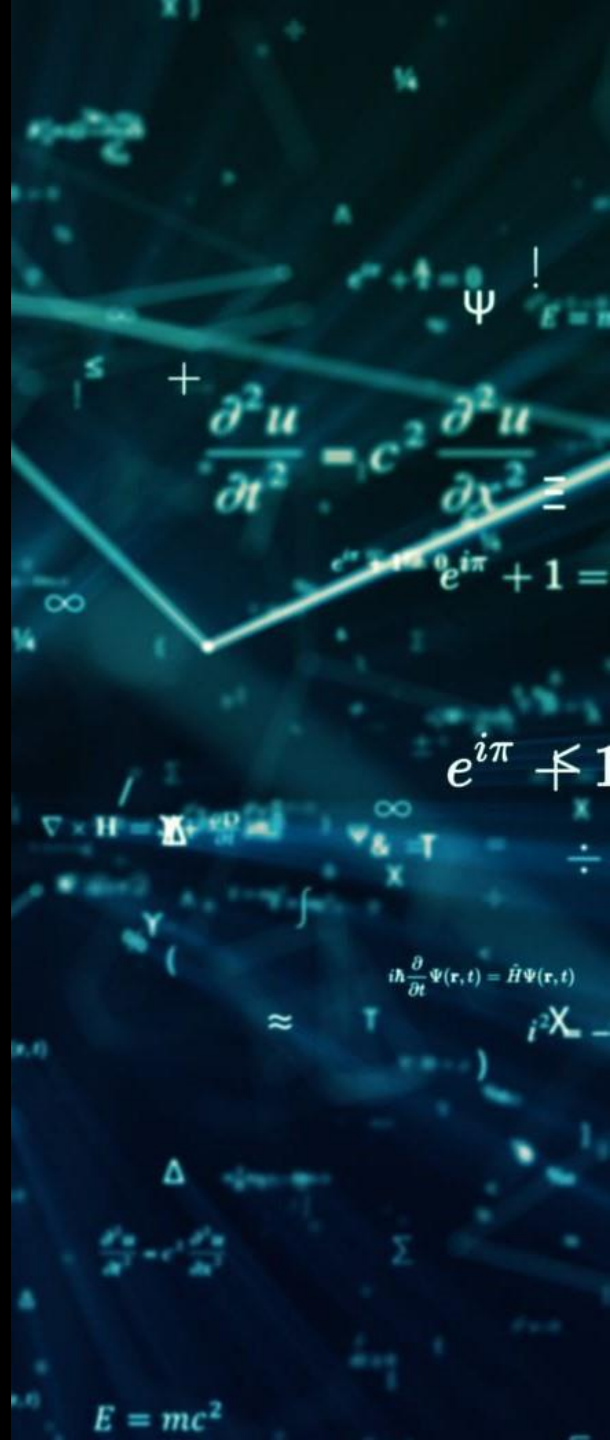
Artificial Intelligence Algorithms and Mathematics

CSCN 8000

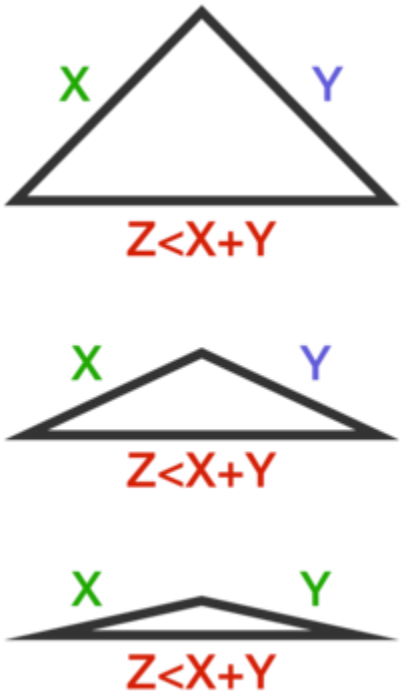


Classification

- KNN
- Decision Trees
- Tree Ensembles

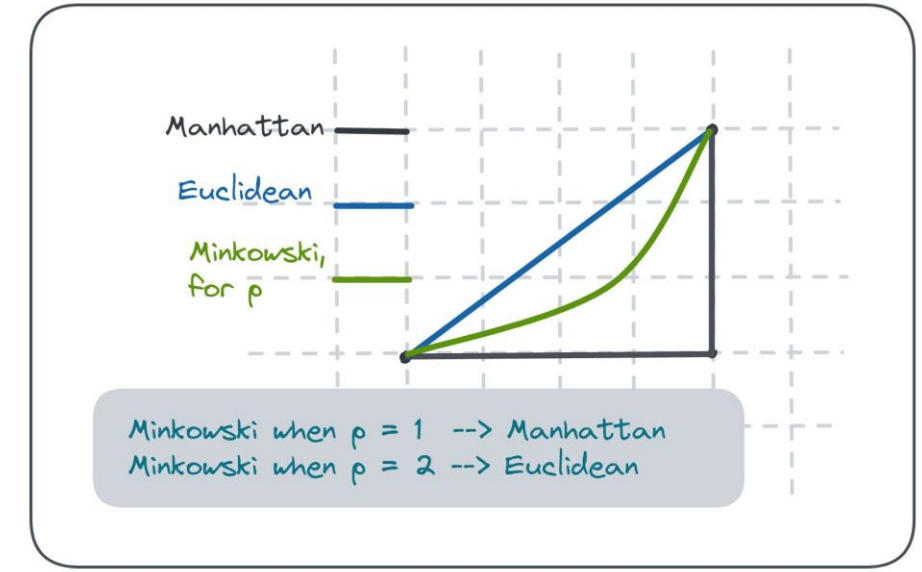
Distance Metrics

- Given two data points a and b in a dataset with M features, we need to quantify how similar/dissimilar those points are from each other.
- A distance metric $d(a, b)$ measures how far or close two points are from each other. All distance metrics must follow the following criteria:
  - Non-negativity: $d(a, b) > 0$
  - Identity: if $d(a, b) = 0, then\ a = b$
  - Symmetry: $d(a, b) = d(b, a)$
  - Triangular Inequality: $d(a, b) \leq d(a, c) + d(b, c)$



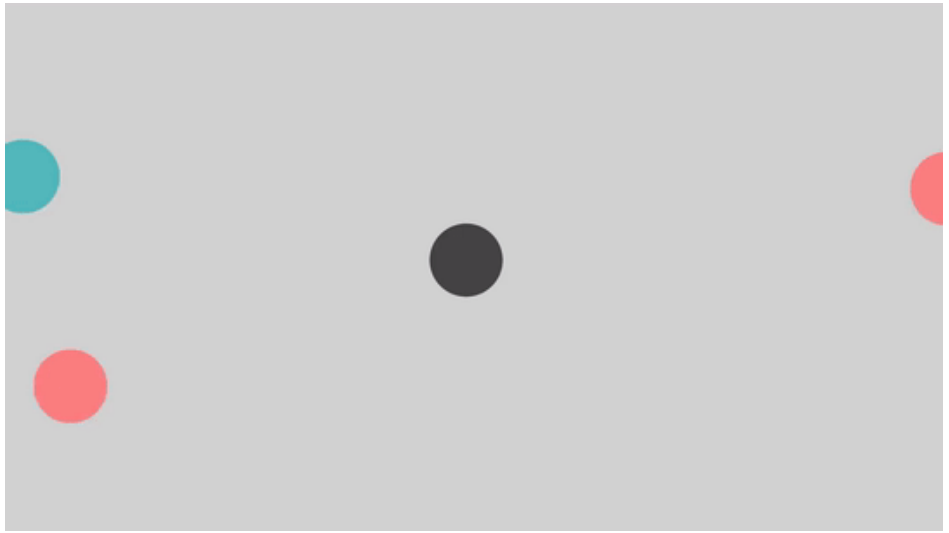
Distance Metrics

- Euclidean Distance:
  - Computes Length of the straight line between two points.
  - Formulation: $d(a,b) = \sqrt{\sum_{i=1}^{M}(a[i]-b[i])^2}$
  - Drawbacks: Could be sensitive to outliers.
- Manhattan Distance:
  - Computes orthogonal (grid line) distances between two points.
  - Formulation: $d(a,b) = \sum_{i=1}^{M}|a[i]-b[i]|$
  - Pros: Less sensitive to outliers.
- Minkowski Distance:
  - Generalization of both Euclidean and Manhattan distance between two points.
  - Formulation: $d(a,b) = \left(\sum_{i=1}^{M}|a[i]-b[i]|^p\right)^{1/p}$
  - Pros: Allows for tuning sensitivity to different dimensions through the parameter p.



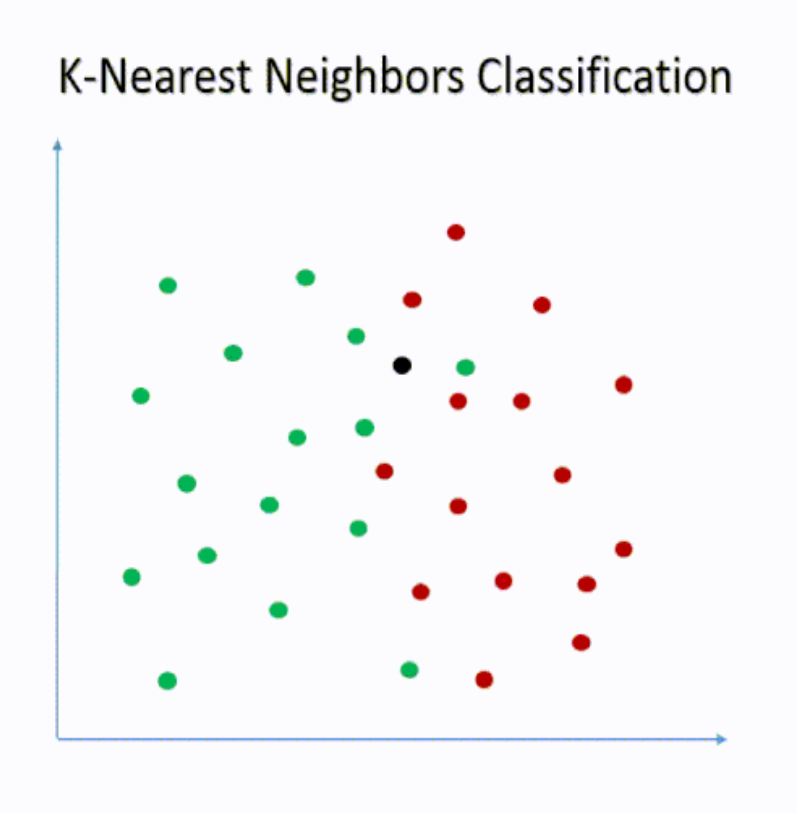
K-Nearest Neighbours (KNN)

- KNN is a **non-linear** machine learning algorithm that makes predictions by finding the K training examples in the dataset that are closest (in terms of some distance metric) to the input data point.
- It uses the labels of those K neighbors to make a prediction.
- K is a manually defined hyperparameter that determines the number of neighbors to observe for each new point.



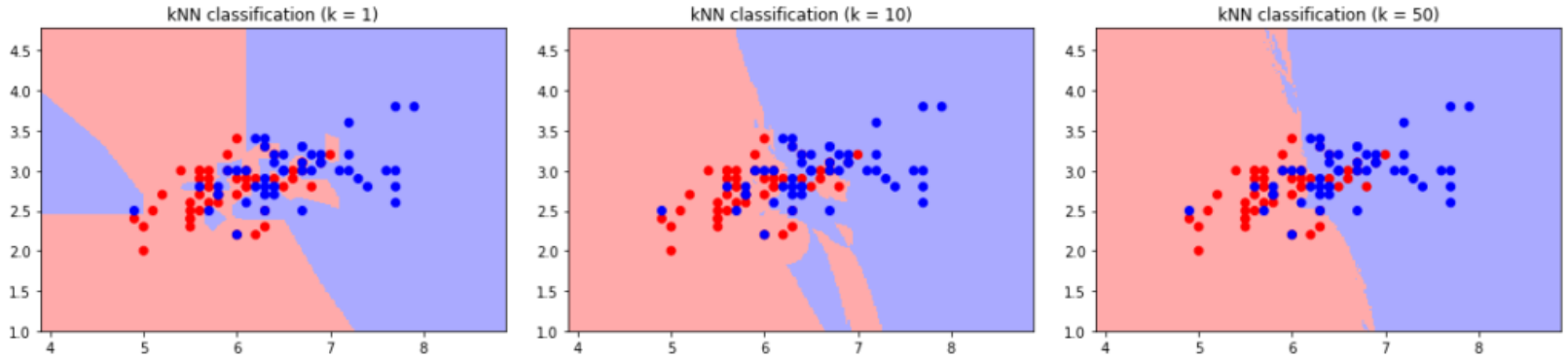
K-Nearest Neighbours (KNN)

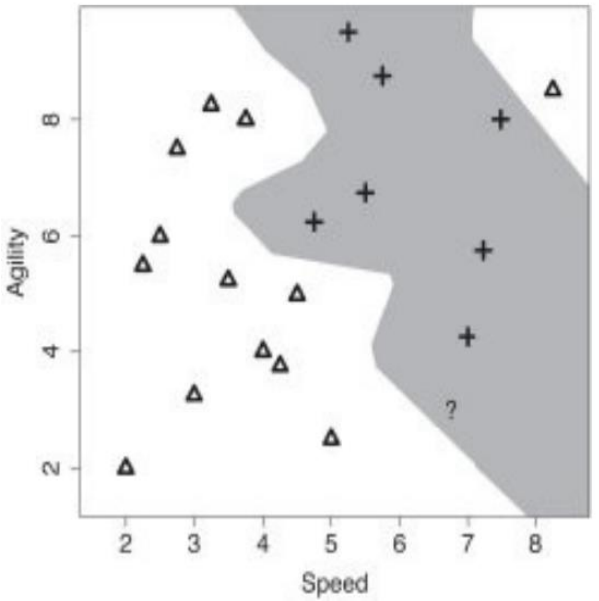
- Given a new query point x to be labeled, a dataset with N data points and a hyperparameter K:
  - Calculate distance $d_i(x, n_i)$ between point x and all points n_i in the dataset.
  - Determine the nearest K points to the point x to be labeled → K points with the smallest distances $d_i(x, n_i)$.
  - The point x will be labeled based on the majority label of the K nearest neighbours.



K-Nearest Neighbours (KNN)

- The choice of the hyperparameter K significantly influences the shape and flexibility of the decision boundary in K-NN.
  - Smaller K:
    - The decision boundary can be highly irregular
    - The model may be sensitive to outliers and local variations in the data.
    - Can lead to overfitting, capturing noise in the data rather than the underlying patterns.
  - Larger K:
    - The decision boundary tends to be smoother
    - Provides a more generalized view of the data, potentially avoiding overfitting

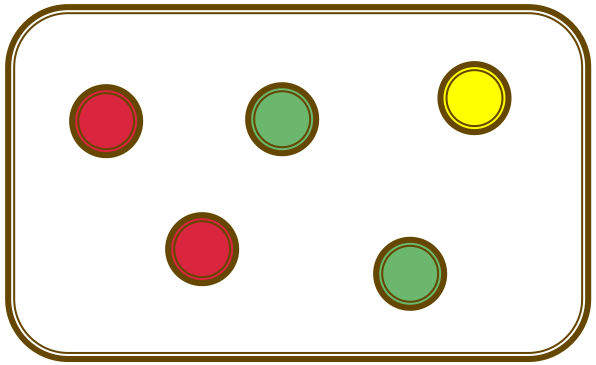




Tree-Based Algorithms

Entropy

- Entropy (H) is used to represent *the amount of uncertainty* that a particular event will occur.
- For example, consider a box with two red balls and two green balls and two yellow balls.
  - Assuming no yellow balls, What is the probability of drawing a red ball?
    - Probability → 50%
    - Uncertainty (intuitively) → $High$
  - What is the probability of drawing a colored ball?
    - Probability → 100%
    - Uncertainty (intuitively) → $Zero$
  - What is the probability of drawing a red or green ball?
    - Probability → 80%
    - Uncertainty (intuitively) → Low
  - What is the probability of drawing a purple ball?
    - Probability → 0%
    - Uncertainty (intuitively) → $Infinitely\ High$



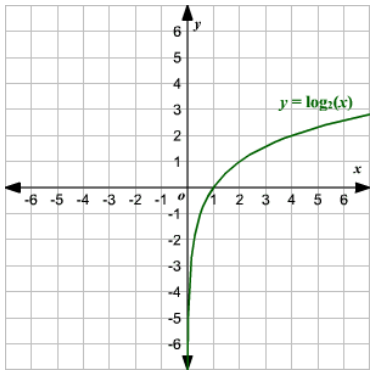
Entropy

- Given a probability p that an event will occur, can we mathematically formulate the entropy/uncertainty (H) ?
  - If $p = 1.0 \rightarrow Entropy = 0$
  - If $p = 0.0 \rightarrow Entropy \rightarrow \infty$
  - If $p = 0.5 \rightarrow Entropy = 1.0$
  - If $p > 0.5 \rightarrow Entropy < 1.0$
  - If $p < 0.5 \rightarrow Entropy > 1.0$

$H = \log(p)$?

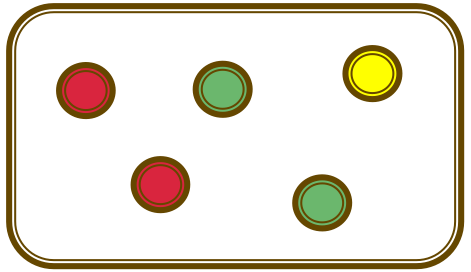
Need to handle –ve log(p) to have +ve Entropy

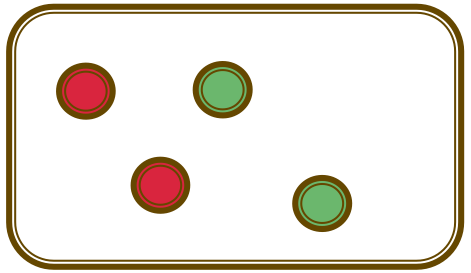
$$H = -\log(p)$$

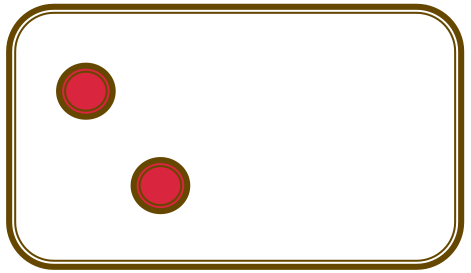


Entropy

- For the previous box example, is there a way to quantify the overall entropy/uncertainty (H) of the whole box?
  - $H(Box) = P(R)H(R) + P(G)H(G) + P(Y)H(Y)$
  - $H(Box) = \frac{2}{5} * -\log\left(\frac{2}{5}\right) + \frac{2}{5} * -\log\left(\frac{2}{5}\right) + \frac{1}{5} * -\log\left(\frac{1}{5}\right)$
  - $H(Box) = 2.3218$
- What if the box had no yellow balls?
  - $H(Box) = P(R)H(R) + P(G)H(G)$
  - $H(Box) = \frac{1}{2} * -\log\left(\frac{1}{2}\right) + \frac{1}{2} * -\log\left(\frac{1}{2}\right)$
  - $H(Box) = 1.0$
- What if the box had only red balls?
  - $H(Box) = P(R)H(R)$
  - $H(Box) = \frac{1}{1} * -\log\left(\frac{1}{1}\right)$
  - $H(Box) = 0.0$







Entropy

- Generally, for a dataset with N classes, the entropy of the full dataset represents that *amount of impurity* in the labels. It is formulated as follows:

$$H(Dataset) = -\sum_{i=1}^{N} P(c_i) * \log_2 P(c_i)$$

- This formulation is called Shannon's model of entropy and represents the foundation of information theory, providing a great measure of impurity.

Decision Trees: Intuition

- Assume we have 4 different people with the features shown below. Can we determine (classify) the name of each person by asking a sequence of questions based on the features?

Brian

John

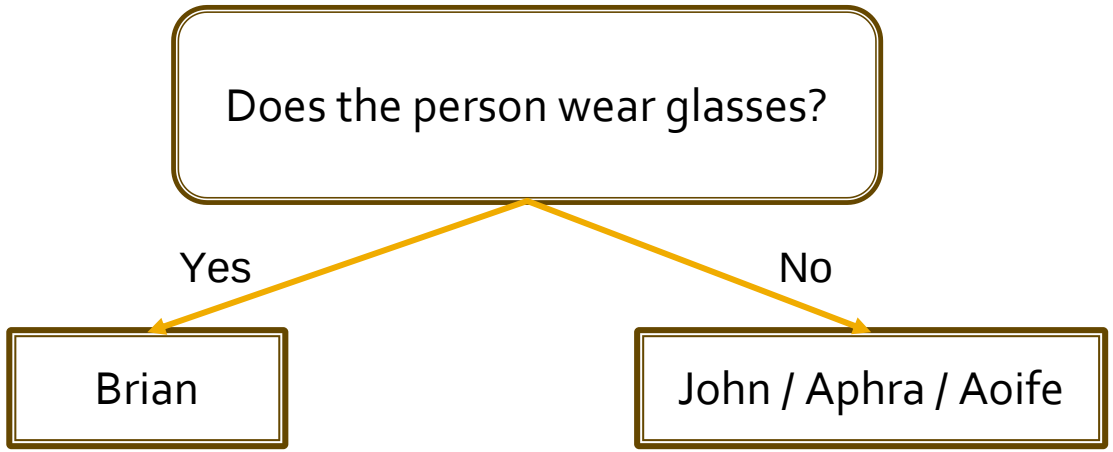
Aphra

Aoife

Man	Long Hair	Glasses	Name
Yes	No	Yes	**Brian**
Yes	No	No	**John**
No	Yes	No	**Aphra**
No	No	No	**Aoife**

Decision Trees: Intuition

- Let's start by asking the following question: "*Does the person wear glasses?*"

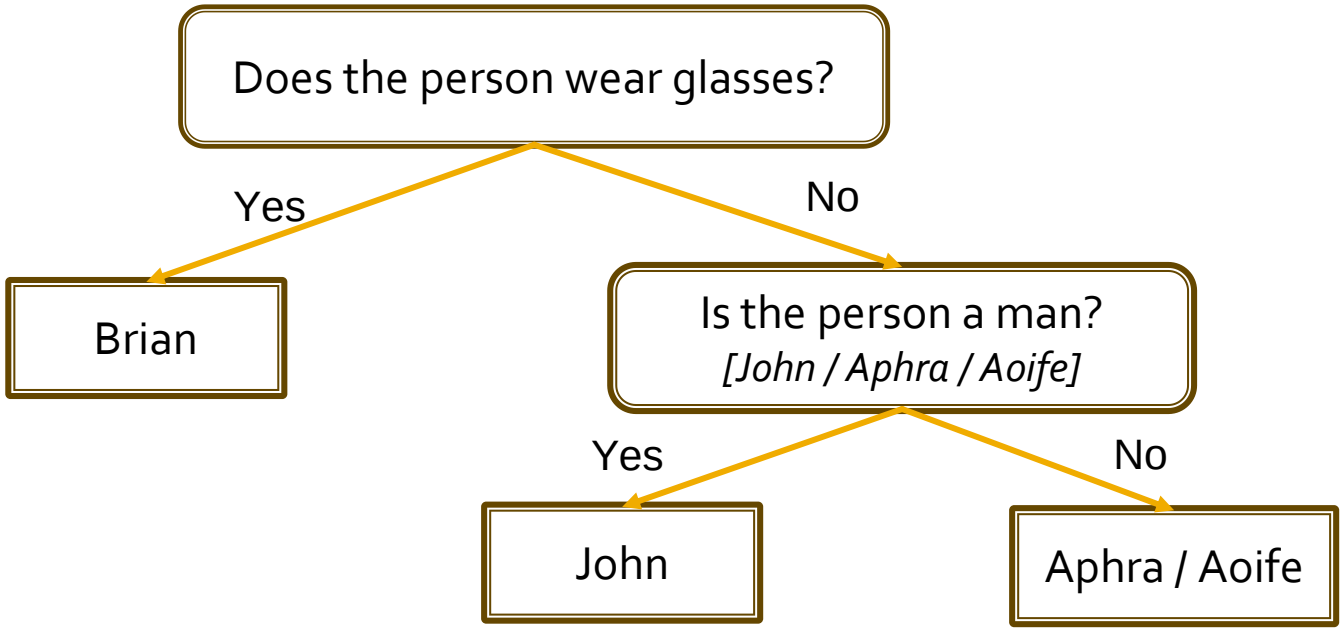


Man	Long Hair	Glasses	Name
Yes	No	Yes	**Brian**
Yes	No	No	**John**
No	Yes	No	**Aphra**
No	No	No	**Aoife**

- "Brian" was correctly classified, three names are remaining for us.

Decision Trees: Intuition

- For the three remaining people, we can ask about the gender: "*Is the person a man?*"

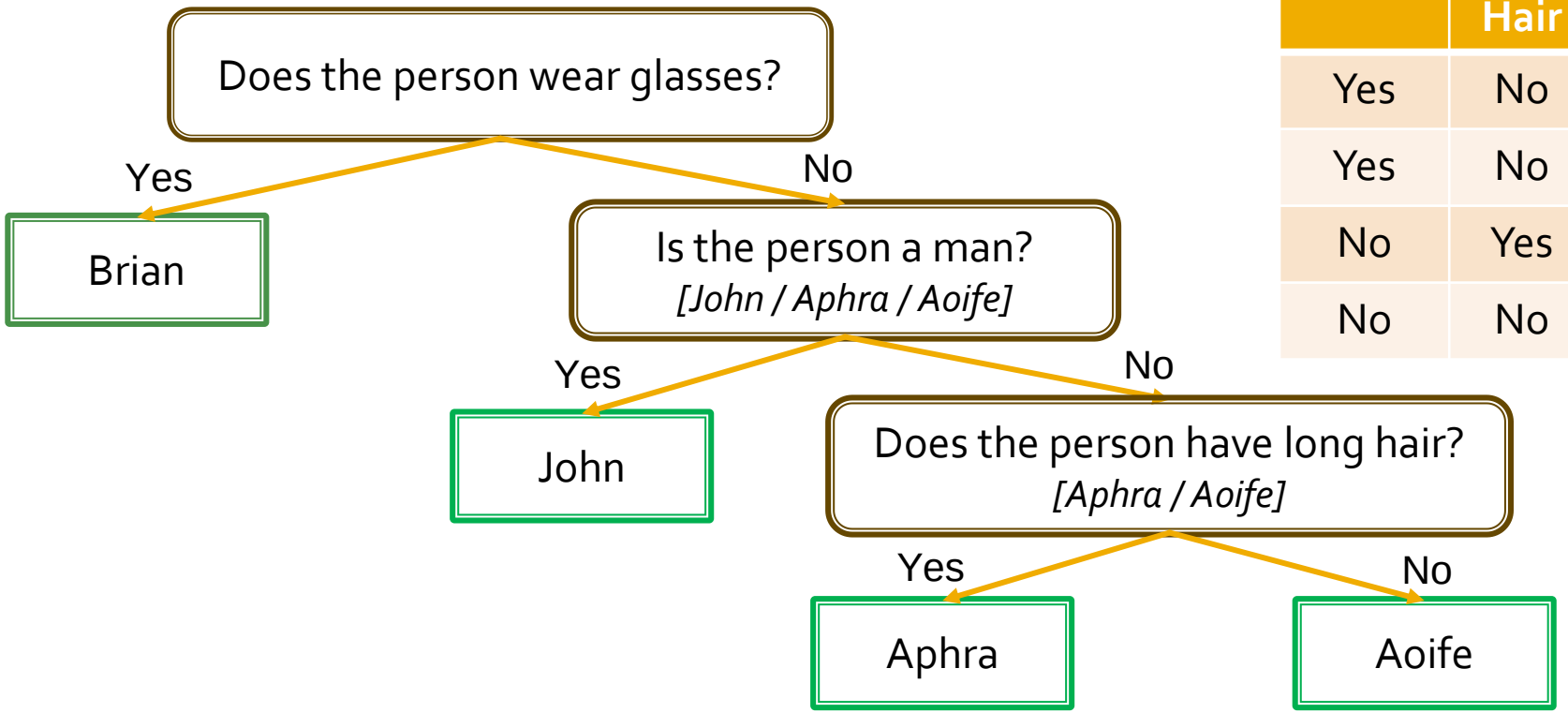


Man	Long Hair	Glasses	Name
Yes	No	Yes	**Brian**
Yes	No	No	**John**
No	Yes	No	**Aphra**
No	No	No	**Aoife**

- "Brian" and "John" are now correctly classified, two names are remaining for us.

Decision Trees: Intuition

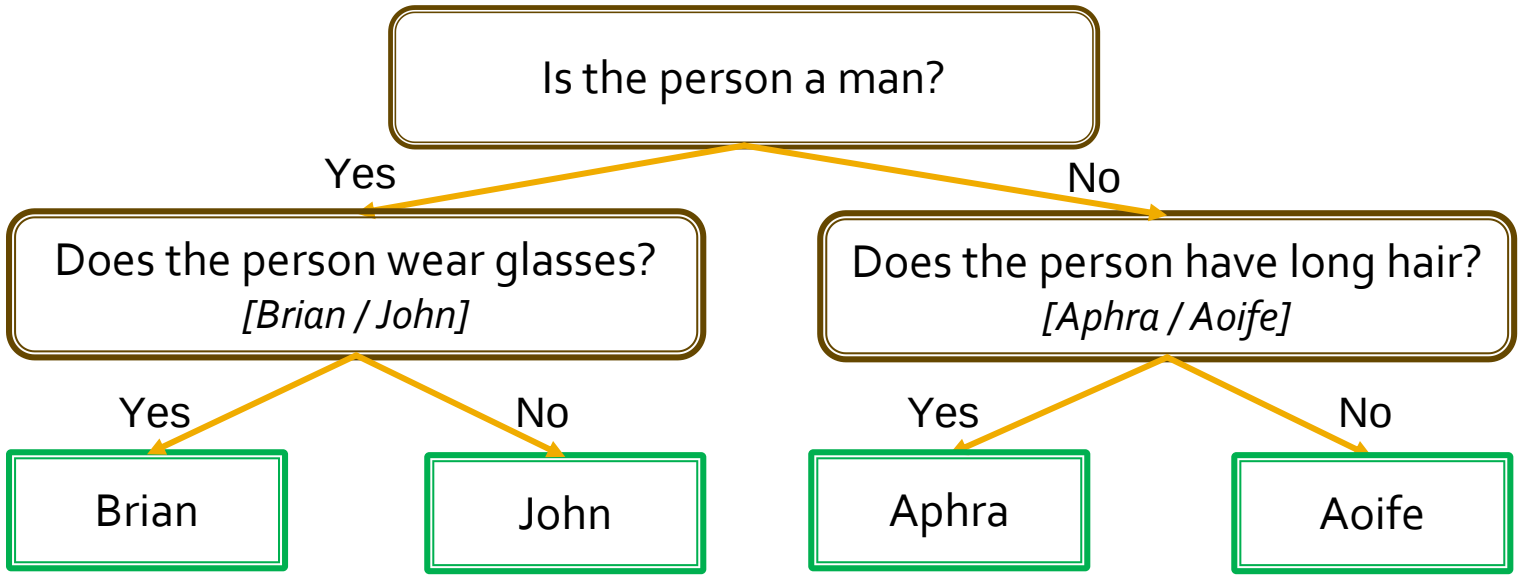
- For the two remaining people, we can ask about the hair: "*Does the person have long hair?*"



Man	Long Hair	Glasses	Name
Yes	No	Yes	**Brian**
Yes	No	No	**John**
No	Yes	No	**Aphra**
No	No	No	**Aoife**

Decision Trees: Intuition

- Is it possible to reach the same classification with less question sequence length *per name*?

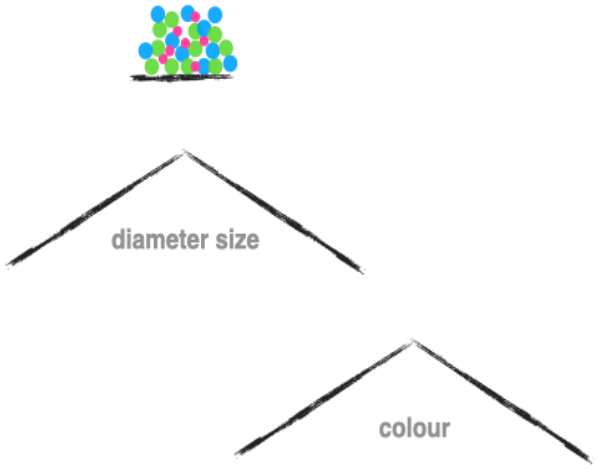


Man	Long Hair	Glasses	Name
Yes	No	Yes	**Brian**
Yes	No	No	**John**
No	Yes	No	**Aphra**
No	No	No	**Aoife**

- Depending on which feature we choose first, we could end up building a shorter tree with less questions to classify each person.
- **Is there a systematic way we can always choose the best feature (question) in every step to reduce the total number of questions per label?**

Decision Tree

- The decision tree is a **non-linear** supervised machine learning algorithm.
- It recursively splits the dataset into subsets based on the most significant attribute at each level.
- It keeps on splitting the dataset based on different attributes, creating a tree-like structure.
- The main goal is to reach a final tree that is as *shallow* as possible and has *homogenous classes* in the leaf nodes (bottom-most nodes of the tree).



Decision Tree

- Consider the following dataset S which predicts whether a person should go for a picnic or not based on weather features.
- Assume we want to build a **decision tree** to do this classification.
- Which is the best feature that we should start splitting by?
  - To calculate the best feature, we will use the concept of **Entropy** to build a new metric called **Information Gain.**

Temp (T)	Hum (H)	Wind (W)	Outside (O)
Hot	High	Weak	**No**
Mild	Normal	Strong	**No**
Hot	High	Strong	**No**
Mild	Normal	Weak	**Yes**
Hot	Normal	Weak	**Yes**
Cool	High	Weak	**No**
Cool	Normal	Strong	**Yes**
Mild	High	Weak	**Yes**
Cool	Normal	Weak	**Yes**

Information Gain

- **Information Gain (IG)** is the reduction in entropy H (impurity) achieved by splitting a set into subsets based on a particular feature F.
- In other words, it measures how much will the total impurity of the classes decrease after splitting the dataset based on a particular feature.
- Mathematically:

$$IG(F) = H(Dataset) - Entropy\ of\ Dataset\ after\ split\ on\ F$$

$$IG(F) = H(Dataset) - \sum_{v\ in\ Values\ (F)} P(F = v) * H(Dataset \mid F = v)$$

- Generally, we want to choose the feature the gives us the **highest** information gain. This is because it is the feature leading to the highest decrease in impurity in the classes.

Decision Tree

- Let's calculate the Information Gain for the Temp/Hum/Wind Features and choose the best one accordingly.
- $H(S) = -[P(O = Yes) * \log(O = Yes) + P(O = No) * \log(O = No)]$

$$H(S) = -\left[\frac{5}{9} * \log\left(\frac{5}{9}\right) + \frac{4}{9} * \log\left(\frac{4}{9}\right)\right] = 0.991$$

- $IG(T) = H(S) - [P(T = Hot) * H(S|T = Hot) + P(T = Mild) * H(S|T = Mild) + P(T = Cool) * H(S|T = Cool)]$
- $IG(T) = 0.991 - [\frac{3}{9} * \left(-\left[\frac{1}{3} * \log\left(\frac{1}{3}\right) + \frac{2}{3} * \log\left(\frac{2}{3}\right)\right]\right) +$

$$\frac{3}{9} * \left(-\left[\frac{1}{3} * \log\left(\frac{1}{3}\right) + \frac{2}{3} * \log\left(\frac{2}{3}\right)\right]\right) +$$

$$\frac{3}{9} * \left(-\left[\frac{2}{3} * \log\left(\frac{2}{3}\right) + \frac{1}{3} * \log\left(\frac{1}{3}\right)\right]\right)] = 0.0727$$

Temp (T)	Hum (H)	Wind (W)	Outside (O)
Hot	High	Weak	**No**
Mild	Normal	Strong	**No**
Hot	High	Strong	**No**
Mild	Normal	Weak	**Yes**
Hot	Normal	Weak	**Yes**
Cool	High	Weak	**No**
Cool	Normal	Strong	**Yes**
Mild	High	Weak	**No**
Cool	Normal	Weak	**Yes**

Decision Tree

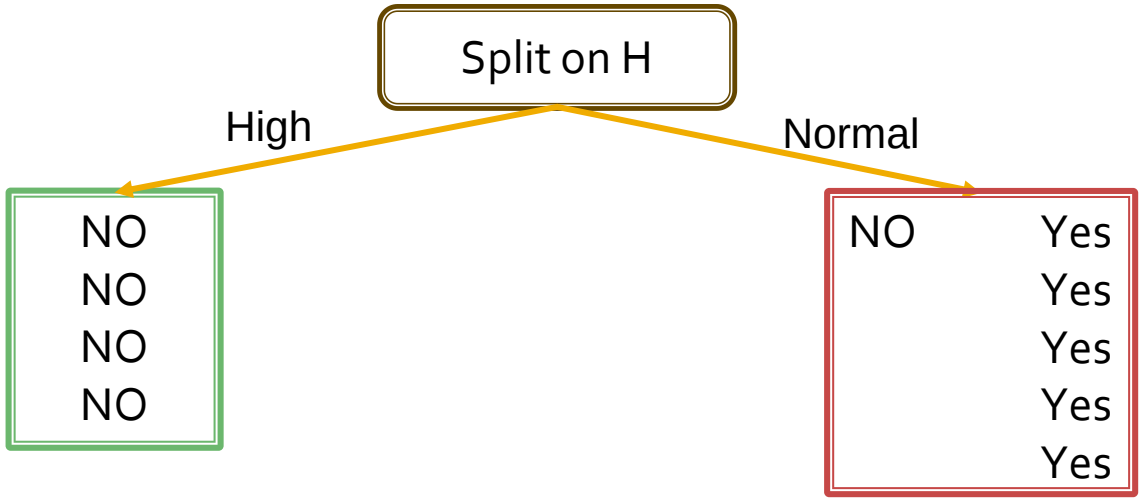
- Let's calculate the Information Gain for the Temp/Hum/Wind Features and choose the best one accordingly.

Temp (T)	Hum (H)	Wind (W)	Outside (O)
Hot	High	Weak	**No**
Mild	Normal	Strong	**No**
Hot	High	Strong	**No**
Mild	Normal	Weak	**Yes**
Hot	Normal	Weak	**Yes**
Cool	High	Weak	**No**
Cool	Normal	Strong	**Yes**
Mild	High	Weak	**No**
Cool	Normal	Weak	**Yes**

- $IG(H) = H(S) - [P(H = High) * H(S|H = High) + P(H = Normal) * H(S|H = Normal)]$
- $IG(H) = 0.991 - [\frac{4}{9} * \left(-\left[\frac{0}{4} * \log\left(\frac{0}{4}\right) + \frac{4}{4} * \log\left(\frac{4}{4}\right)\right]\right) + \frac{5}{9} * \left(-\left[\frac{4}{5} * \log\left(\frac{4}{5}\right) + \frac{1}{5} * \log\left(\frac{1}{5}\right)\right]\right) = 1.1699$
- $IG(W) = H(S) - [P(W = Weak) * H(S|W = Weak) + P(W = Strong) * H(S|W = Strong)]$
- $IG(W) = 0.991 - [\frac{6}{9} * \left(-\left[\frac{3}{6} * \log\left(\frac{3}{6}\right) + \frac{3}{6} * \log\left(\frac{3}{6}\right)\right]\right) + \frac{3}{9} * \left(-\left[\frac{1}{3} * \log\left(\frac{1}{3}\right) + \frac{2}{3} * \log\left(\frac{2}{3}\right)\right]\right) = 0.9183$

Decision Tree

- Based on the previous calculations, it looks like Humidity (H) is the best feature to split on in the initial step as it has the highest Information Gain.



Temp (T)	Hum (H)	Wind (W)	Outside (O)
Hot	High	Weak	**No**
Mild	Normal	Strong	**No**
Hot	High	Strong	**No**
Mild	Normal	Weak	**Yes**
Hot	Normal	Weak	**Yes**
Cool	High	Weak	**No**
Cool	Normal	Strong	**Yes**
Mild	High	Weak	**No**
Cool	Normal	Weak	**Yes**

- "High" branch is now fully homogenous, no need to further split it.
- "Normal" branch is not yet homogenous, we need to split it further. We will repeat the same process to choose between "Wind" and "Temp" features.

Decision Tree

- Our dataset now is only consisting of the split where "$Hum = Normal$". We will choose between T and W.
- $H(S|H = Normal) = -[P(O = Yes) * \log(O = Yes) + P(O = No) * \log(O = No)]$
- $H(S|H = Normal) = -\left[\frac{4}{5} * \log\left(\frac{4}{5}\right) + \frac{1}{5} * \log\left(\frac{1}{5}\right)\right] = 0.720$
- $IG(T) = H(S|H = Normal) - [P(T = Hot) * H(S|T = Hot) + P(T = Mild) * H(S|T = Mild) + P(T = Cool) * H(S|T = Cool)]$
- $IG(T) = 0.720 - [\frac{1}{5} * \left(-\left[\frac{1}{1} * \log\left(\frac{1}{1}\right) + \frac{0}{1} * \log\left(\frac{0}{1}\right)\right]\right) + \frac{2}{5} * \left(-\left[\frac{1}{2} * \log\left(\frac{1}{2}\right) + \frac{1}{2} * \log\left(\frac{1}{2}\right)\right]\right) + \frac{2}{5} * \left(-\left[\frac{2}{2} * \log\left(\frac{2}{2}\right) + \frac{0}{2} * \log\left(\frac{0}{2}\right)\right]\right)] = 0.320$

Temp (T)	Hum (H)	Wind (W)	Outside (O)
Mild	Normal	Strong	**No**
Mild	Normal	Weak	**Yes**
Hot	Normal	Weak	**Yes**
Cool	Normal	Strong	**Yes**
Cool	Normal	Weak	**Yes**

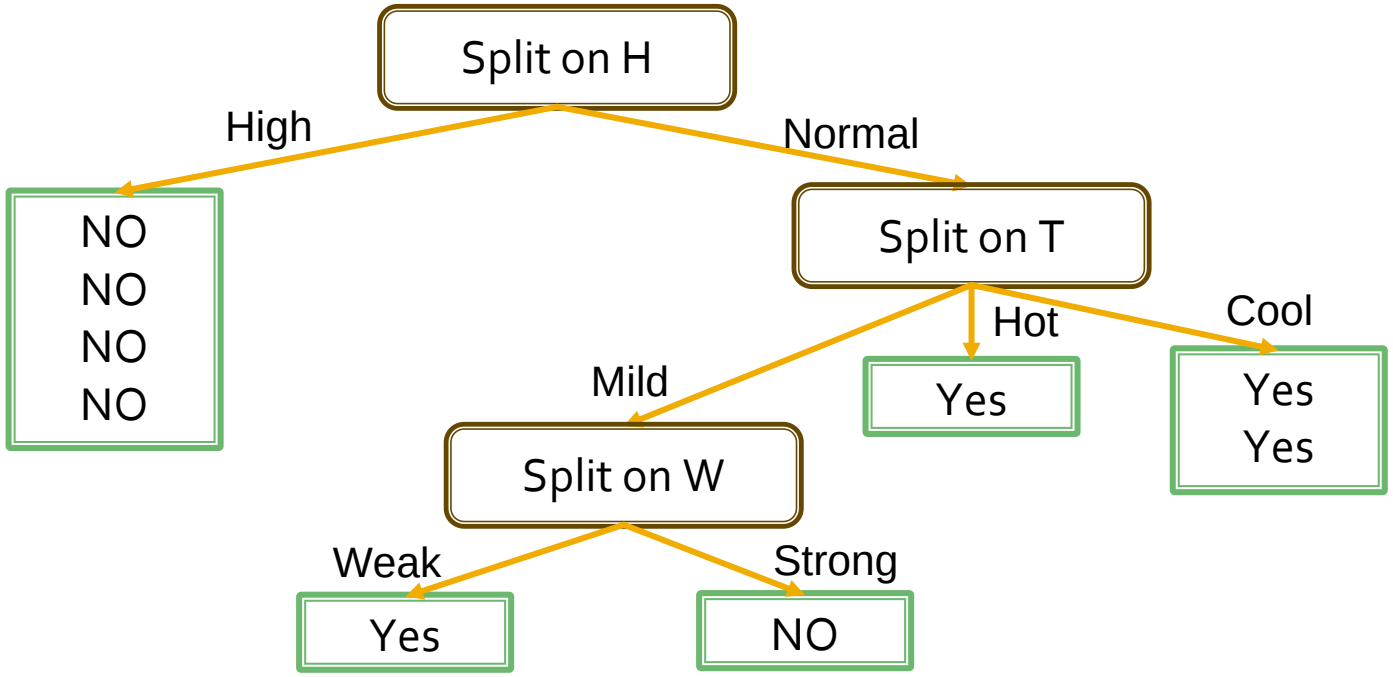
Decision Tree

- $IG(W) = H(S|H = Normal) - [P(W = Weak) * H(S|W = Weak) + P(W = Strong) * H(S|W = Strong)]$
- $IG(W) = 0.720 - [\frac{3}{5} * \left(-\left[\frac{3}{3} * \log\left(\frac{3}{3}\right) + \frac{0}{3} * \log\left(\frac{0}{3}\right)\right]\right) + \frac{2}{5} * \left(-\left[\frac{1}{2} * \log\left(\frac{1}{2}\right) + \frac{1}{2} * \log\left(\frac{1}{2}\right)\right]\right) = 0.320$

Temp (T)	Hum (H)	Wind (W)	Outside (O)
Mild	Normal	Strong	**No**
Mild	Normal	Weak	**Yes**
Hot	Normal	Weak	**Yes**
Cool	Normal	Strong	**Yes**
Cool	Normal	Weak	**Yes**

Decision Tree

- Based on the previous calculations, it looks like both Temp and Wind have equal Information Gains, we can choose any of them to split on. Let's go with Temp. "Mild" branch only has Wind to split on, we go with it.



Temp (T)	Hum (H)	Wind (W)	Outside (O)
Hot	High	Weak	**No**
Mild	Normal	Strong	**No**
Hot	High	Strong	**No**
Mild	Normal	Weak	**Yes**
Hot	Normal	Weak	**Yes**
Cool	High	Weak	**No**
Cool	Normal	Strong	**Yes**
Mild	High	Weak	**No**
Cool	Normal	Weak	**Yes**

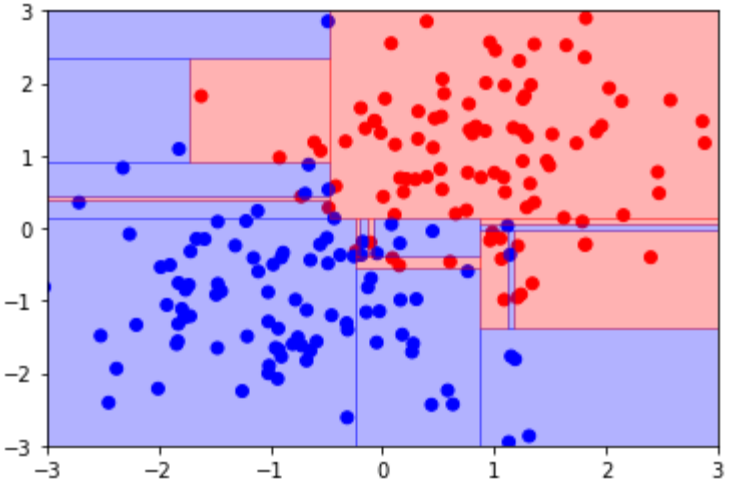
- The resulting Decision Tree has homogenous leaf nodes, therefore no further splitting needs to be done.
- We are guaranteed that the final decision tree is the "shallowest" one possible, since we used highest Information gains while splitting.

Decision Tree: ID3 Algorithm

- The previous steps we followed are defined as the **ID3 algorithm**, which is a classic algorithm for building decision trees. The steps of the algorithm are as follows:
  - Calculate the entropy of every attribute of the data set S.
  - Partition ("split") the set S into subsets using the attribute for which the resulting entropy after splitting has maximum information gain.
  - Make the selected attribute an internal node and divide the subset of records that have the value of that attribute into smaller subsets.
  - Recurse on each subset until one of the following conditions holds:
    - Every element in the subset belongs to the same class.
    - There are no more attributes to be selected.
    - There are no examples in the subset.
- Other algorithms like the $C4.5$ build on the $ID3$ algorithm to handle continuous numerical features, missing features, and generate a smaller overall tree.

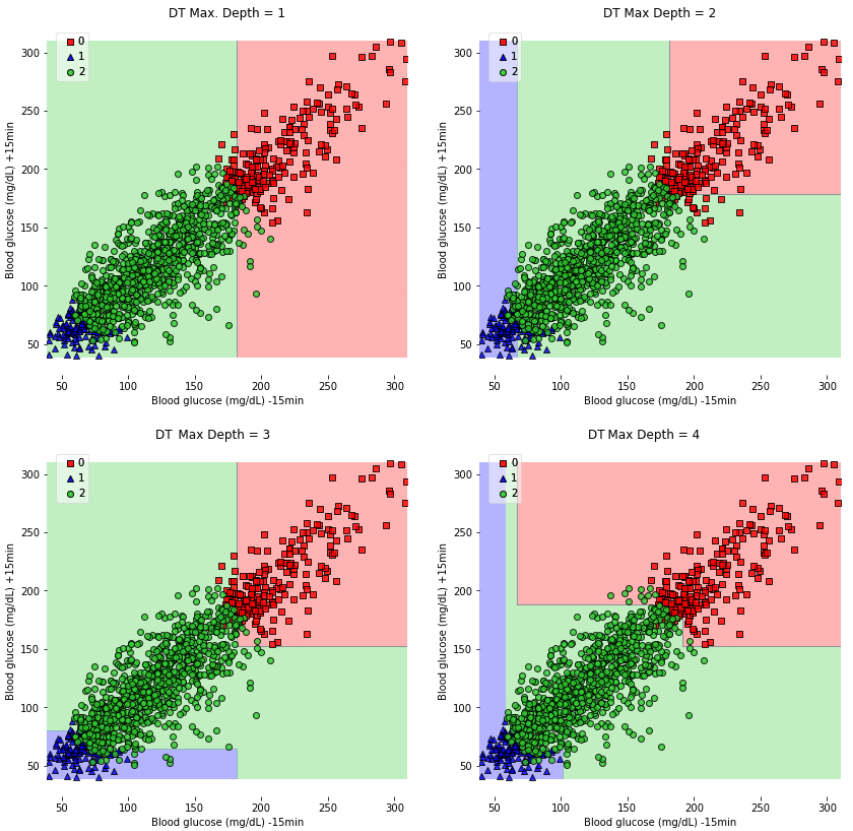
Decision Tree: Decision Boundary

- The decision boundary in a decision tree represents the regions in the input feature space where the decision tree assigns a particular class label or output value.
- In other words, it partitions the feature space into regions, and each region is associated with a different class label or output value.
- In 2D, it can be visualized as a set of lines or curves that separate different classes.
- A very deep decision tree can impose problems as it could overfit on the training data and capture noise and fluctuations in the data.
  - It would also lead to an overly complex decision boundary that doesn't capture the real underlying patterns.
- Sometimes, we would want to impose external **stopping criteria** to enforce having a shallow tree which would generalize better to new data and capture the underlying patterns.



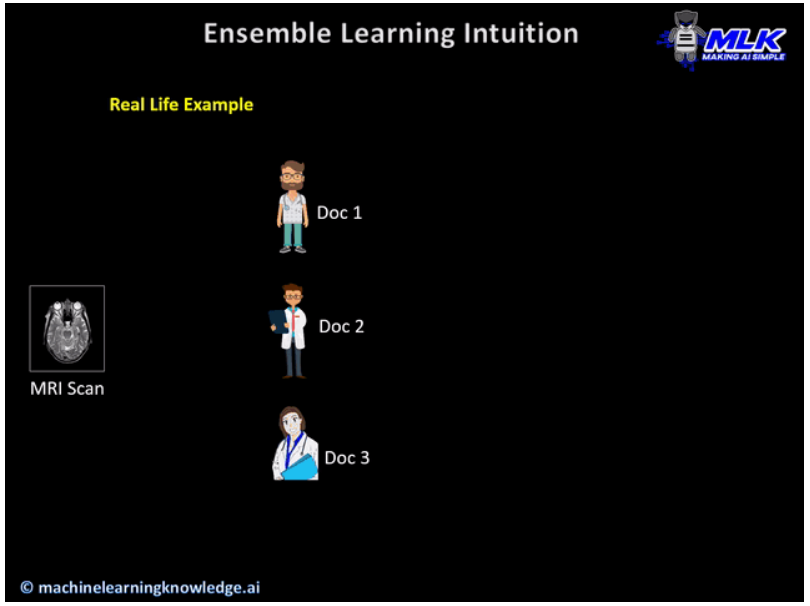
Decision Tree: Stopping Criteria

- Stopping criteria are imposed on decision trees to prevent them from becoming too deep, a condition known as overfitting. Overfitting occurs when a decision tree learns the training data too well, capturing noise. Examples of different stopping criteria:
  - Maximum Depth:
    - Stop growing the tree when a specified maximum depth is reached.
  - Minimum Samples per Leaf:
    - Stop growing the tree when the number of samples in a leaf node falls below a specified threshold.
  - Impurity Threshold:
    - Stop splitting nodes when the impurity (Information Gain) falls below a specified threshold.
  - Pruning:
    - Grow the tree fully, and then prune it by removing branches that do not significantly improve performance on a validation set.



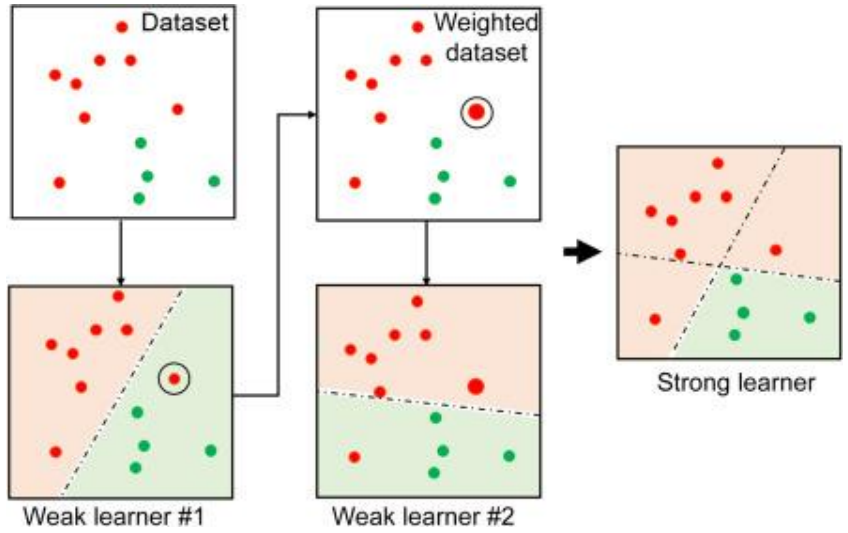
Ensemble Methods

- Given a specific problem at hand, it is believed that that a committee of experts working together on a problem are more likely to solve it successfully than a single expert working alone.
- A **model ensemble** generate a set of models and then make predictions by aggregating the outputs of these models, rather than creating a single model only.
- Given a large population of independent models, an ensemble can be very accurate even if the individual models in the ensemble perform only marginally better than random guessing.
- The two famous types of ensemble algorithms are: **Boosting** and **Bagging**.



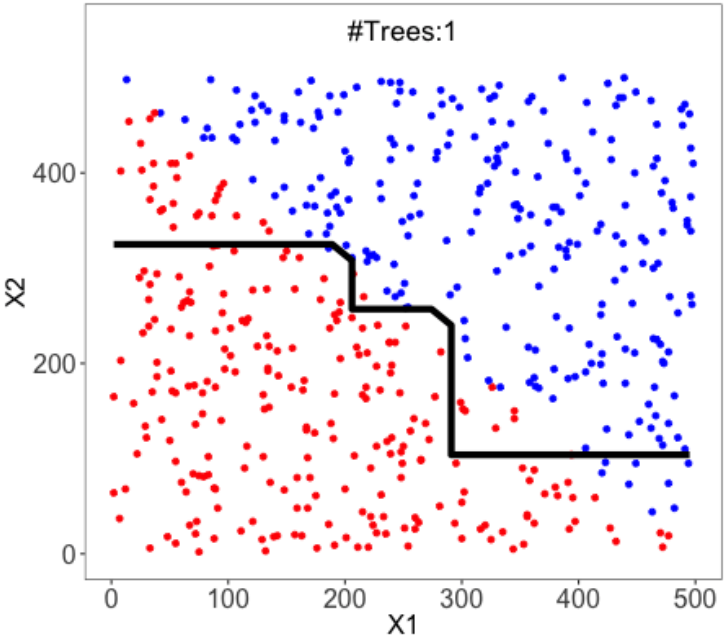
Boosting Algorithms

- **Boosting** is an ensemble learning technique that combines multiple weak learners to create a strong learner.
- It aims to improve predictive performance by sequentially training models, with each subsequent model giving more weight to instances that were misclassified by the previous models.
- Steps of a typical boosting algorithm are as follows:
  - Assign equal weights to all data points in the training set.
  - Train a weak learner (usually a simple model, e.g., a shallow decision tree) on the training data, considering the instance weights.
  - Compute the error of the weak learner, emphasizing misclassified instances.
  - Increase the weights of misclassified instances to make them more influential in the next iteration.
  - Train another weak learner on the updated dataset, giving higher importance to previously misclassified instances.
  - Repeat steps 3-5 for a predefined number of iterations or until a satisfactory performance is achieved.
  - Combine the predictions of all weak learners to produce the final strong model.
- Examples of Tree-based Boosting Models: AdaBoost / Gradient Boosting (GBM) / XGBoost / LightGBM

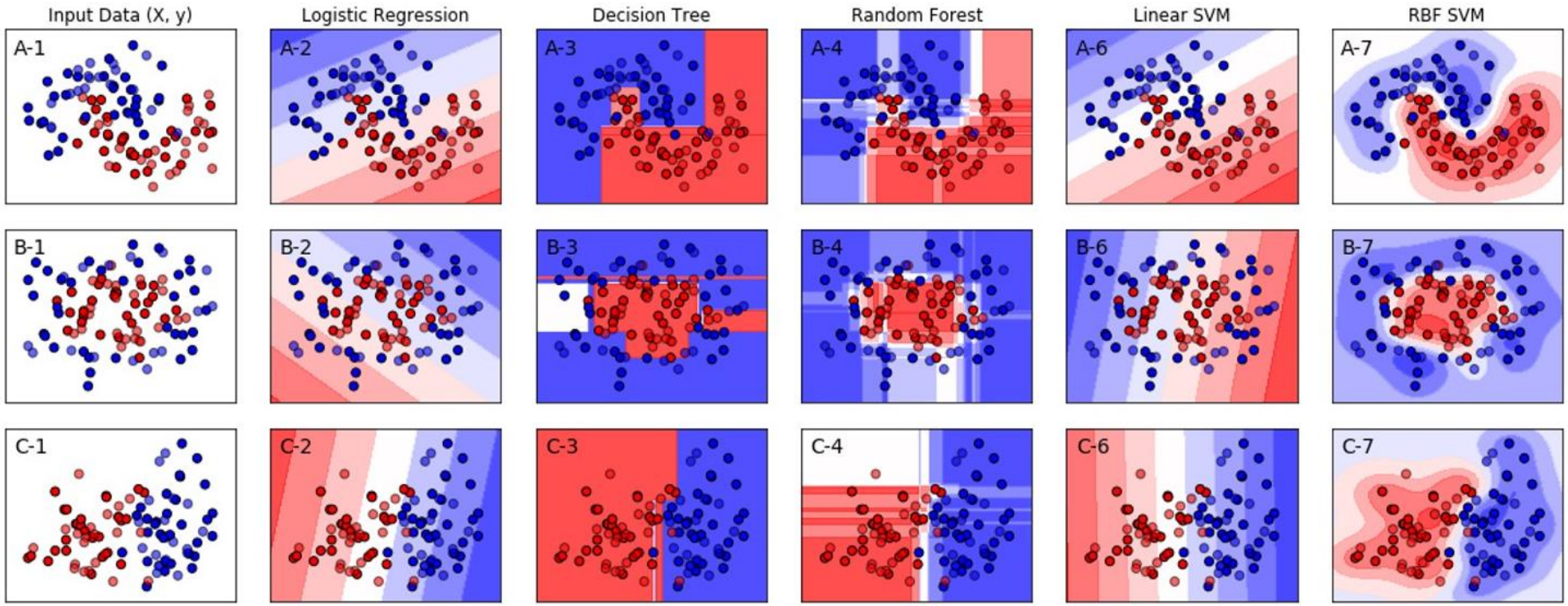


Bagging Algorithms

- **Bagging (Bootstrap Aggregating)** is an ensemble learning technique that combines multiple base learners.
- Each model in the ensemble is trained on a random sample of the dataset where, importantly, each random sample is the same size as the dataset and **sampling with replacement** is used.
- These type of methods are well-suited for Decision Trees since single trees are very sensitive to changes in the dataset. Therefore, having multiple trees trained on different samples of the dataset could result in a more powerful ensemble model.
- Steps of a Bagging approach are simple:
  - Generate multiple random samples with replacement from the training dataset.
  - Train a base learner (e.g., decision tree) on each bootstrap sample independently.
  - Combine predictions from all base models by voting for classification.
- The vanilla bagging approach mentioned above is called a "Random Forest" model.
- Extremely randomized trees (Extra Trees) take it a step further by also randomizing the feature to split on in every step. In other words, each weak learner (Decision Tree) doesn't choose the best feature with highest Information Gain, but chooses it randomly.



Comparison of Models So Far



Example adapted from Scikit-learn open-source developer guide, Code source: Gaël Varoquaux, Andreas Müller; 2018
https://scikit-learn.org/stable/auto_examples/classification/plot_classifier_comparison.html

Extra Resources

- More information regarding the Decision Trees:
  - Chapter 4: John D. Kelleher, Brian Mac Namee, and Aoife D'Arcy. 2015. Fundamentals of Machine Learning for Predictive Data Analytics: Algorithms, Worked Examples, and Case Studies. The MIT Press.

Thank you!

- Any questions?

Disclaimer

Due to nature of the course, various materials have compiled from different open source resources with some moderation. I sincerely acknowledge their hard work and contribution

CONESTOGA

Thank You

Youssef Abdelkareem

yabdelkareem@conestogac.on.ca

**** Credits for Dr. Anasuya Bhima for assisting in the slides structure***