# TREES & FORESTS

Anurag Srivastava

# AGENDA

Why do we need AI / ML?

CART

Decision Trees

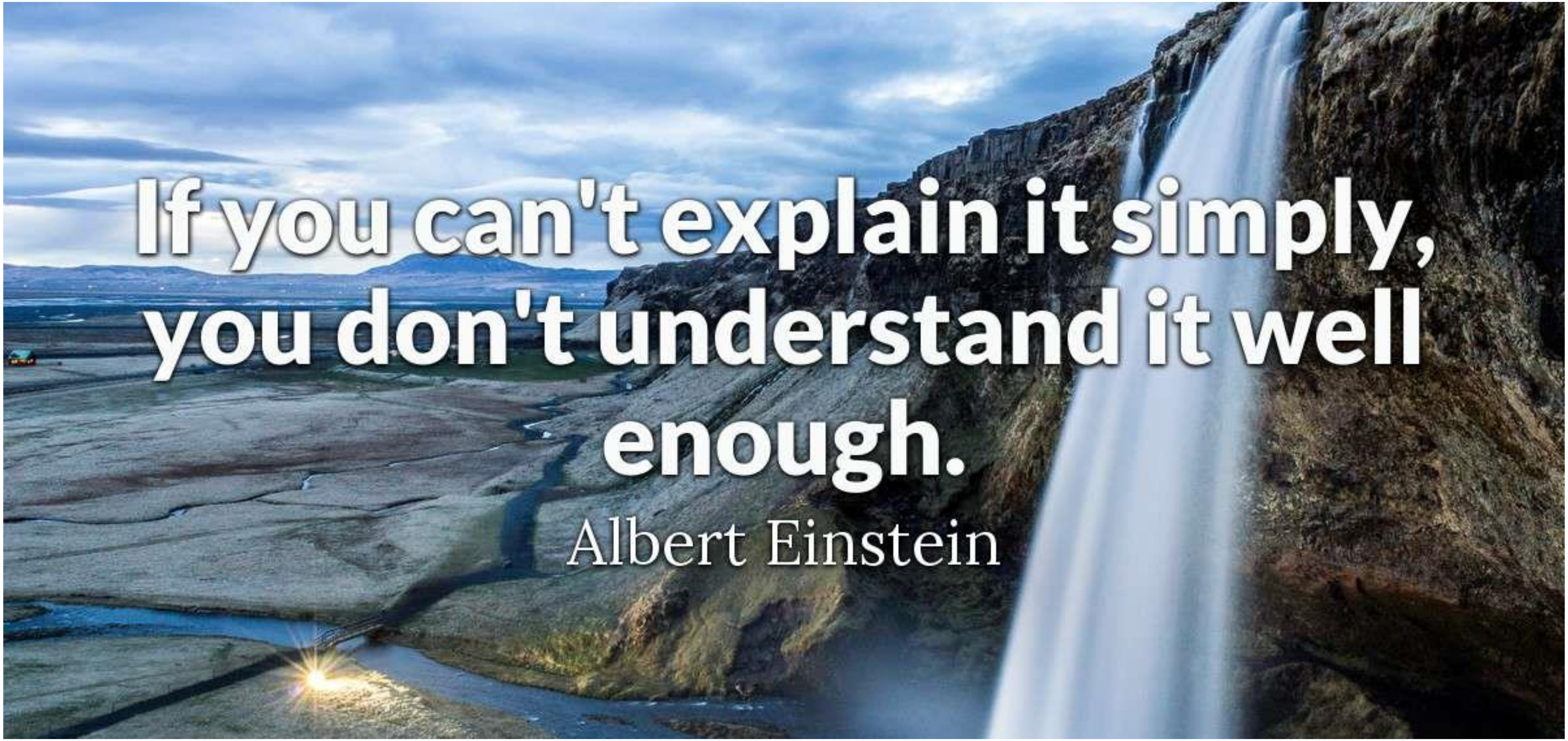
Random Forests

Ensembles (Bagging / Boosting / Stacking)

1 image summary

When to use which algorithm

Sklearn vs. Statsmodels


If you can't explain it simply, you don't understand it well enough.
Albert Einstein

# APPROACH TO THE COURSE

## Applications of AI / ML

**Image Processing**
- Image tagging / Image Recognition
- OCR or Optical Character Recognition
- Self-driving cars

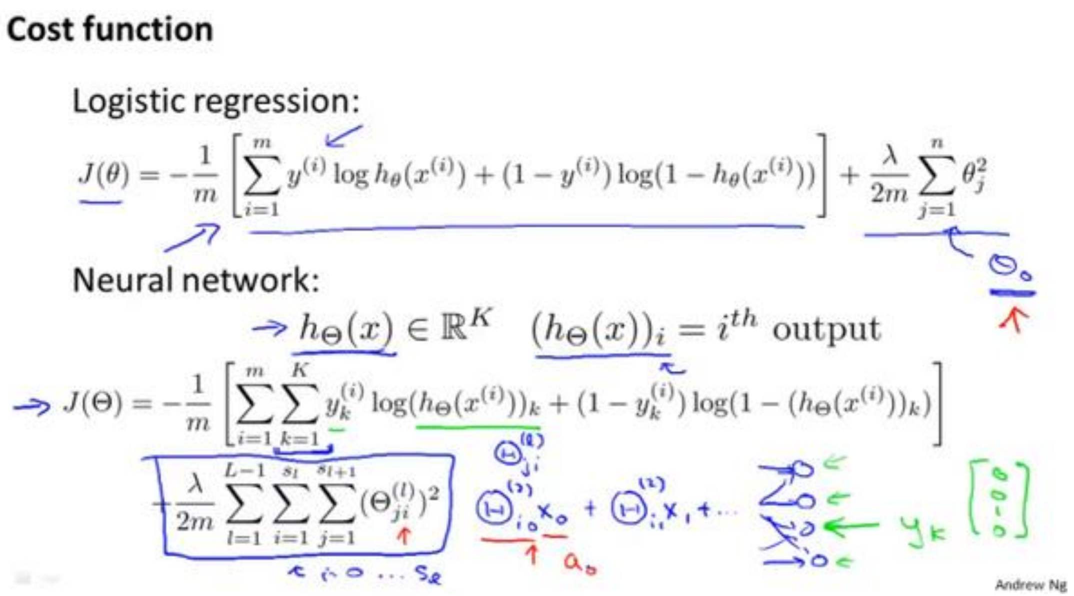
**Text Analysis**
- Spam Filtering
- Sentiment Analysis
- Information Extraction

**Data Mining**
- Anomaly Detection
- Association Rules
- Grouping
- Predictions

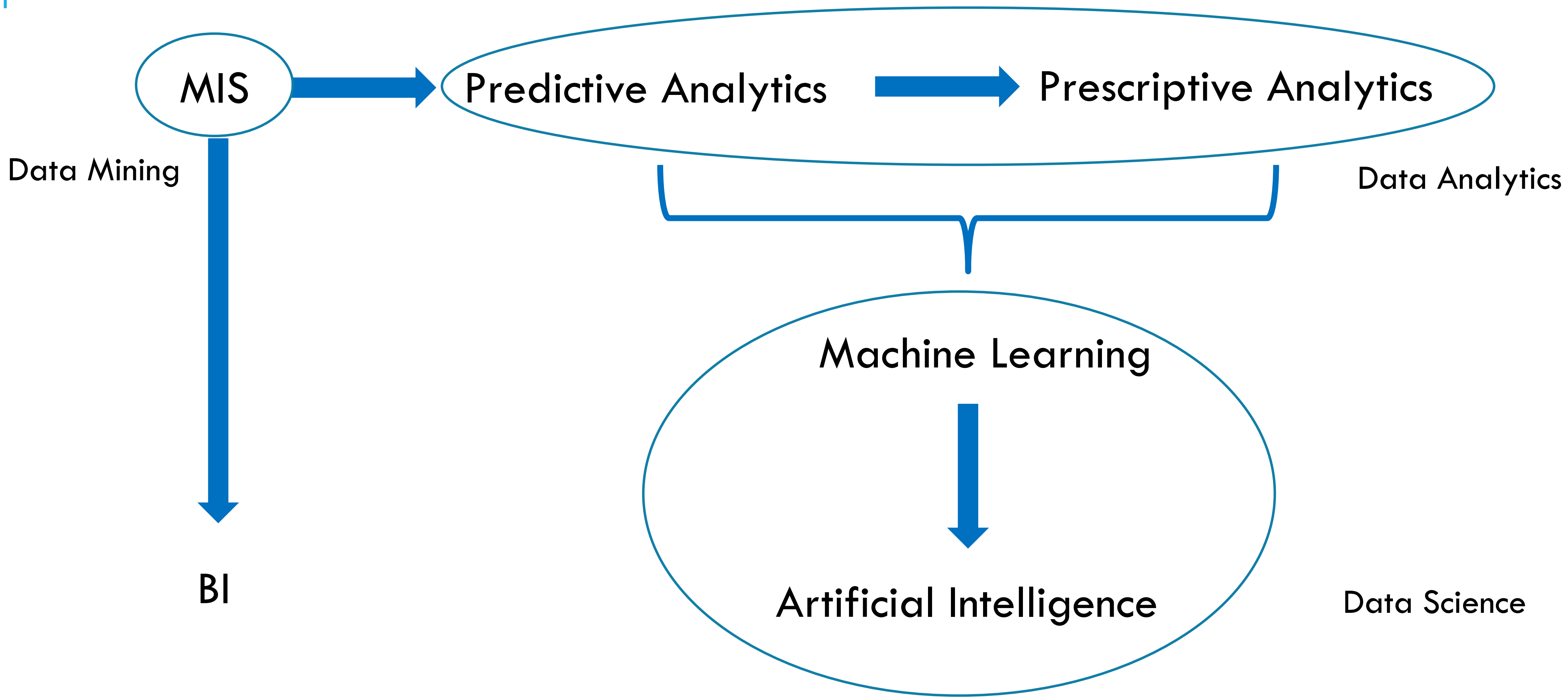
**Healthcare**
- Medical Diagnosis
- Imaging Diagnosis
- Oncology
- Drug Trials

**Video Games**
- Reinforcement Learning

**Robotics**
- Industrial tasks
- Human simulations



# THE PROGRESSION



# ROLES / PROFILES / DESIGNATION

- AI Researcher
  - Client / CXO Interface
  - Product Owner
  - SME Coordination
  - Lead Data Scientist
  - Proof of Concept (PoC)
- AI Developer
  - Execution of PoC
  - Scaling up
  - Development, Delivery & QA
- AI DBA
  - Hadoop Stack Administration (Ambari)

Data Scientist

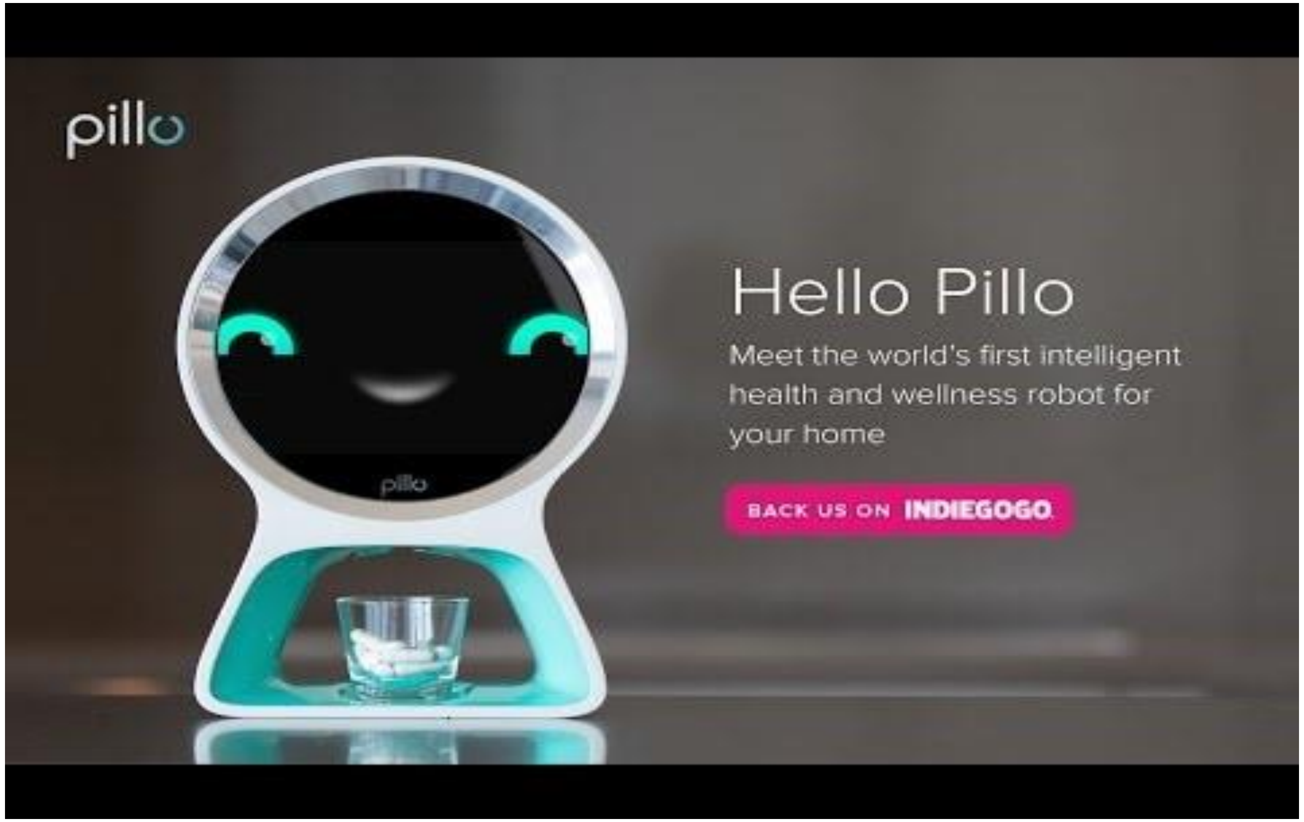
# WHY AI? LET US INTUIT.

- Would you cross this bridge?

# EXAMPLES OF AI SYSTEMS - BITBITE


https://www.youtube.com/watch?v=qU2w_qiP4Ck

# EXAMPLES OF AI SYSTEMS - PILLO




https://www.youtube.com/watch?v=GfjGPTKBFB4&t=1s

# EXAMPLES OF AI SYSTEMS - COZMO


https://www.youtube.com/watch?v=cjFA531qJNE

# AI IN APPLICATION

In [50]:
```
evidences = ["The patient has lower jaw pain."]

hypotheses = ["The patient has heart attack."]

sentence1 = [fit_to_size(np.vstack(sentence2sequence(evidence)[0]),
                         (30, 50)) for evidence in evidences]

sentence2 = [fit_to_size(np.vstack(sentence2sequence(hypothesis)[0]),
                         (30,50)) for hypothesis in hypotheses]

prediction = sess.run(classification_scores, feed_dict={hyp: (sentence1 * N),
                                                        evi: (sentence2 * N),
                                                        y: [[0,0,0]]*N})
print(["Positive", "Neutral", "Negative"][np.argmax(prediction[0])]+
      " entailment")
```

```
Negative entailment
```

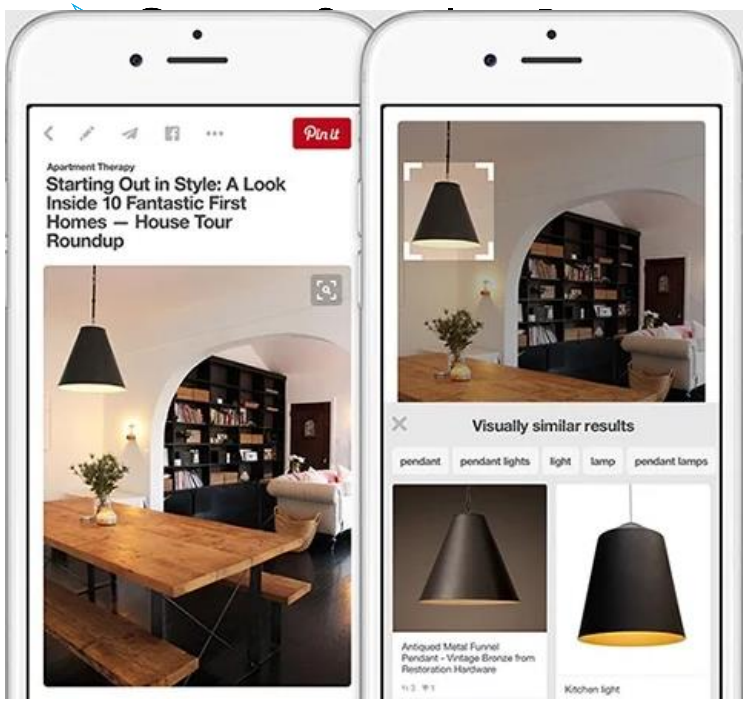
# AI IN APPLICATION

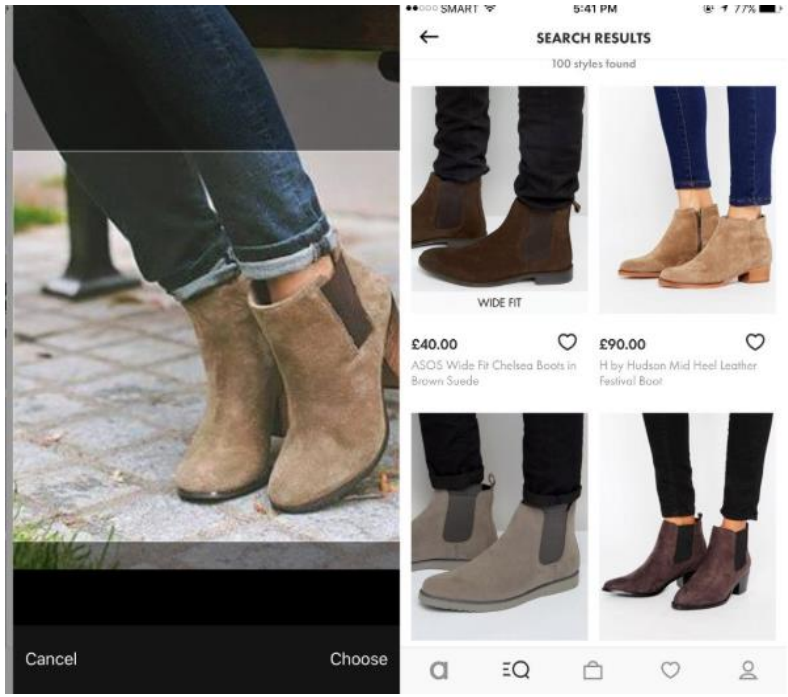
```
In [49]: evidences = ["The patient has diabetes and lower jaw pain."]

         hypotheses = ["The patient has heart attack."]

         sentence1 = [fit_to_size(np.vstack(sentence2sequence(evidence)[0]),
                                  (30, 50)) for evidence in evidences]

         sentence2 = [fit_to_size(np.vstack(sentence2sequence(hypothesis)[0]),
                                  (30,50)) for hypothesis in hypotheses]

         prediction = sess.run(classification_scores, feed_dict={hyp: (sentence1 * N),
                                                                 evi: (sentence2 * N),
                                                                 y: [[0,0,0]]*N})
         print(["Positive", "Neutral", "Negative"][np.argmax(prediction[0])]+
               " entailment")
```

```
Positive entailment
```

# AI USE CASE – E COMMERCE

- App that suggests recipes using the inventory of your fridge – Chefling



# AI USE CASE – E COMMERCE

- Chatbots
- Virtual Assistants
  - By using Alexa on Amazon's Echo device, customers can discover local gigs for the upcoming weekend through StubHub, arrange transport to and from the event via Uber, or even order pre-event dinner from Domino's (and track the order status in real time).
- Wish
  - On the spot personal pricing
- Tackle fake reviews
  - www.fakespot.com
- Inventory Management & Sales Forecasting
- Generating product descriptions in any writing style

# IMAGINE REAL TIME MODULES

- Deception detection in speech
- Image verification – KYC – document photo and selfie
- Reading handwritten forms / applications (any language)
- Intuitive client risk profiling (social media data + financial data)
- Voice based initial screening of loan applicants - https://www.youtube.com/watch?v=oUf9_rS1fYg
- Transactional bots - digital assistants helping users navigate their finance plans, savings, and spending based on web footprints.
- OCR can be used to digitize hard copy documents. An NLP model with layered business logic can then interpret, record, and correct contracts at high speed.

# TRANSACTIONAL BOTS

- Automated Claims Processes –

```
> Hi
Hi, how can I help?
> I'd like to submit a claim
Sure. Which area is this regarding: Home, Car or Health?
> Home
Ok. I'm going to ask you a series of questions to help you fill the claim.
What is the nature of the incident?
>
```

- It then moves on to the adjustment model where it provides a range of values for payout. Once all data is set, human intervention can be included for auditing purposes. The bot can at this point calculate and propose payout amounts, based on a payout predictor model it has been trained on.

# TRANSACTIONAL BOTS

- AI assistance with claims settlement is fast and more successful.
- And fast means — 3 seconds from claim submission to payment. This is the result of the independent work of Lemonade's AI - Jim.
- Lemonade is a New York based Insurance startup.

- What AI can do is to assist, advise, and create a better customer experience. At the moment, AI complements human work and makes it more efficient, but it isn't aimed to replace a human.

# CLASSIFICATION

Classification is a data mining technique used for systematic placement of group membership of data.

It maps the data into predefined groups or classes and searches for new patterns.

For example, you may wish to use classification to predict whether the weather on a particular day will be “sunny”, “rainy”, or “cloudy”.

# REGRESSION

Used to predict for individuals on the basis of information gained from a previous sample of similar individuals.

For example, A person wants do some savings for future and then It will be based on his current values and several past values. He uses a linear regression formula to predict his future savings.

It may also be used in modelling the effect of doses in medicines or agriculture, response of a customer to a mail and evaluate the risk that the client will not pay back the loan taken from the bank.

# WHAT IS CART?

Classification And Regression Trees

Developed by Breiman, Friedman, Olshen, Stone in early 80's.

Introduced tree-based modeling into the statistical mainstream, rigorous approach involving cross-validation to select the optimal tree.

One of many tree-based modeling techniques.

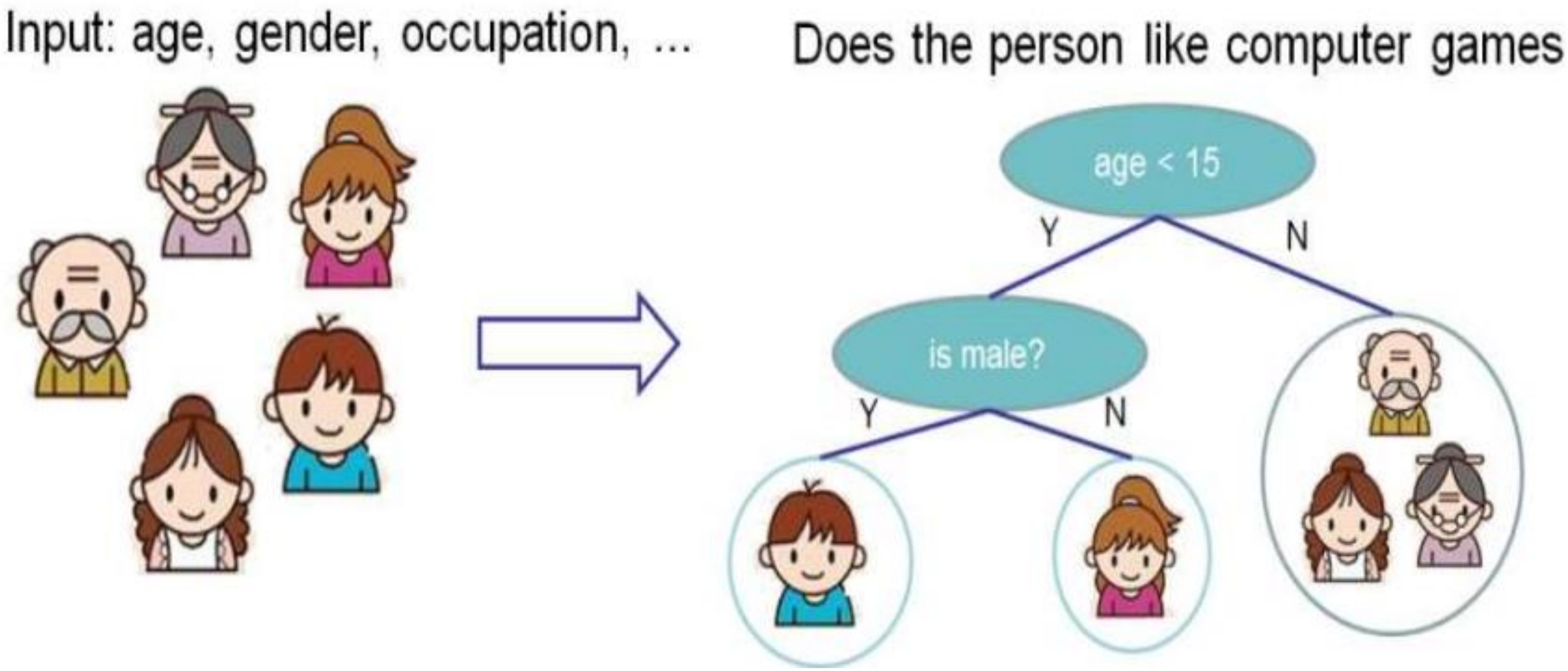
CART -- the classic

CHAID

C4.5

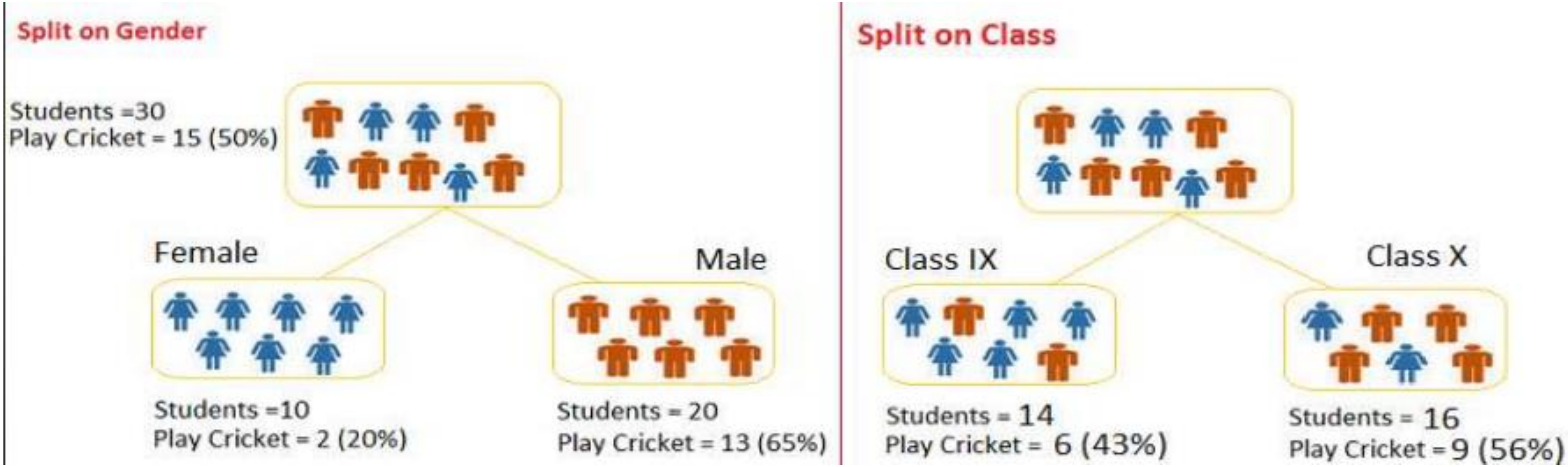
C5.0

# HOW DO THEY WORK?



# WHAT?

If the dependent variable is categorical, CART produces a classification tree. And if the variable is continuous, it produces a regression tree.



# THE KEY IDEA

Take all of your data.

Consider all possible values of all variables.

Select the variable/value ($X=t_1$) that produces the greatest "separation" in the target.

($X=t_1$) is called a "split".

If ($X< t_1$) then send the data to the "left"; otherwise, send data point to the "right".

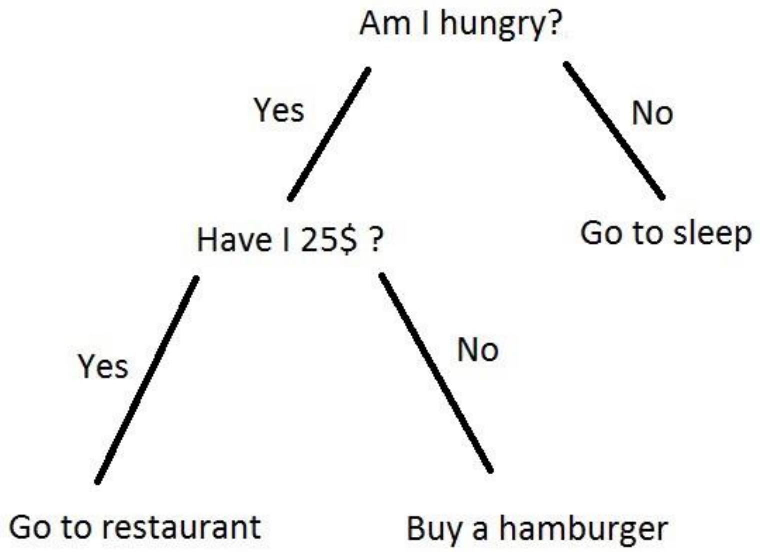
Now repeat same process on these two "nodes".

You get a "tree"

Note: CART only uses binary splits.

# DECISION TREES

- A decision tree is a decision support tool that uses a tree-like model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility.
- Simply, A **decision tree** is a graphical representation of possible solutions to a **decision** based on certain conditions. It's called a **decision tree**because it starts with a single box (or root), which then branches off into a number of solutions, just like a **tree**.



# WAIT, CART VS. DECISION TREES?

Difference?

No difference….

Decision Tree – old name

CART – fancy modern name

# ADVANTAGES OF DECISION TREES

### Advantages of a Decision Tree

- **Simple** Representation for any one to understand
- **Framework** of business rules that can be used going forward
- Ease of **explanation** to business
- **Flexibility to modify** the Decision Tree to make it balanced and more practical in usage
- It is the only **multivariate algorithm** that be represented in a diagram form so that **anyone can understand**
- Can be applied to **categorical** and continuous target variables as well.

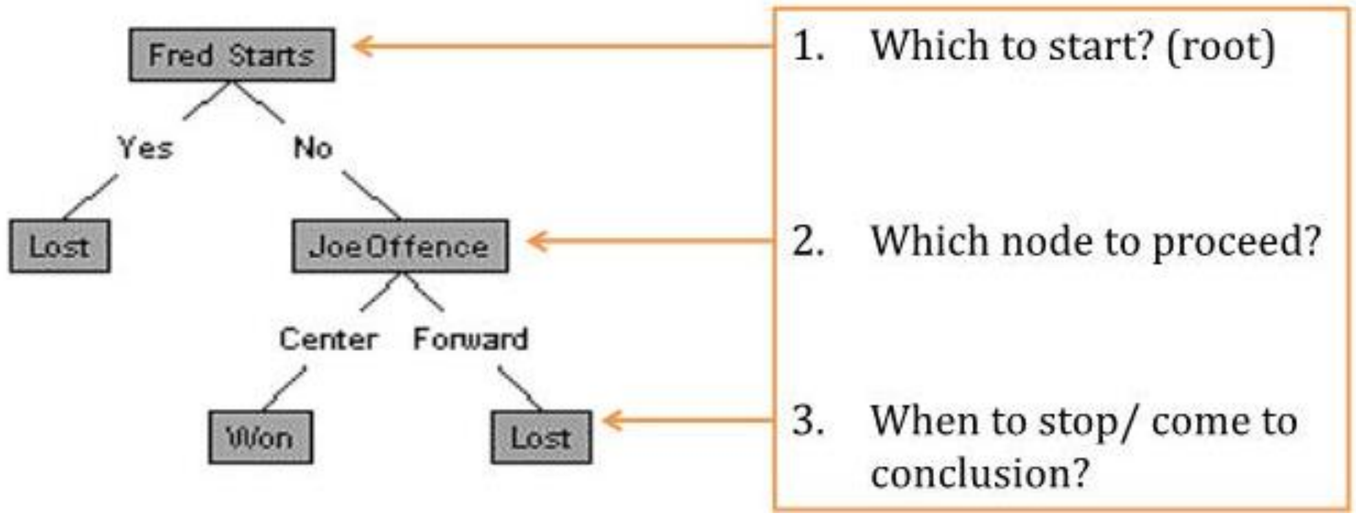
### Key Requirements

- **Attribute-Value Description:** Object or case must be expressible in terms of a fixed collection of properties or attributes (e.g., hot, mild, cold).
- **Predefined Classes (target values):** The target function has discrete output values (bollean or multiclass)
- **Sufficient Data:** Enough training cases should be provided to learn the model.

# COMPONENTS OF DECISION TREES

- Decision node: Specifies a test on a single attribute
- Leaf node: Indicates the value of the target attribute
- Arc/edge: Split of one attribute
- Path: A disjunction of test to make the final decision

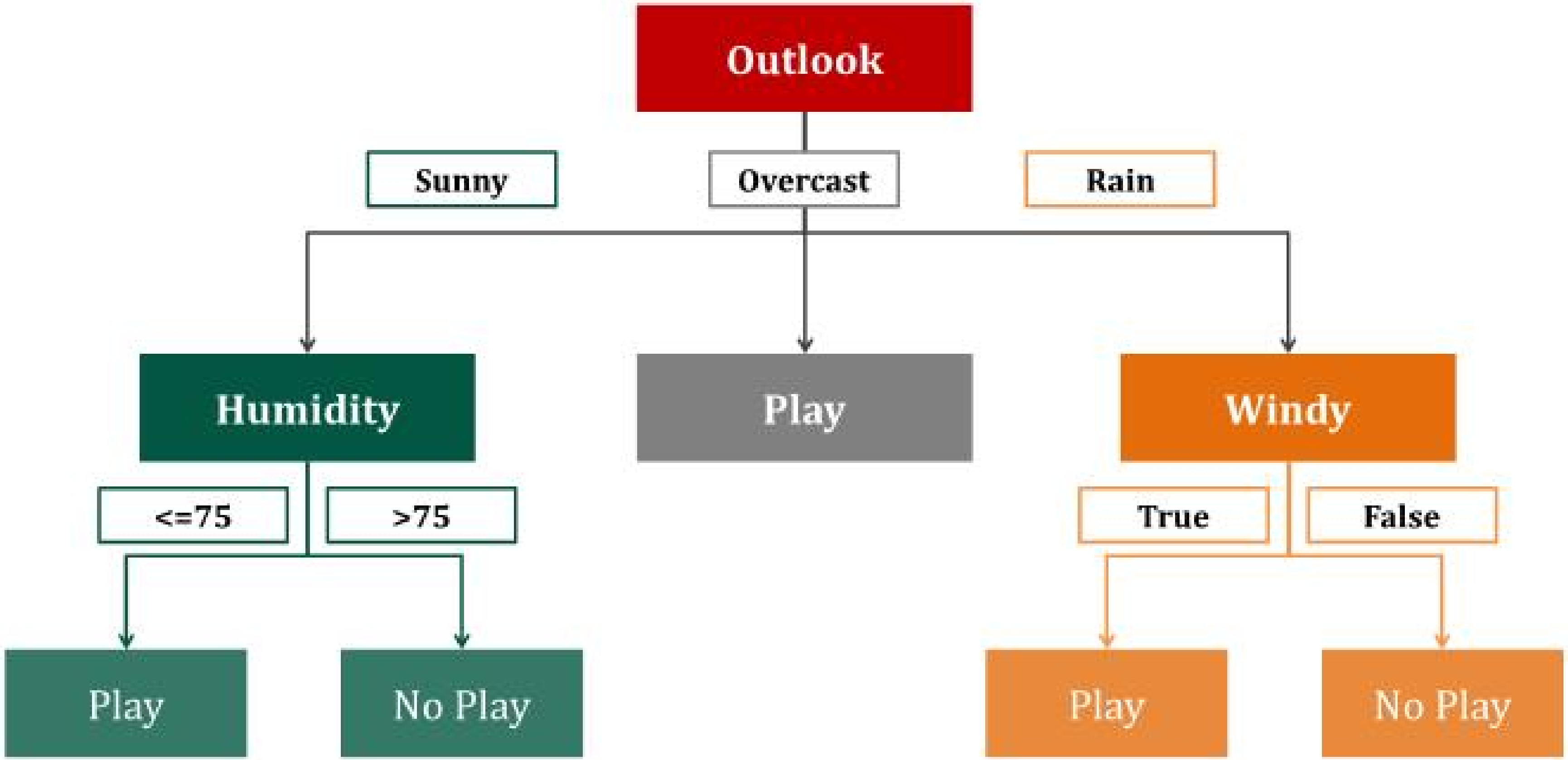
Decision trees **classify** instances or examples **by starting at the root** of the tree and **moving through it until a leaf node**.



# TRAINING DATA

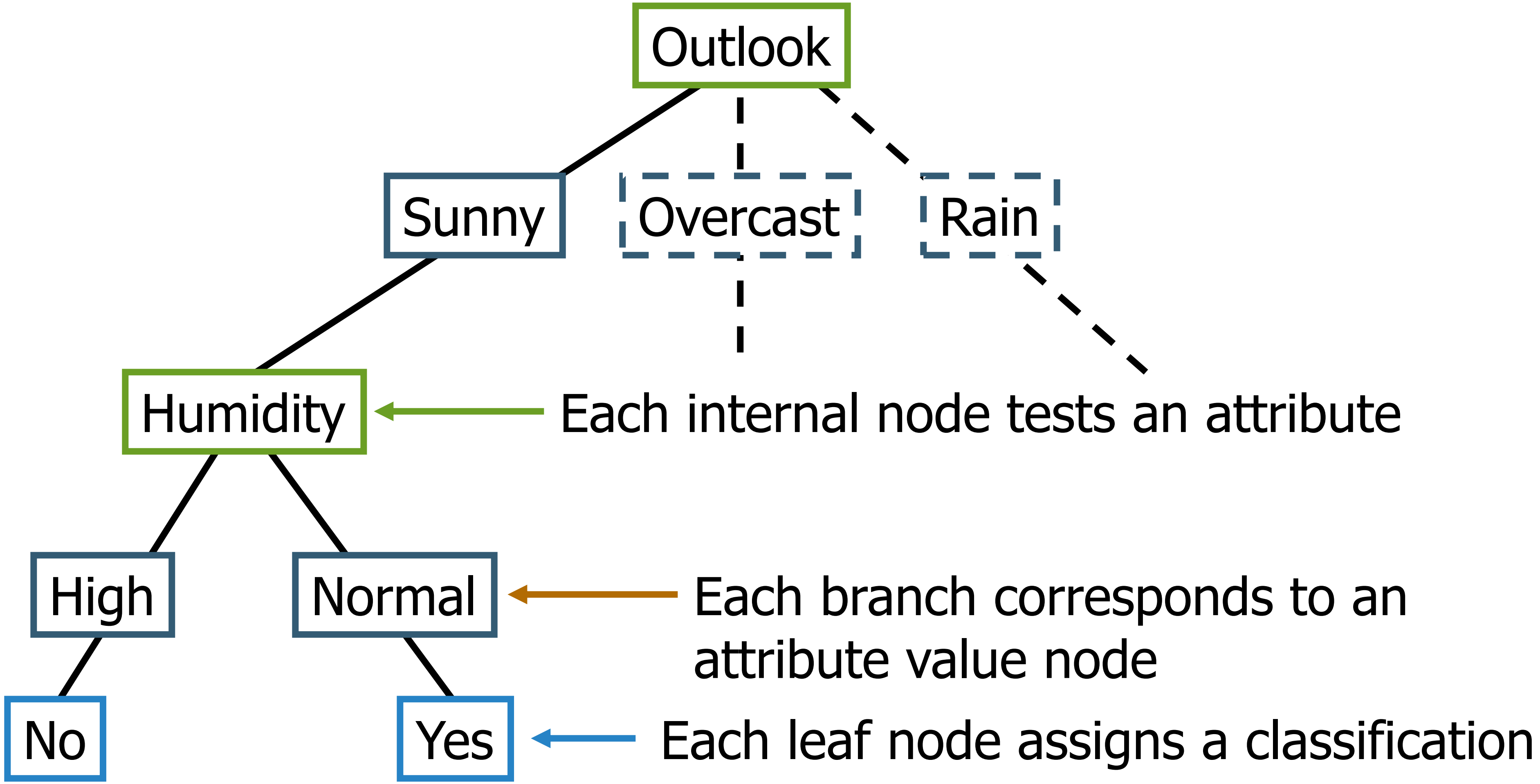
| Day | Outlook | Temp. | Humidity | Wind | Play Tennis |
|---|---|---|---|---|---|
| D1 | Sunny | Hot | High | Weak | No |
| D2 | Sunny | Hot | High | Strong | No |
| D3 | Overcast | Hot | High | Weak | Yes |
| D4 | Rain | Mild | High | Weak | Yes |
| D5 | Rain | Cool | Normal | Weak | Yes |
| D6 | Rain | Cool | Normal | Strong | No |
| D7 | Overcast | Cool | Normal | Weak | Yes |
| D8 | Sunny | Mild | High | Weak | No |
| D9 | Sunny | Cold | Normal | Weak | Yes |
| D10 | Rain | Mild | Normal | Strong | Yes |
| D11 | Sunny | Mild | Normal | Strong | Yes |
| D12 | Overcast | Mild | High | Strong | Yes |
| D13 | Overcast | Hot | Normal | Weak | Yes |
| D14 | Rain | Mild | High | Strong | No |

# DECISION TREES RULES: SAMPLE




Five leaf nodes – Each represents a rule

# DECISION TREE FOR PLAYTENNIS



# WORD SENSE DISAMBIGUATION

Given an occurrence of a word, decide which sense, or meaning, was intended.

Example: "run"

- run1: move swiftly (I ran to the store.)
- run2: operate (I run a store.)
- run3: flow (Water runs from the spring.)
- run4: length of torn stitches (Her stockings had a run.)
- etc.

# WORD SENSE DISAMBIGUATION

Categories

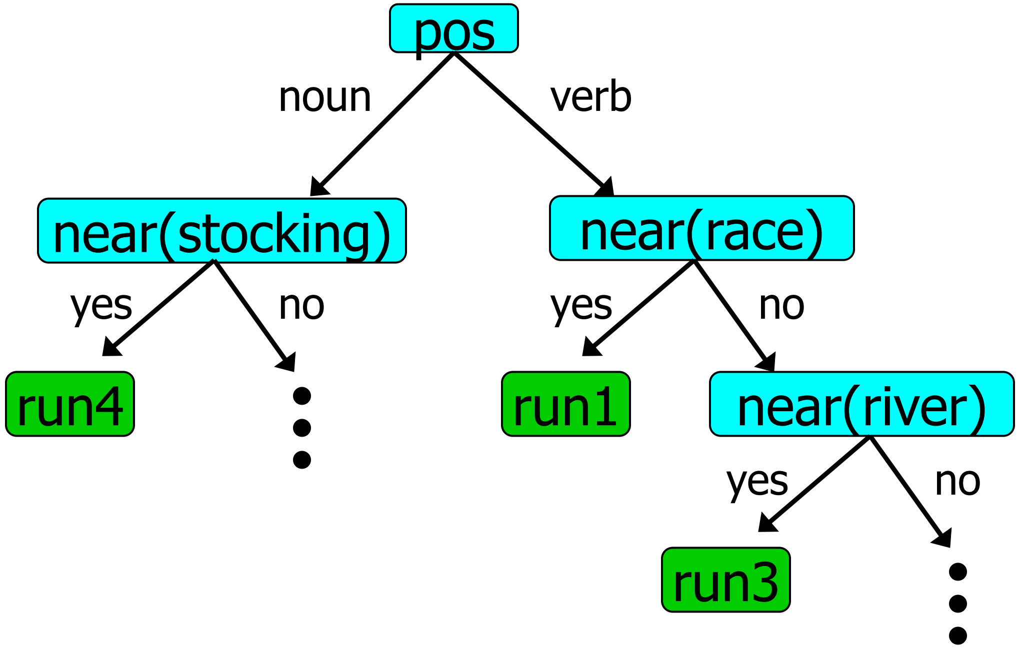
- Use word sense labels (run1, run2, etc.) to name the possible categories.

Features

- Features describe the *context* of the word we want to disambiguate.
- Possible features include:
  - near(w): is the given word near an occurrence of word w?
  - pos: the word's part of speech
  - left(w): is the word immediately preceded by the word w?
  - etc.

# WORD SENSE DISAMBIGUATION

Example decision tree:




(Note: Decision trees for WSD tend to be quite large)

# HOW IS A TREE CREATED?

- GINI impurity

- Random Forest uses the gini index taken from the CART learning system to construct decision trees. The gini index of node impurity is the measure most commonly chosen for classification-type problems. If a dataset T contains examples from n classes,

gini index, Gini(T) is defined as:

$$Gini(T) = 1 - \sum_{j=1}^{n} (p_j)^2$$

where pj is the relative frequency of class j in T.

# HOW IS A TREE CREATED?

- If a dataset T is split into two subsets T1 and T2 with sizes N1 and N2 respectively, the gini index of the split data contains examples from n classes, the gini index (T) is defined as:

$$Gini_{split}(T) = \frac{N_1}{N} gini(T_1) + \frac{N_2}{N} gini(T_2)$$

**The attribute value that provides the smallest SPLIT Gini (T) is chosen to split the node.

# ALTERNATIVELY YOU CAN USE

Entropy and Information Gain but they work with different algorithms

https://thatascience.com/learn-machine-learning/gini-entropy/#:~:text=Decision%20tree%20algorithms%20use%20information,thermodynamics%20where%20it%20signifies%20disorder.

# DECISION MAKING

- Knowing the ``when'' attribute values provides larger information gain than ``where''.
- Therefore the ``when'' attribute should be chosen for testing prior to the ``where'' attribute.
- Similarly, we can compute the information gain for other attributes.
- At each node, choose the attribute with the largest information gain.

# STRENGTH

- Can generate understandable rules
- Perform classification without much computation
- Can handle continuous and categorical variables
- Provide a clear indication of which fields are most important for prediction or classification

# WEAKNESS

- Not suitable for prediction of continuous attribute.
- Perform poorly with many class and small data.
- Computationally expensive to train.
  - At each node, each candidate splitting field must be sorted before its best split can be found.
  - In some algorithms, combinations of fields are used and a search must be made for optimal combining weights.
  - Pruning algorithms can also be expensive since many candidate sub-trees must be formed and compared.

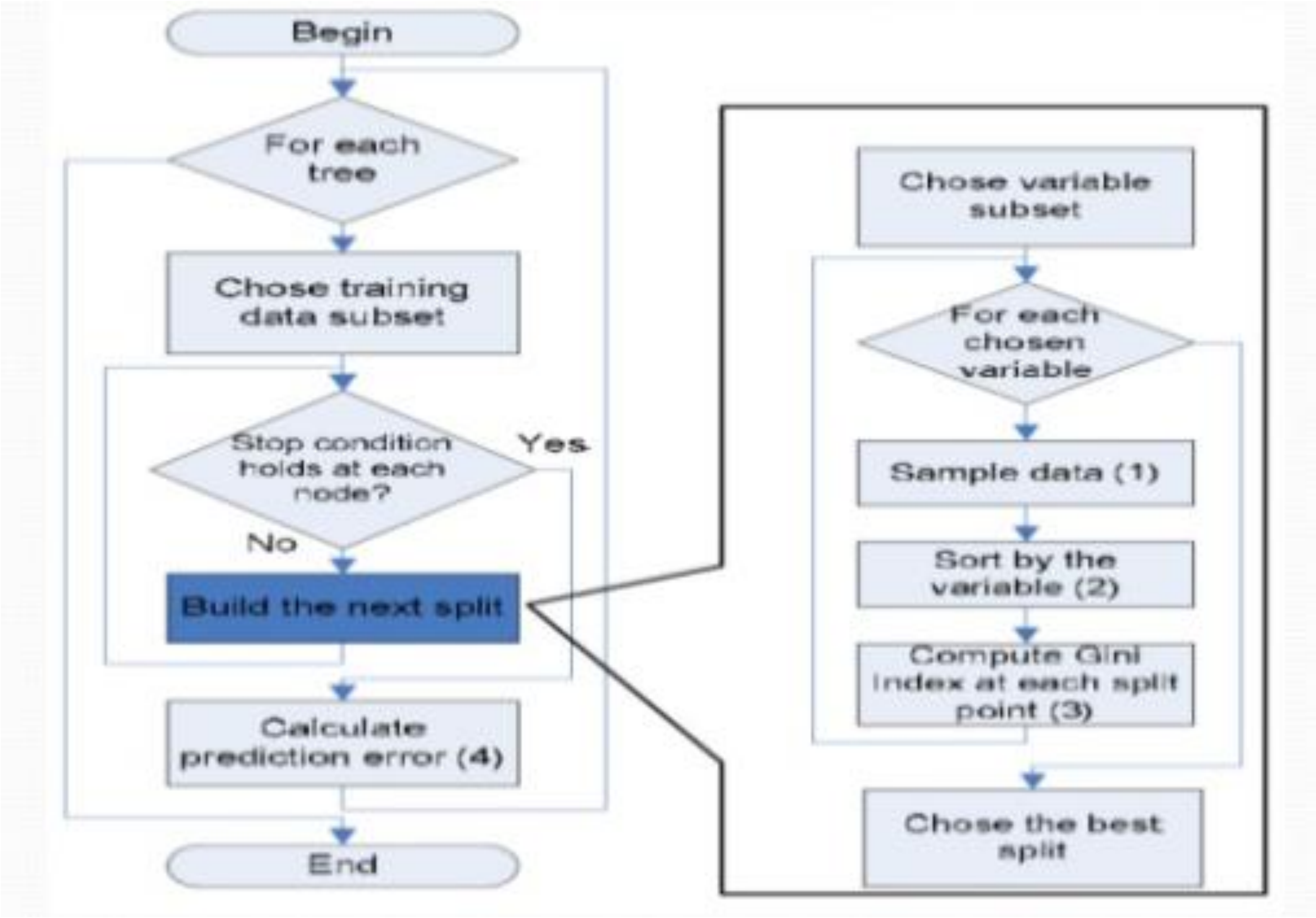
# RANDOM FORESTS

- An ensemble of decision trees.
  During learning tree nodes are split using a random subset of data features.
  All trees vote to produce a final answer.
- Why do this?
  - It was found that optimal cut points can depend strongly on the training set used  high variance
  - This led to the idea of using multiple trees to vote for a result.
  - For the use of multiple trees to be most effec9ve the trees should be independent as possible.
  - Splitting using a random subset of features hopefully achieves this.
  - Averaging the outputs of trees reduces overfinng to noise.
  - Pruning is not needed.

# RANDOM FORESTS

- Typically 5 – 100 trees are used.
- Often only a few trees are needed.
- Results seem fairly insensitive to the number of random attributes that are tested for each split.
- A common default is to use the square root of the number of attributes.
- Trees are fast to generate because fewer attributes have to be tested for each split and no pruning is needed.
- Memory needed to store the trees can be large.

# THE WORKING


Begin
For each tree
Chose training data subset
Stop condition holds at each node?
Yes
No
Build the next split
Calculate prediction error (4)
End
Chose variable subset
For each chosen variable
Sample data (1)
Sort by the variable (2)
Compute Gini index at each split point (3)
Chose the best split

# RANDOM FOREST

- A Random Forest is a classifier consisting of a collection of tree-structured classifiers {h(x, Θk ), k = 1....}where the Θk are independently, identically distributed random trees and each tree casts a unit vote for the final classification of input x. Like CART, Random Forest uses the gini index for determining the final class in each tree.

- The final class of each tree is aggregated and voted by weighted values to construct the final classifier.

# EXAMPLE – HOW DOES IT WORK?

1. A random seed is chosen which pulls out at random a collection of samples from the training dataset while maintaining the class distribution.

2. With this selected data set, a random set of attributes from the original data set is chosen based on user defined values. All the input variables are not considered because of enormous computation and high chances of over fitting.

# EXAMPLE – HOW DOES IT WORK?

3. In a dataset where M is the total number of input attributes in the dataset, only R attributes are chosen at random for each tree where R< M.

4. The attributes from this set creates the best possible split using the gini index to develop a decision tree model. The process repeats for each of the branches until the termination condition stating that leaves are the nodes that are too small to split.

# EXAMPLE – HOW DOES IT WORK?

- The example below shows the construction of a single tree using the abridged dataset .
- Only two of the original four attributes are chosen for this tree construction.

| RECORD | ATTRIBUTES | | CLASS |
|---|---|---|---|
| | HOME_TYPE | SALARY | |
| 1 | 31 | 3 | 1 |
| 2 | 30 | 1 | 0 |
| 3 | 6 | 2 | 0 |
| 4 | 15 | 4 | 1 |
| 5 | 10 | 4 | 0 |

# EXAMPLE – HOW DOES IT WORK?

- Assume that the first attribute to be split is HOME_TYPE attribute.
- The possible splits for HOME_TYPE attribute in the left node range from 6 <= x < 31, where x is the split value.
- All the other values at each split form the right child node. The possible splits for the HOME_TYPE attributes in the dataset are HOME_TYPE <=6, HOME_TYPE <=10, HOME_TYPE <= 15, HOME_TYPE <= 30, and HOME_TYPE <= 31.
- Taking the first split, the gini index is calculated as follows:

# EXAMPLE – HOW DOES IT WORK?

- Partitions after the Binary Split on HOME_TYPE <=6 by the Random Forest

| Attribute | Number of records | | |
|---|---|---|---|
| | Zero(0) | One (1) | N = 5 |
| HOME_TYPE <=6 | 1 | 0 | n1 = 1 |
| HOME_TYPE > 6 | 2 | 2 | n2 = 4 |

- Then Gini(D1) , Gini (D2) , and $Gini_{SPLIT}$ are calculated as follows:

# EXAMPLE – HOW DOES IT WORK?

Then $Gini(D_1)$, $Gini(D_2)$, and $Gini_{SPLIT}$ are calculated as follows:

$$Gini(HOME\_TYPE \leq 6) = 1 - (1^2 + 0^2) = 0$$

$$Gini(HOME\_TYPE > 6) = 1 - \left(\left(\frac{2}{4}\right)^2 + \left(\frac{2}{4}\right)^2\right) = 0.5$$

$$Gini_{SPLIT} = \left(\frac{1}{5}\right) \times 0 + \left(\frac{4}{5}\right) \times 0.5 = 0.4$$

# EXAMPLE – HOW DOES IT WORK?

- In the next step, the data set at HOME_TYPE <=10 is split and tabulated in Table.

- Partitions after the Binary Split on HOME_TYPE <=10 by the Random Forest:

| Attribute | Number of records | | |
|---|---|---|---|
| | Zero(0) | One (1) | N = 5 |
| HOME_TYPE <=10 | 2 | 0 | n1 = 2 |
| HOME_TYPE > 10 | 1 | 2 | n2 = 3 |

- Then Gini (D1) , Gini (D2) , and $\text{Gini}_{\text{SPLIT}}$ are calculated as follows:

# EXAMPLE – HOW DOES IT WORK?

$$Gini(HOME\_TYPE \leq 10) = 1 - (1^2 + 0^2) = 0$$

$$Gini(HOME\_TYPE > 10) = 1 - \left( \left( \frac{1}{3} \right)^2 + \left( \frac{2}{3} \right)^2 \right) = 0.4452$$

$$Gini_{SPLIT} = \left( \frac{2}{5} \right) \times 0 + \left( \frac{3}{5} \right) \times 0.4452 = 0.2671$$

# EXAMPLE – HOW DOES IT WORK?

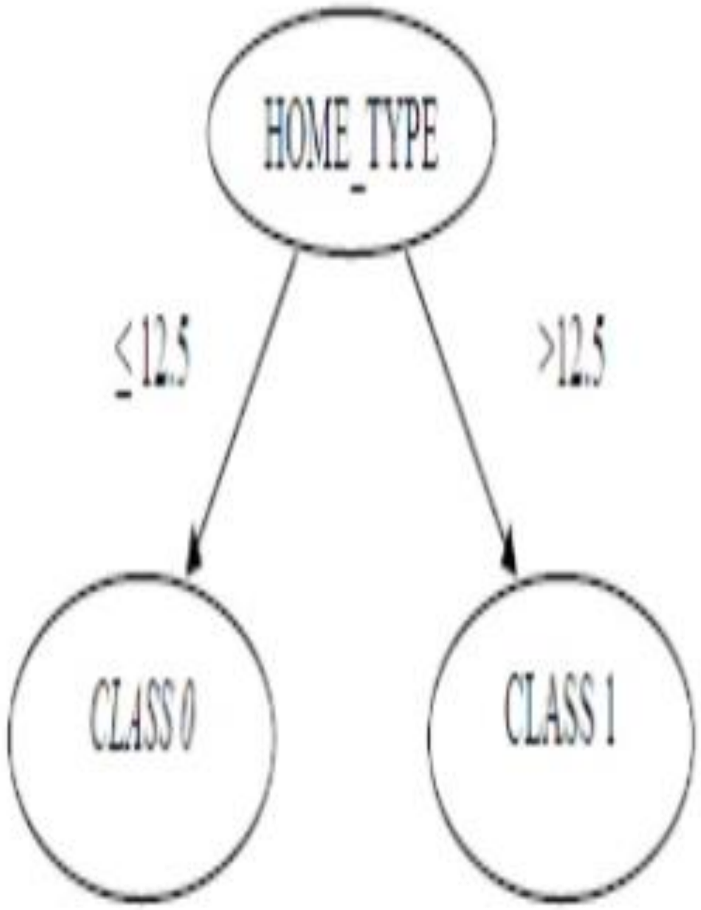
- tabulates the gini index value for the HOME_TYPE attribute at all possible splits.

| Gini SPILT | Value |
|---|---|
| Gini$_{SPLT}$(HOME_TYPE<=6) | 0.4000 |
| Gini$_{SPLT}$(HOME_TYPE<=10) | 0.2671 |
| Gini$_{SPLT}$(HOME_TYPE<=15) | 0.4671 |
| Gini$_{SPLT}$(HOME_TYPE<=30) | 0.3000 |
| Gini$_{SPLT}$(HOME_TYPE<=31) | 0.4800 |

- the split HOME_TYPE <= 10 has the lowest value

# EXAMPLE – HOW DOES IT WORK?

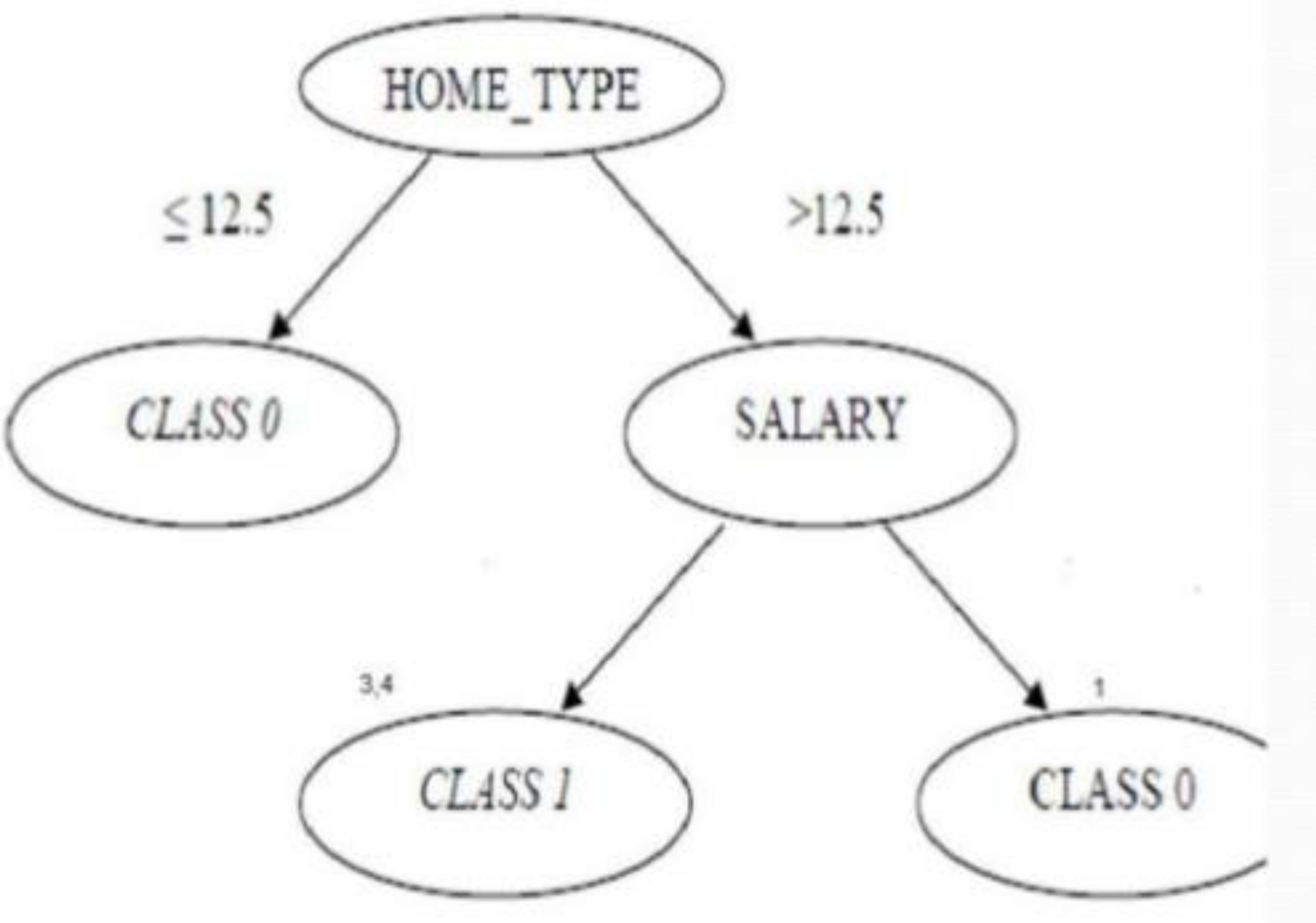
- In Random Forest, the split at which the gini index is lowest is chosen at the split value.
- However, since the values of the HOME_TYPE attribute are continuous in nature, the midpoint of every pair of consecutive values is chosen as the best split point.
- The best split in our example, therefore, is at HOME TYPE =(10+15)/2=12.5 instead of at HOME _TYPE<= 10 . The decision tree after the first split is shown in:



# EXAMPLE – HOW DOES IT WORK?

- This procedure is repeated for the remaining attributes in the dataset.
- In this example, the gini index values of the second attribute SALARY are calculated.
- The lowest value of the gini index is chosen as the best split for the attribute.
- The final decision trees shown in:

# EXAMPLE – HOW DOES IT WORK?



# EXAMPLE – HOW DOES IT WORK?

- The decision rules for the decision tree illustrated are:

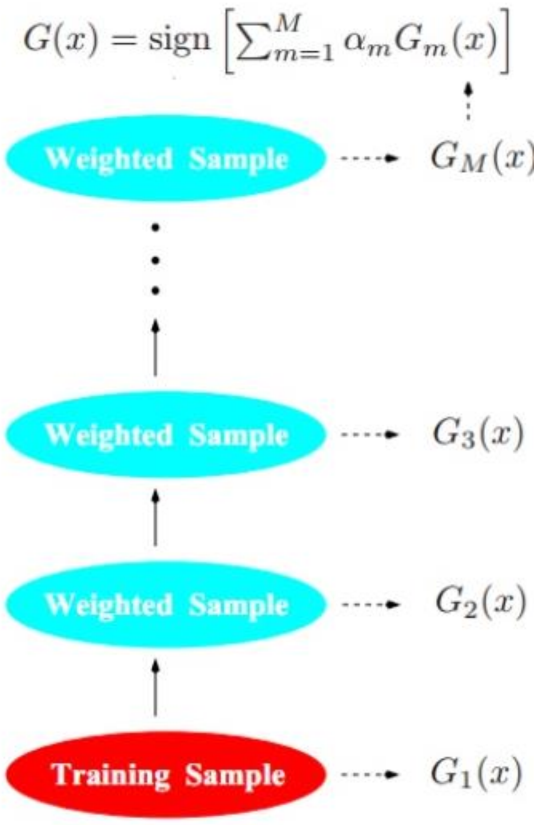
- If HOME_TYPE <= 12.5, then
  Class value is 0.

- If HOME_TYPE is > 12.5 and SALARY is 3/4, then
  Class value is 1.

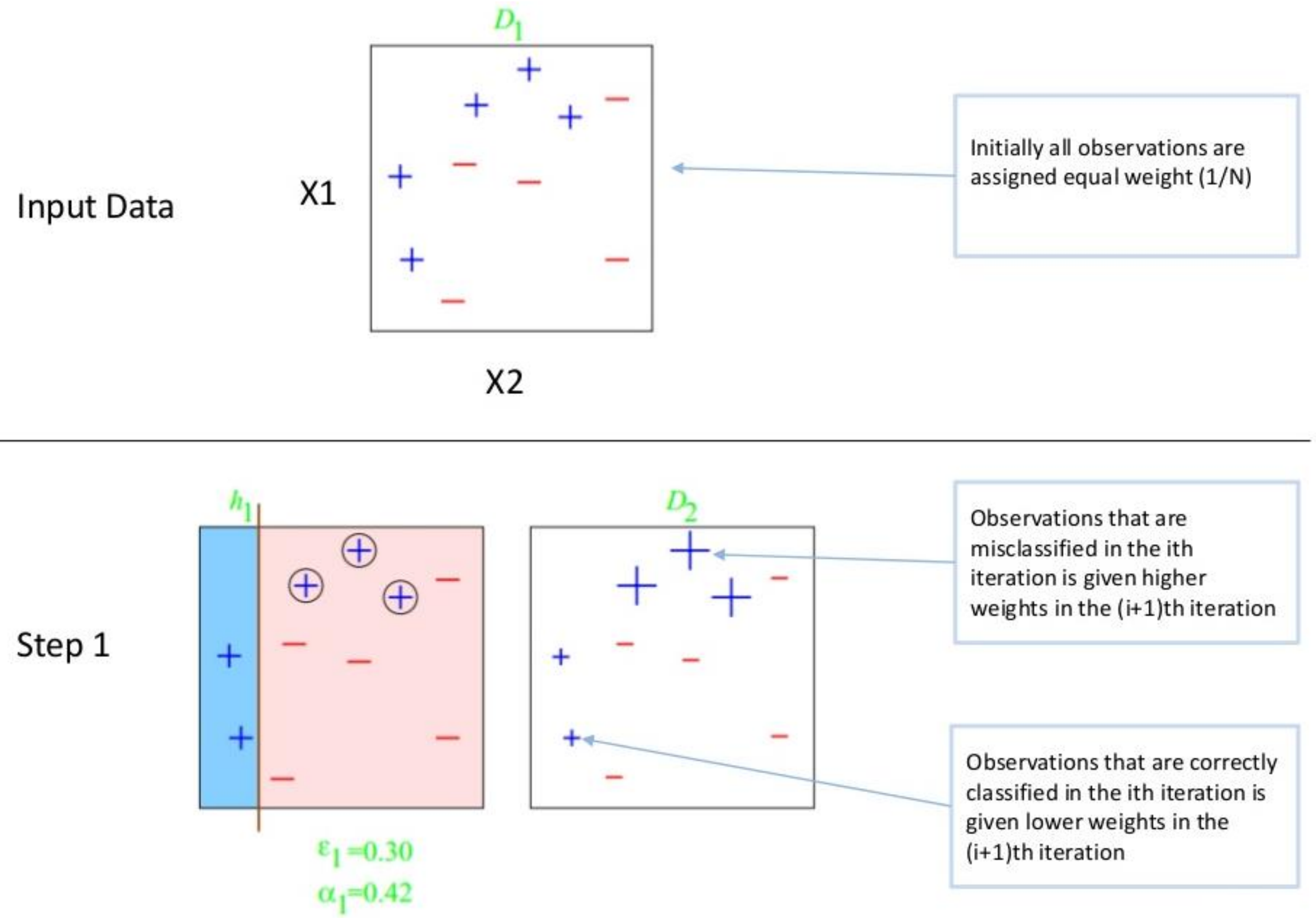
- If HOME_TYPE is > 12.5 and SALARY is 1, then Class value is 0.

# BOOSTING

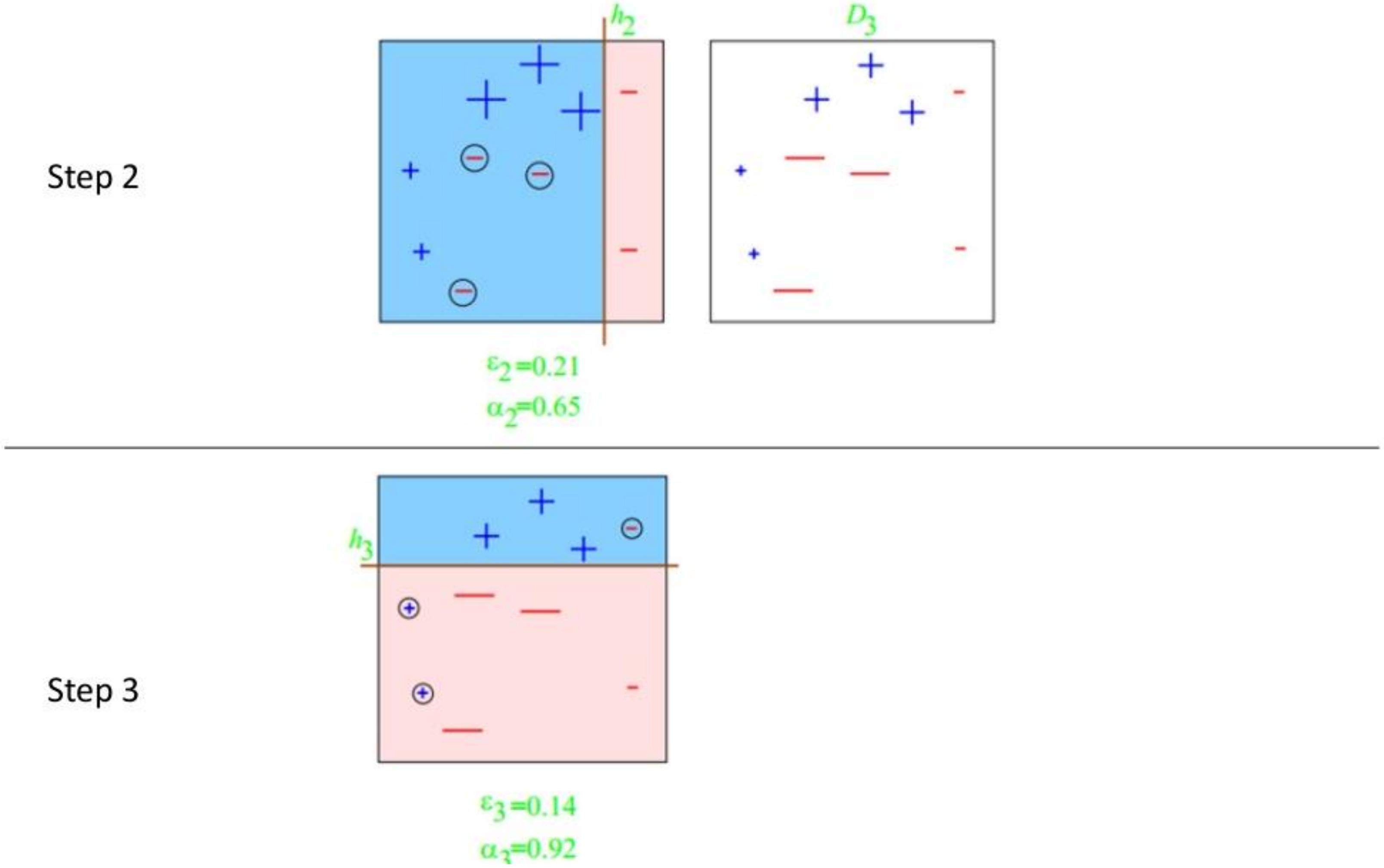
- Intuition: Ensemble many "weak" classifiers (typically decision trees) to produce a final "strong" classifier
  - Weak classifier → Error rate is only slightly better than random guessing.
- Boosting is a Forward Stagewise Additive model
- Boosting sequentially apply the weak classifiers one by one to repeatedly reweighted versions of the data.
- Each new weak learner in the sequence tries to correct the misclassification/error made by the previous weak learners.
  - Initially all of the weights are set to Wi = 1/N
  - For each successive step the observation weights are individually modified and a new weak learner is fitted on the reweighted observations.
  - At step m, those observations that were misclassified by the classifier Gm−1(x) induced at the previous step have their weights increased, whereas the weights are decreased for those that were classified correctly.
- Final "strong" classifier is based on weighted vote of weak classifiers



# BOOSTING

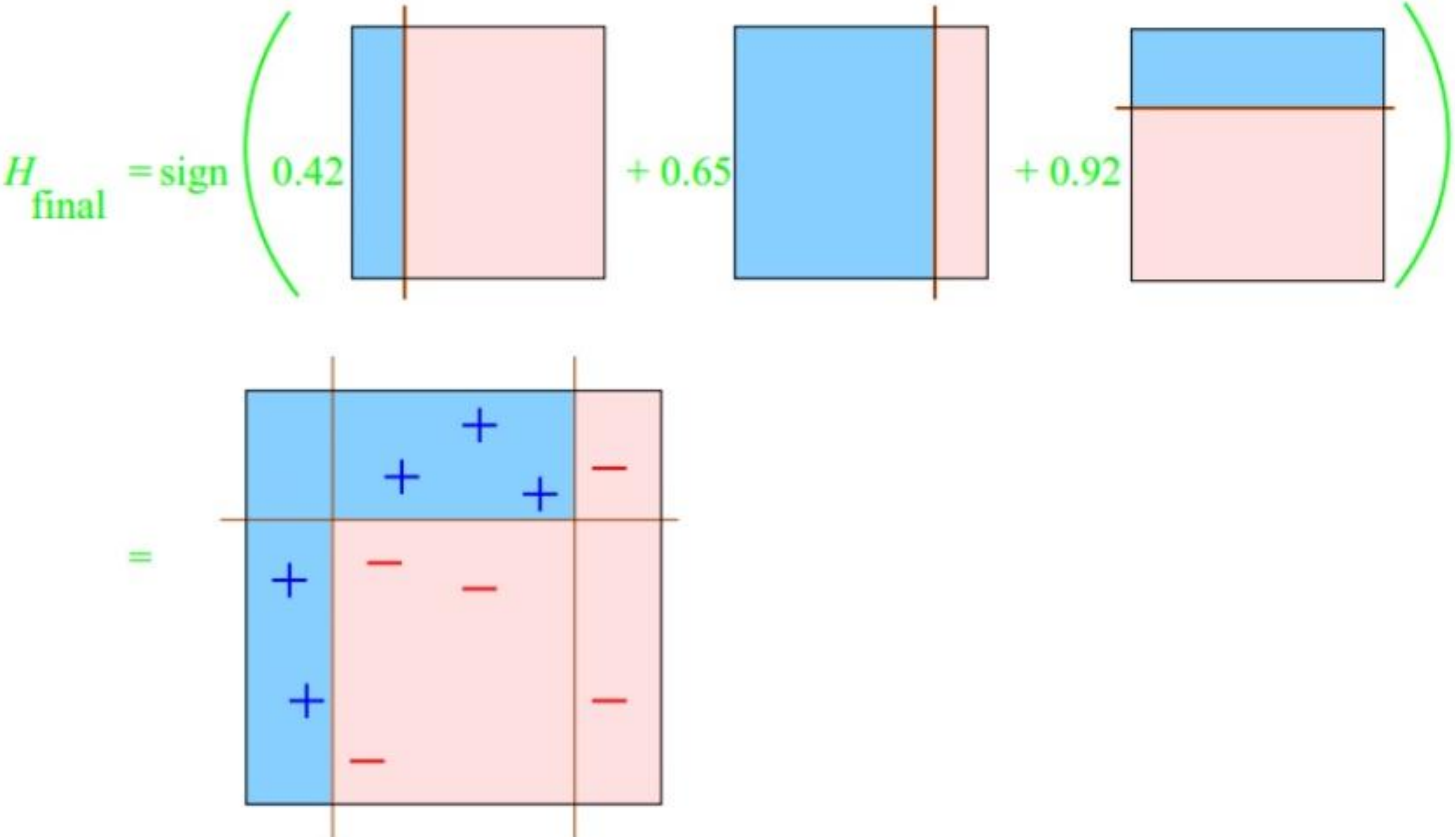

$D_1$
Input Data
X1
X2
Initially all observations are assigned equal weight (1/N)
Step 1
$h_1$
$\varepsilon_1$=0.30
$\alpha_1$=0.42
$D_2$
Observations that are misclassified in the ith iteration is given higher weights in the (i+1)th iteration
Observations that are correctly classified in the ith iteration is given lower weights in the (i+1)th iteration

# BOOSTING



# BOOSTING

Final Ensemble/Model



# ADDITIONAL RESOURCES

https://www.datasciencecentral.com/profiles/blogs/decision-tree-vs-random-forest-vs-boosted-trees-explained

https://victorzhou.com/blog/intro-to-random-forests/

https://web.csulb.edu/~tebert/teaching/lectures/551/random_forest.pdf

https://towardsdatascience.com/the-ultimate-guide-to-adaboost-random-forests-and-xgboost-7f9327061c4f

https://medium.com/analytics-vidhya/comparing-random-forest-and-xgboost-be98578479c3

https://analyticsindiamag.com/random-forest-vs-xgboost-comparing-tree-based-algorithms-with-codes/

# WHEN TO USE WHICH ALGO?

See text file.

# GET HOLISTIC UNDERSTANDING USING SINGLE DATASET

https://machinelearningmastery.com/compare-machine-learning-algorithms-python-scikit-learn/

# 1 IMAGE SUMMARIES

Attached

# SCIKIT LEARN VS. STATSMODELS

- https://blog.thedataincubator.com/2017/11/scikit-learn-vs-statsmodels/

# QUESTIONS?

Thank you!