 127 without heart disease).

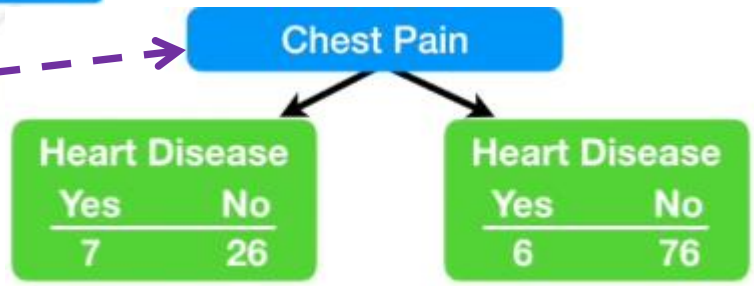


Gini impurity for Chest Pain = 0.3



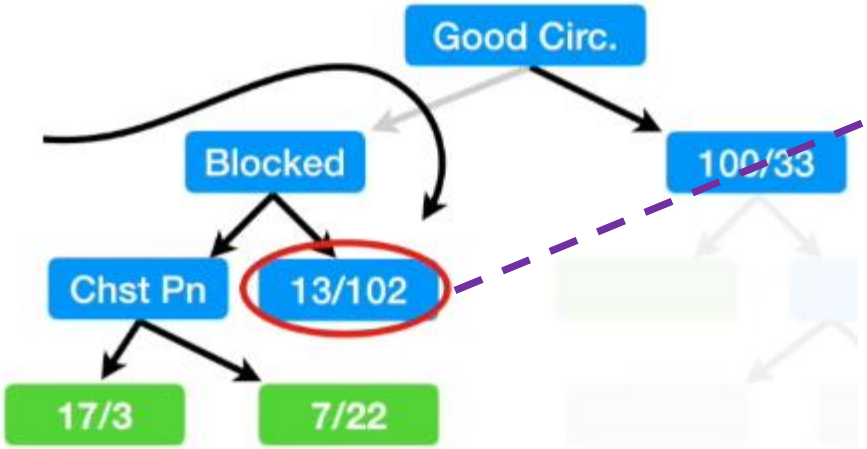
Gini impurity for Blocked Arteries = 0.290

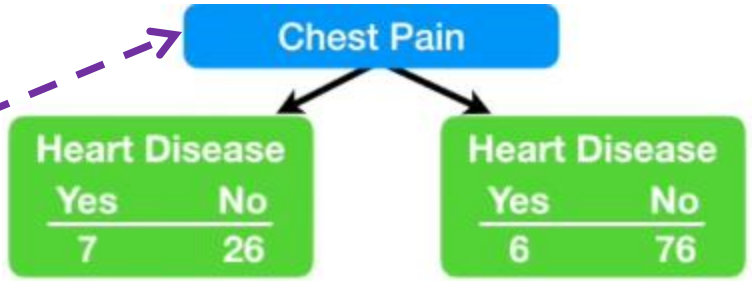




Gini impurity for Chest Pain = 0.29

Decision Trees





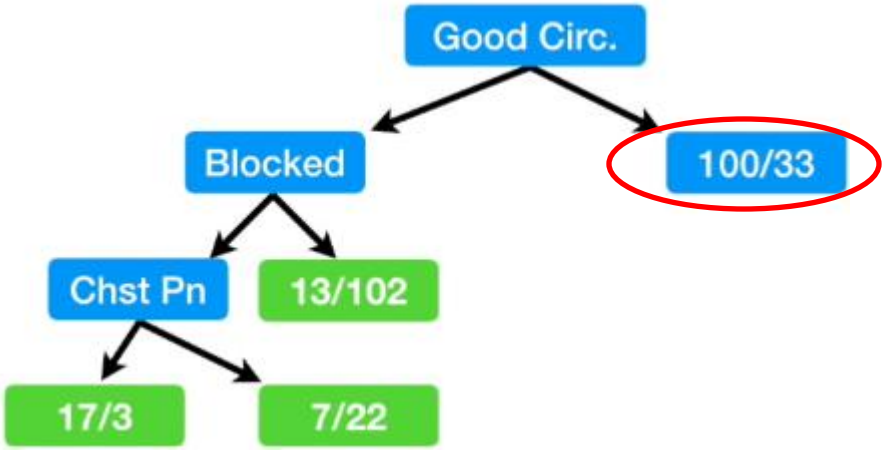
Gini impurity for Chest Pain = 0.29

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

= 1 - (the probability of "yes")^2 - (the probability of "no")^2

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

= 0.2



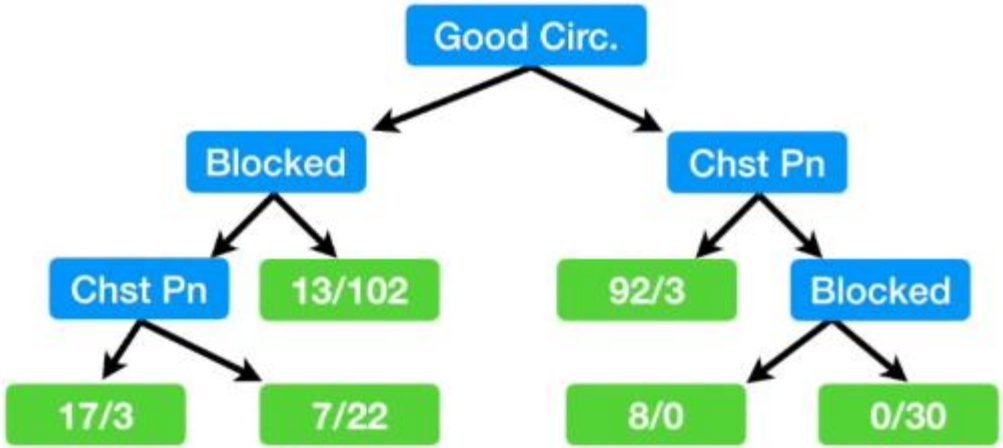
Steps:

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
Lowest	155	No
	180	Yes
	190	No
	220	Yes
Highest	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		Gini impurity = ?
180	Yes	
185		Gini impurity = ?
190	No	
205		Gini impurity = ?
220	Yes	
222.5		Gini impurity = ?
225	Yes	

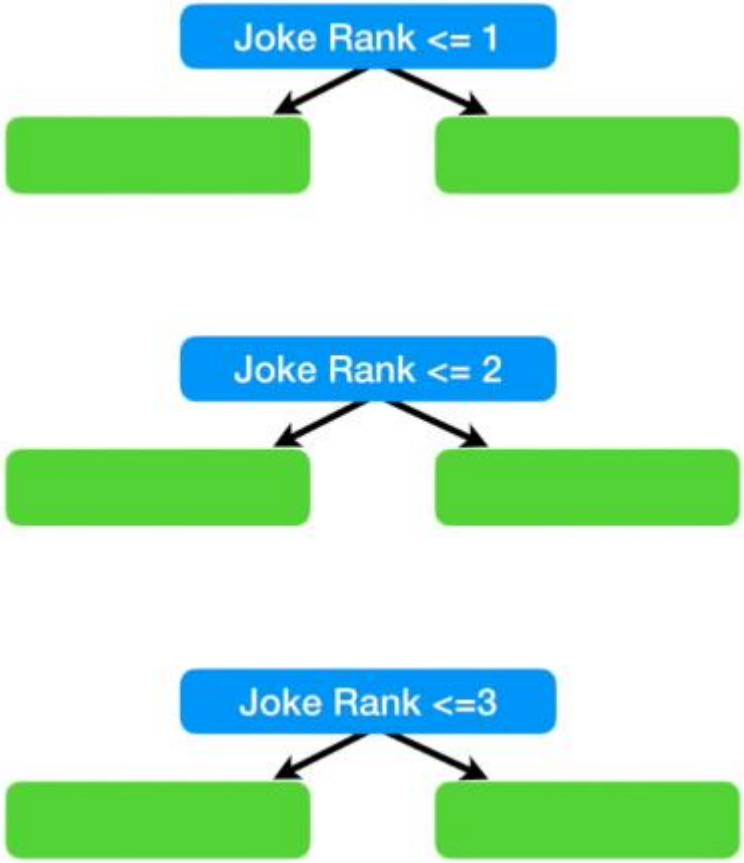
Weight	Heart Disease	
155	No	
167.5		Gini impurity = 0.3
180	Yes	
185		Gini impurity = 0.47
190	No	
205		Gini impurity = 0.27
220	Yes	
222.5		Gini impurity = 0.4
225	Yes	

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...

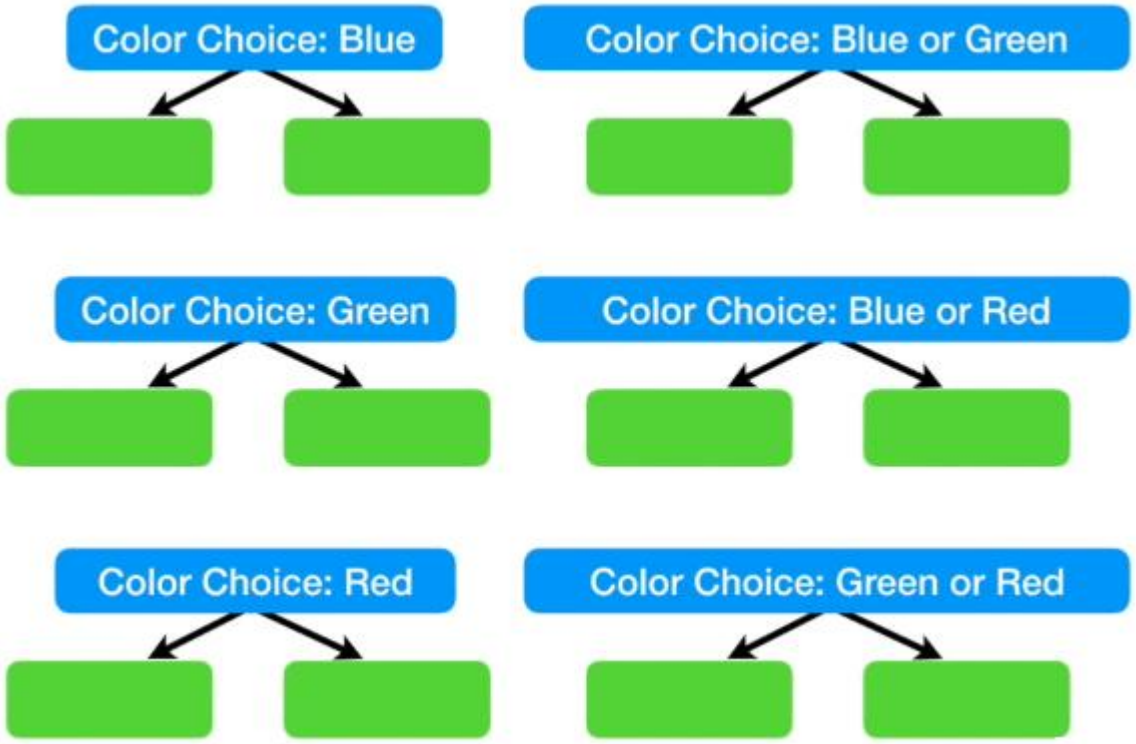


***Note:* Do not calculate for Rank <= 4.**
Otherwise, all samples will fall in that category.

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*

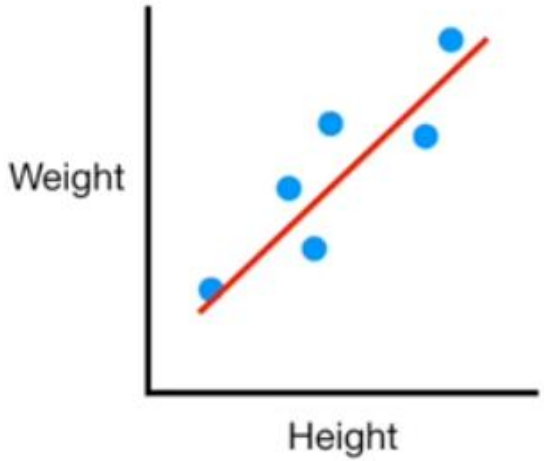
Handling Missing Data

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

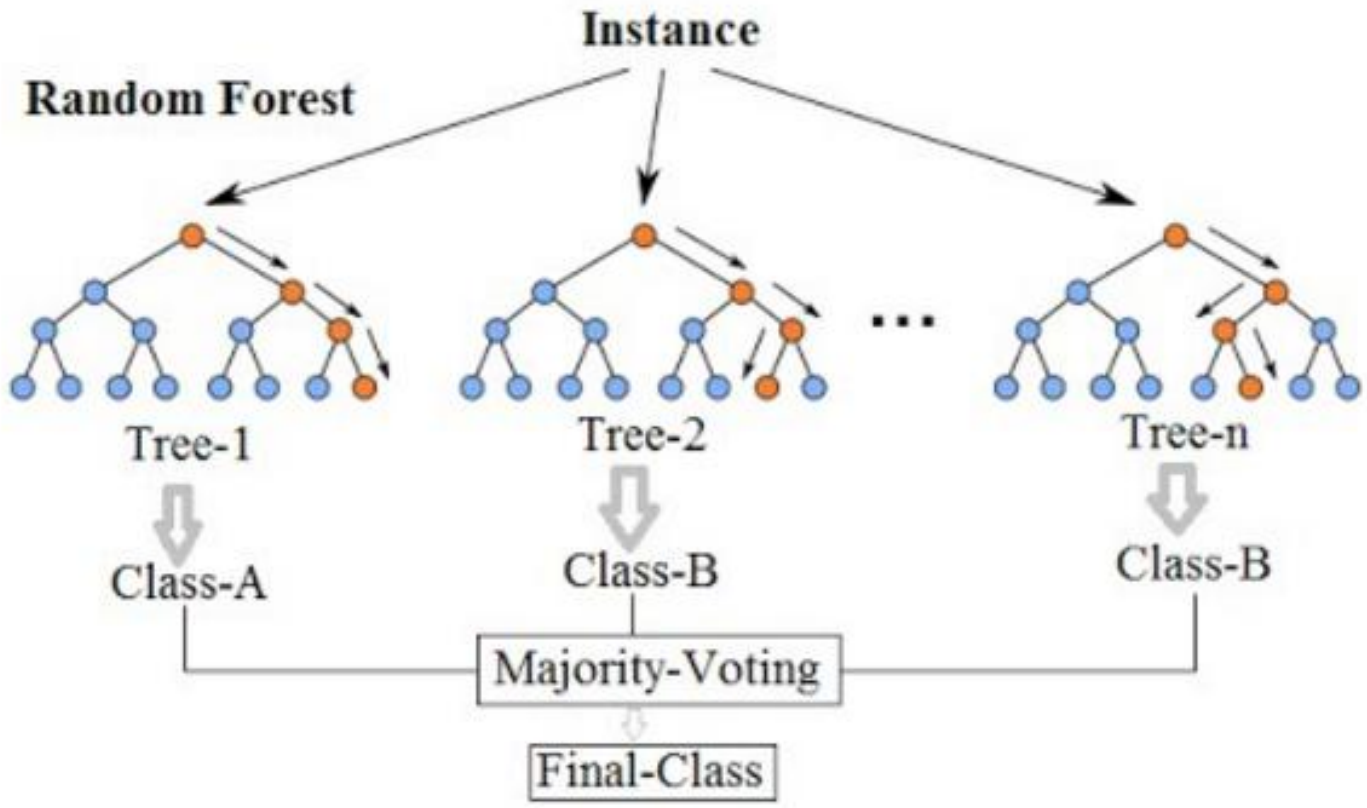
Chest Pain	Good Blood Circulation	Blocked Arteries	Heart Disease
No	No	No	No
Yes	Yes	Yes	Yes
Yes	Yes	No	No
Yes	No	???	Yes
etc...	etc...	etc...	etc...

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

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

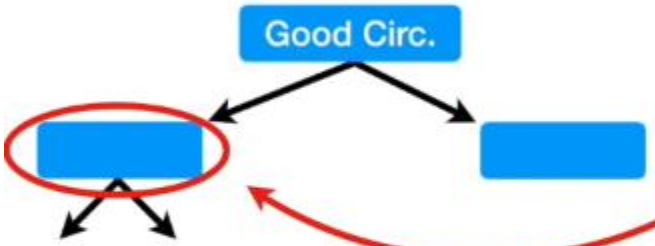
???

…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

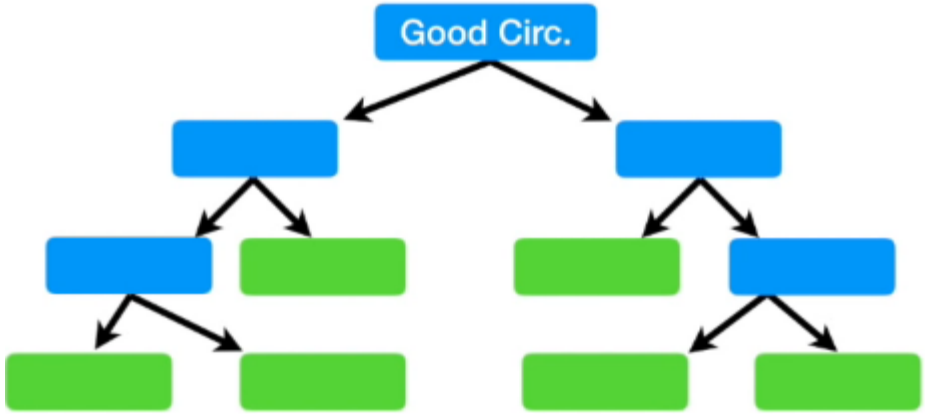
Random Forest



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.



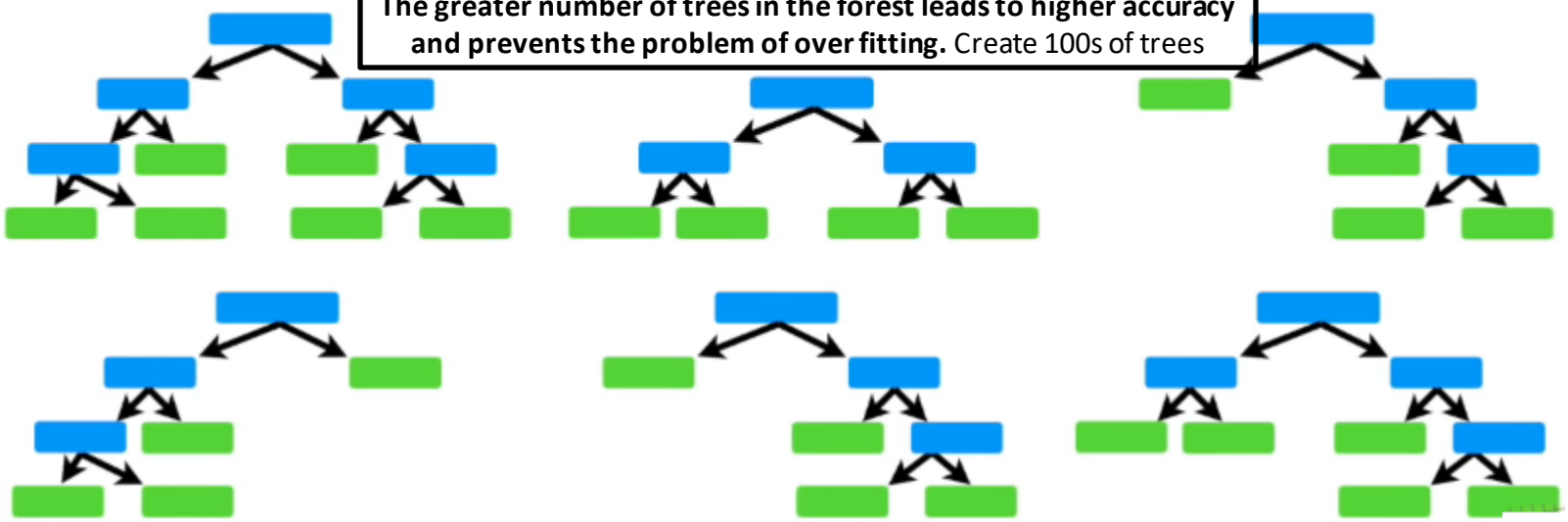
We built a tree…

1) Using a bootstrapped dataset

2) Only considering a random a subset of variables at each step.

Random Forest

The greater number of trees in the forest leads to higher accuracy and prevents the problem of over fitting. Create 100s of trees



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

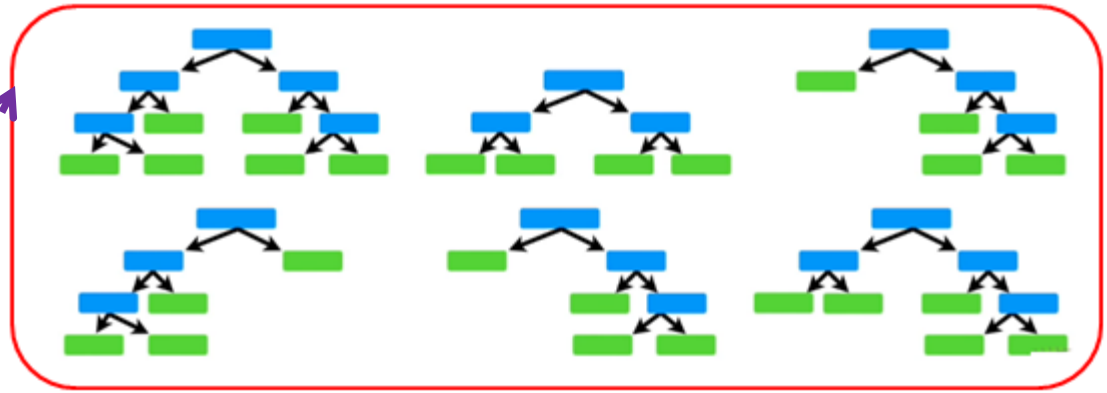
Bootstrapped Dataset

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	No
1	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	No
4	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	No
3	1

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

OK, we now know how to:

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

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

…change the number of variables used per step…

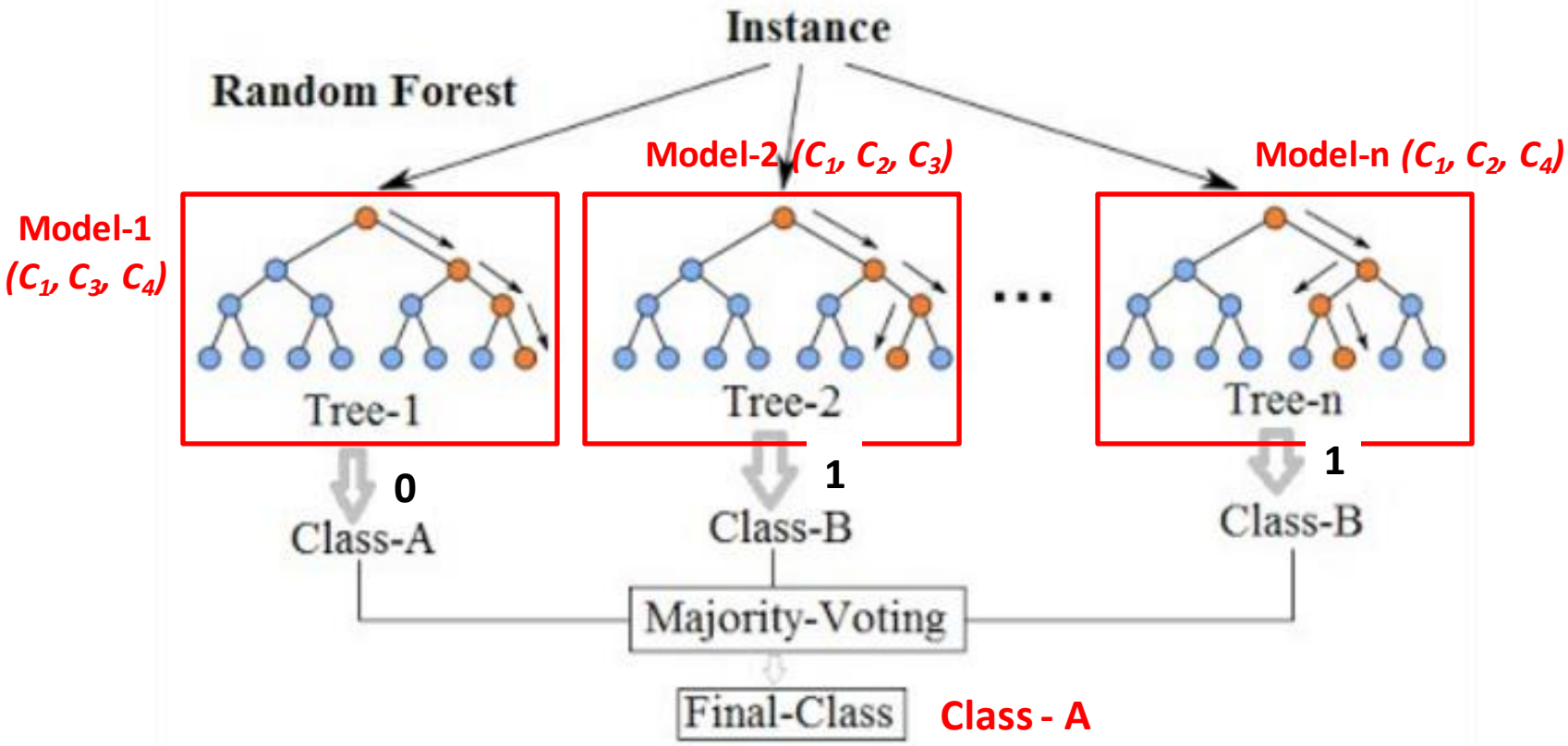
1) Build a Random Forest
2) Estimate the accuracy of a Random Forest.

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

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

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

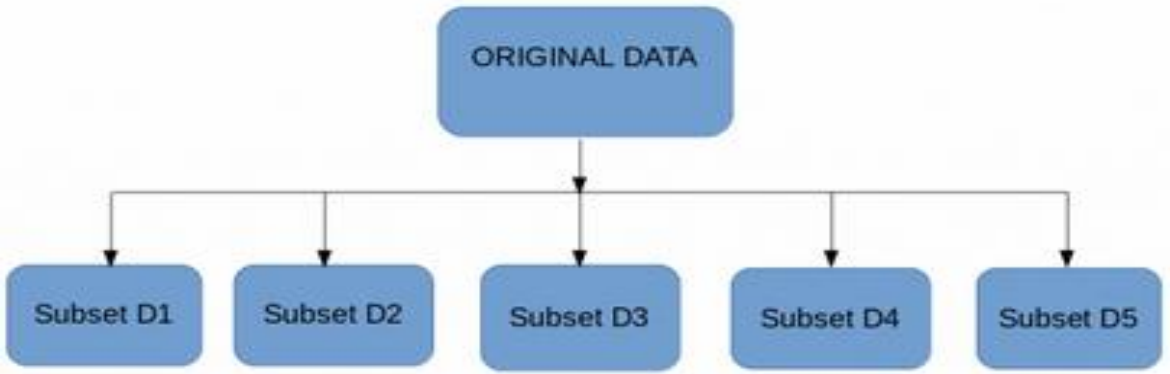
- *Combining Classifiers*
  - Combine the results to make a final prediction
  - Simple majority to Weighted majority voting
- Ensemble models are really exciting
- Offers potential for breakthroughs in classification problems

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

https://www.ibm.com/cloud/learn/bagging

Bagging Meta-Estimator

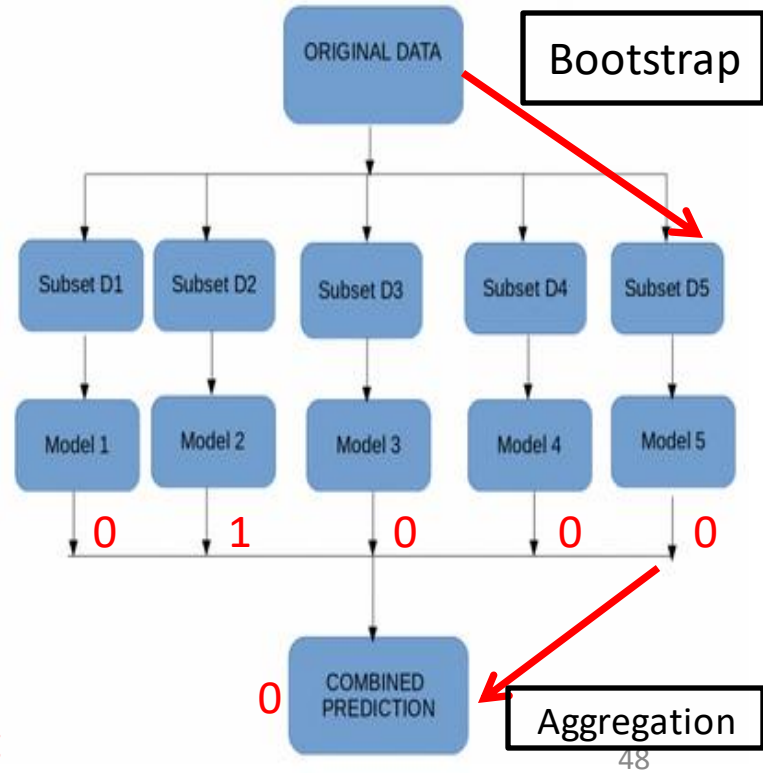


Row Sampling with Replacement

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

Train Dataset

Test Dataset



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

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

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

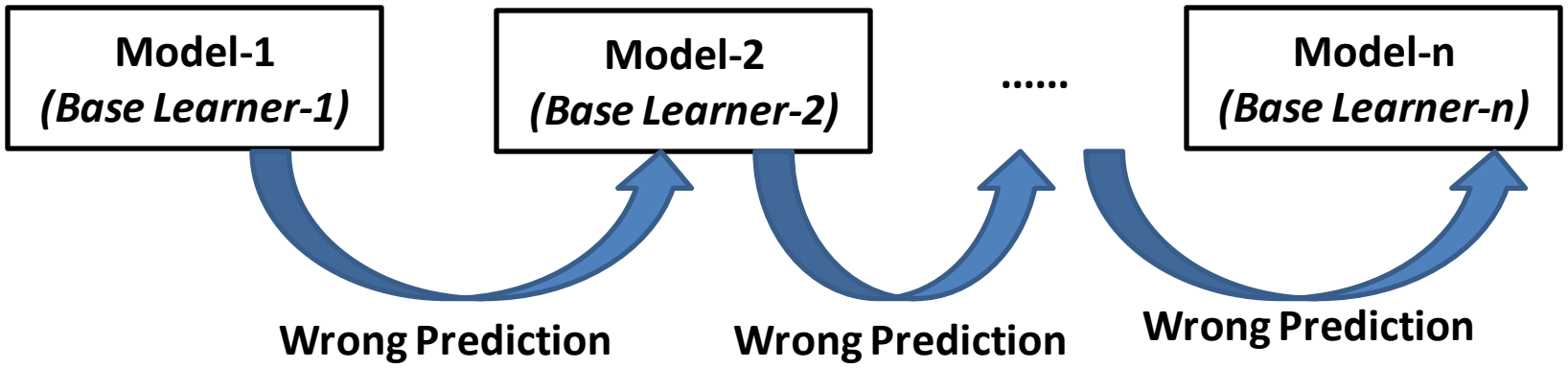
https://scikit-learn.org/stable/modules/ensemble.html

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

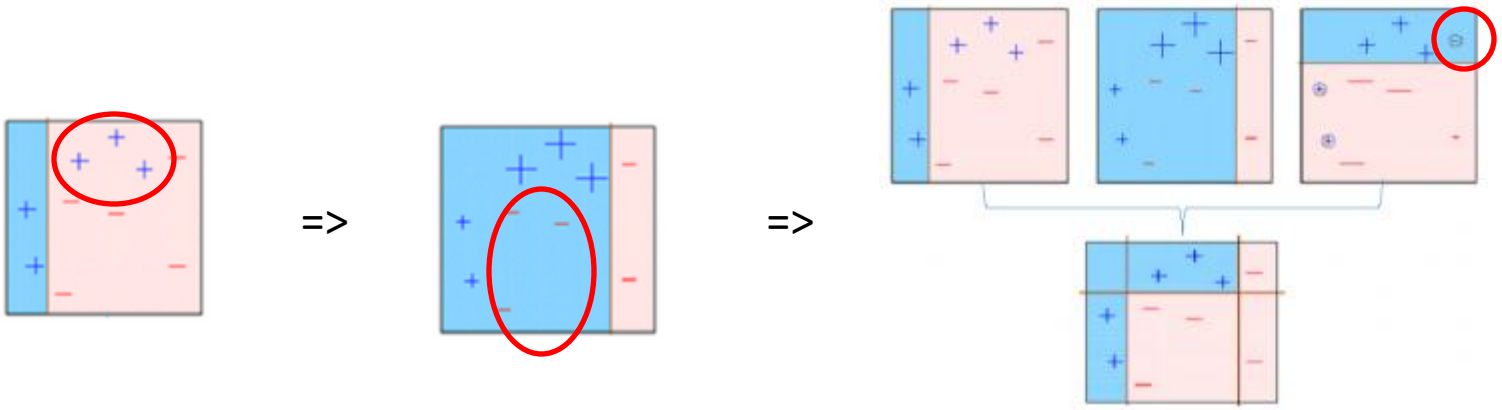
Boosting

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?

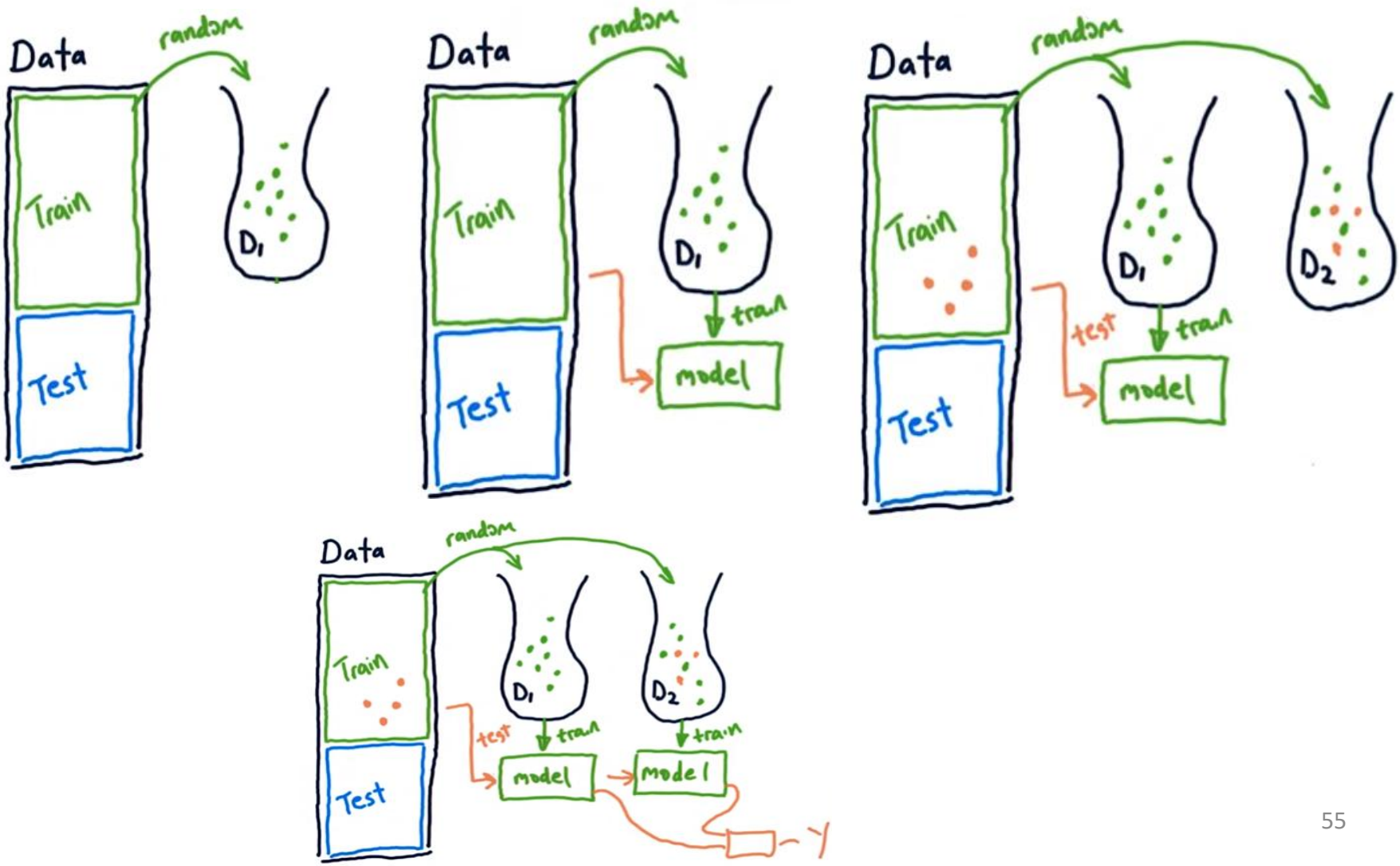


Boosting vs Bagging

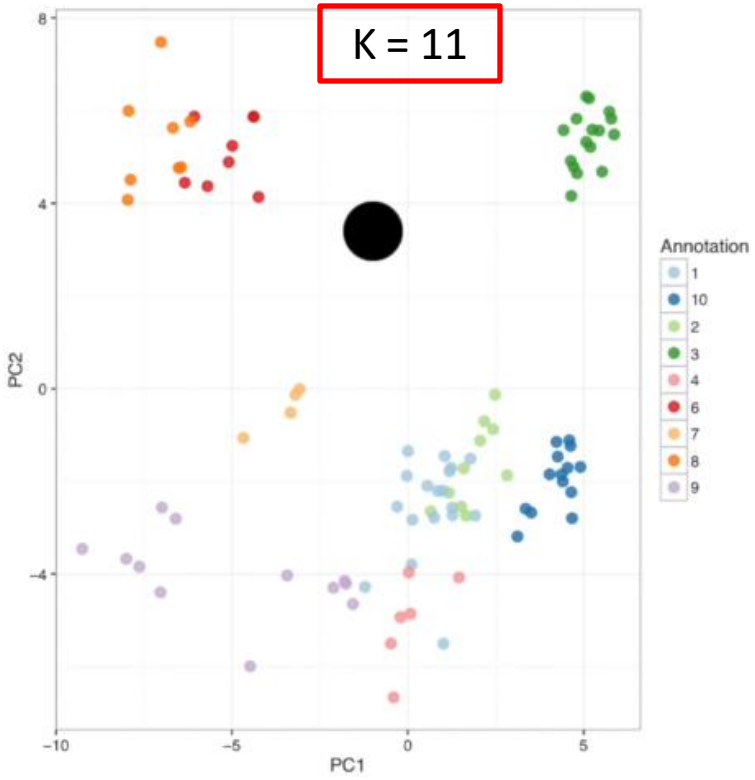
Sl. No.	Boosting	Bagging
1.	Boosting methods are leveraged when low variance and high bias is observed	Bagging methods are typically used on weak learners which exhibit high variance and low bias
2.	• Learn sequentially • Means that a series of models are constructed and with each new model iteration, 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

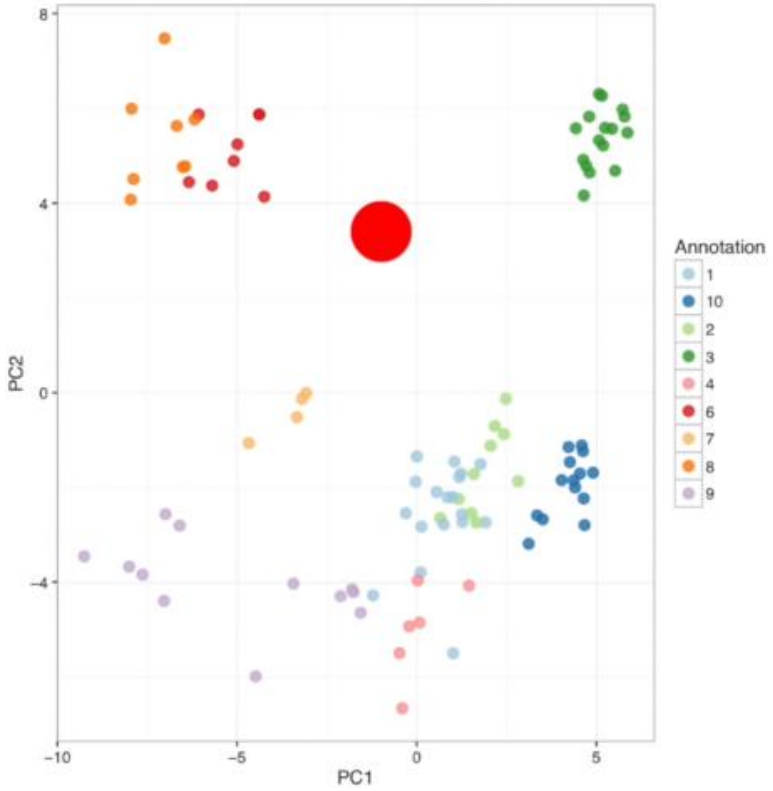
AdaBoost

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 out voted 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
```

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

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

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

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

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?

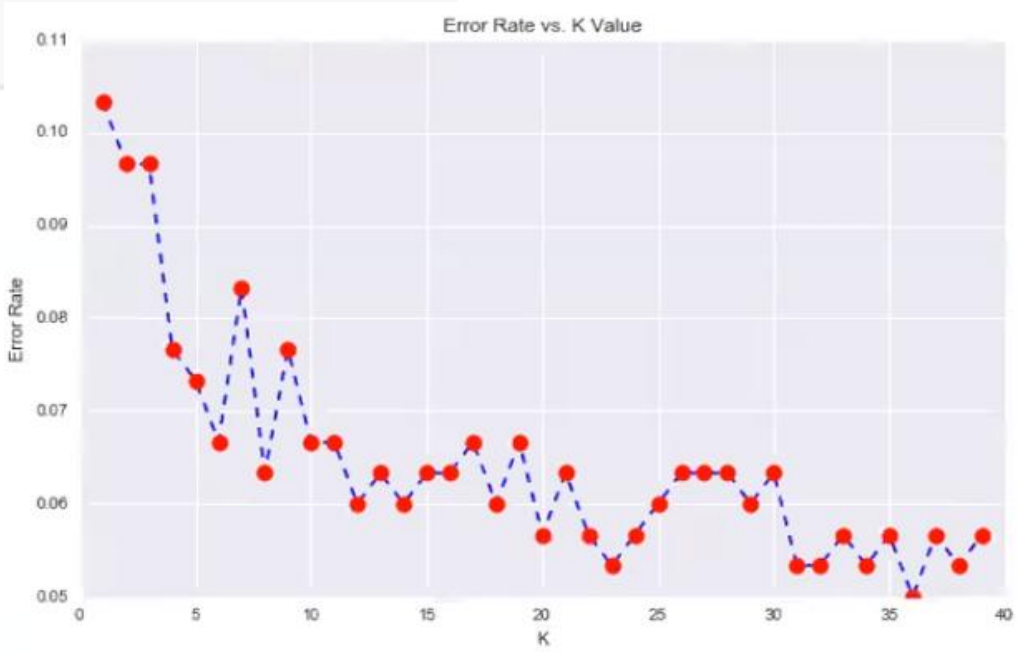
Error Rate = Mean(Predicted_Class != Y_Actual_Class)

```
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]]
```

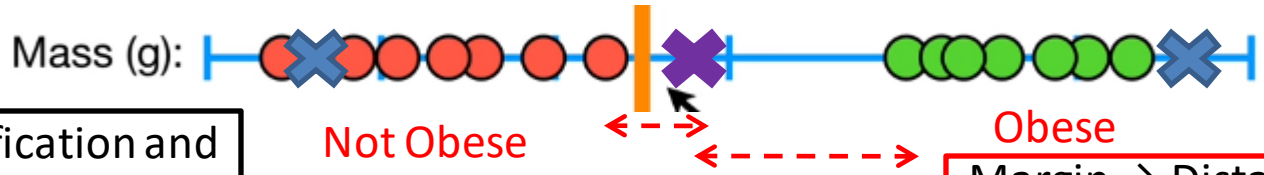
	precision	recall	f1-score	support
0	0.96	0.92	0.94	143
1	0.93	0.97	0.95	157
avg / total	0.95	0.95	0.95	300

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
  - Kernal
    - Polynomial Kernal
    - RDF Kernal
    - Sigmoid Kernal

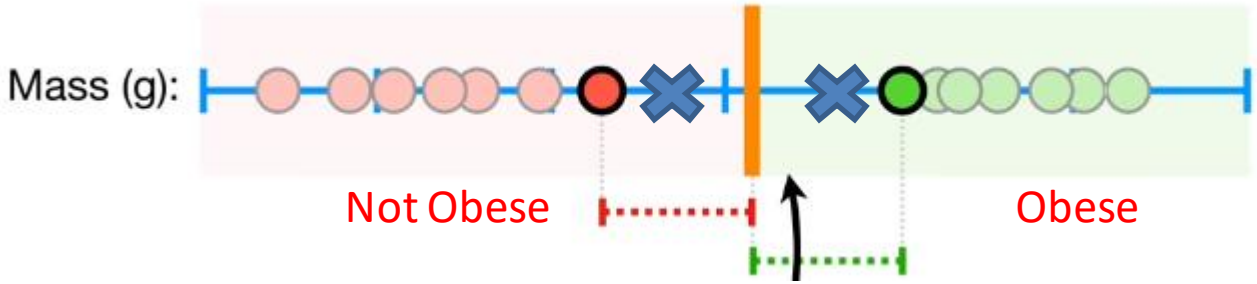
- https://www.javatpoint.com/machine-learning-support-vector-machine-algorithm
- https://en.wikipedia.org/wiki/Support-vector_machine
- https://monkeylearn.com/blog/introduction-to-support-vector-machines-svm/

Support Vector Machine

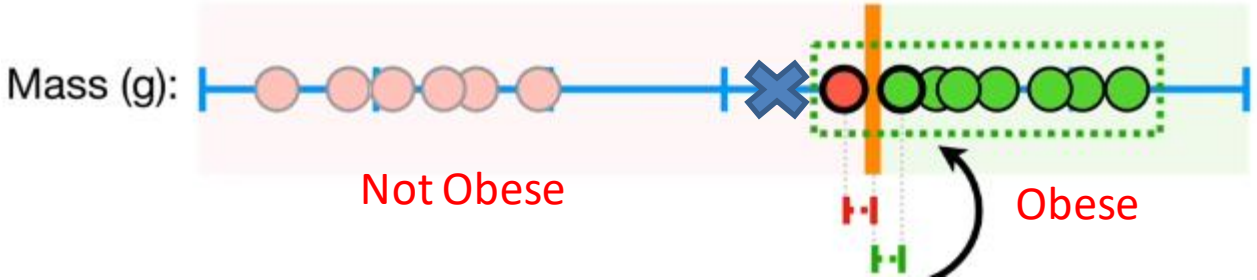


Both classification and Regression problems

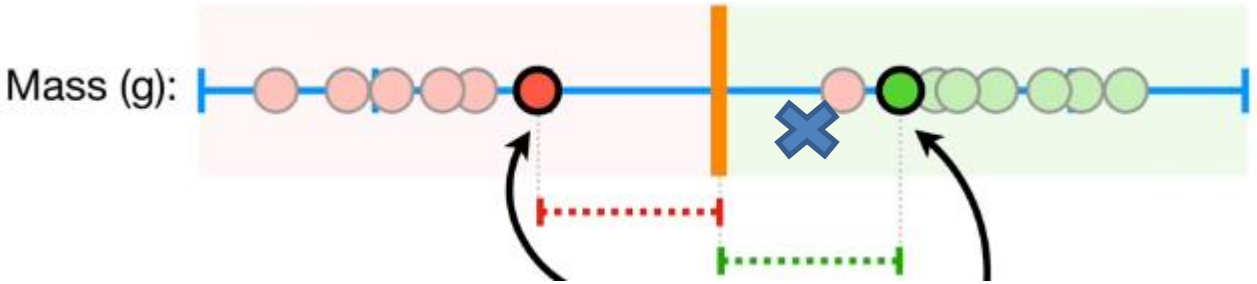
Margin → Distance between the observation and threshold



Maximal Margin Classifier use Maximal Margin



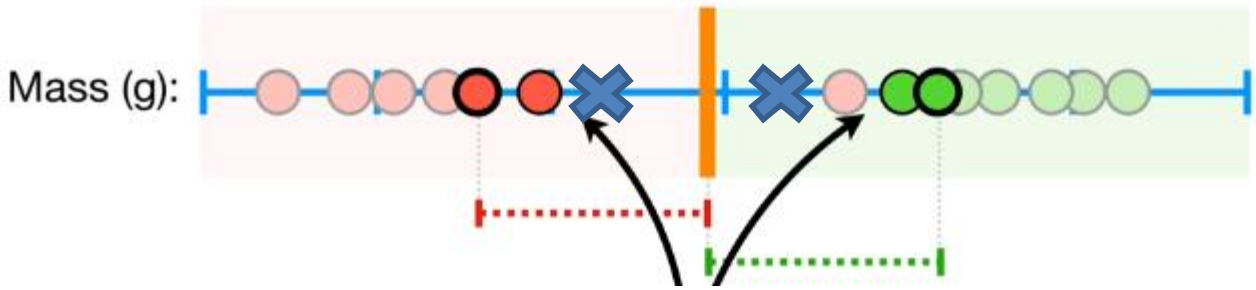
Maximal Margin Classifier → Sensitive to outliers



Soft Margin -> Allow some miscalculations

Bias / Variance Trade-off

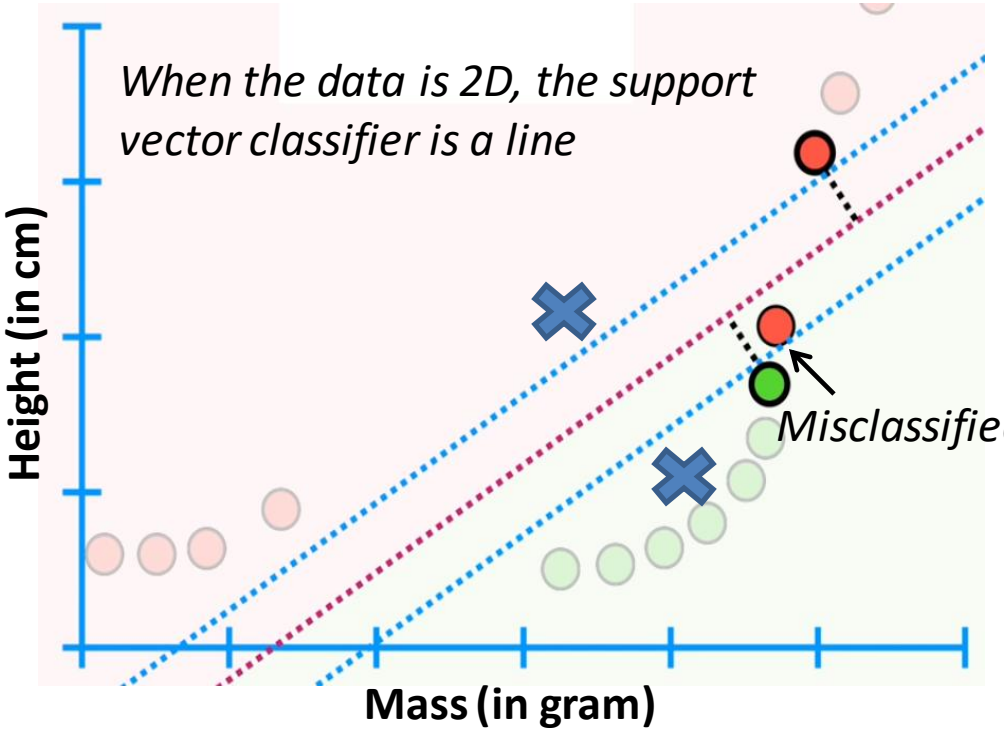
Support Vector Machine

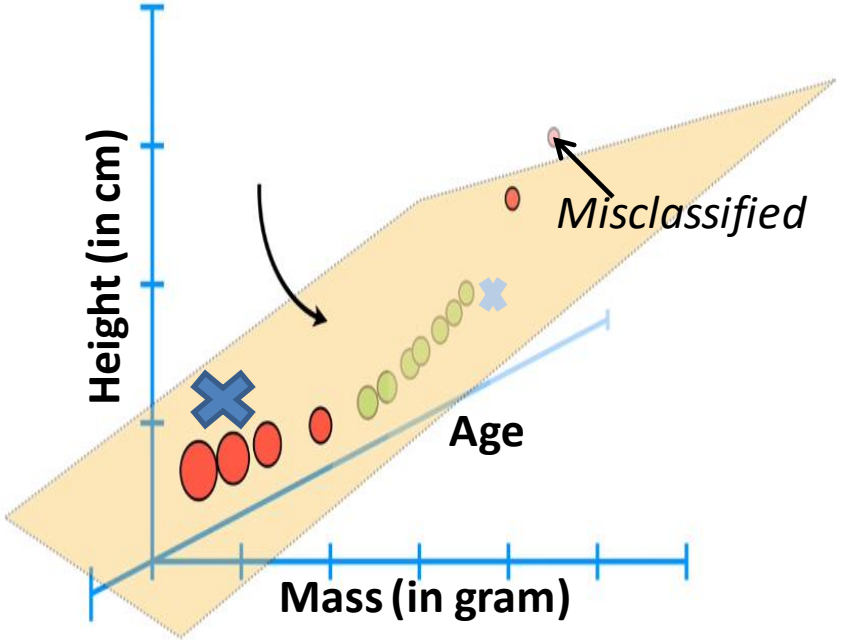


Soft Margin Classifier or Support Vector Classifier

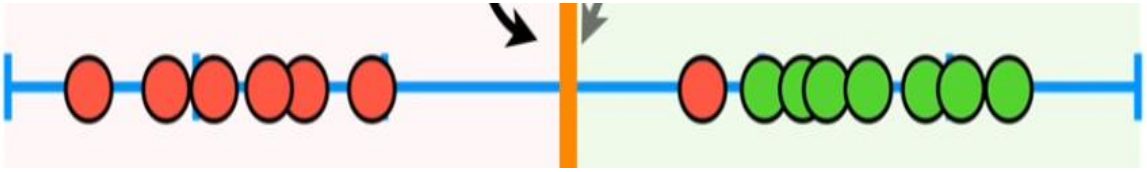
Suppose if we use cross-validation to find the soft margin, then we allow 1 miscalculation and 2 correct observations to be correctly classified within the soft margin

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

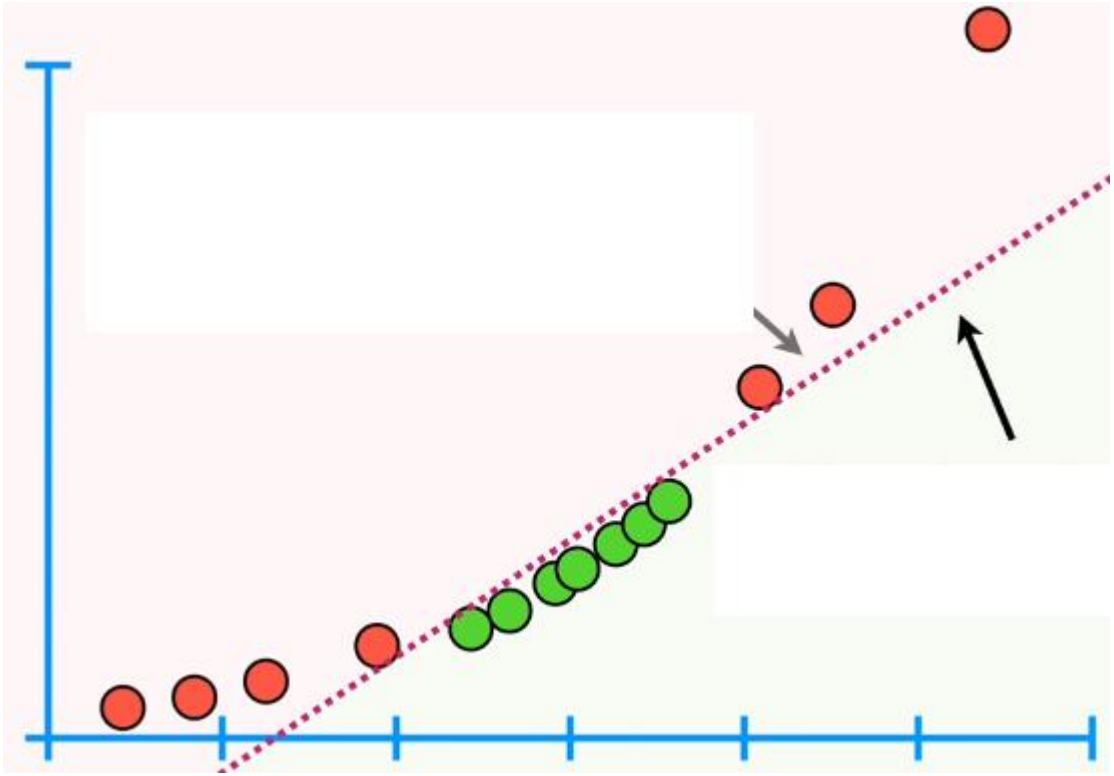




Support Vector Machine

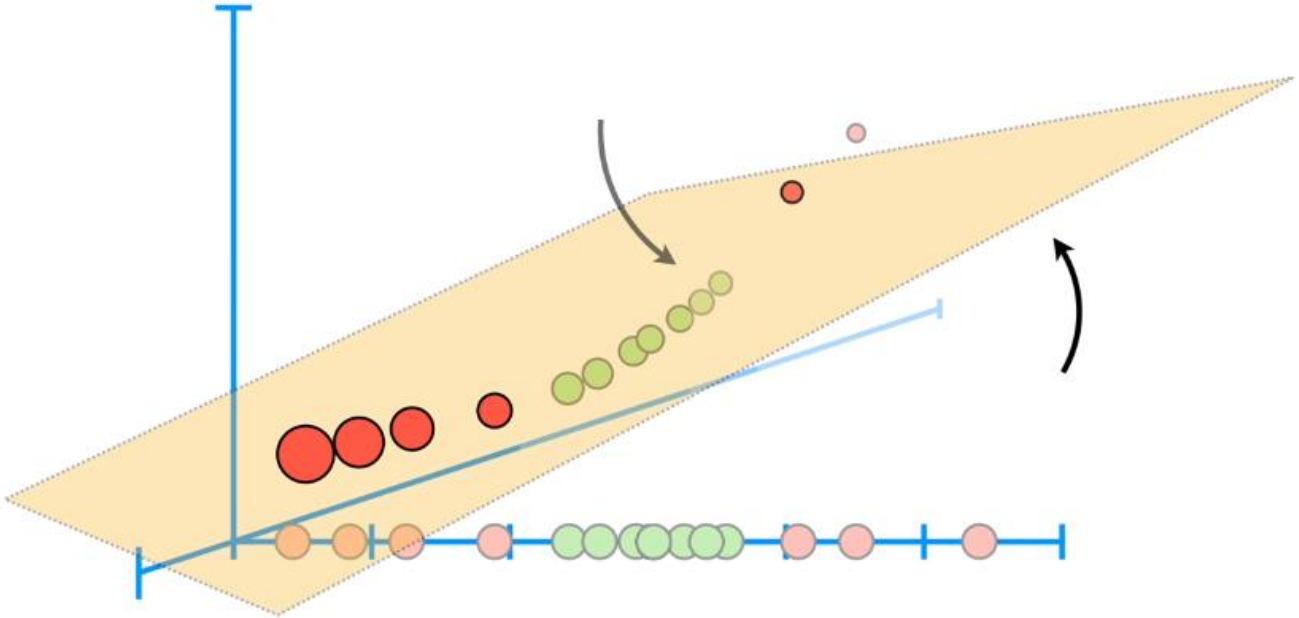


- *If the data is 1D, then the Support Vector Classifier is a single point on a 1D number line*

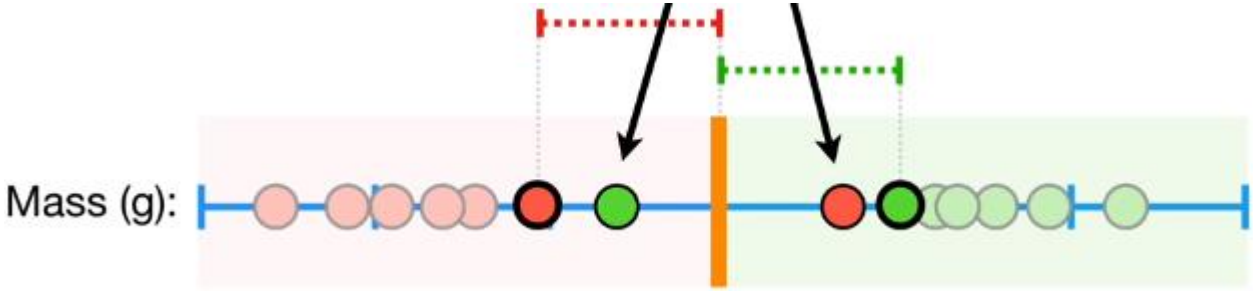


- *If the data is 2D, then the Support Vector Classifier is a line on a 2D space*

Support Vector Machine



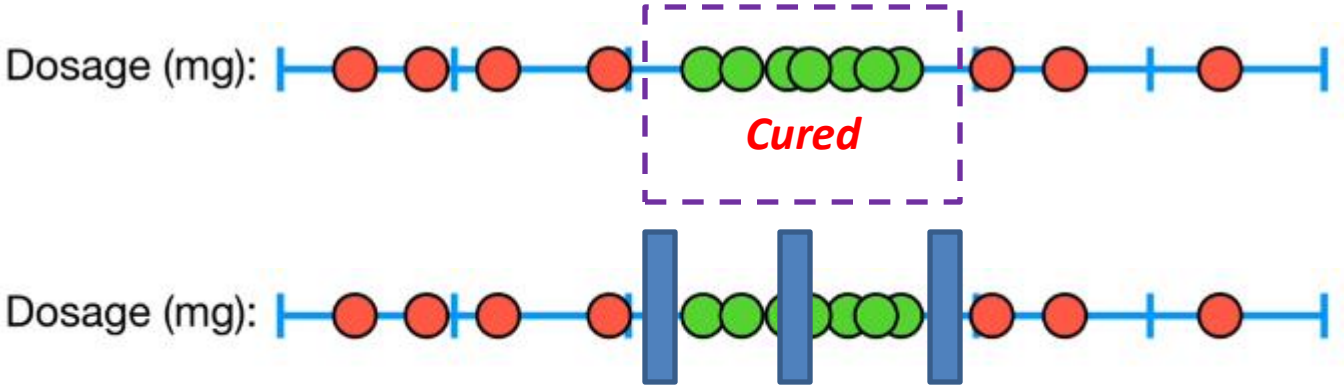
- *If the data is 3D, then the Support Vector Classifier is a plane*
- *If the data is 4D or more, then the Support Vector Classifier is a hyperplane*



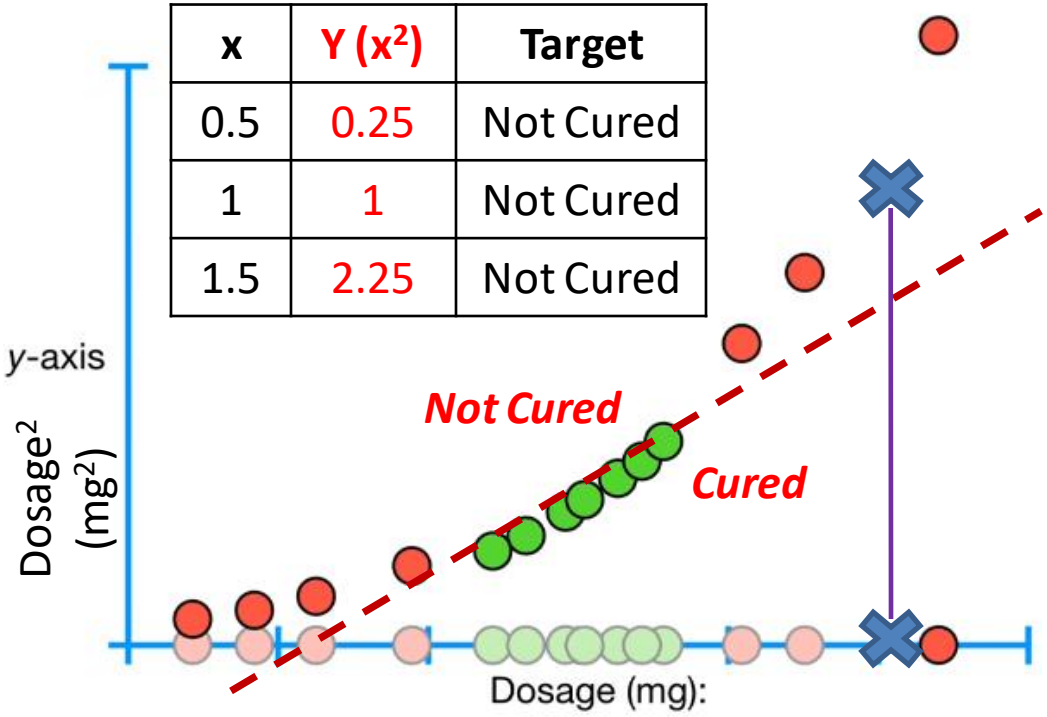
Adv of Support Vector Classifiers:

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

Support Vector Machine

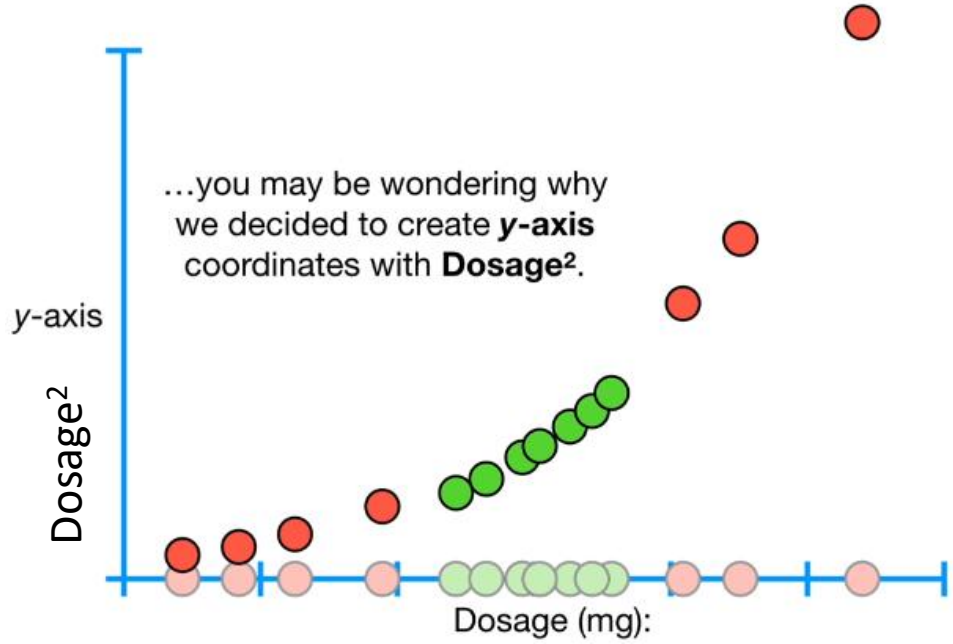


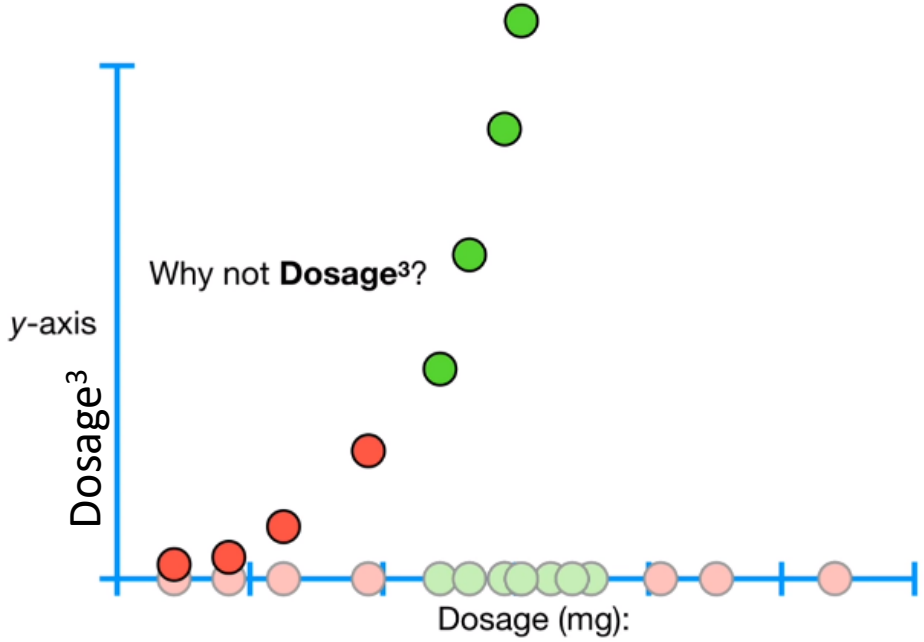
x	Y (x^2)	Target
0.5	0.25	Not Cured
1	1	Not Cured
1.5	2.25	Not Cured

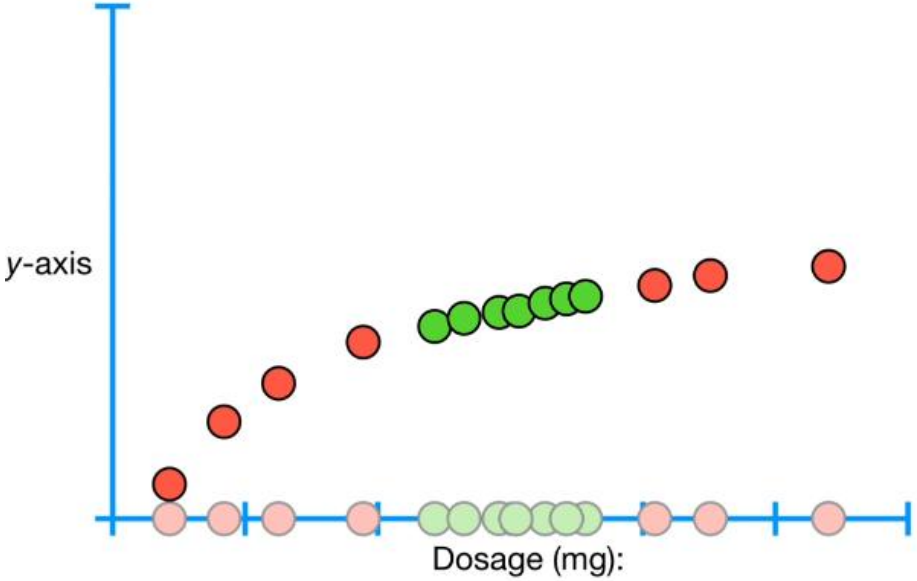


1. Start with data in low dimension
2. Move the data into a higher dimension
3. Find a support vector classifier that separates the data into 2 groups

Support Vector Machine

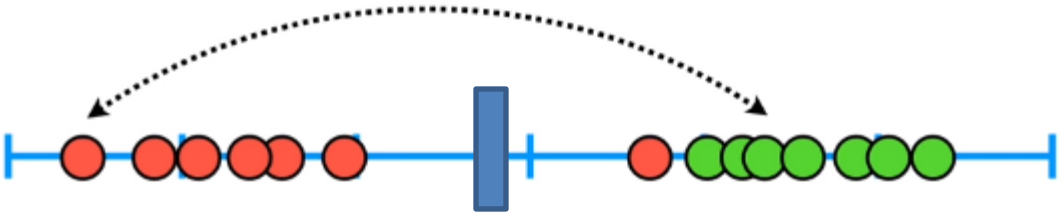




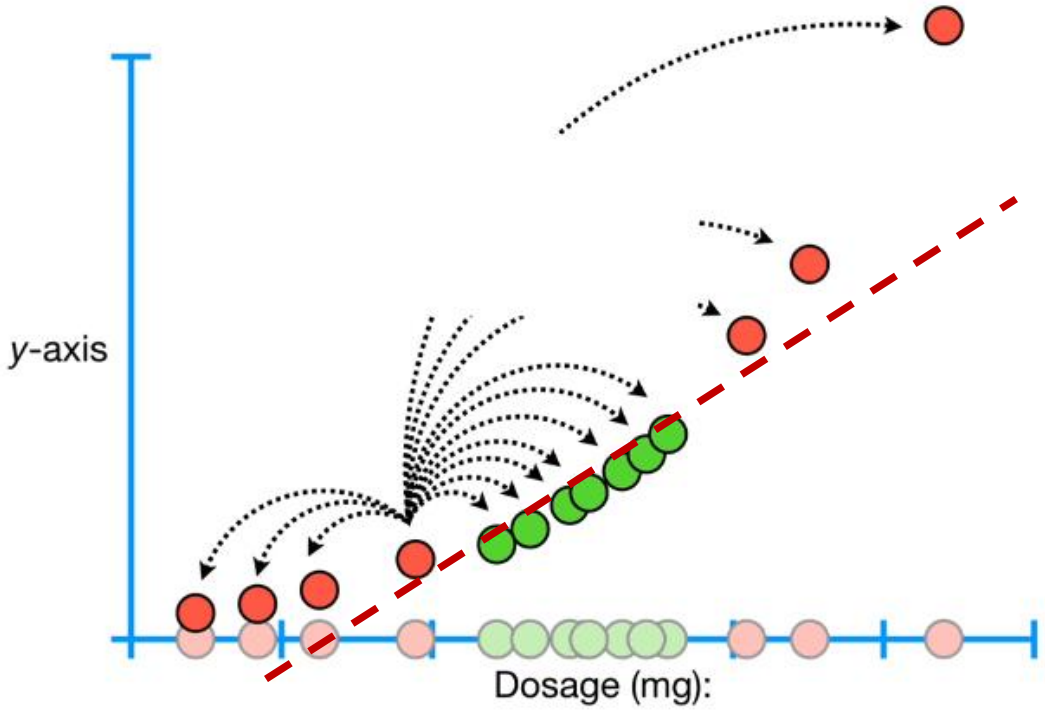


SVM use Kernal Functions to systematically find Support Vector Classifiers in higher dimensions

Support Vector Machine

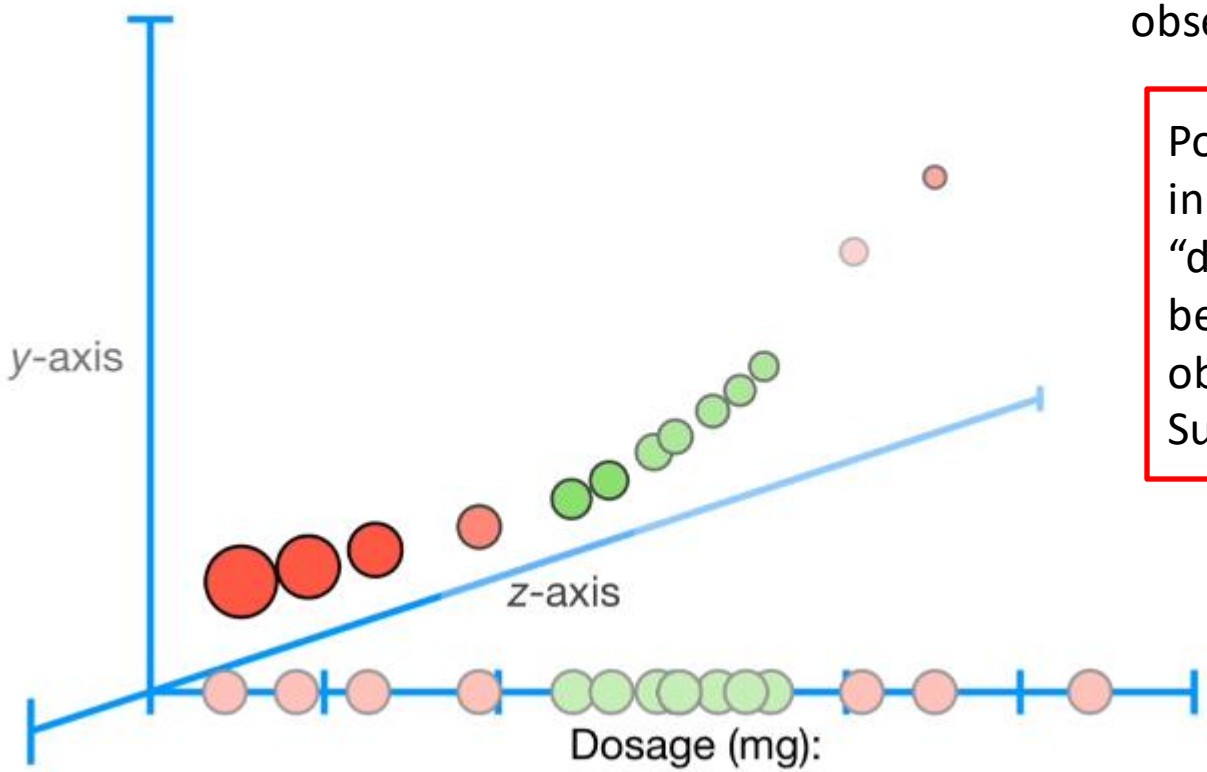


- Polynomial Kernal → Parameter "d" which stands for the degree of the polynomial
- When d = 1, it computes the relationship between each pair of observations in 1D



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

Support Vector Machine



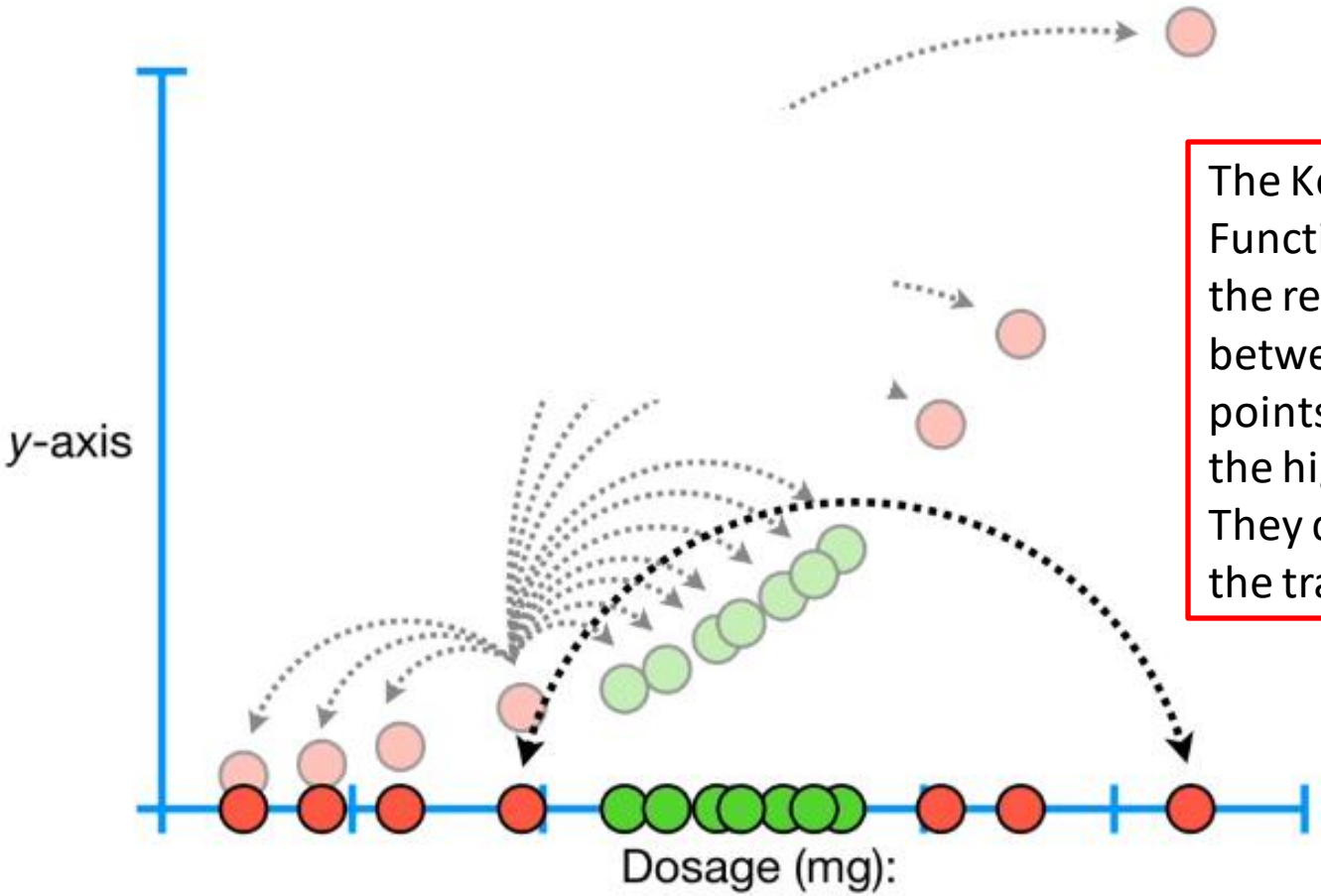
- When d = 3, it computes the relationship between each pair of observations in 3D

Polynomial Kernal 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 Kernal → Infinite Dimensions

Support Vector Machine



The Kernal Trick → Kernal Functions only calculate the relationships between every pair of points as if they are in the higher dimensions. They don't actually do the transformation

Clustering Algorithm

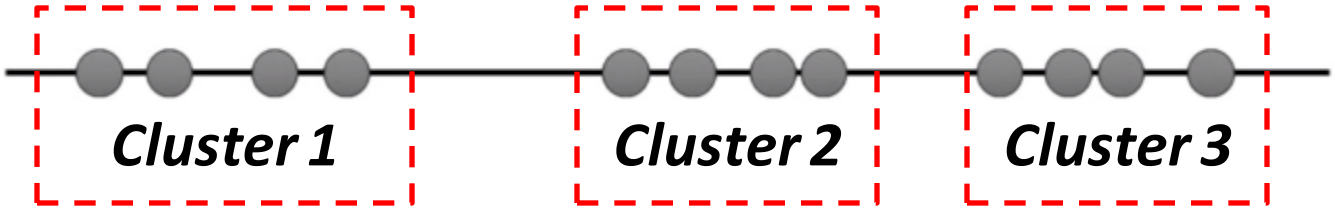
Unsupervised Learning Algorithms

- 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

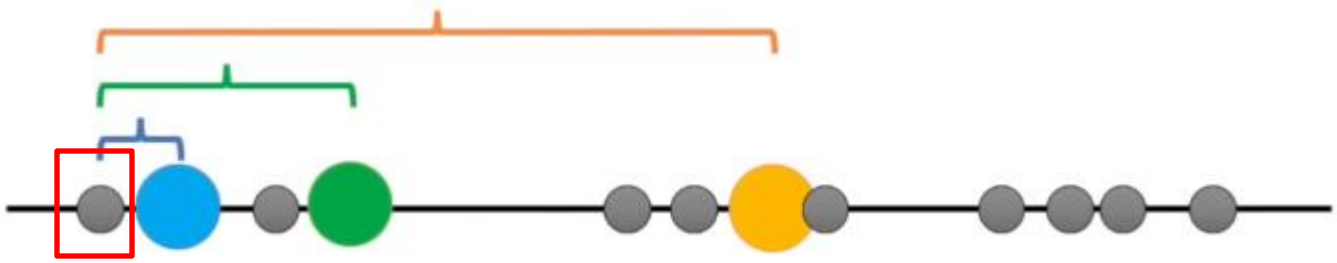
K-Means Clustering

K -> No. of Clusters in your data



Initial Observation

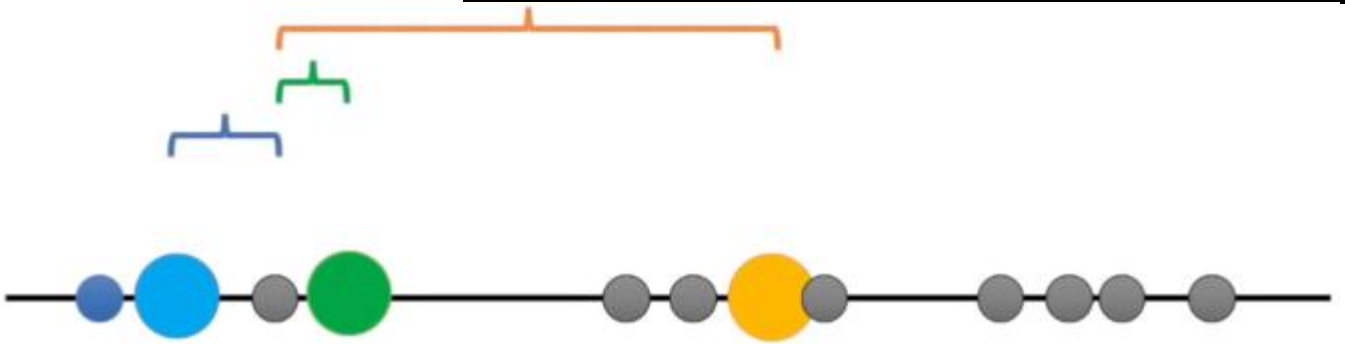
Randomly choose the points equal to the value of “k”



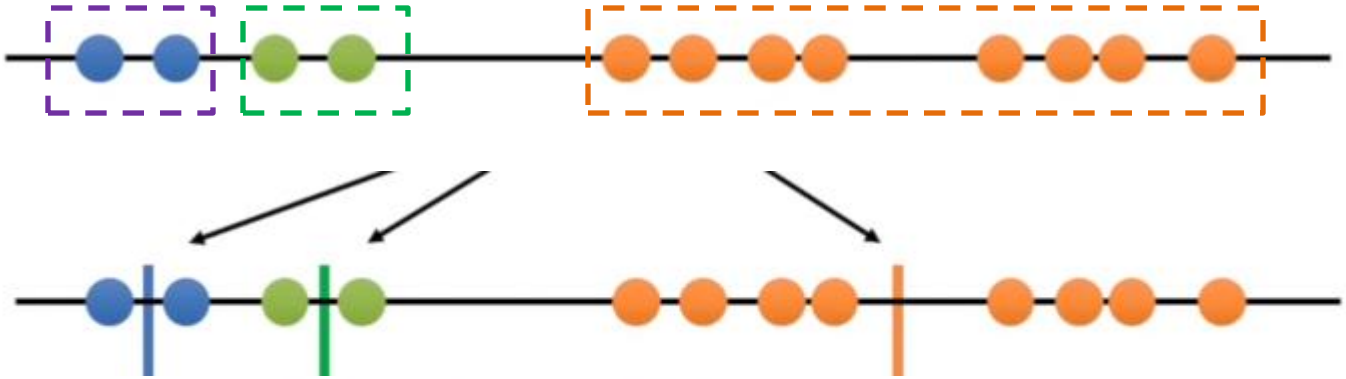
Find the distance between the first point and the three chosen points

The first point belongs to the closest one

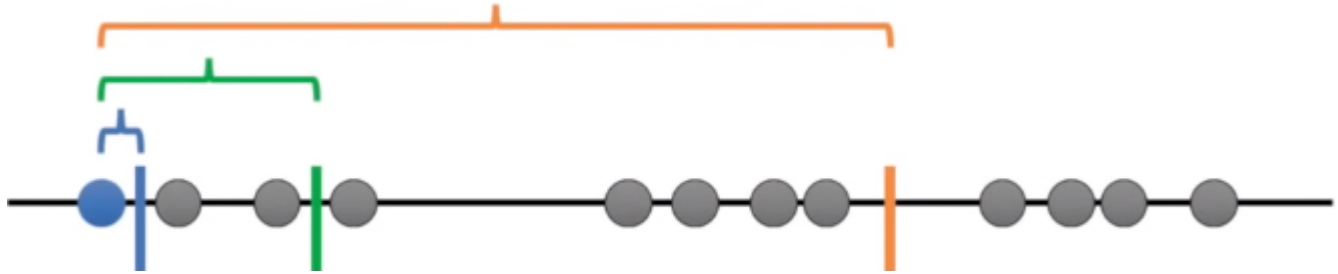
K-Means Clustering



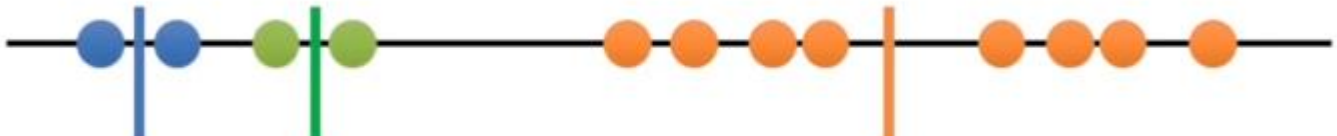
Find the distance between the second point and the three chosen points



Calculate the mean for each cluster

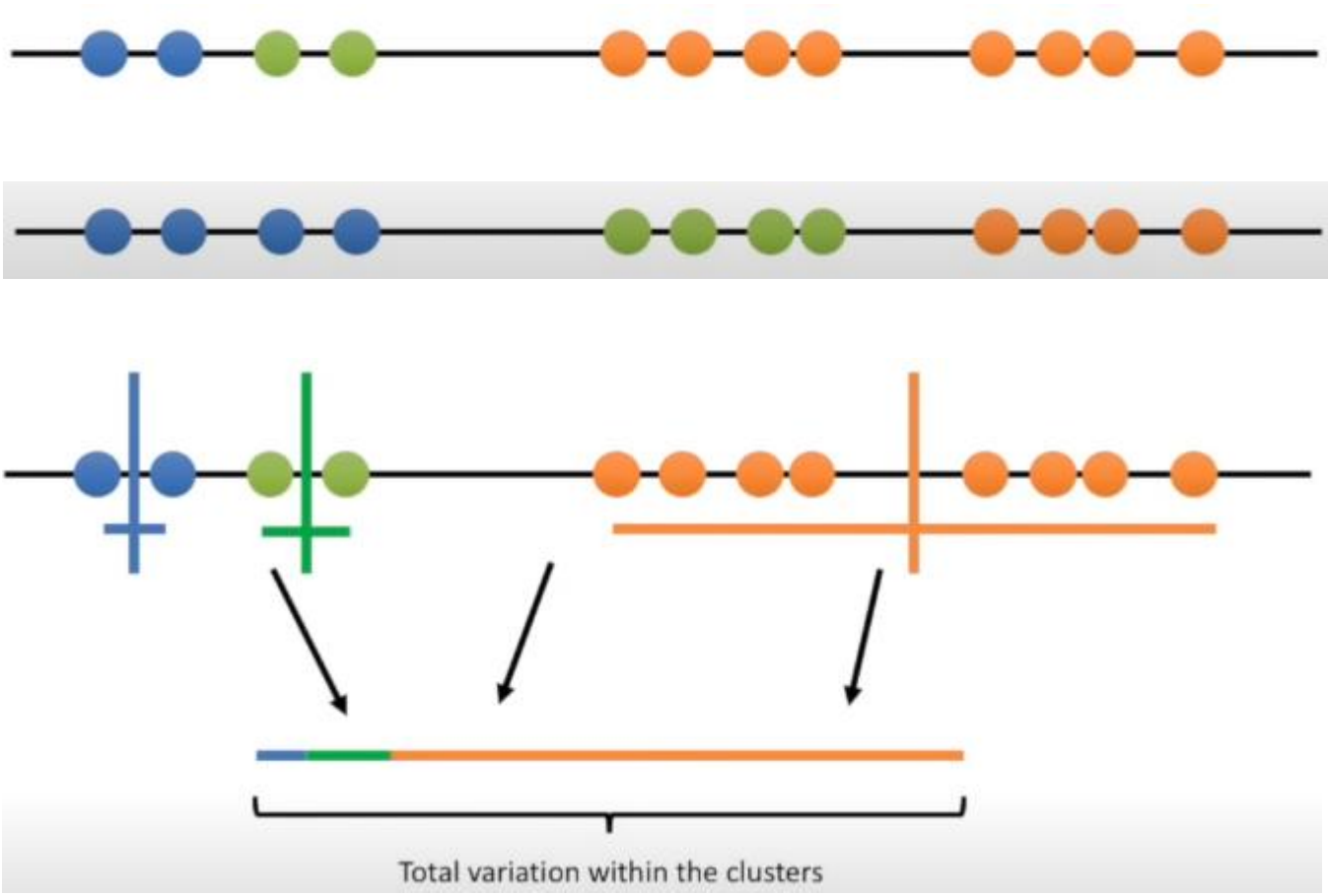


Repeat the process with the identified mean



Since there is no change, we are done

K-Means Clustering



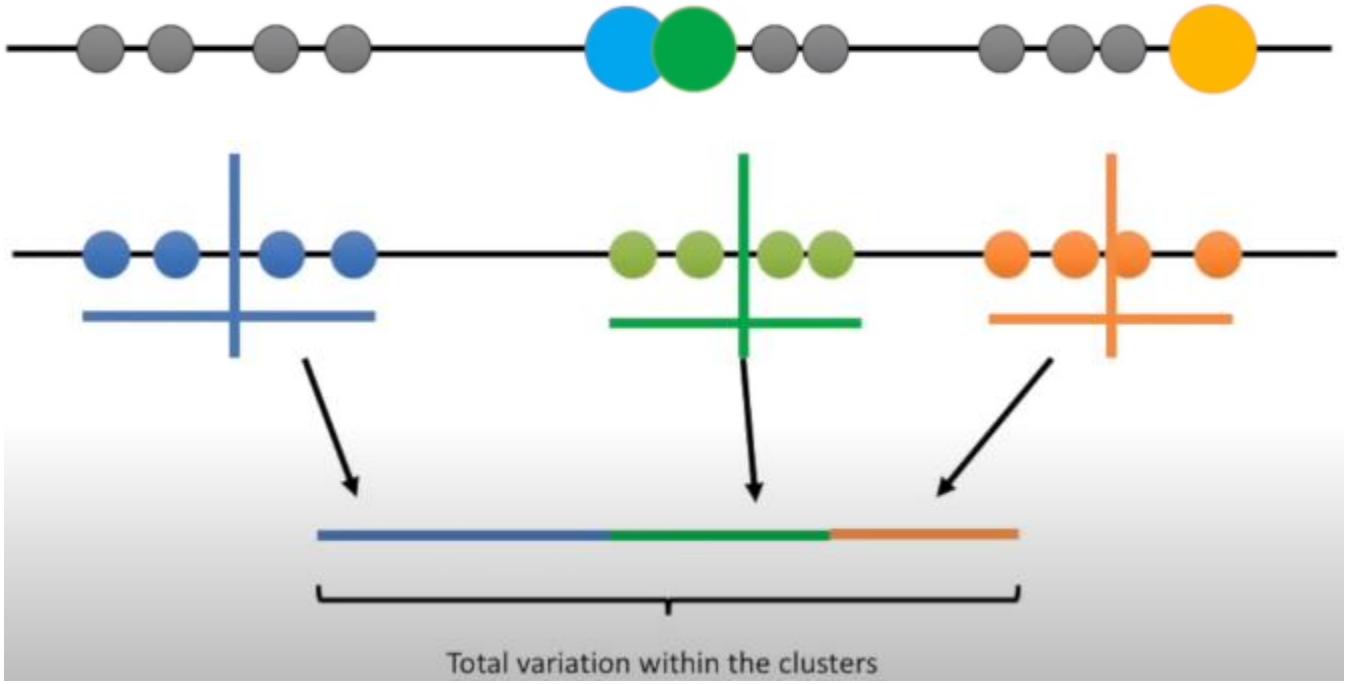
Calculated

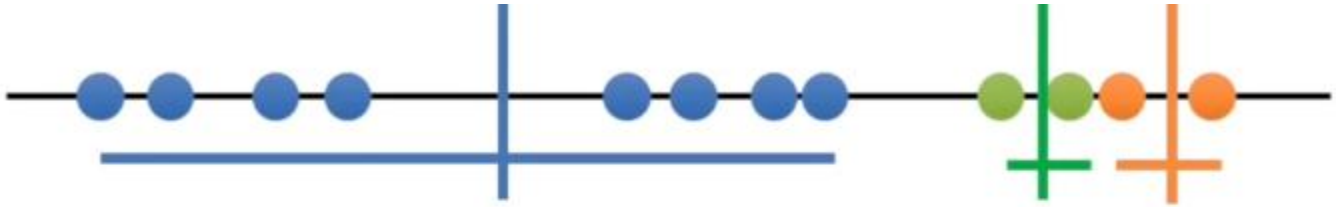
Initial Observation

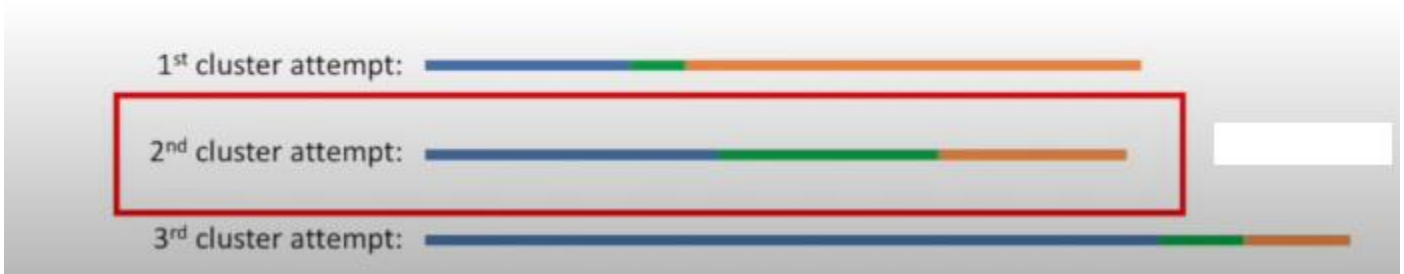
Assess the quality of clustering by calculating the variation within each cluster

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

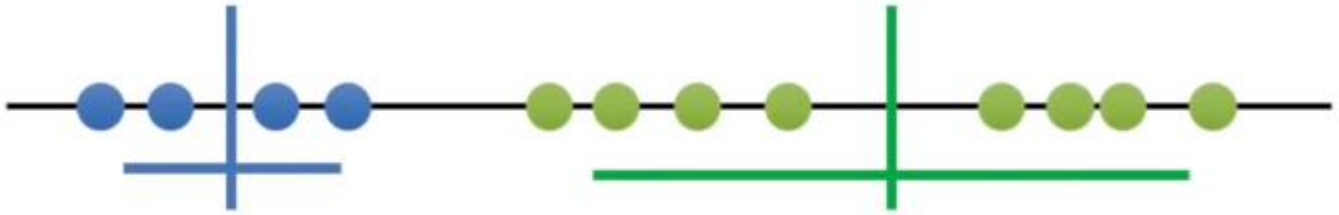
K-Means Clustering



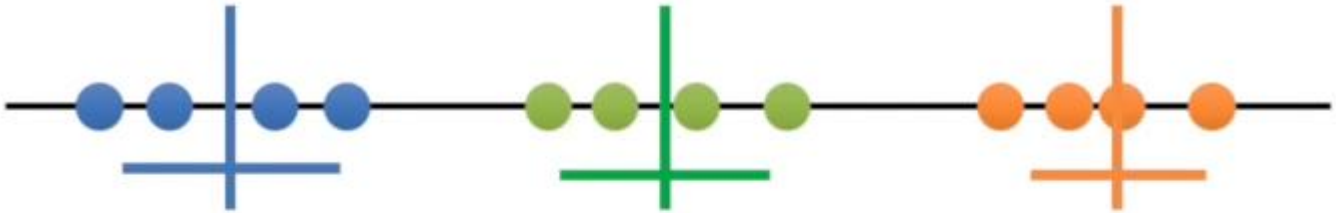




K-Means Clustering *(Choosing k)*

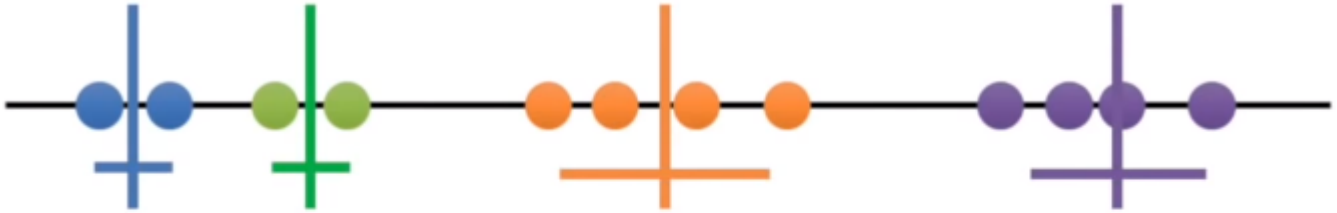


K = 2 is better, and we can quantify how much better by comparing the total variation within the 2 clusters to K = 1



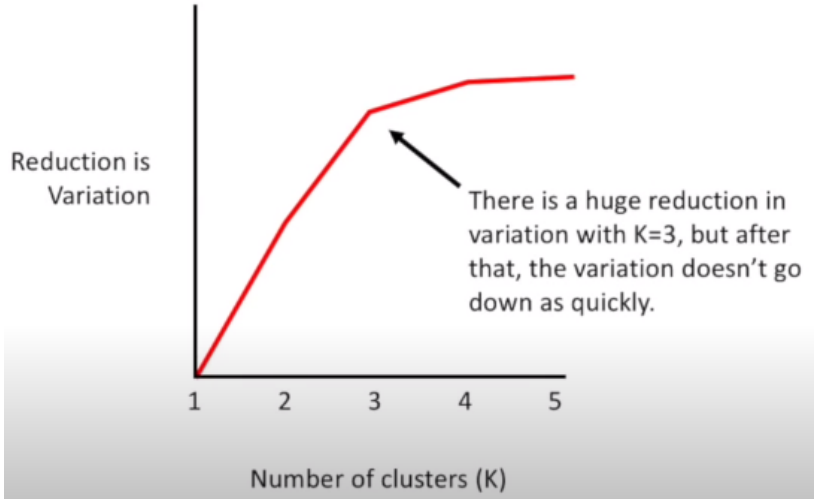
K = 3 is even better! We can quantify how much better by comparing the total variation within the 3 clusters to K = 2

K-Means Clustering *(choosing k)*



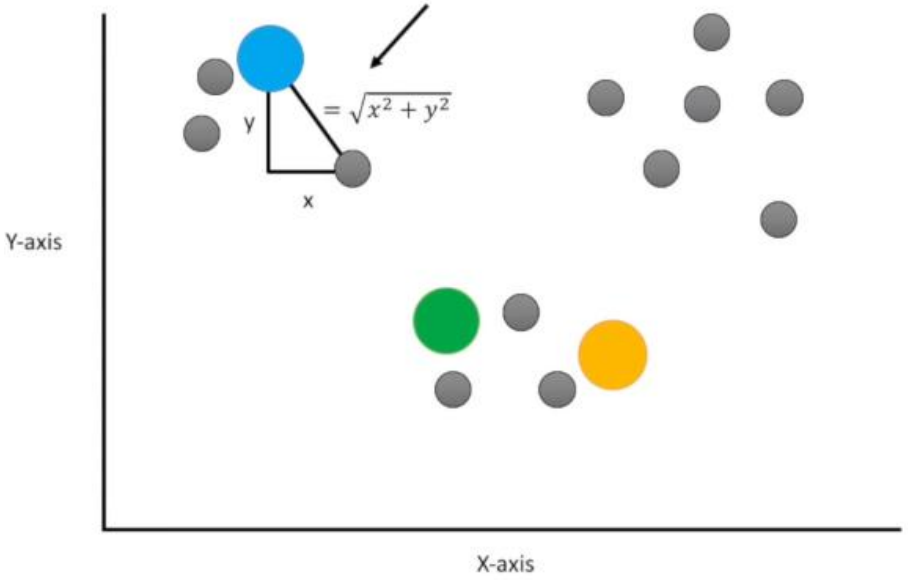
Total variance of k = 4 is lesser when compared with the total variance when k = 3

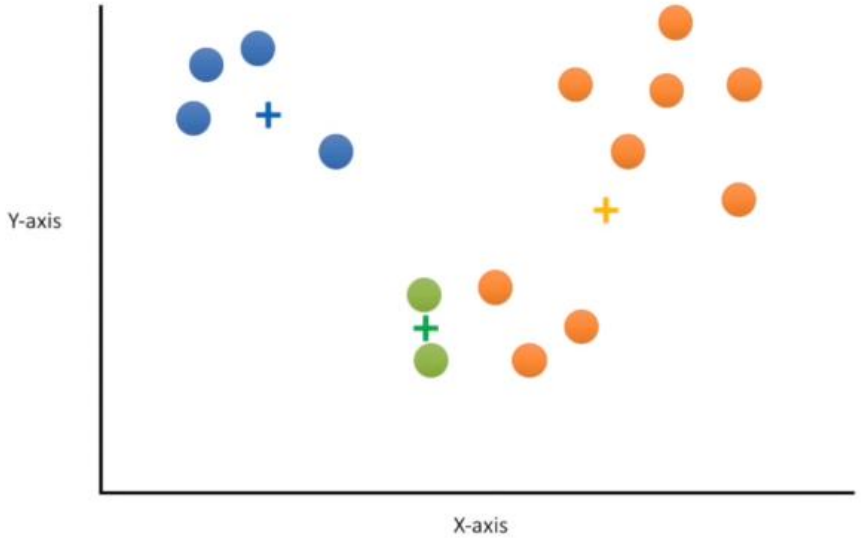
Each time we add a new cluster, the total variation within each cluster is smaller than before. And, when there is only one point per cluster, the variation = 0

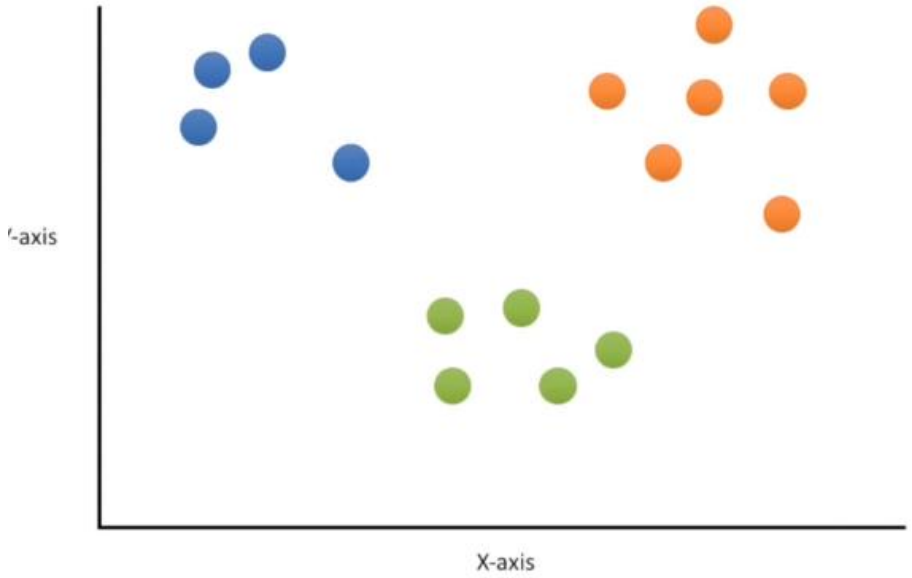


This is called an "Elbow Plot". You can pick "K" by finding the "elbow" in the plot

K-Means Clustering





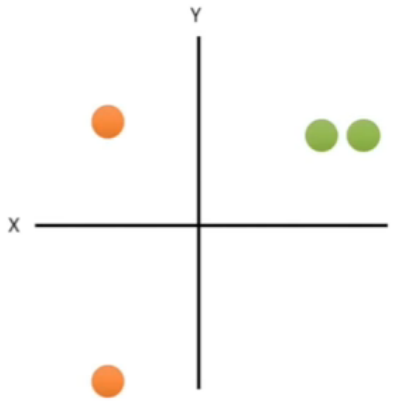


K-Means Clustering

	Samples: #1	#2
Gene 1	12	6
Gene 2	-7	-13
Gene 3	8	6
Gene 4	7	-8

→

	Axis: X	Y
	12	6
	-7	-13
	8	6
	7	-8



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

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

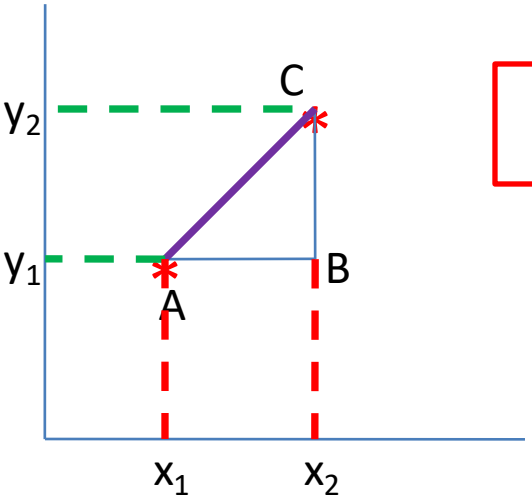
When we have 3 samples, or 3 axes, the Euclidean distance is:

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

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

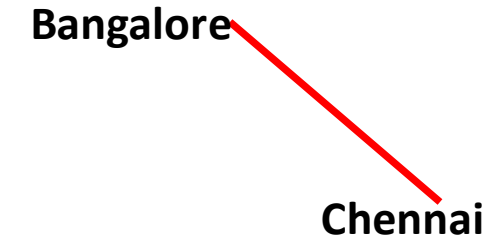
$$\sqrt{x^2 + y^2 + z^2 + a^2}$$

Euclidean Distance vs Manhattan Distance



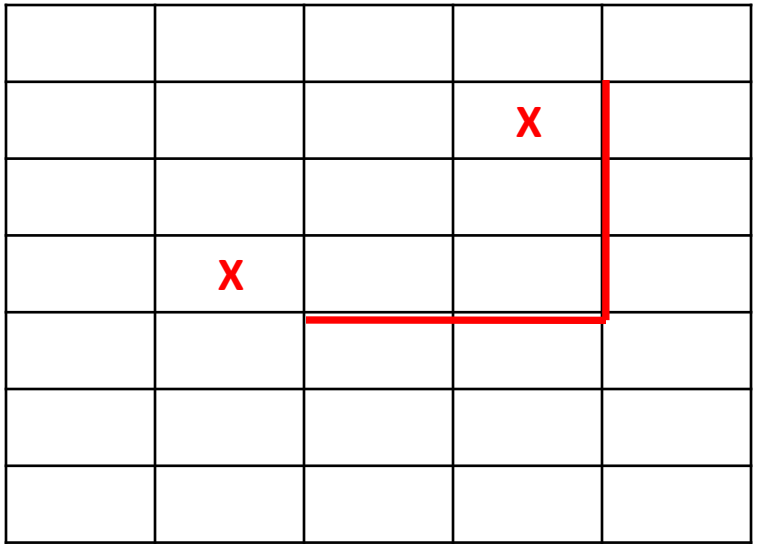
Euclidean Distance

$$AC^2 = AB^2 + BC^2$$
$$= \text{sqrt}((x_2 - x_1)^2 + (y_2 - y_1)^2$$



Manhattan Distance

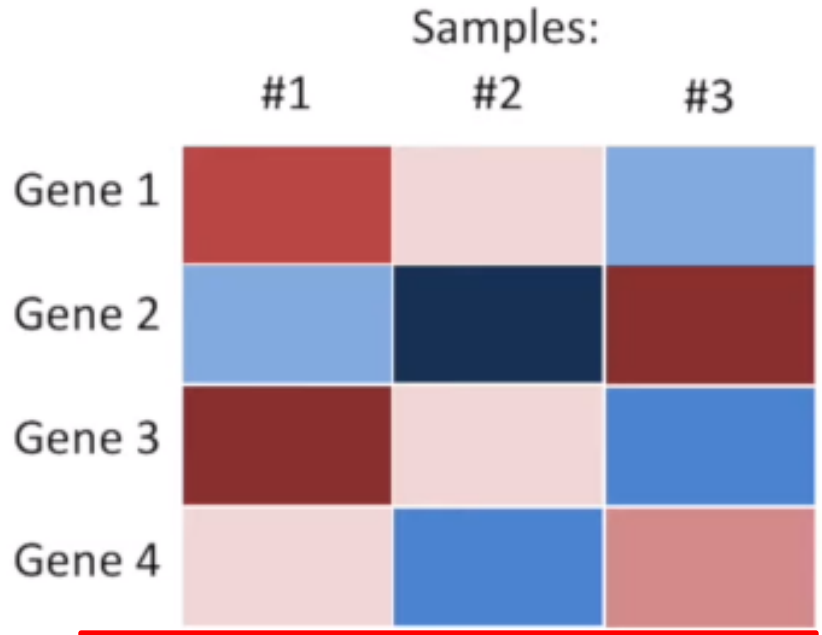
$$= |(x_2 - x_1)| + |(y_2 - y_1)|$$



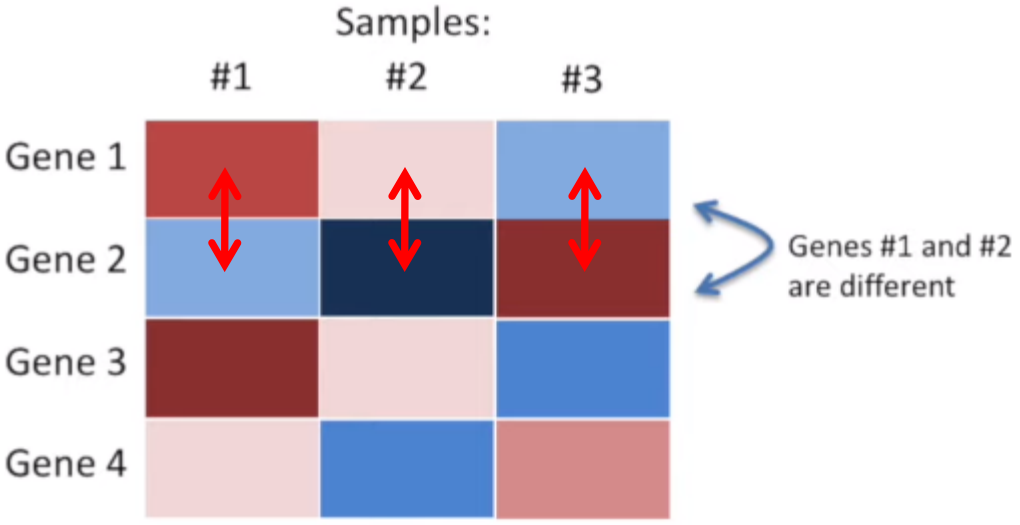
Hierarchical Clustering

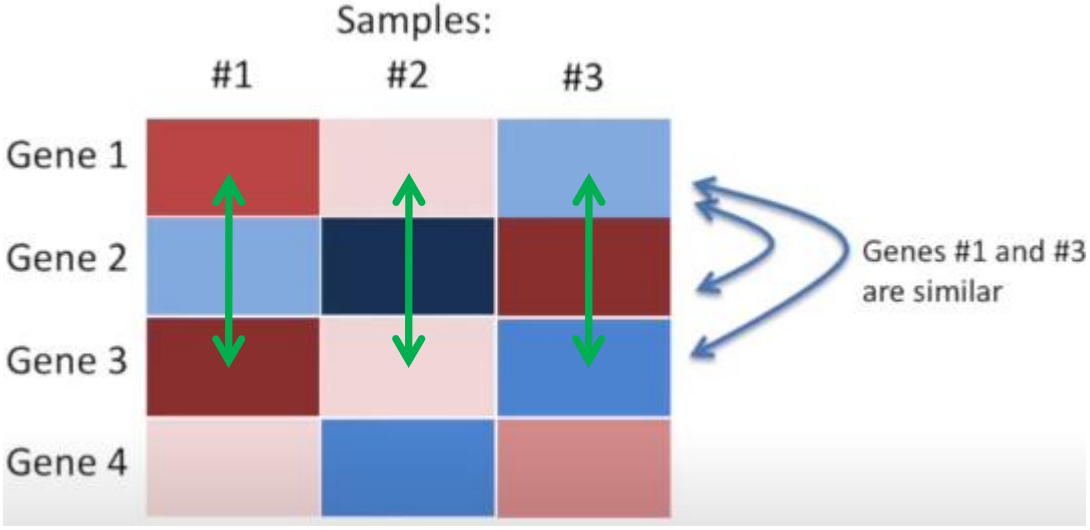
- Hierarchical clustering orders the rows and/or the columns based on similarity
- Makes it easy to see correlations in the data
- Types
  - Divisive (Top-Down)
  - Agglomerative (Bottom-Up)

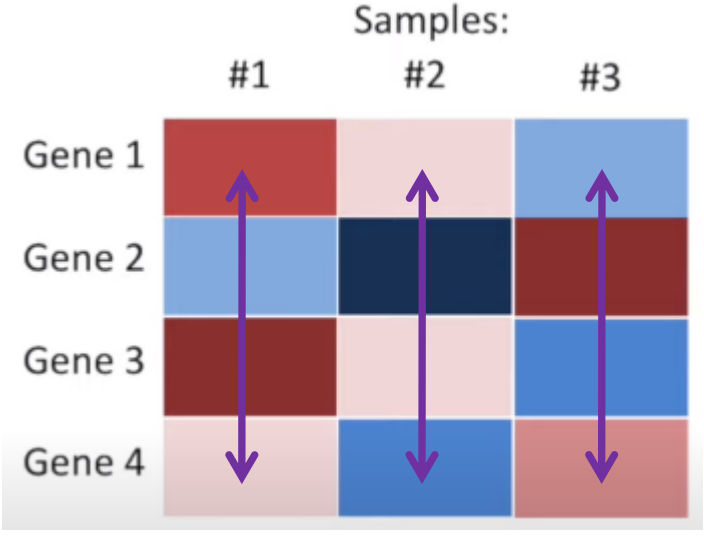
Hierarchical Clustering



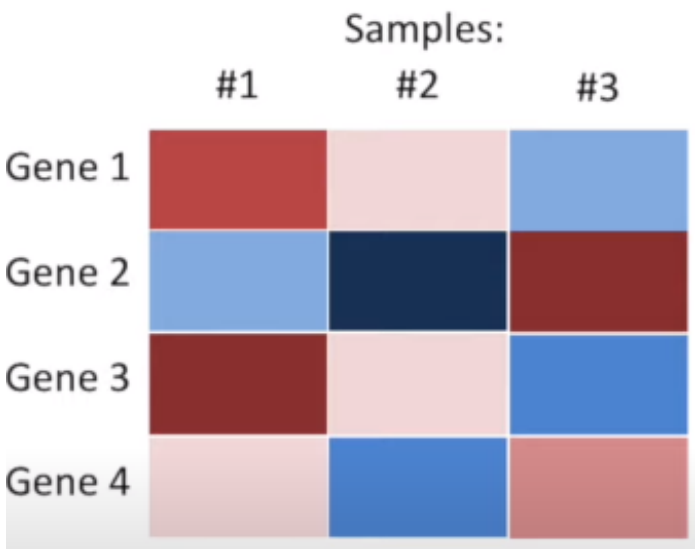
Cluster (Reorder) the rows (Genes)

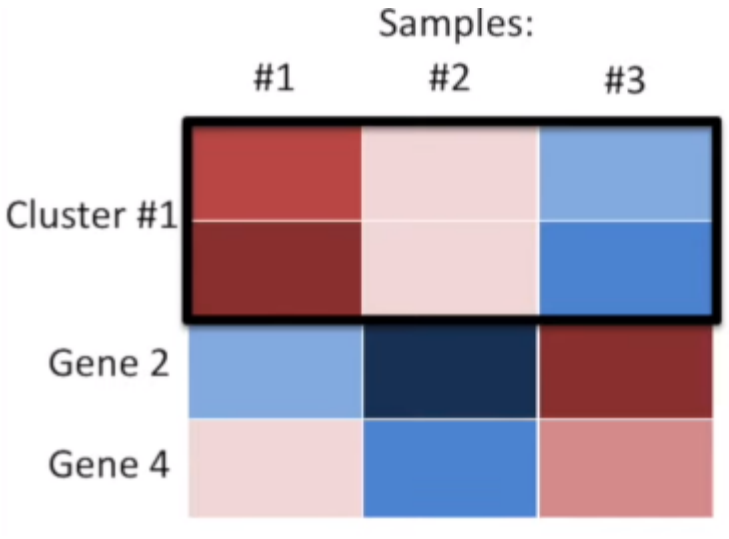




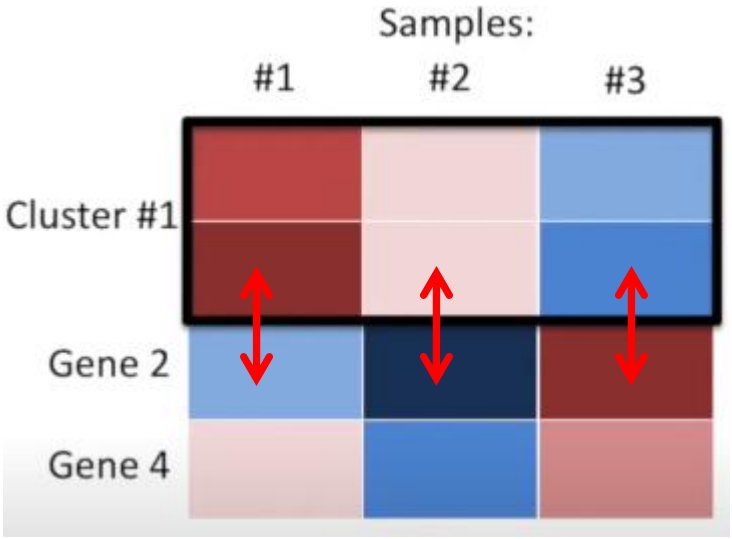


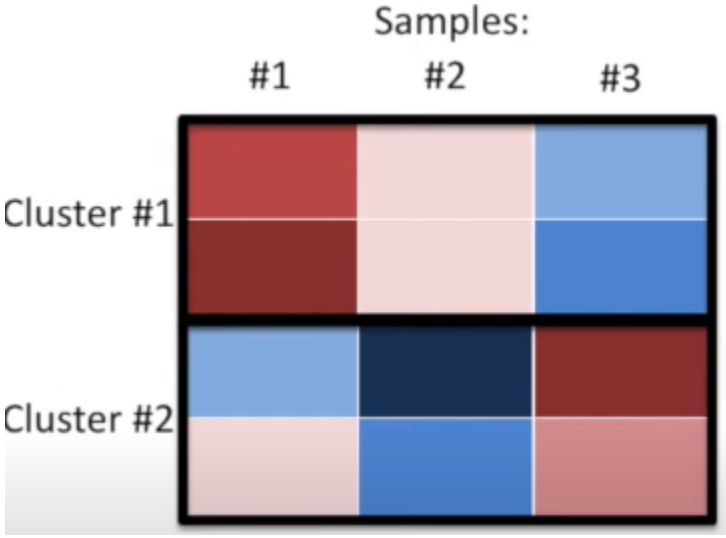
Hierarchical Clustering



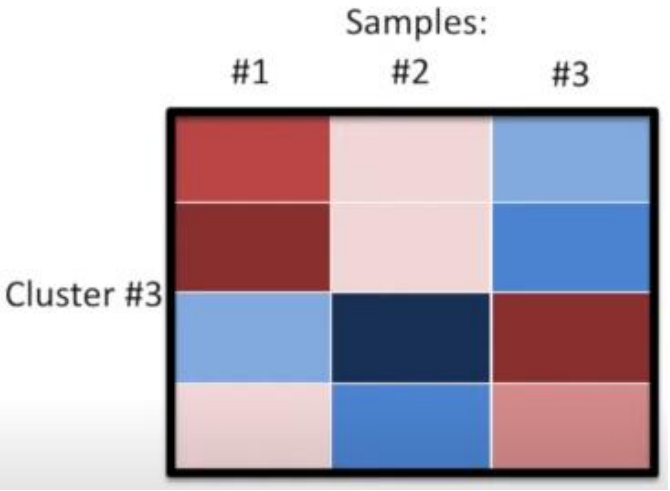


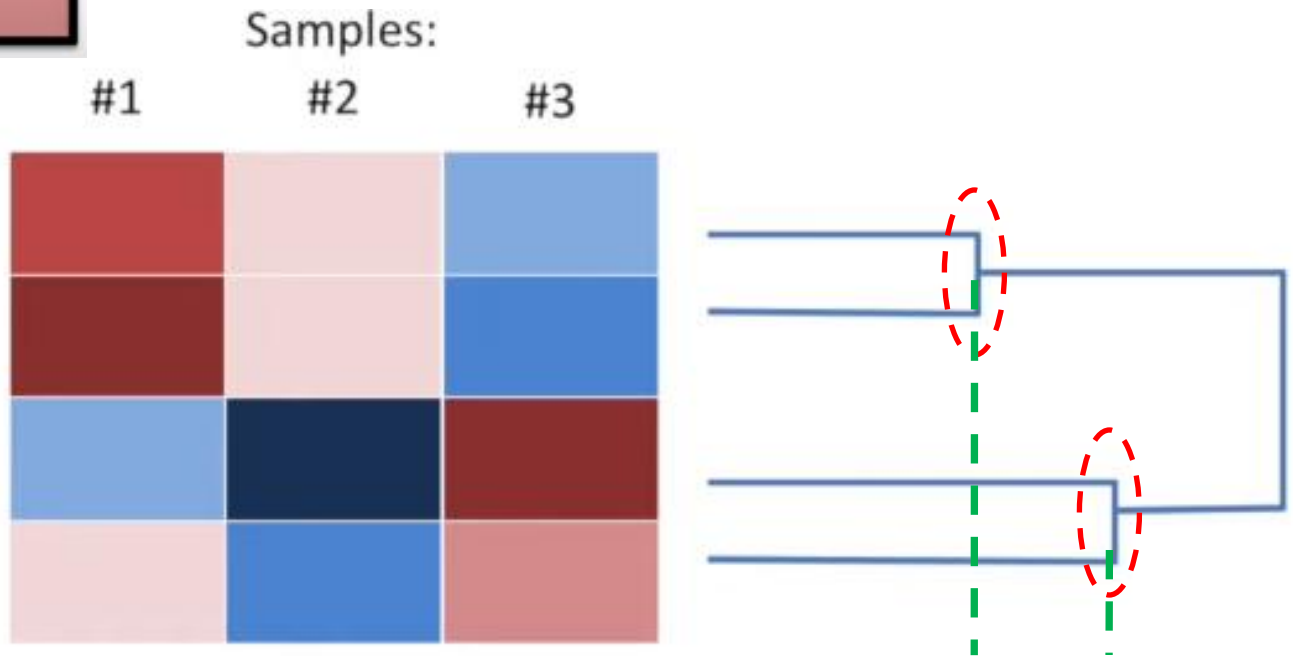
Genes #1 and #3 are now cluster #1.





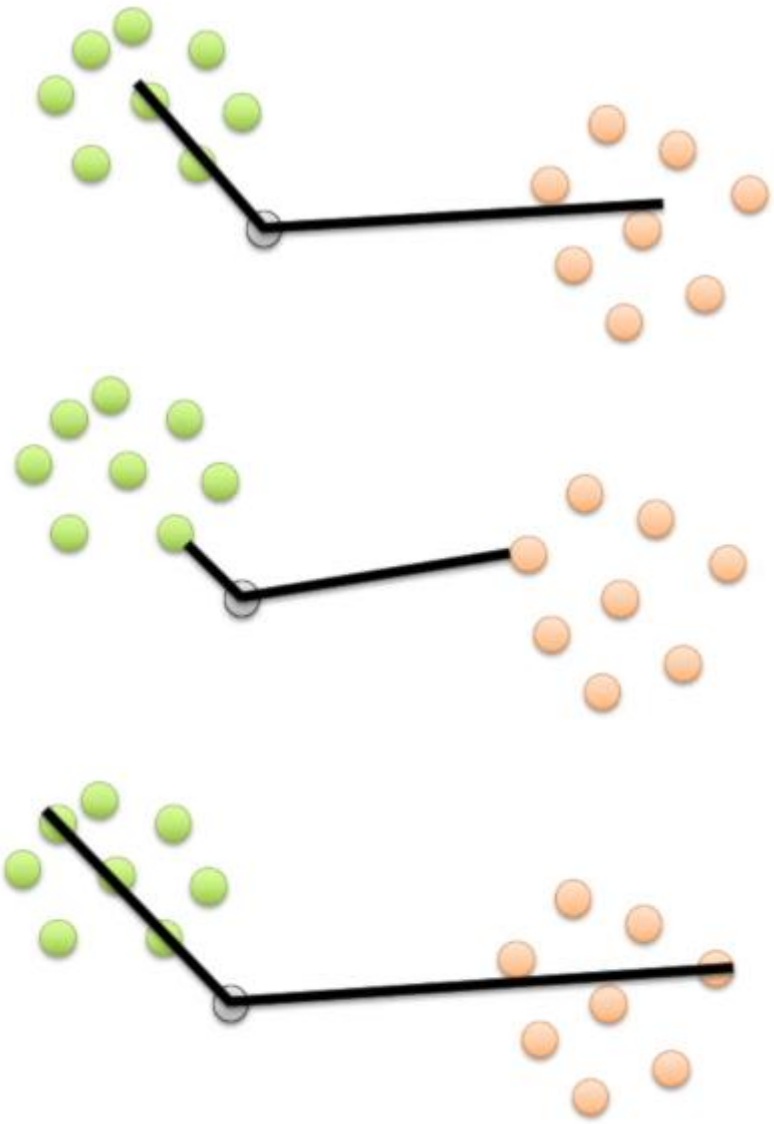
Hierarchical Clustering





- Hierarchical clustering is often associated with “dendrogram”
- Dendrogram indicates the similarity and the order that the clusters are formed

Hierarchical Clustering

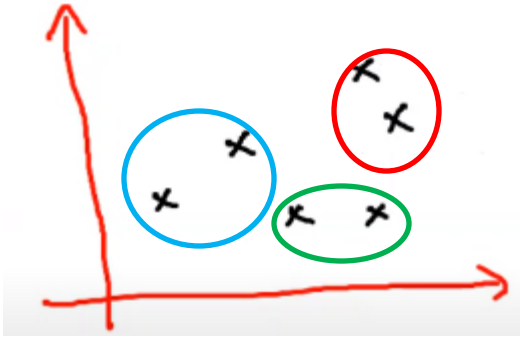


Centroid
(Average)

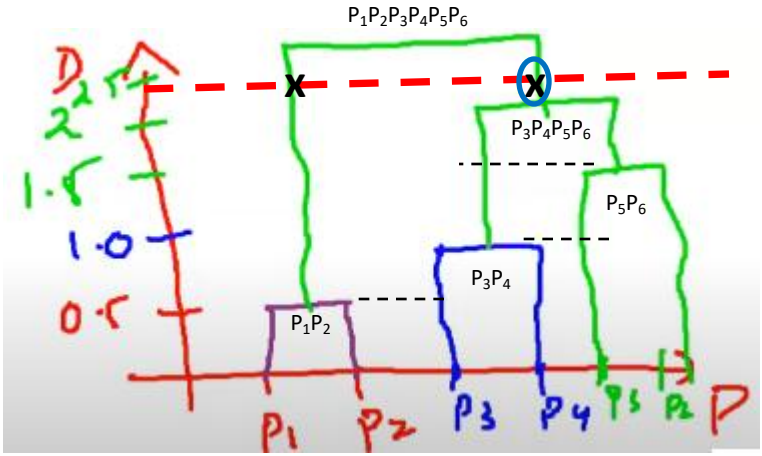
Single-Linkage
(Closest Point)

Complete-Linkage
(Farthest Point)

Hierarchical Clustering - Agglomerative



Euclidean Distance = Height of that Cluster



Find the longest vertical line that is not being cut by any other horizontal lines

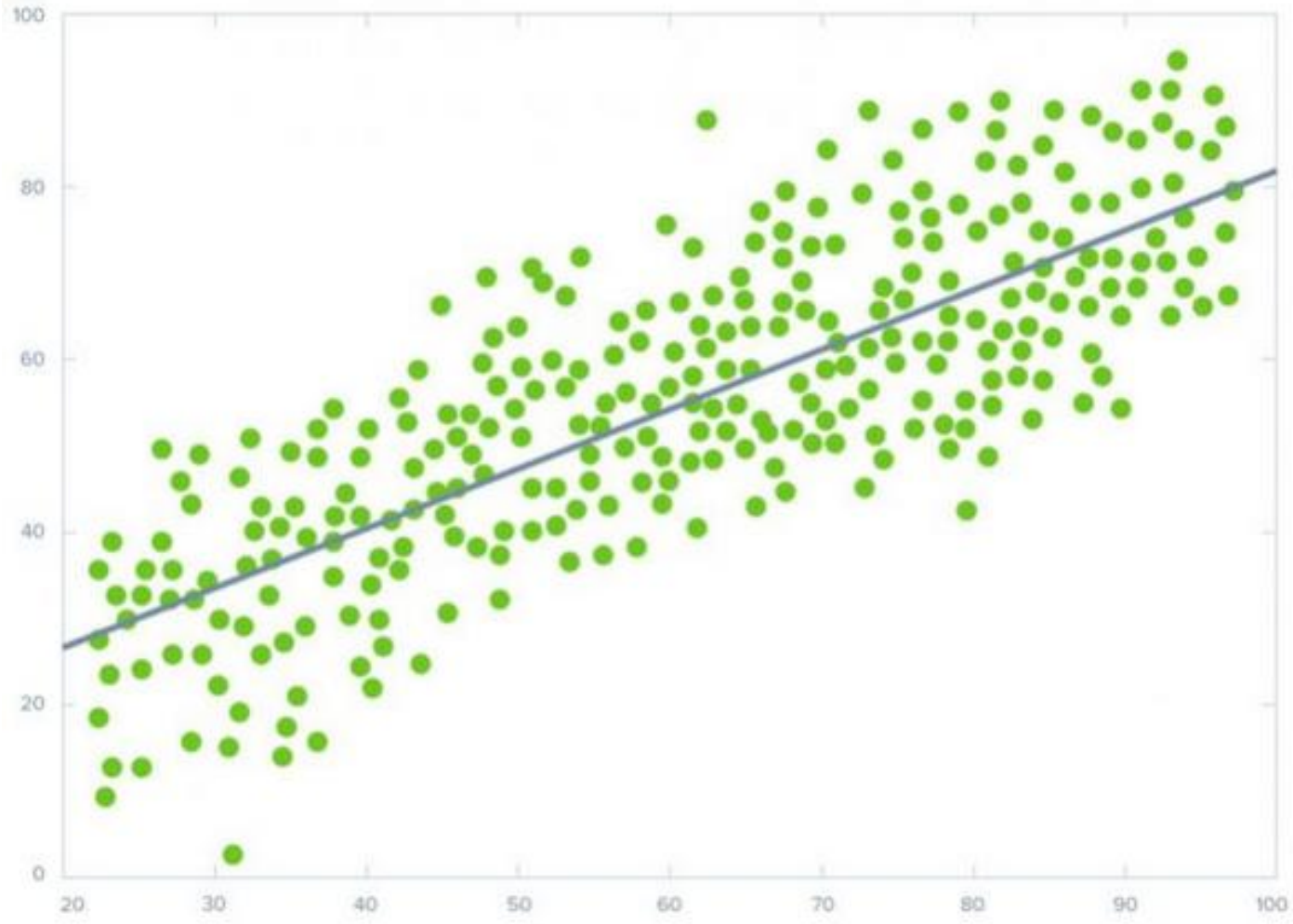
K-Means vs Hierarchical Clustering

- K-Means clustering specifically tries to put the data into the number of clusters you tell it to
- Hierarchical clustering just tells you, pairwise, what two things are similar

Regression Algorithms

- Used to predict response variables with numerical outcomes
- Eg: Wireless carrier can use to predict call volumes at customer service centres
    - Allocate right no. of staffs to meet the demand
- Values of input variables -> numeric or categorical
- Values of response variable -> numeric
- Simple linear regression model is a linear function
    - Only one input variable -> Best line that fits the data
    - Two or more variables -> Best hyperplane that fits the data

Regression Algorithms – Linear Regression

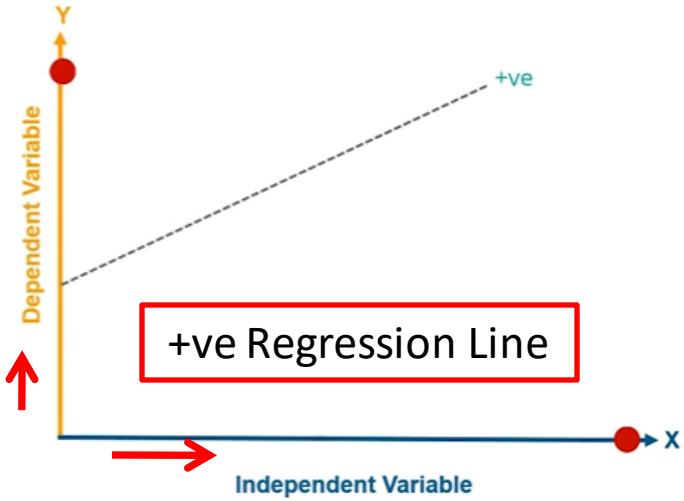


Regression

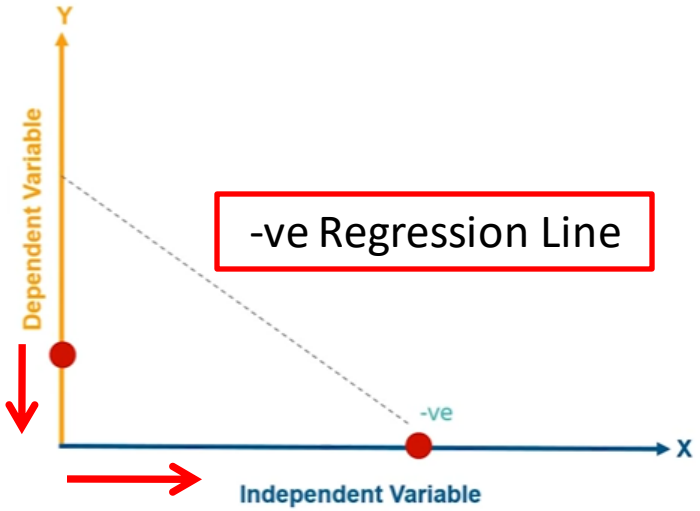
- Answer whether and how some phenomenon influences the other or how several variables are related
- Useful when you want to forecast a response using a new set of predictors
- Types
  - Linear Regression
  - Logistic Regression

https://realpython.com/linear-regression-in-python/

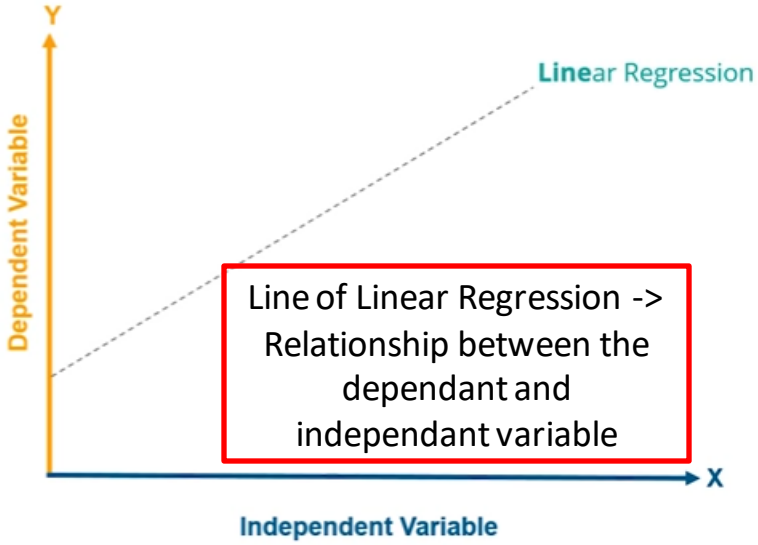
Linear Regression



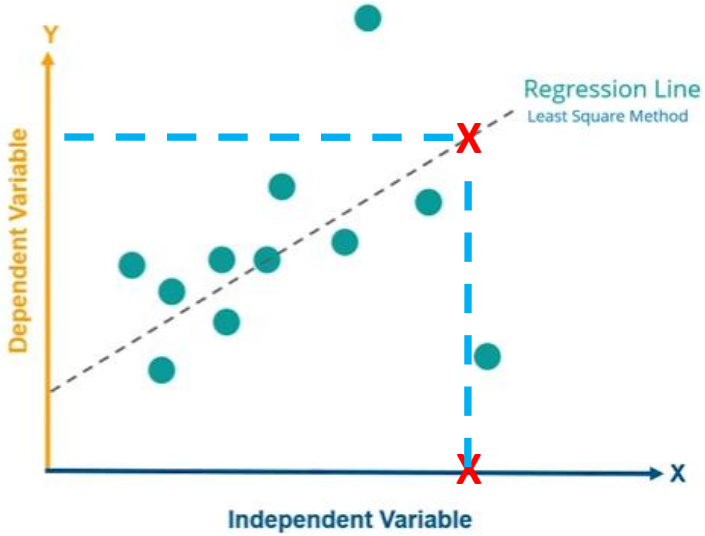
+ve Regression Line



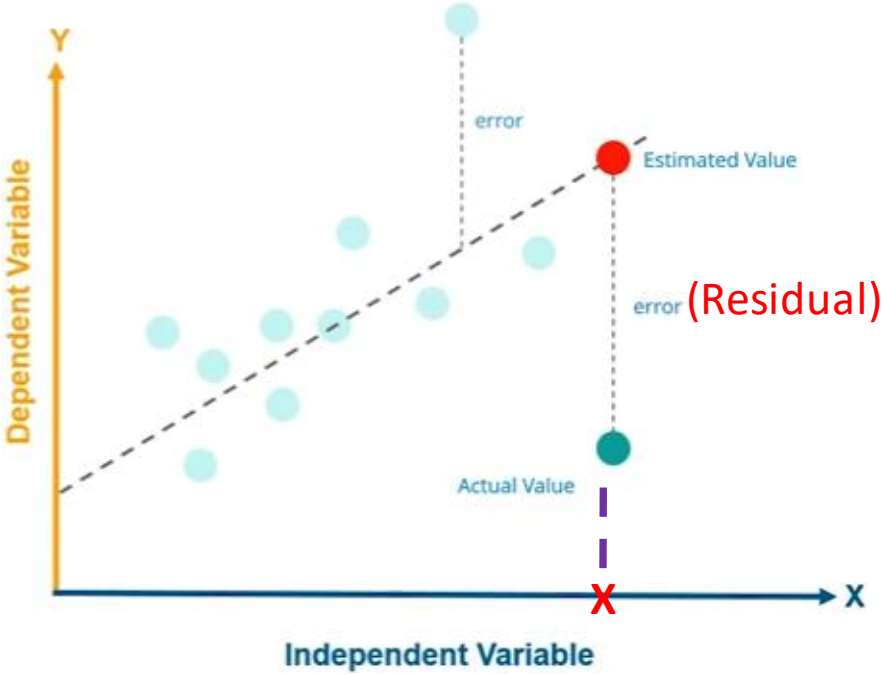
-ve Regression Line

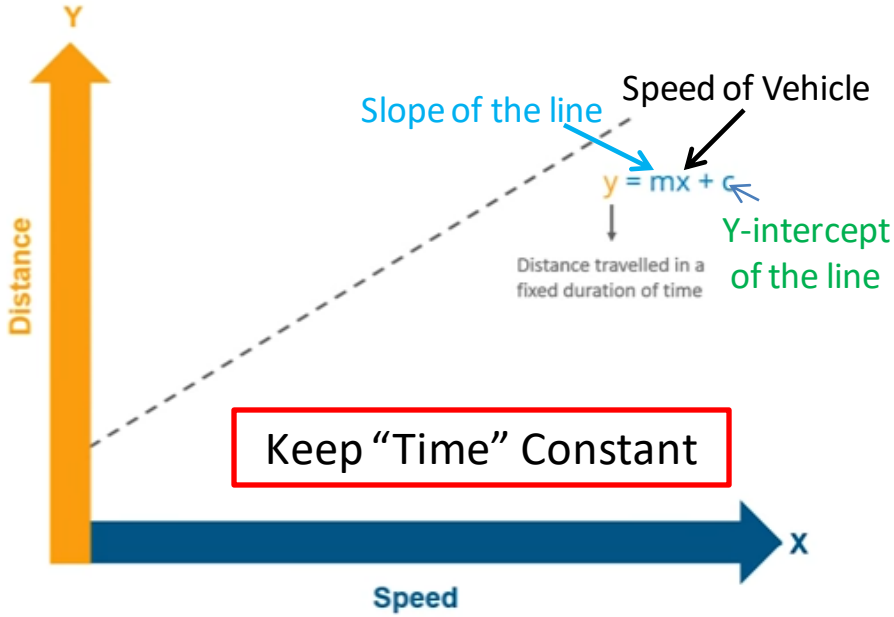


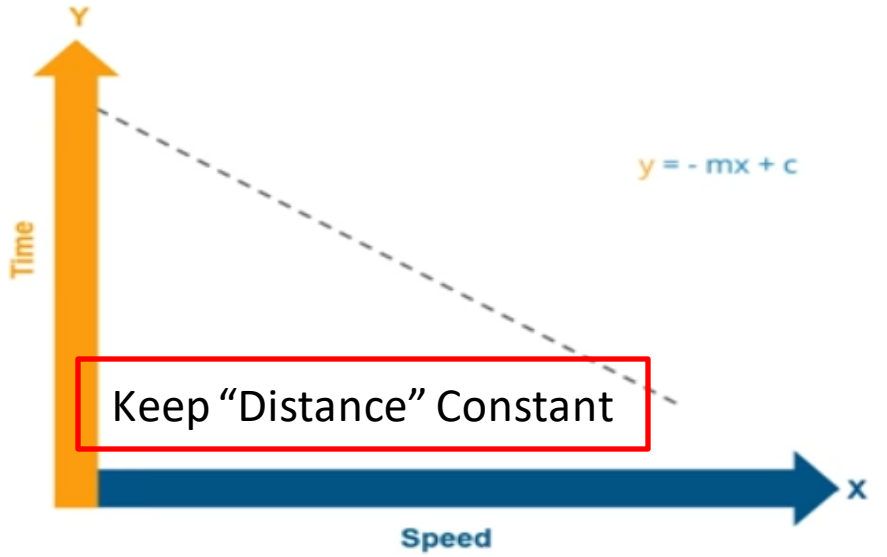
Line of Linear Regression -> Relationship between the dependant and independant variable



Linear Regression

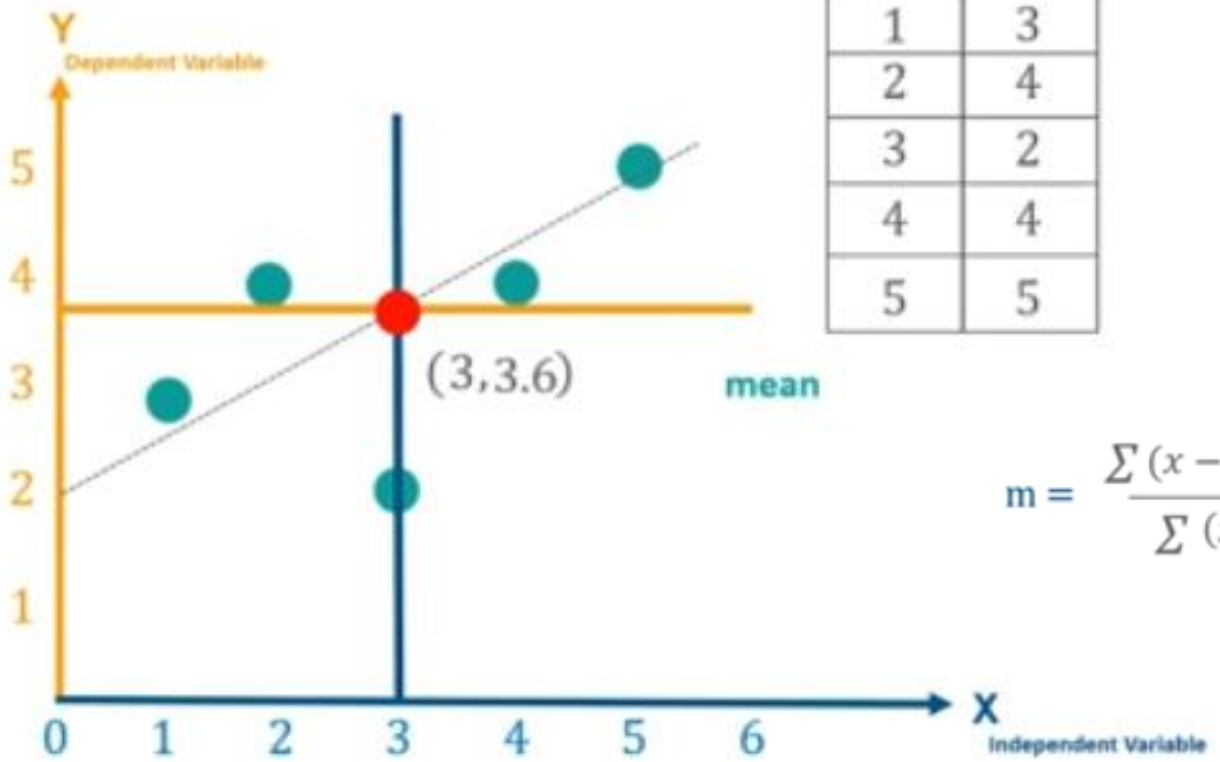






You usually minimize the sum of squared residuals (SSR) for all observations $i = 1, \ldots, n$: SSR = $\Sigma_i(y_i - f(\mathbf{x}_i))^2$. This approach is called the method of ordinary least squares.

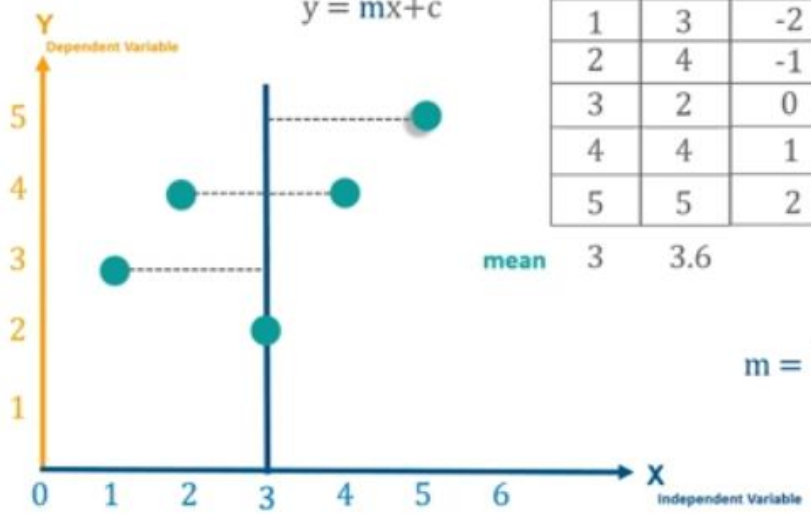
Linear Regression



x	y
1	3
2	4
3	2
4	4
5	5

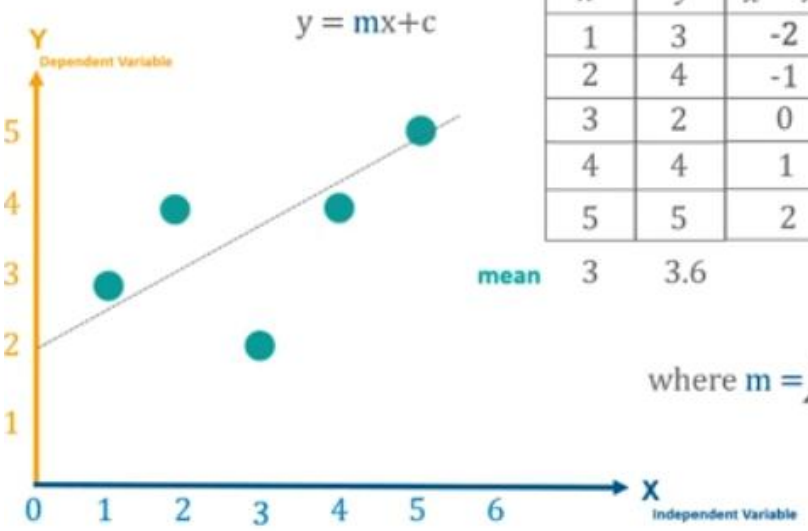
$$m = \frac{\sum (x - \dot{x})(y - \dot{y})}{\sum (x - \dot{x})^2}$$

Linear Regression



x	y	$x-\bar{x}$
1	3	-2
2	4	-1
3	2	0
4	4	1
5	5	2
mean	3	3.6

$$m = \frac{\sum (x-\bar{x})(y-\bar{y})}{\sum (x-\bar{x})^2}$$



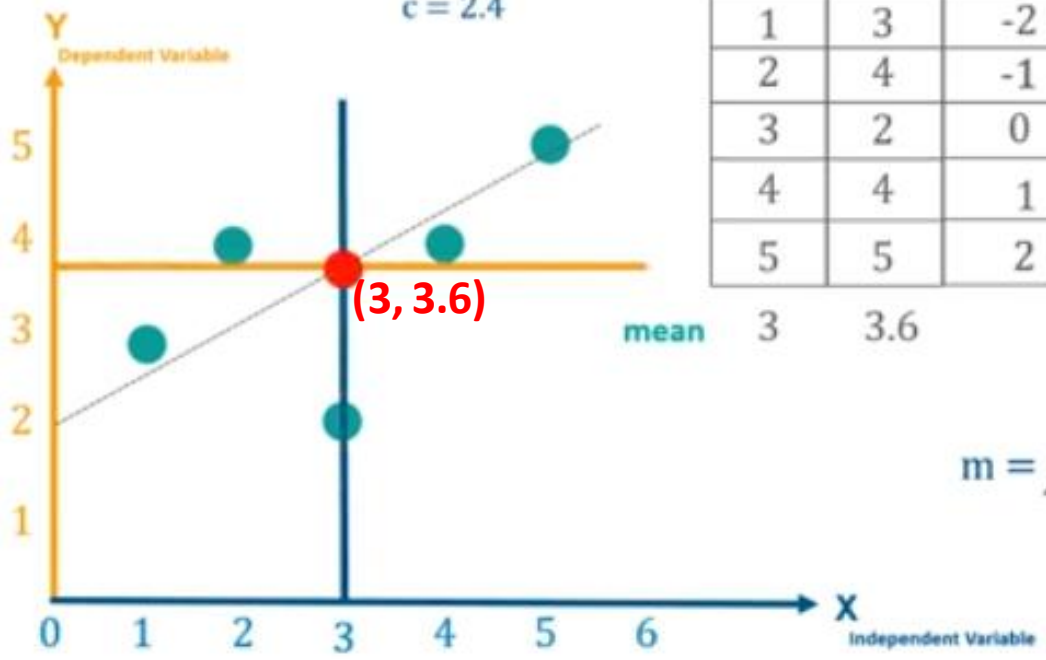
x	y	$x-\dot{x}$	$y-\dot{y}$	$(x-\dot{x})^2$
1	3	-2	-0.6	
2	4	-1	0.4	
3	2	0	-1.6	
4	4	1	0.4	
5	5	2	1.4	
mean	3	3.6		

$$\text{where } m = \sum \frac{(x-\dot{x})(y-\dot{y})}{(x-\dot{x})^2}$$

Linear Regression

$y = mx+c$

$c = 2.4$



x	y	$x-\bar{x}$	$y-\bar{y}$	$(x-\bar{x})^2$	$(x-\bar{x})(y-\bar{y})$
1	3	-2	-0.6	4	1.2
2	4	-1	0.4	1	-0.4
3	2	0	-1.6	0	0
4	4	1	0.4	1	0.4
5	5	2	1.4	4	2.8
mean	3	3.6		$\Sigma = 10$	$\Sigma = 4$

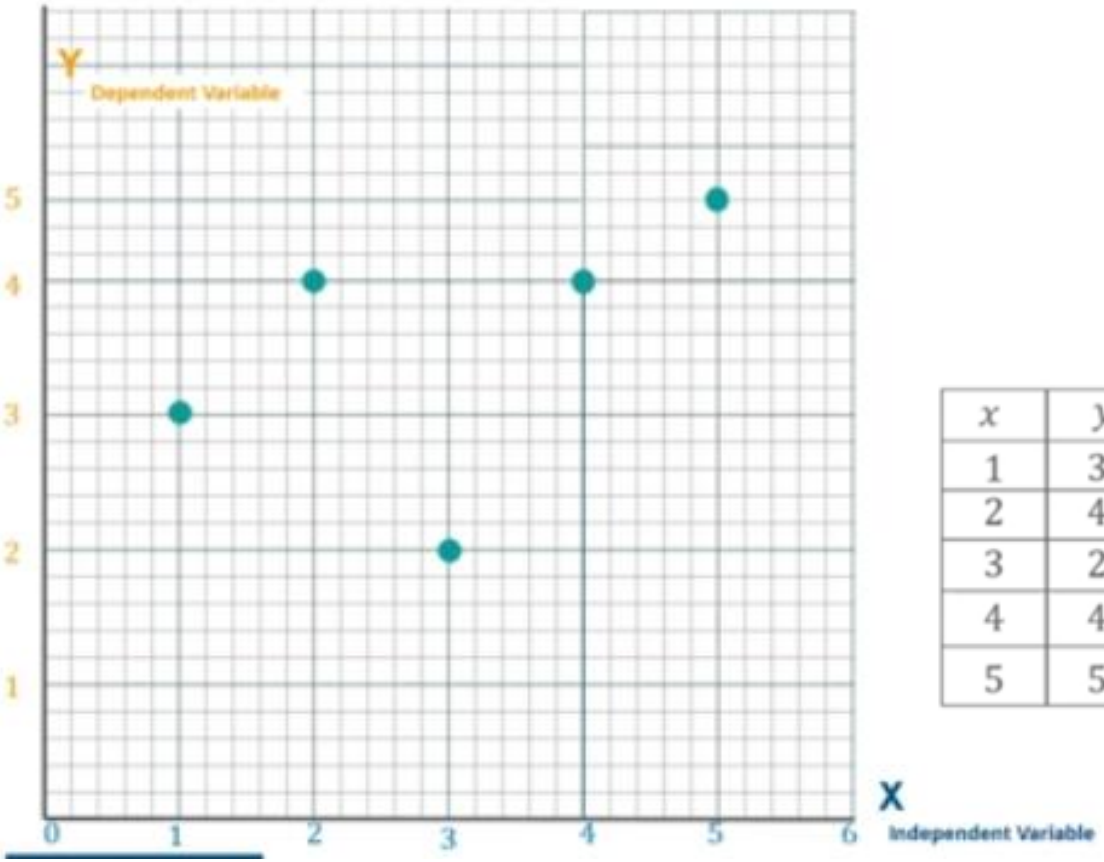
$$m = \sum \frac{(x-\bar{x})(y-\bar{y})}{(x-\bar{x})^2} = \frac{4}{10}$$

$m = 0.4$

$c = 2.4$

$y = 0.4x + 2.4$

Linear Regression



x	y
1	3
2	4
3	2
4	4
5	5

$m = 0.4$

$c = 2.4$

$y = 0.4x + 2.4$

For given $m = 0.4$ & $c = 2.4$, lets predict values for y for $x = \{1,2,3,4,5\}$

$y = 0.4 \times 1 + 2.4 = 2.8$

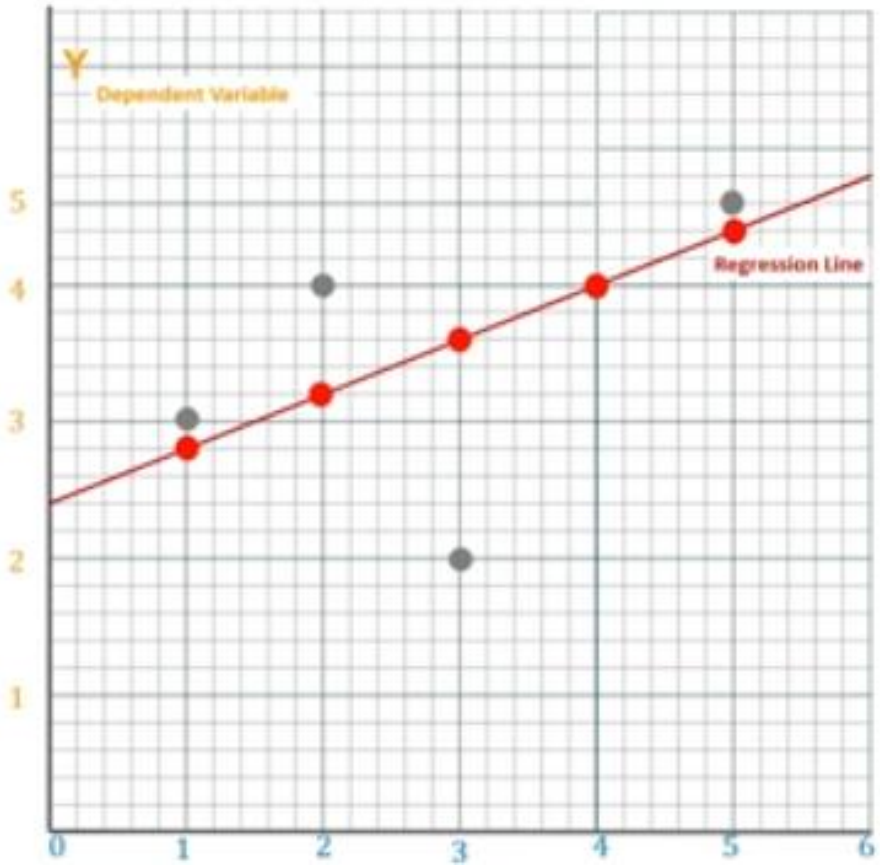
$y = 0.4 \times 2 + 2.4 = 3.2$

$y = 0.4 \times 3 + 2.4 = 3.6$

$y = 0.4 \times 4 + 2.4 = 4.0$

$y = 0.4 \times 5 + 2.4 = 4.4$

Linear Regression



x	y
1	3
2	4
3	2
4	4
5	5

$m = 0.4$

$c = 2.4$

$y = 0.4x + 2.4$

For given $m = 0.4$ & $c = 2.4$, lets predict values for y for $x = \{1,2,3,4,5\}$

$y = 0.4 \times 1 + 2.4 = 2.8$

$y = 0.4 \times 2 + 2.4 = 3.2$

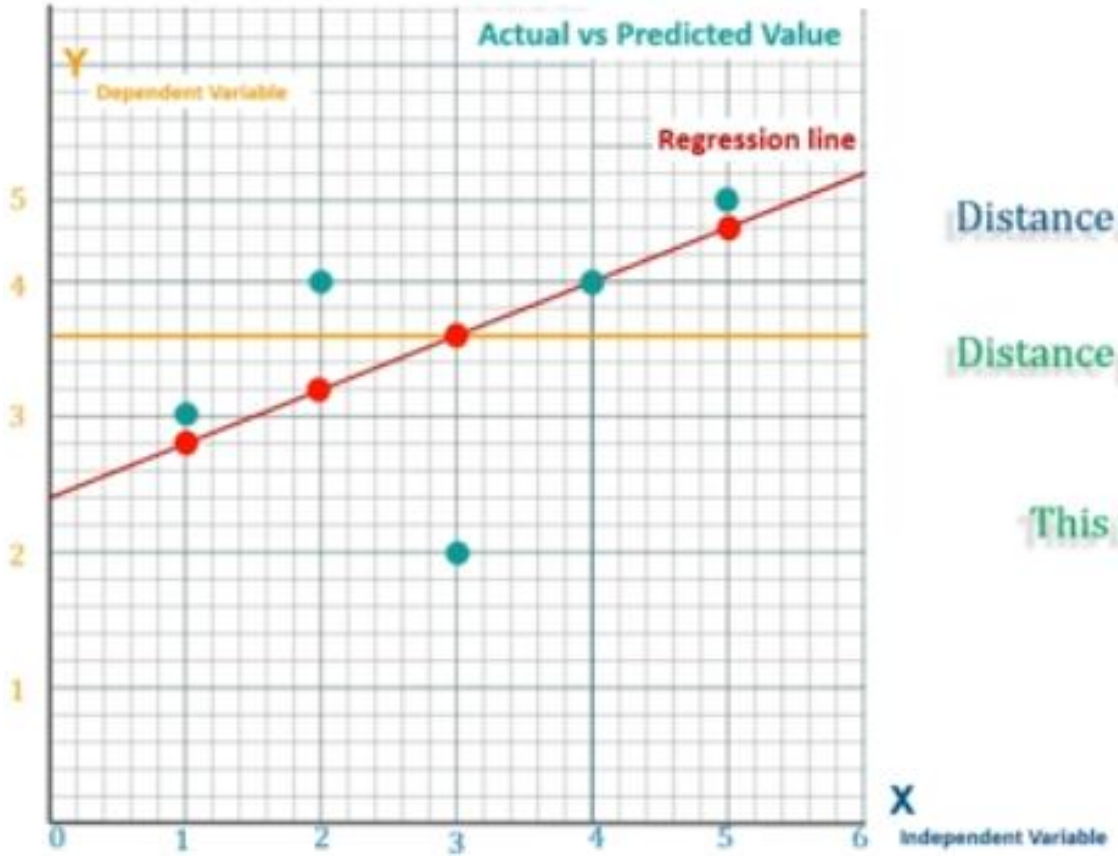
$y = 0.4 \times 3 + 2.4 = 3.6$

$y = 0.4 \times 4 + 2.4 = 4.0$

$y = 0.4 \times 5 + 2.4 = 4.4$

R-Squared Method

- R-Squared value is a statistical measure of how close the data are to the fitted regression line
- Coefficient of Determination or Co-efficient of Multiple Determination



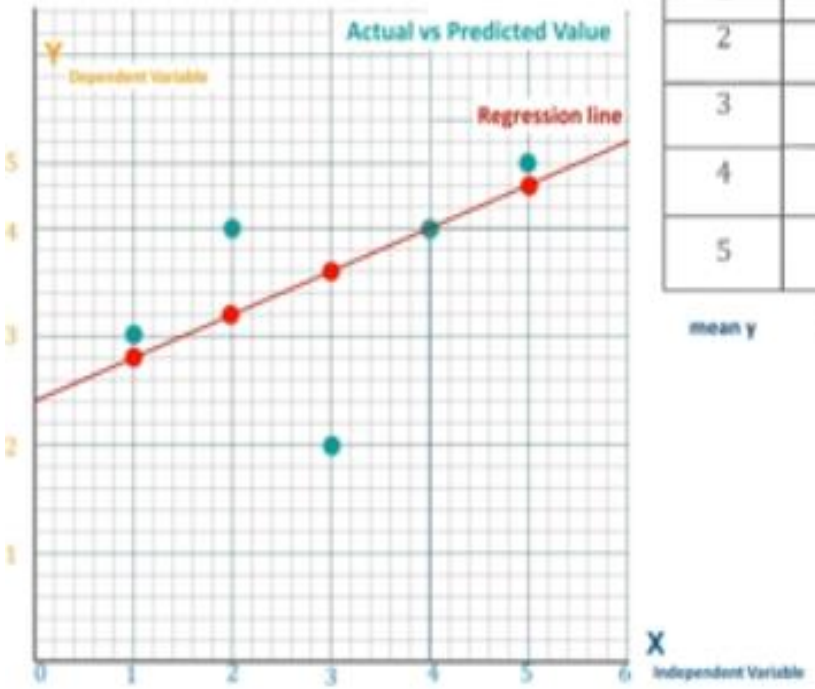
Distance actual - mean

vs

Distance predicted - mean

This is nothing but $R^2 = \frac{\sum (y_p - \bar{y})^2}{\sum (y - \bar{y})^2}$

R-Squared Method

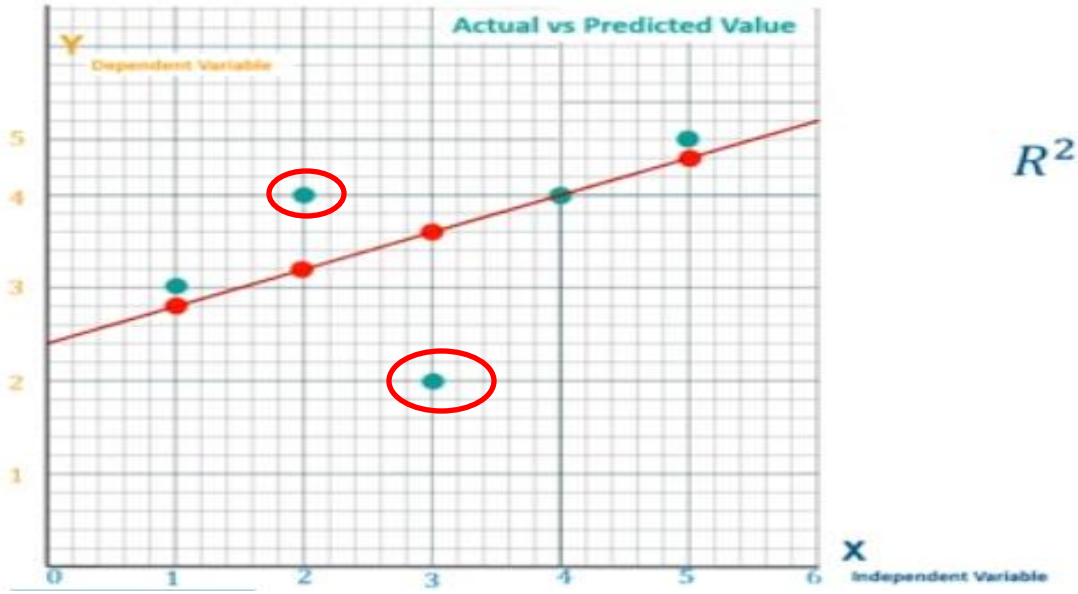


x	y	$y-\bar{y}$	$(y-\bar{y})^2$	y_p	$(y_p-\bar{y})$	$(y_p-\bar{y})^2$
1	3	-0.6	0.36	2.8	-0.8	0.64
2	4	0.4	0.16	3.2	-0.4	0.16
3	2	-1.6	2.56	3.6	0	0
4	4	0.4	0.16	4.0	0.4	0.16
5	5	1.4	1.96	4.4	0.8	0.64
mean y	3.6		$\sum$ 5.2			$\sum$ 1.6

$$R^2 = \frac{1.6}{5.2} = \frac{\sum (y_p - \bar{y})^2}{\sum (y - \bar{y})^2}$$

= 0.3

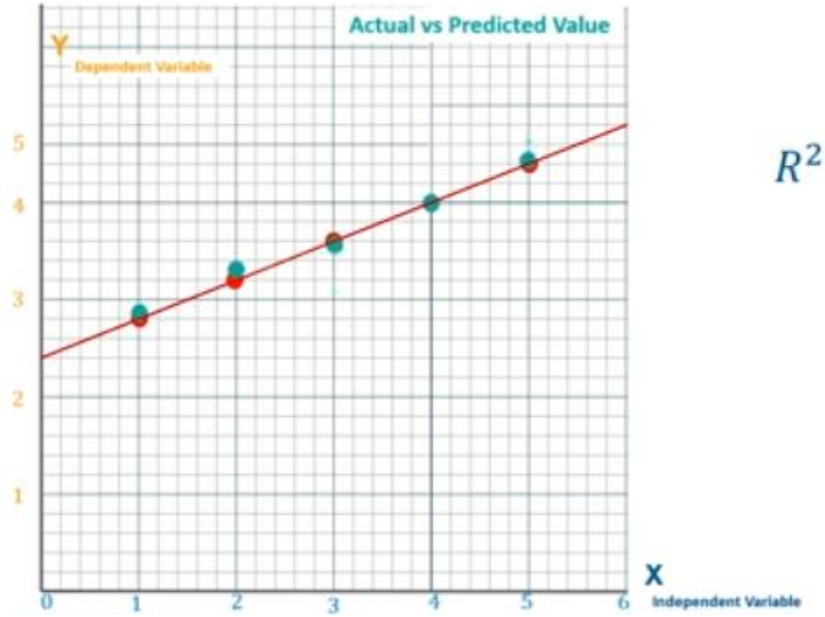
R-Squared Method



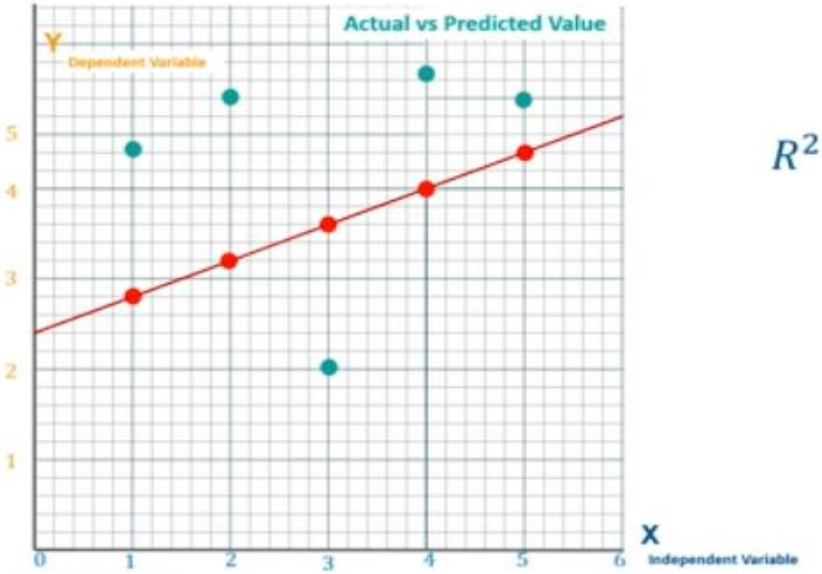
$R^2 \approx 0.3$

$R^2 \approx 0.7$

R-Squared Method



$R^2 \approx 0.9$

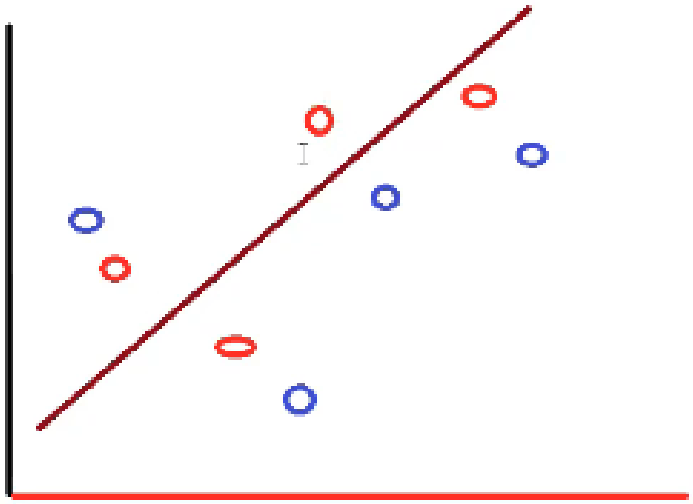


$R^2 \approx 0.02$

Regression Metrics

- Mean Absolute Error (MAE)
- Mean Squared Error (MSE)
- Root Mean Squared Error (RMSE)

Residual Error = Y - y



Red = Actual Values
Blue = Predicted Values

THANK YOU