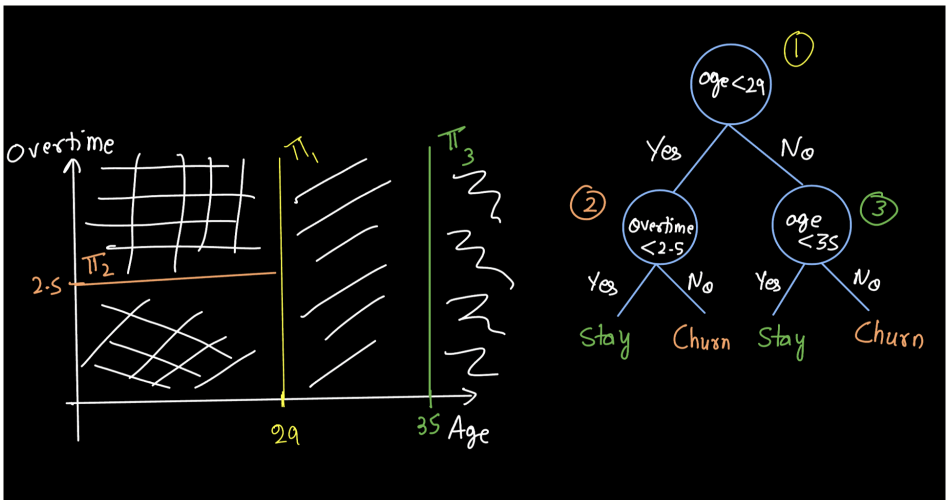
What are decision trees?

- Decision Trees are powerful and interpretable supervised learning models
- They operate by recursively partitioning the feature space into smaller regions based on the values of input features, aiming to create homogeneous subsets with respect to the target variable.
- At each internal node of the tree, a decision is made based on a feature's value, leading to the traversal down different branches until a leaf node, which represents the final prediction or decision.



- A decision tree is a bunch of nested if-else statements (rules).
- Topmost node -> root node
- Bottom nodes -> leaf nodes.

Some Advantages of using decision trees

1. Useful for predicting non-linear boundaries
   - Decision-boundary in DTs are made up of axis-parallel hyperplanes
   - For slanted lines, DT uses multiple axis-parallel lines in a staircase manner

2. Decision trees are easily interpretable. How?

   - Each node can be considered as a rule for an if-else-based condition

- As an example, the above mentioned decision tree can be broken down into rules:

```
If age < 29:
  If overtime < 2.5hrs:
     Employee will stay
  else:
     Employee will churn
else:
  if age < 35:
     Employee will stay
  else:
     Employee will churn
```

Splitting of nodes

Pure nodes: Nodes that are purely homogeneous i.e. contain data points belonging to only one class

Impure nodes: Nodes that are heterogeneous i.e. contain data points belonging to multiple classes

So what is the purpose of splitting a node?
- To create pure nodes
- Pure nodes need not be split further
  Why?
    Because in this case uncertainty of prediction would be the lowest

Impurity Measures

How to decide which features to use for splitting nodes?

- Using impurity measures
- Impurity Measures:
    - They are used to measure the heterogeneity of a node
    - Impurity of pure node = 0

Some Properties of impurity measures for binary classification

- N = #Points belonging to class 1
- M = #Points belonging to class 2

1. Impurity of a pure node = 0
2. It has 2 minimas (N=0, M=0)
3. N=M: Impurity is maximum
4. Impurity is Symmetric around the maxima
5. It increases from minima to maxima then decreases from maxima to minima
6. Should be non-negative for all points

Measuring impurity

1. Entropy

Imagine Y be a discrete random variable:

We define entropy (H) as
- $H(Y) = - \sum^{k}_{i=1} p(y_i)\log(p(y_i))$

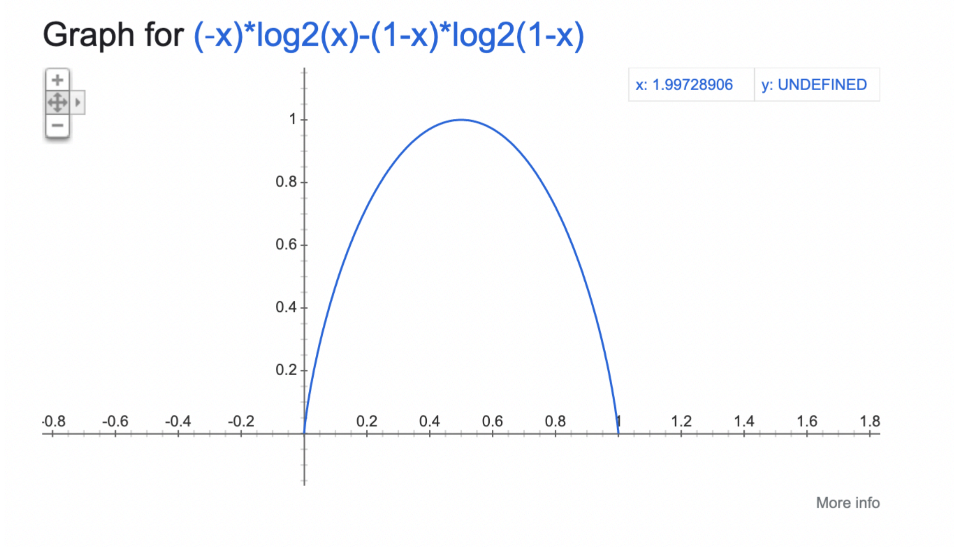
where $p(y_i)$ is the probability that random variable $Y = y_i$

Entropy for binary classification
- $H(Y) = -P(Y=0)\log(P(Y=0)) - P(Y=1)\log(P(Y=1))$

where
- $P(Y=0)$ is the probability $Y = 0$
- $P(Y=1)$ is the probability $Y = 1$

For $P(Y=1) = p$ entropy becomes:
- $H(Y) = -p\log(p) - (1-p)\log(1-p)$

Plot of entropy



Some properties that can be observed from entropy's (for Binary classification) formula/plot

1. The values of the plot range from 0 to 1.
2. The maxima lies at x = 0.5
3. Maximum value of entropy for binary case will be 1 (log base 2).

How to use Entropy for node splitting?

At each node, we try to find that split which minimizes the entropy.

- This **reduction in entropy** i.e. Parent - the weight entropy of the child is termed **Information gain**

Why is there a need to minimize the entropy?

- Assuming that recursively splitting in such a manner can lead to pure leaves in all cases
- Greedy in nature. Doesn't necessarily happen