Introduction to Machine Learning

Module 5:
Decision trees

Module Checklist:

- ❑ Decision trees
  - ❑ Intuition
  - ❑ The “best” split
  - ❑ Model performance
  - ❑ Model optimization (pruning)

Where are we?

We’ve laid much of the groundwork for machine learning, and introduced our very first algorithm of linear regression.

Now, we can focus on expanding our algorithm toolkit. **Decision trees** are another type of algorithm that can accomplish the same objective of prediction that linear regression can. We will also go deeper into the pros and cons of decision trees in this module.

Where are we?

Supervised Algorithms

Linear regression

Decision tree

Ensemble Algorithms

Let's do a quick intro to decision trees and ensembles!

Decision Tree

Today, we will start by looking at a <u>decision tree</u> algorithm.

A decision tree is a set of rules we can use to classify data into categories **(also can be used for regression tasks)**.

Humans often use a similar approach to arrive at a conclusion. **For example, doctors ask a series of questions to diagnose a disease. A doctor's goal is to ask the minimum number of questions needed to arrive at the correct diagnosis.**

What is wrong with my patient?

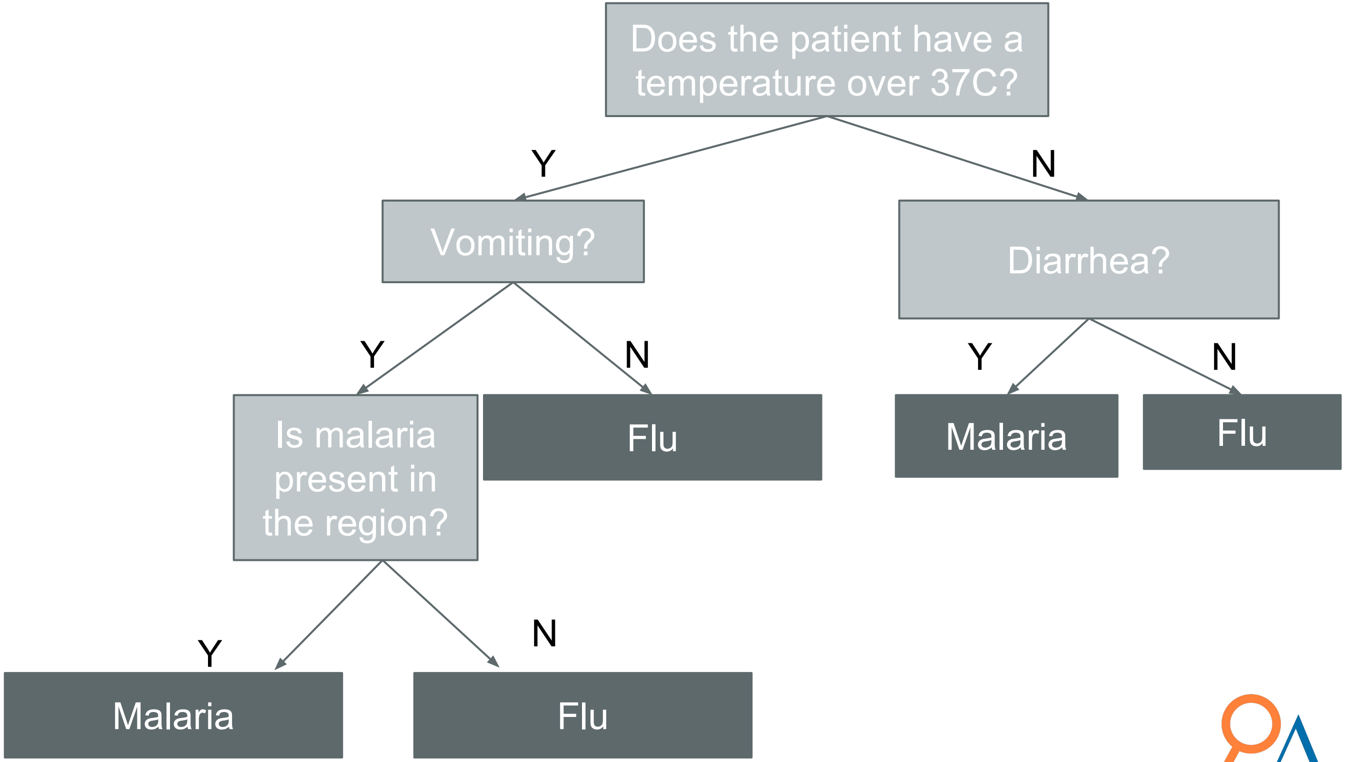
Help me put together some questions I can use to diagnose patients correctly.

Decision Tree

Decision trees are intuitive because they are similar to how we make many decisions.

My mental diagnosis decision tree might look something like this.

How is the way I think about this different from a machine learning algorithm?

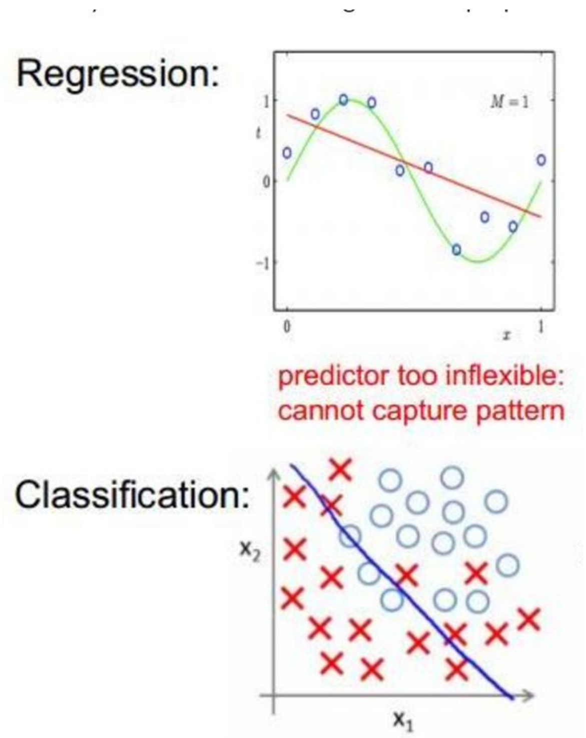


Decision Tree

Decision trees are intuitive and can handle more complex relationships than linear regression can.

A linear regression is a single global trend line.

This makes it inflexible for more sophisticated relationships.



Sources:
https://www.slideshare.net/ANITALOKITA/winnow-vs-perceptron, http://www.turingfinance.com/regression-analysis-using-python-statsmodels-and-quandl/

We will use our now familiar framework to discuss both decision trees and ensemble algorithms:

Task	What is the problem we want our model to solve?
Learning Methodology	ML algorithms can be supervised or unsupervised. This determines the learning methodology.
Performance Measure	Quantitative measure we use to evaluate the model's performance.

Source: Deep Learning Book - Chapter 5: Introduction to Machine Learning

Task	What is the problem we want our model to solve?
Defining f(x)	What is f(x) for a decision tree and random forest?
Feature engineering & selection	What is x? How do we decide what explanatory features to include in our model?
Is our f(x) correct for this problem?	What assumptions does a decision tree make about the data? Do we have to transform the data?

Learning Methodology

Decision tree models are supervised. How does that affect the learning processing?

How does our ML model learn?

Overview of how the model teaches itself.

What is our loss function?

Every supervised model has a loss function it wants to minimize.

Optimization process

How does the model minimize the loss function?

Performance: Quantitative measure we use to evaluate the model's performance.

- **Measures of performance**: Metrics we use to evaluate our model
- **Feature Performance**: Determination of feature importance
- **Ability to generalize to unseen data**: Overfitting, underfitting, bias, variance

Decision Tree

Decision tree: model cheat sheet

Pros

- Mimics human intuition closely (we make decisions in the same way!)
- Not *prescriptive* (i.e., decision trees do not assume a normal distribution)
- Can be intuitively understood and interpreted as a flow chart, or a division of feature space.
- Can handle nonlinear data (no need to transform data)

Cons

- **Susceptible to overfitting** (poor performance on test set)
- High variance between data sets
- Can become unstable: small variations in the training data result in completely different trees being generated

Task

Task

What are we predicting?

How does a doctor diagnose her patients based on their symptoms?

Task

A doctor builds a diagnosis from your symptoms.

I start by asking patients a series of questions about their symptoms.

I use that information to build a diagnosis.

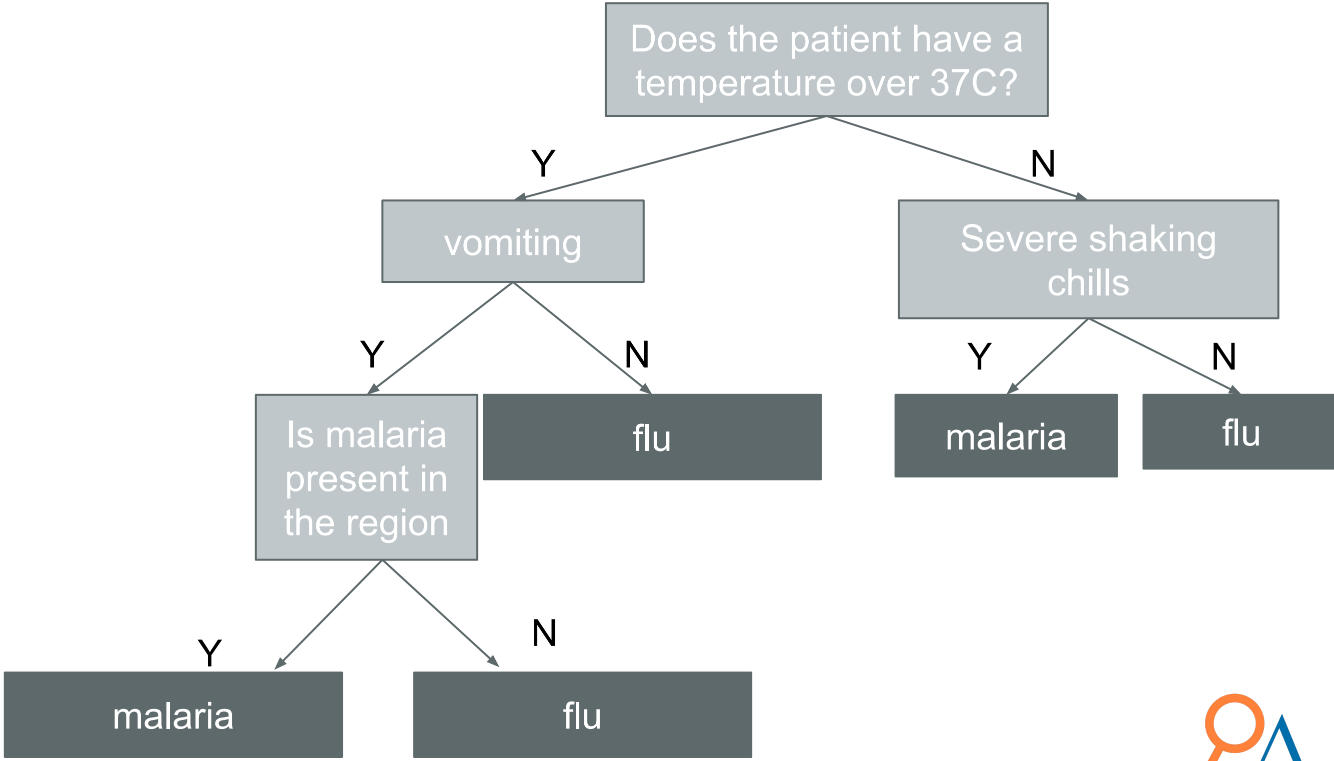
temp	vomiting	Shaking chills	diagnosis	predicted diagnosis
X1	X2	X3	Y	Y*
39.5°C	Yes	Severe	Flu	Flu
37.8°C	No	Severe	Malaria	Malaria
37.2°C	No	Mild	Flu	Malaria
37.2°C	Yes	None	Flu	Flu

There are some questions the doctor does not need to ask because she learns it from her environment. (For example, a doctor would know if malaria present in this region.) **A machine, however, would have to be explicitly taught this feature.**

Task

A doctor makes a mental diagnosis path to arrive at her conclusion. She is building a decision tree!

How does the way I think about how to decide the value of the split compare to a machine learning algorithm?



Human Intuition

Based upon my experience as a doctor, I know there are certain questions whose answers quickly separate flu from malaria.

Decision Tree

At each split, we can determine the best question to maximize the number of observations correctly grouped under the right category.

- Both a doctor and decision tree try to arrive at the minimum number of questions (splits) that needs to be asked to correctly diagnose the patient.
- Key difference: how the order of the questions and the split value are determined. A doctor will do this based upon **experience**, a decision tree will **formalize** this concept using a loss function.

Task

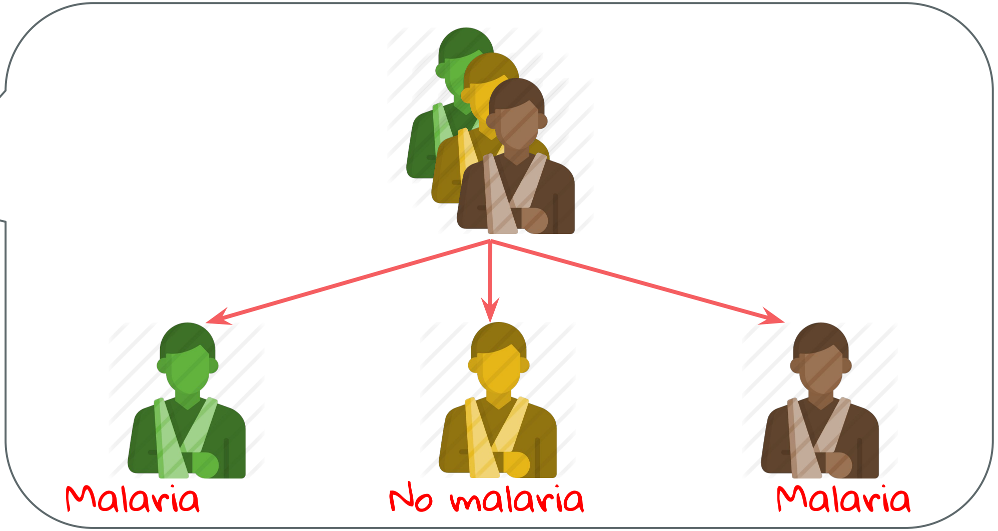
Machine learning adds the power of data to our doctor's task.

To determine whether a new patient has malaria, a doctor uses her experience and education, and machine learning creates a model using data.

By using a machine learning model, our doctor is able to use both her experience **and** data to make the best decision!

Task

Decision trees act in the same manner as human categorization, they (split) the data based upon answers to questions.



Where do I go?

Mr. Model creates a flow chart (the model) by separating our entire dataset into **distinct categories**. Once we have this model, we'll know what category our **new patient** falls into!

To get a sense of how this "splitting" works, let's play a guessing game. **I am thinking of something that is blue.**

You may ask 10 questions to guess what it is.

The first few questions you would probably ask may include:

- Is it alive? (No)
- Is it in nature? (Yes)

Then, as you got closer to the answer, your questions would become more specific.

- Is it solid or liquid? (Liquid)
- Is it the **ocean**? (Yes!)

This process is your mental decision tree. In fact, this strategy captures many of the qualities of our algorithm!

Some winning strategies include:

- Use questions early on that eliminate the most possibilities.
- Questions often start broad and become more granular.

You created your own decision tree **based on your own experience of what you know is blue** to make an educated guess as to what I was thinking of (**the ocean**).

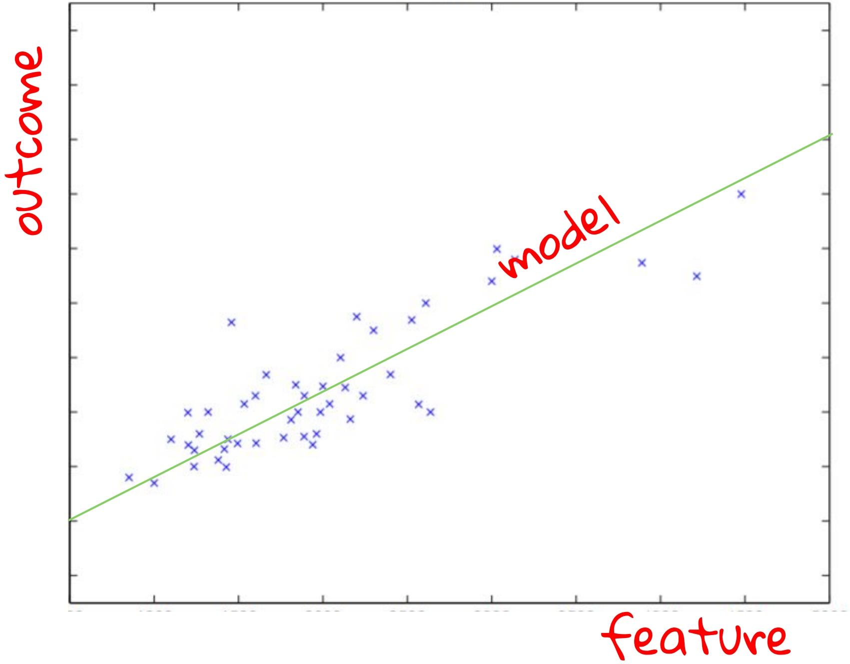
Similarly, a decision tree model will make an educated guess, but instead of using experience, it uses **data**.

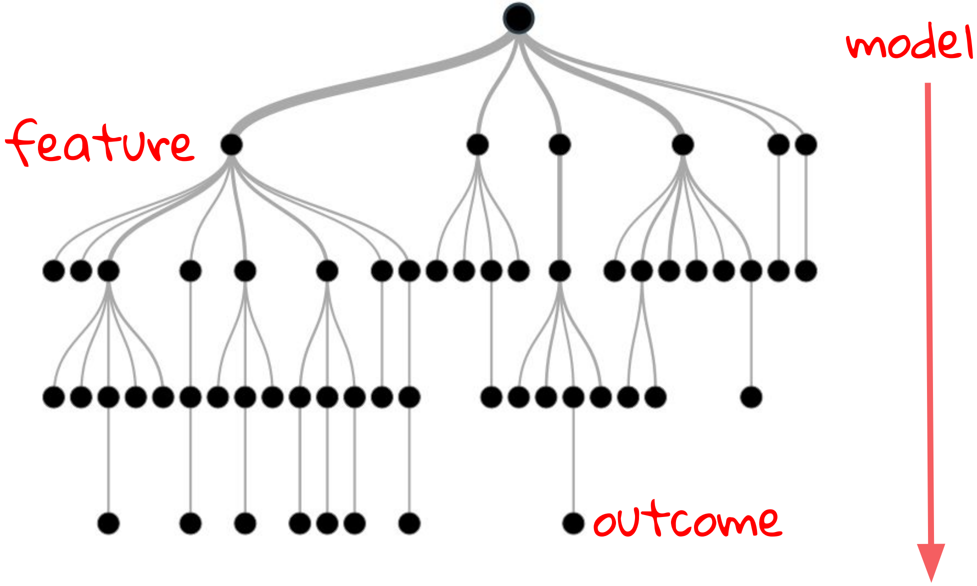
Now, let's build a formal vocabulary for discussing decision trees.

Decision Tree

Defining f(x)

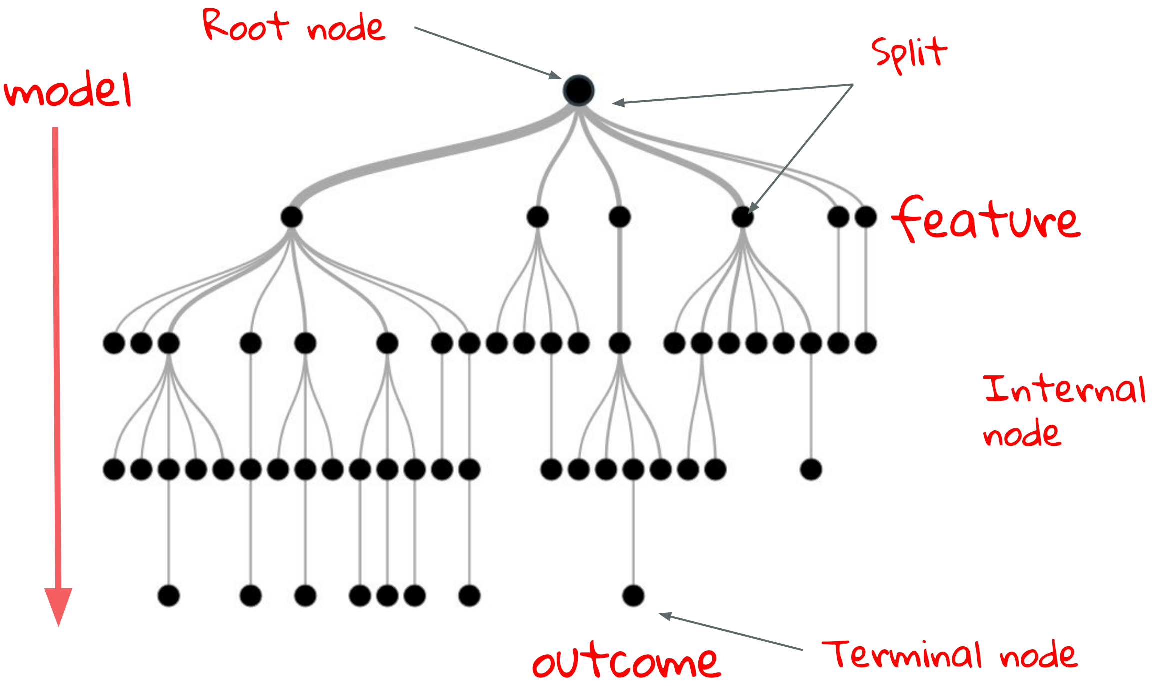
Like a linear regression, a decision tree has explanatory features and an outcome. Our f(x) is the decision path from the top of the tree to the final outcome.





Decision Tree

New vocabulary

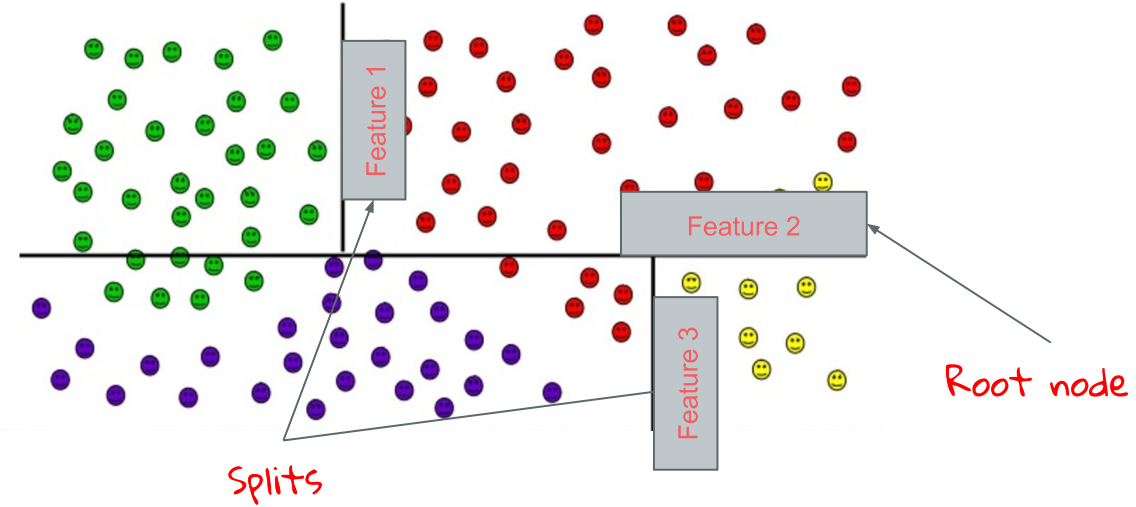


Split	The "decisions" of the decision tree model. The model decides which features to use to split the data.
Root node	Our starting point. The first split occurs here along the feature that the model decides is **most important**.
Internal node	Intermediate splits. Corresponds to a subset of the entire dataset.
Terminal node	Predicted outcome. Corresponds to a smaller subset of the entire dataset.

Decision Tree

New vocabulary

A decision tree splits the feature space (the dataset) along features. The order of splits are determined by the algorithm.

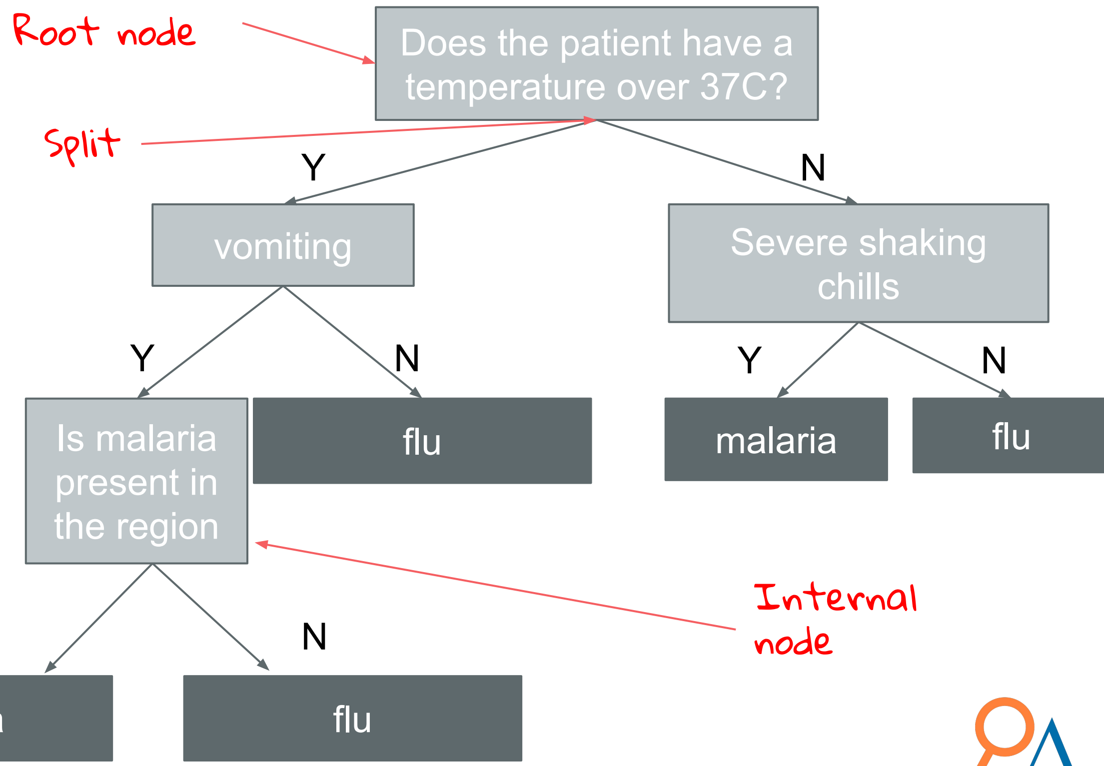


This visualization of splits in feature space is a different but equally accurate way to conceptualize decision trees as the flow chart in the previous slide.

Task

Some additional terminology

Splits at the top of the tree have an outsized importance on final outcome.



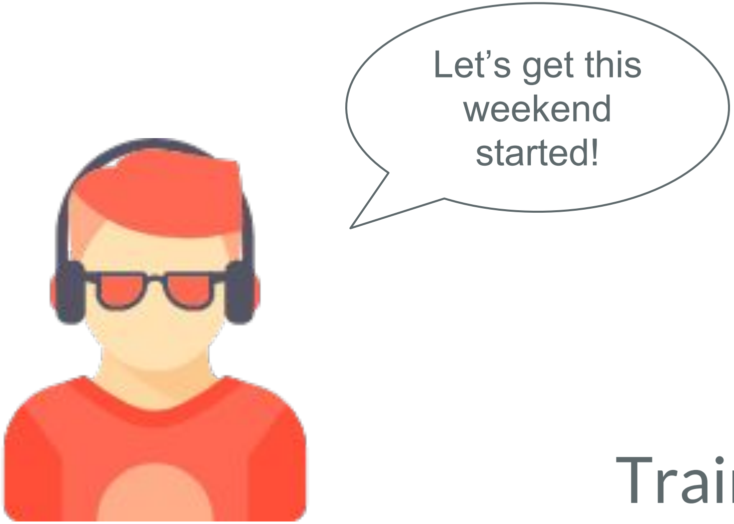
Terminal node

What is the root node, split, internal node and terminal node in this example?

Decision Tree

Defining f(x)

Let's look at another question to combine our intuition with formal vocabulary. **What should Sam do this weekend?**



Y = Choice of weekend activity

- Dancing
- Cooking dinner at home
- Eating at fancy restaurant
- Music concert
- Walk in the park

Training data set: You have a dataset of what he has done every weekend for the last two years.

Decision Tree Task

What will Sam do this weekend?

Has a girlfriend	Parents in town	Savings ($)
X1	X2	X3
Yes	No	$80

What will Sam do?

Y

Dancing
Cooking dinner at home
Eating out
Music concert
Walk in the park
Walk in the park
Dancing
Eating out

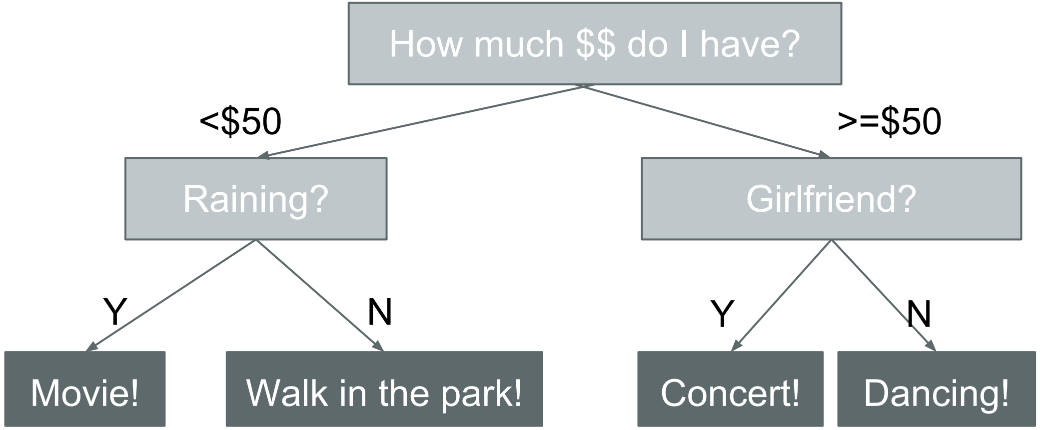
We have historical data on what Sam has done on past weekends (Y), as well some explanatory variables.

Decision Tree Task

Defining f(x)

The decision tree f(x) predicts the value of a target variable by learning simple decision rules inferred from the data features.

In this example, we predict Sam's weekend activity using decision rules trained on historical weekend behavior.



Our most important predictive feature is Sam's budget. How do we know this? Because it is the **root node**.

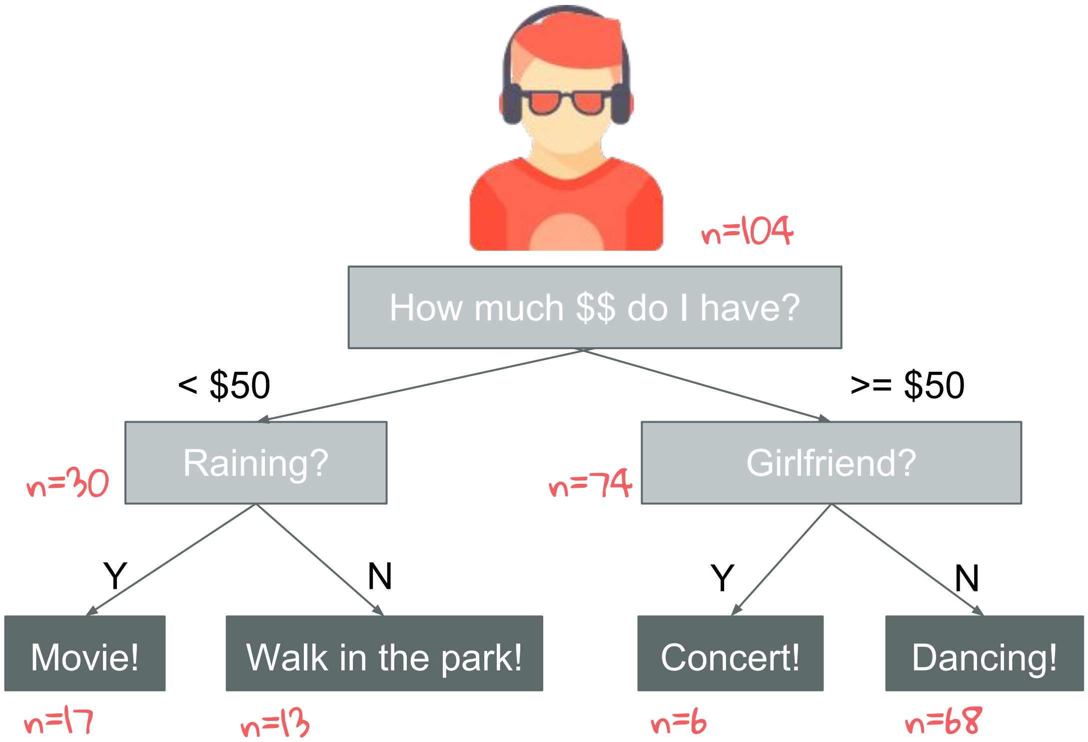
Decision Tree Task

f(x) as a spatial function

An example of how the algorithm works:

A decision tree splits the dataset ("**feature space**"). Here, the entire dataset = data from 104 weekends. You can see how each split subsets the data into progressively smaller groups.

Now, when we want to predict what will happen **this weekend**, we can!

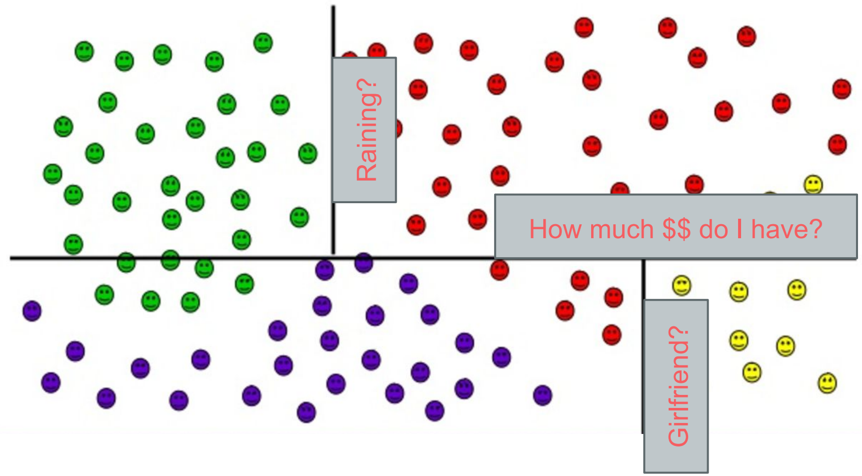


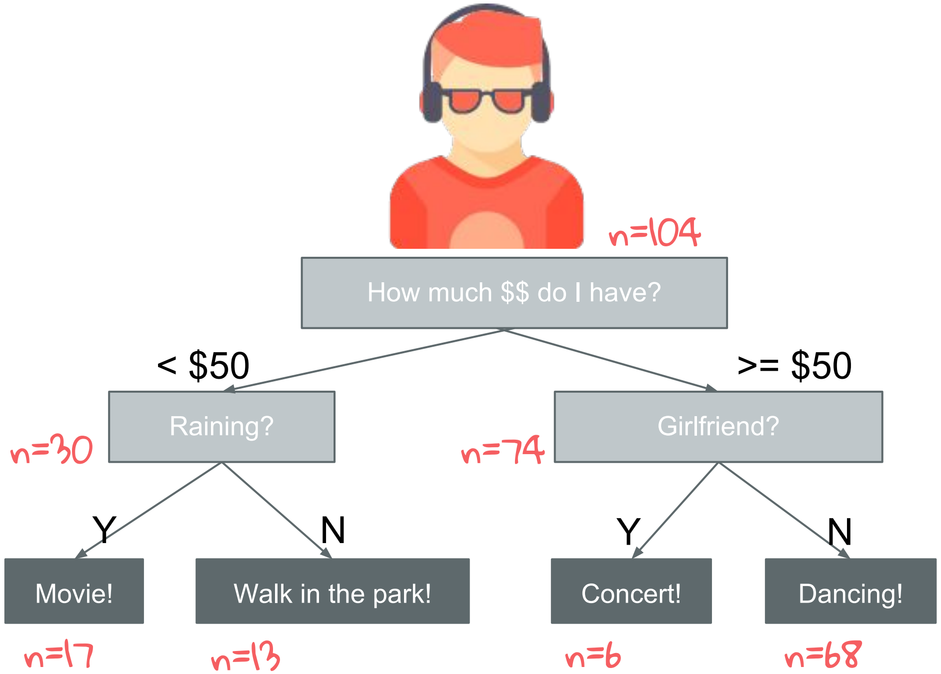
Decision Tree Task

f(x) as a spatial function

Each of the four weekend options are grouped spatially by our splits.

Visualized in the data:





Now we understand the mechanics of the decision tree task. **But how does a decision tree learn**? How does it know where to split the data, and in what order?

We turn now to examine decision trees' learning methodology.

Learning Methodology

Learning Methodology

How does our f(x) learn?

Remember that the key difference between a doctor and our model is how the order of the questions and the split value are determined.

Human Intuition

Based upon my experience as a doctor, I know there are certain questions whose answers quickly separate flu from malaria.

Decision Tree

At each split, we can determine the best question to ask to maximize the number of observations correctly grouped under the right category.

Learning Methodology

How does our f(x) learn?

key difference between a doctor and our model is how the order of the questions and the split value are determined.

The two key levers that can be changed to improve accuracy of our medical diagnosis are:

- The order of the questions
- The split value at each node. (For example the temperature boundary)

A doctor will make these decisions based upon experience, a decision tree will set these parameters to minimize our loss function by learning from the data.

Recap: Loss function

A loss function quantifies how unhappy you would be if you used f(x) to predict Y* when the correct output is y. It is the object we want to minimize.

Linear Regression → Sound familiar? We just went over a similar optimization process for linear regression (our loss function there was MSE).

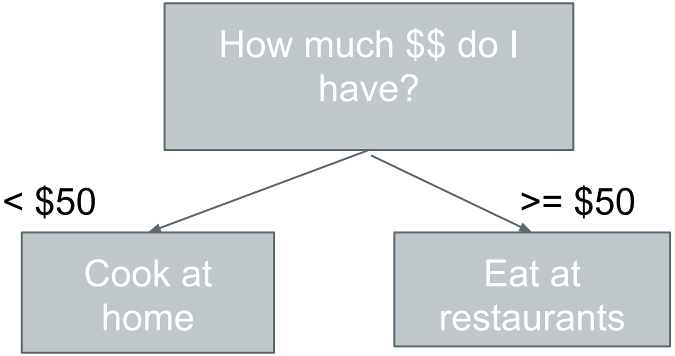
Remember: All supervised models have a loss function (sometimes also known as the cost function) they must optimize by changing the model parameters to minimize this function.

Source: Stanford ML Lecture 1

Learning Methodology

How does our decision tree learn?

Parameters of a model are values that Mr. Model controls to minimize our loss function. Mr. Model sets these parameters by learning from the data.



One of the parameters Mr. Model controls are the split values. For example, Mr. Model learns that $50 is the best split to predict whether Sam eats out or stays at home.

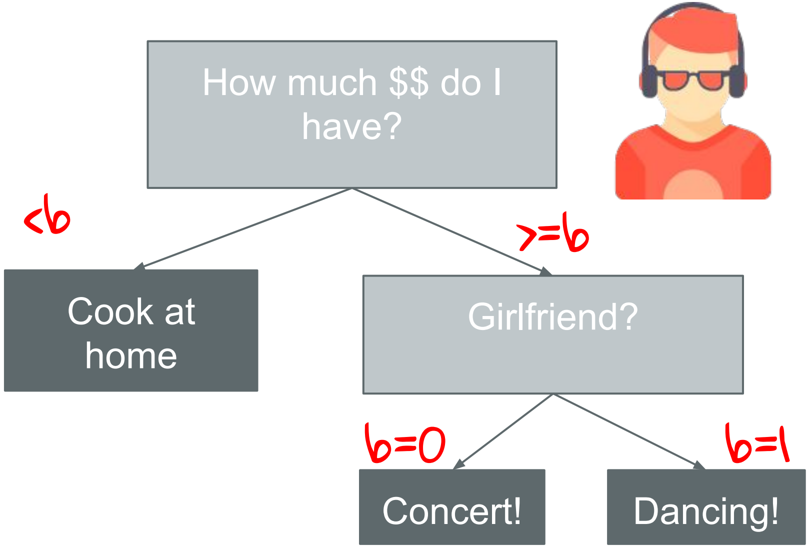
Parameters

Values that control the behavior of the model. The model learns what parameters are from data.

Learning Methodology

How does our decision tree learn?

We have two decision tree parameters: split decision rule (value of b) and the ordering of features



Mr. Model controls B and the ordering of the features.

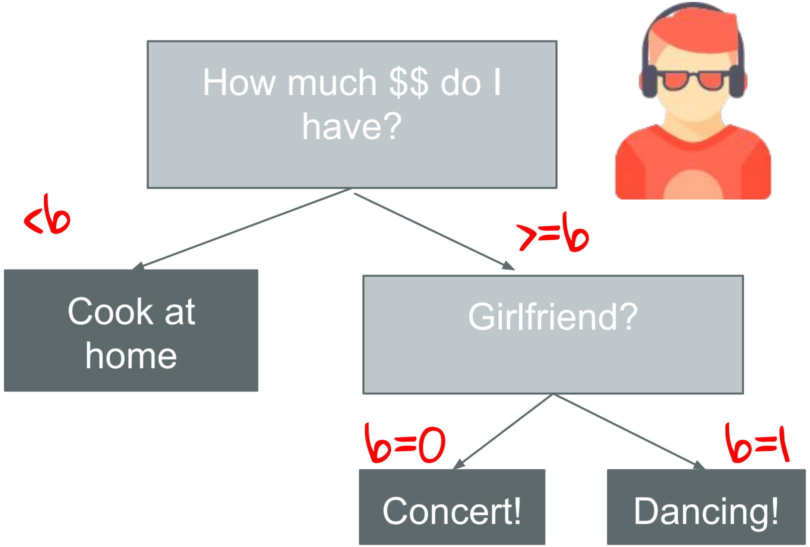
Mr. Model learns what best fits the data by moving b up or down and by trying many combinations of decision rule ordering.

Central problem: How do we learn the “***best***” split?

Learning Methodology

How does our decision tree learn?

There are so many possible decision paths and split values - in fact, an infinite amount! How does Mr. Model choose?



Oh no! That made it worse. Let's try something else.

Mr. Model checks his parameter choices against the loss function every time he makes a change.

He checks whether his error went up, down or stayed the same. If it went down, he knows he did something right!

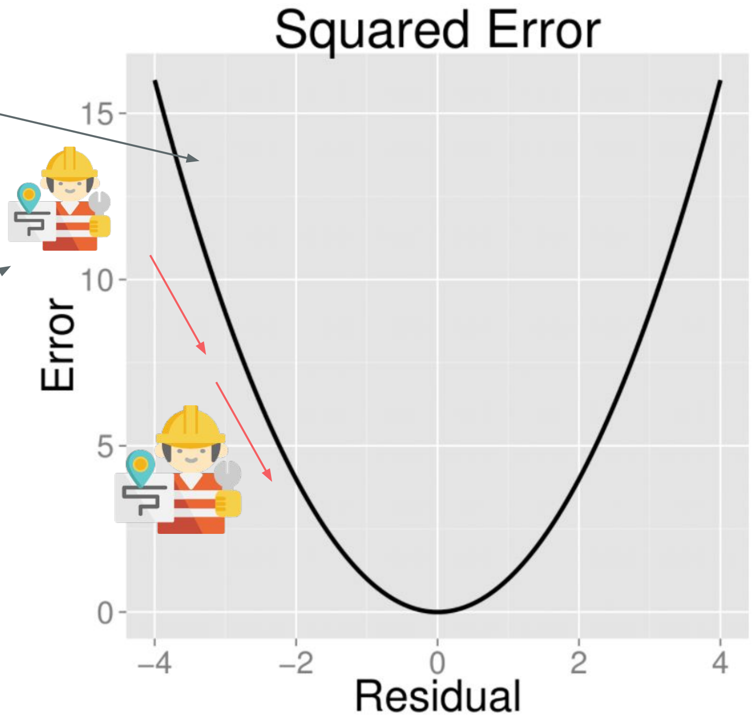
Learning Methodology

How does our decision tree learn?

Imagine that is a game. We start with a **random** ordering of features and set b to completes random values.

Our **random** initialization of parameters give us a unsurprisingly high initial error

Mr. Model's job is to change the parameters so that every time he updates the loss goes down.



The game is over when Mr.Model is able to reduce the error to e, the irreducible error.

Recap of tasks:

Decision trees also can be used for two types of tasks!

Regression

Continuous variable

A regression problem is when we are trying to predict a numerical value given some input, such as "dollars" or "weight".

Classification

Categorical variable

A classification problem is when are trying to predict whether something belongs to a category, such as "red" or "blue" or "disease" and "no disease".

Source: Andrew Ng , Stanford CS229 Machine Learning Course

Learning Methodology | What is our loss function?

Our decision tree loss function depends on the type of task.

The most common loss functions for decision trees include:

For classification trees:

1. Gini Impurity
2. **Entropy**

For regression trees:

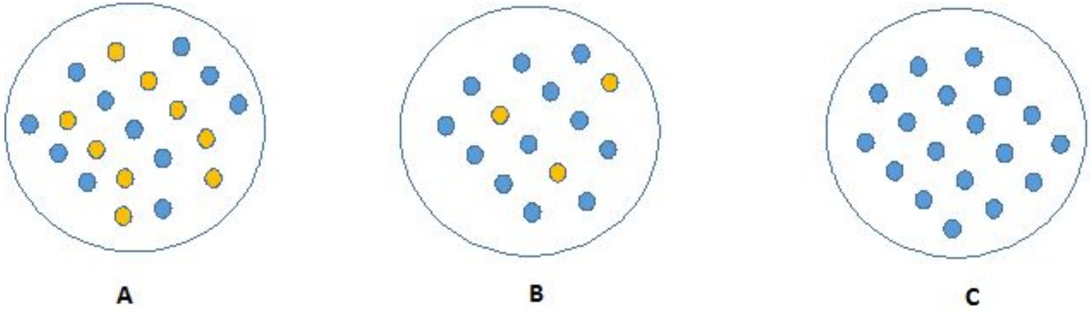
1. Mean Squared Error

For example, the loss function for a regression decision tree should feel familiar. It is the same loss function we used for linear regression!

Learning Methodology

For classification trees, we can use information gain!

How would you rank the entropy of these circles?



Source

Information gain attempts to minimize **entropy** in each subnode. Entropy is defined as a degree of disorganization.

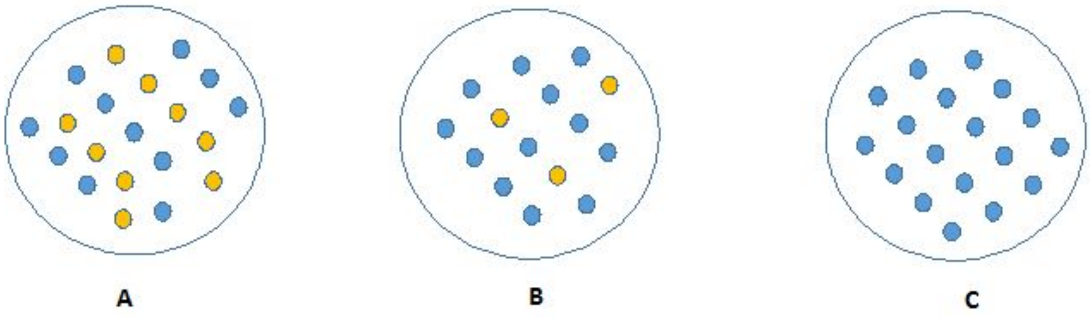
If the sample is completely homogeneous, then the entropy is zero.

If the sample is an equally divided (50% – 50%), it has entropy of one.

Learning Methodology

Information Theory is a neat freak and values organization.

How would you rank the circles in terms of entropy?



All things that are the same need to be put away in the same terminal node.

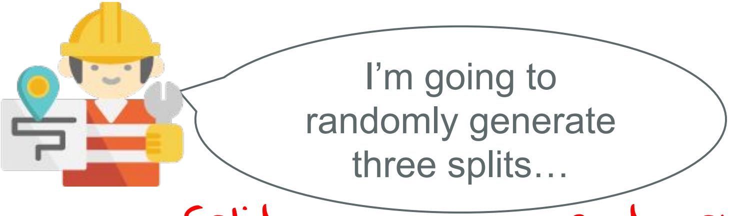
High Entropy	Medium Entropy	Low Entropy

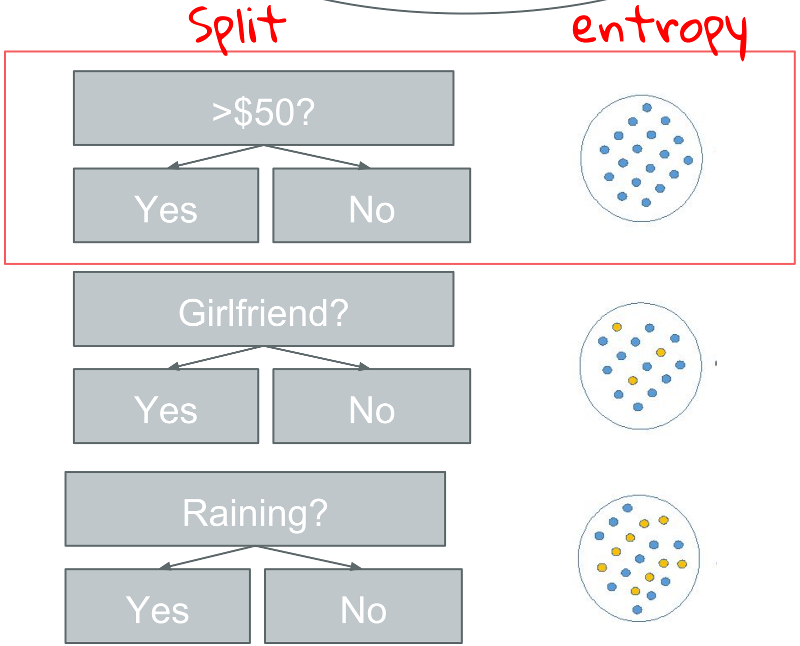
Our low entropy population is entirely homogenous: the entropy is 0!

Learning Methodology

What is our loss function?

So, how does it work in our decision tree model?





The **Information Gain** algorithm:

1. Calculates entropy of parent node
2. Calculates entropy of each individual node of split, and calculates weighted average of all sub-nodes available in split.

The algorithm then chooses the split that has the lowest entropy.

Learning Methodology

What is our loss function?

Regression trees use Mean Squared Error.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

Y-Y*	For every point in our dataset, measure the difference between true Y and predicted Y.
^2	Square each Y-Y* to get the absolute distance, so positive values don't cancel out negative ones when we sum.
Sum	Sum across all observations so we get the total error.
mean	Divide the sum by the number of observations we have.

This may look familiar - look back at our discussion of linear regression! Decision trees also use MSE.

The split with lower MSE is selected as the criteria to split the population.

Learning Methodology → What is our loss function?

Regression trees use Mean Squared Error.

I'm going to randomly generate three splits…

Split	RMSE
Amount of $ (Yes / No)	3.83
# of girlfriends (Yes / No)	7.28
Likelihood of rain (Yes / No)	10.3

The RMSE algorithm proceeds by:

1. Calculating variance for each node.
2. Calculating variance for each split as **weighted average** of each node variance.

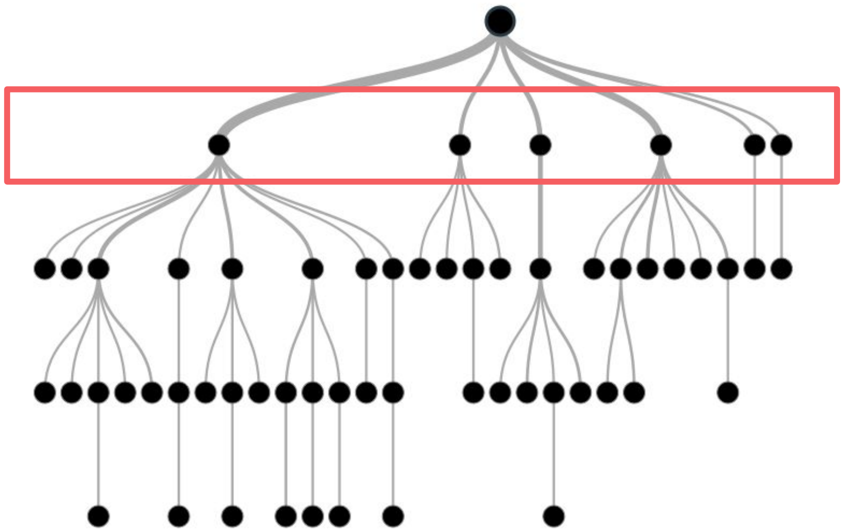
The algorithm then selects the split that has the lowest variance.

Model Performance

Performance

Feature Performance

Decision trees provide us with understanding of what features are most important for predicting Y*



The intuition is provided by the understanding that the most important splits are at the first nodes.

Recall our intuition: Important splits happen closer to the root node.

*The decision tree tells us what the important splits are (which features cause splits that reduce cost function the **most**).*

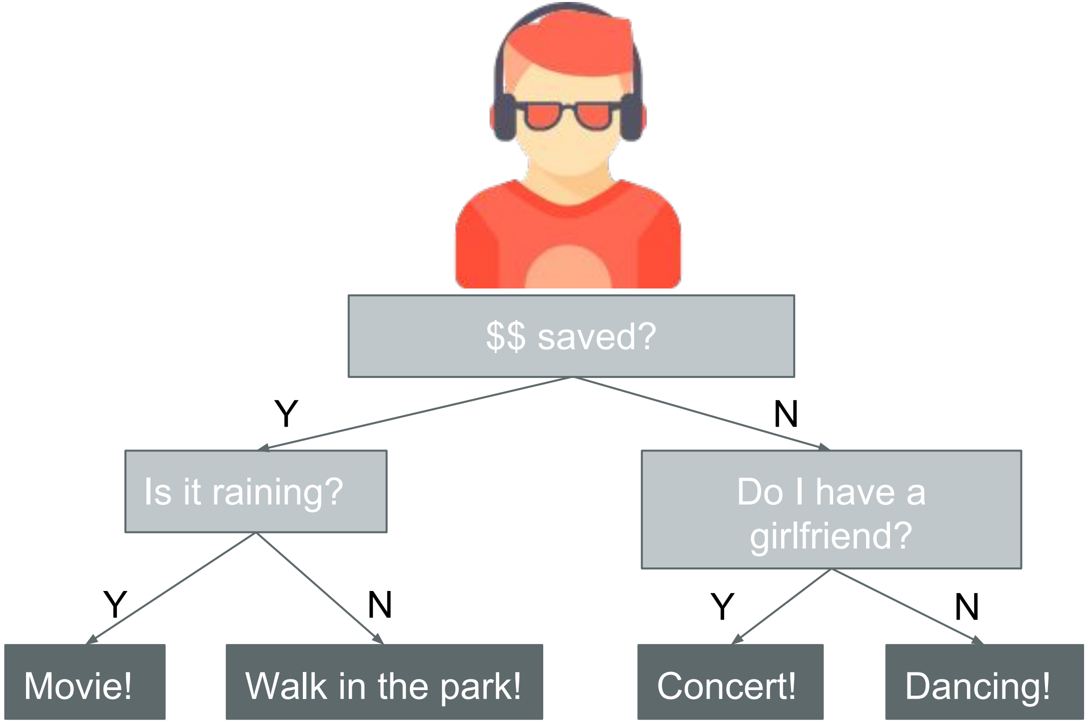
Performance

Feature Performance

Recall our intuition: Important splits happen earlier

In Sam's case, our algorithm determined that Sam's budget was the most important feature in predicting his weekend plans.

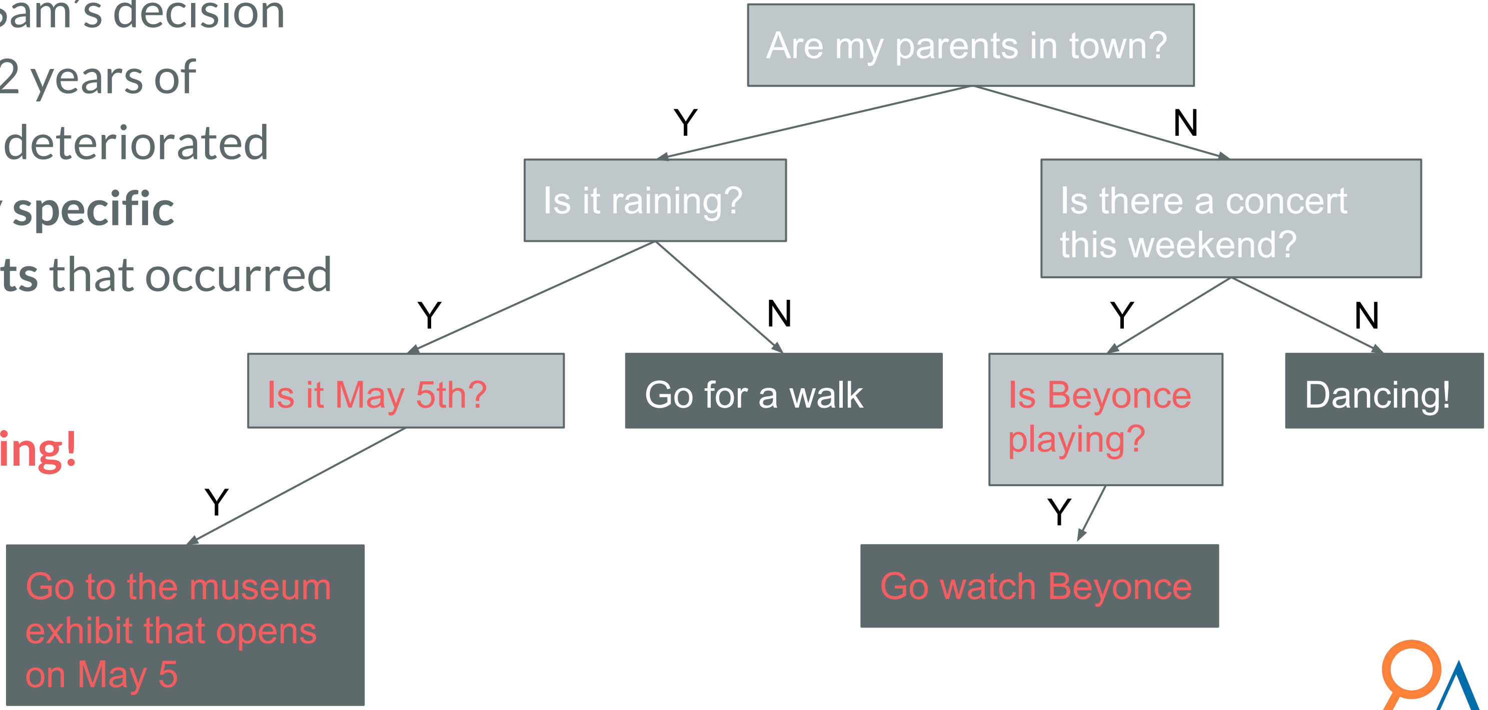
This makes a lot of sense!



Performance | Overfitting

Imagine if the Sam's decision tree, based on 2 years of weekend data, deteriorated into **extremely specific weekend events** that occurred in the data.

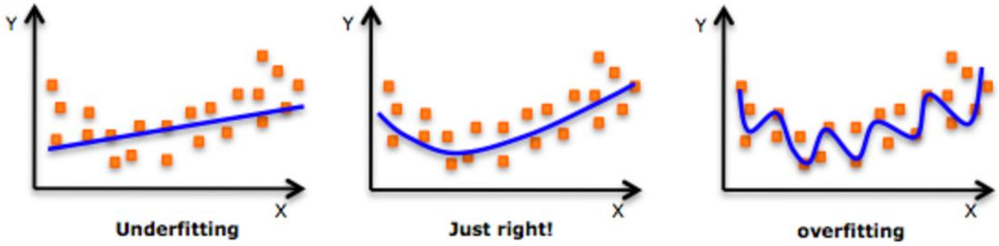
This is overfitting!



Performance

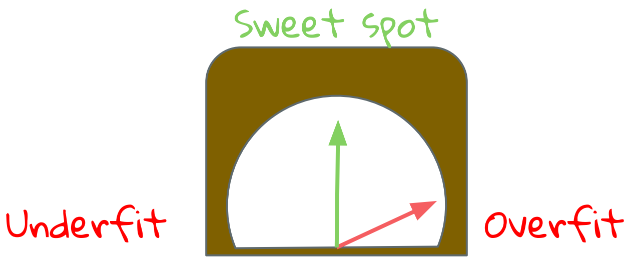
Ability to generalize to unseen data

Remember that the most important goal we have is to build a model that will generalize well to unseen data.



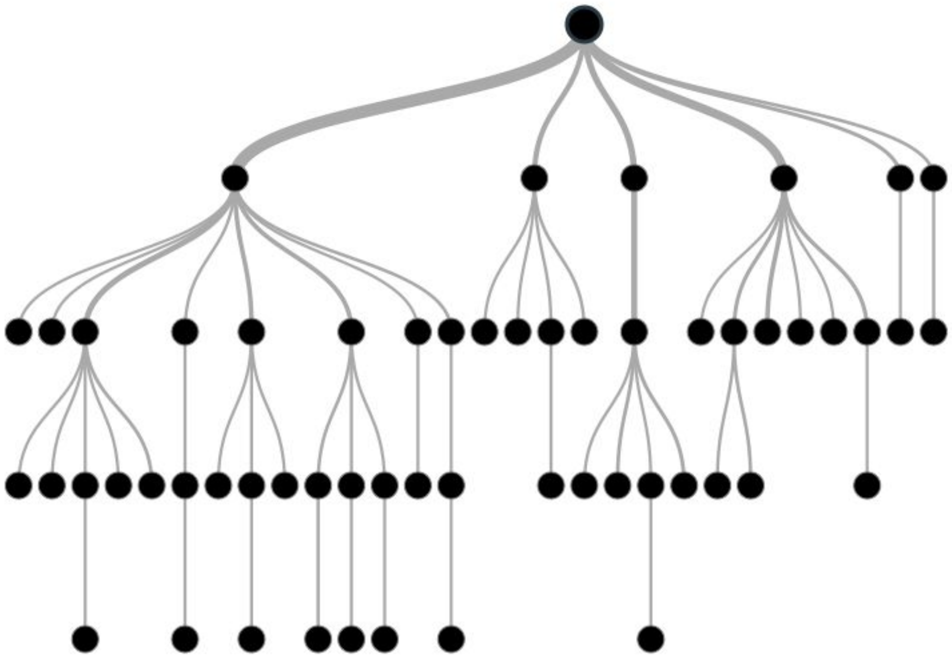
If our train data set overfits, it will not generalize to our test set (unseen data) well.

However, if it underfits, we are not capturing the complexity of the relationship and our loss will be high!



Our goal in evaluating performance is to find a sweet spot between overfitting and underfitting.

Unfortunately, **decision trees are very prone to overfitting**. We have to be very careful in evaluating this.



Each time we add a node, we fit additional models to subsets of the data . This means we start to get to know our train data really well but it will impair our ability to generalize to our test data.

Overfitting: If each terminal node is an individual observation, it is overfit.

Let's take a look at a concrete example of optimizing our model!

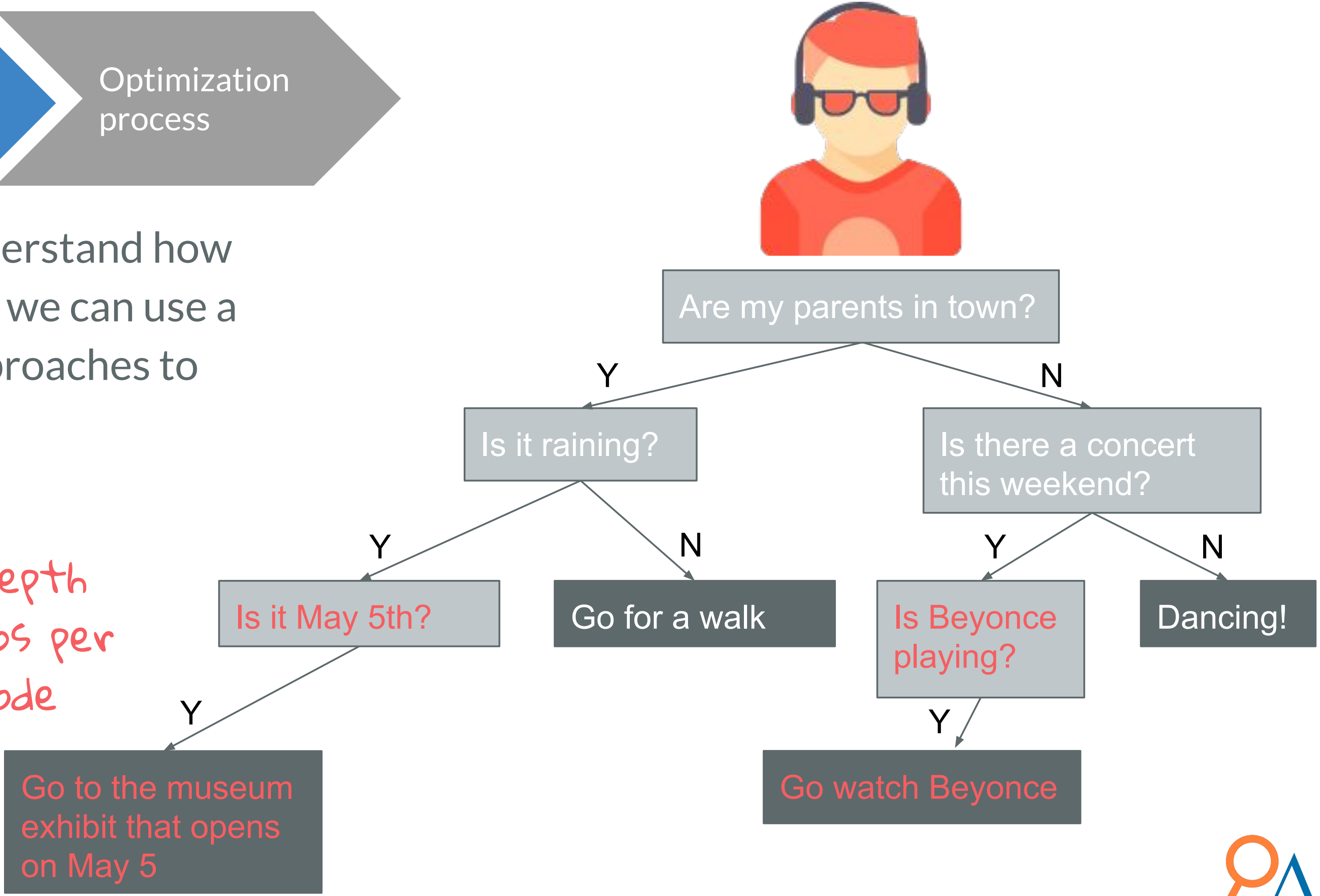
Model Optimization

Learning Methodology

Optimization process

Now that we understand how overfitting is bad we can use a few different approaches to avoid it:

- Pruning
- Maximum Depth
- Minimum obs per terminal node



Learning Methodology

Pruning, max depth and n_ obs in a terminal node are all examples of hyperparameters.

Hyperparameters are higher level settings of a model that are fixed before training begins.

Pruning, max depth and n_ obs in a terminal node are all decision tree hyperparameters set before training.

Their values are **not learned** from the data so Mr. Model cannot say anything about them.

You, not Mr.Model decide what the hyperparameters are!

Learning Methodology

Hyperparameters

Recap: What is a hyperparameter?

- In linear regression, the coefficients were **parameters.**

$$Y = a + b_1x_1 + b_2x_2 + \dots + e$$

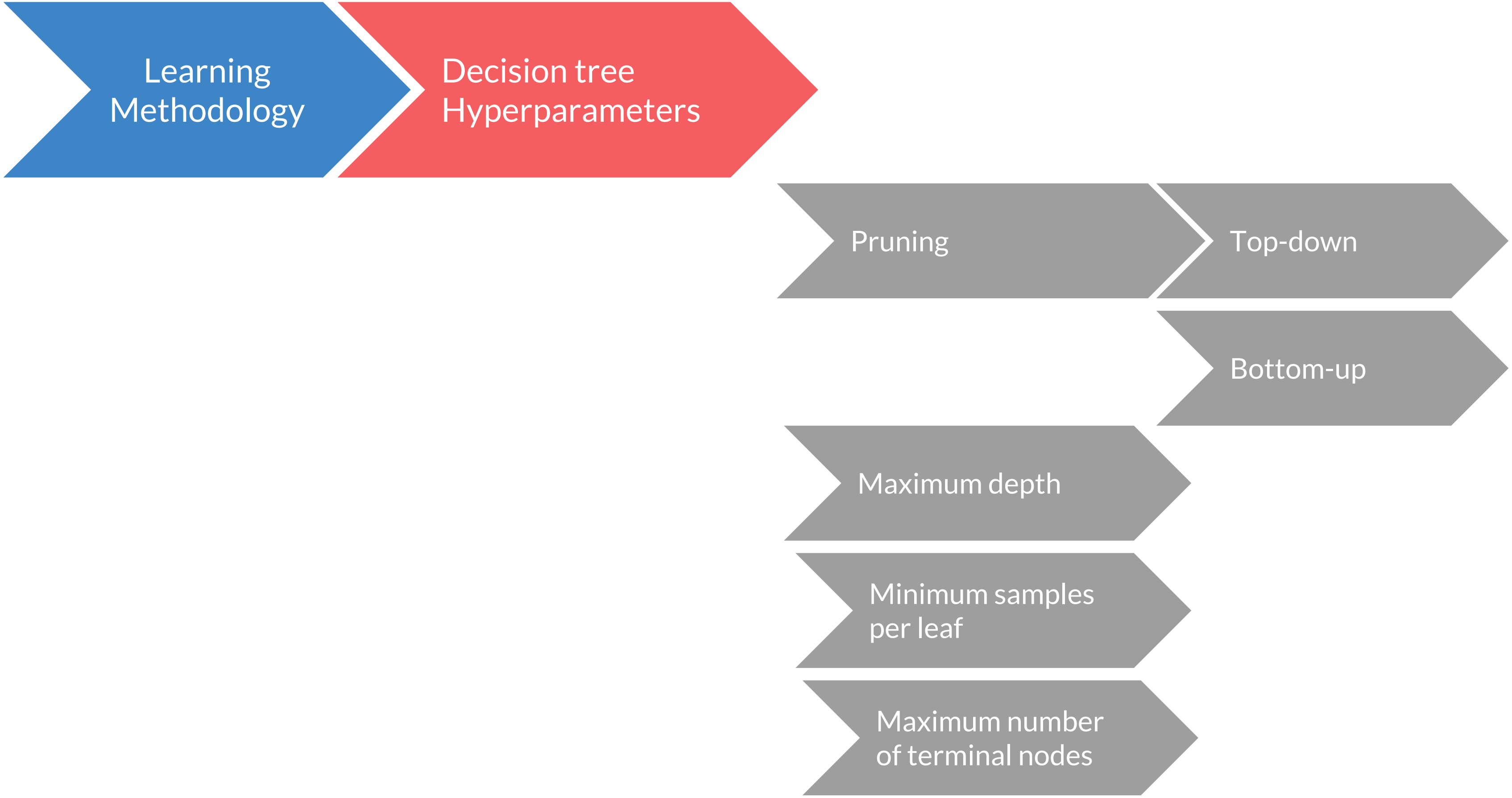
The model decides!

  - Parameters are **learned** from the training data using the chosen algorithm.

- Hyperparameters cannot be learned from the training process. They express "higher-level" properties of the model such as its **complexity** or **how fast it should learn.**

We decide!

Source: Quora


Learning Methodology
Decision tree Hyperparameters
Pruning
Top-down
Bottom-up
Maximum depth
Minimum samples per leaf
Maximum number of terminal nodes

Hyperparameters

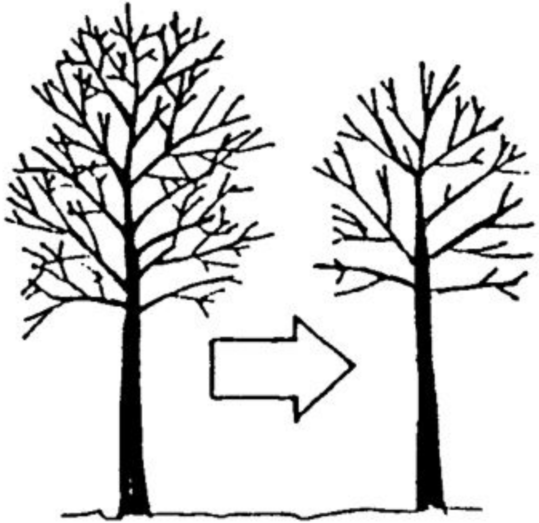
Pruning

Pruning reduces the number of splits in the decision tree.

Limiting the number of splits minimizes the problem of **overfitting**.

Two approaches:

- Pre-pruning (top-down)
- Post-pruning (bottom-up)



Read more: [Notre Dame's Data Mining class, CSE 40647]

Hyperparameters

pre-Pruning

Pre-pruning is a top down approach where the tree is limited in depth before it fully grows.

Pre-pruning slows the algorithm before it becomes a fully grown tree. This prevents irrelevant splits.

- E.g. In the malaria diagnosis example, it probably wouldn’t make sense to include questions about a person’s favorite color.
  - It might cause a split, but it likely wouldn’t be a meaningful split. May lead to overfitting

A useful analogy in nature is a bonsai tree. This is a tree whose growth is slowed starting from when it's a sapling.

Hyperparameters | post-Pruning

Post-pruning is a bottom up approach where the tree is limited in depth after it fully grows.

Grow the full tree, then prune by merging terminal nodes together.

1. Split data into training and validation set
2. Using the decision tree yielded from the training set, merge two terminal nodes together
3. Calculate error of tree with merged nodes and tree without merged nodes
4. If tree with merged nodes has lower error, merge leaf nodes and repeat 2-4.

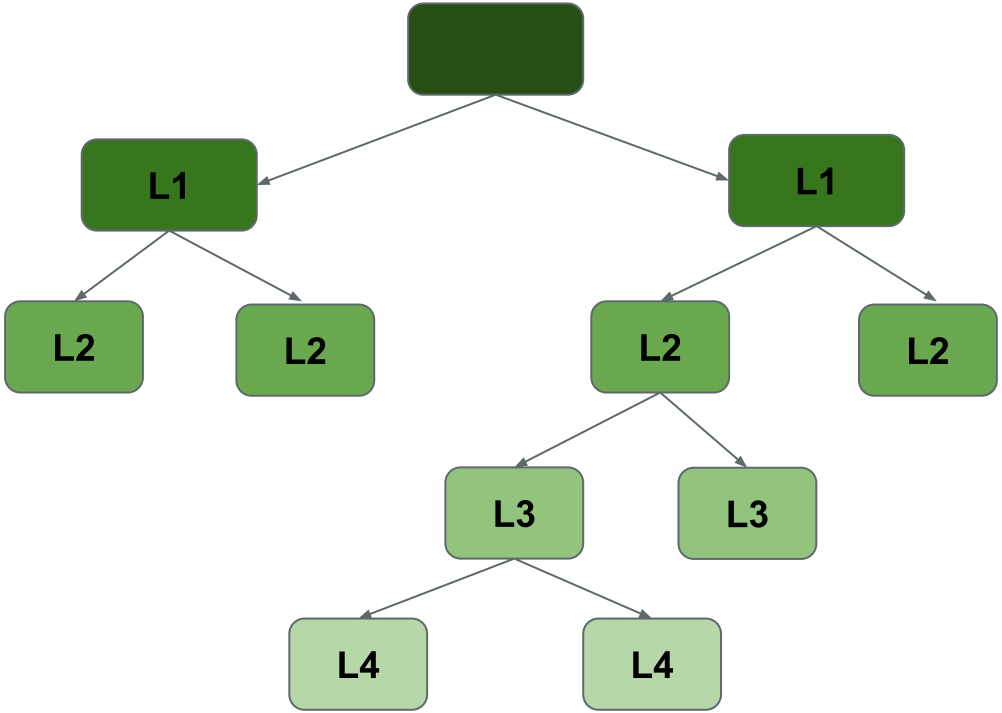
A useful analogy in nature is pruned bushes. Bushes are grown to their full potential, **then** are cut and shaped.

Hyperparameters

Maximum depth

Maximum depth defines the maximum number of layers a tree can have.

For example, a maximum depth of 4 means a tree can be split between 1 to 4 times.

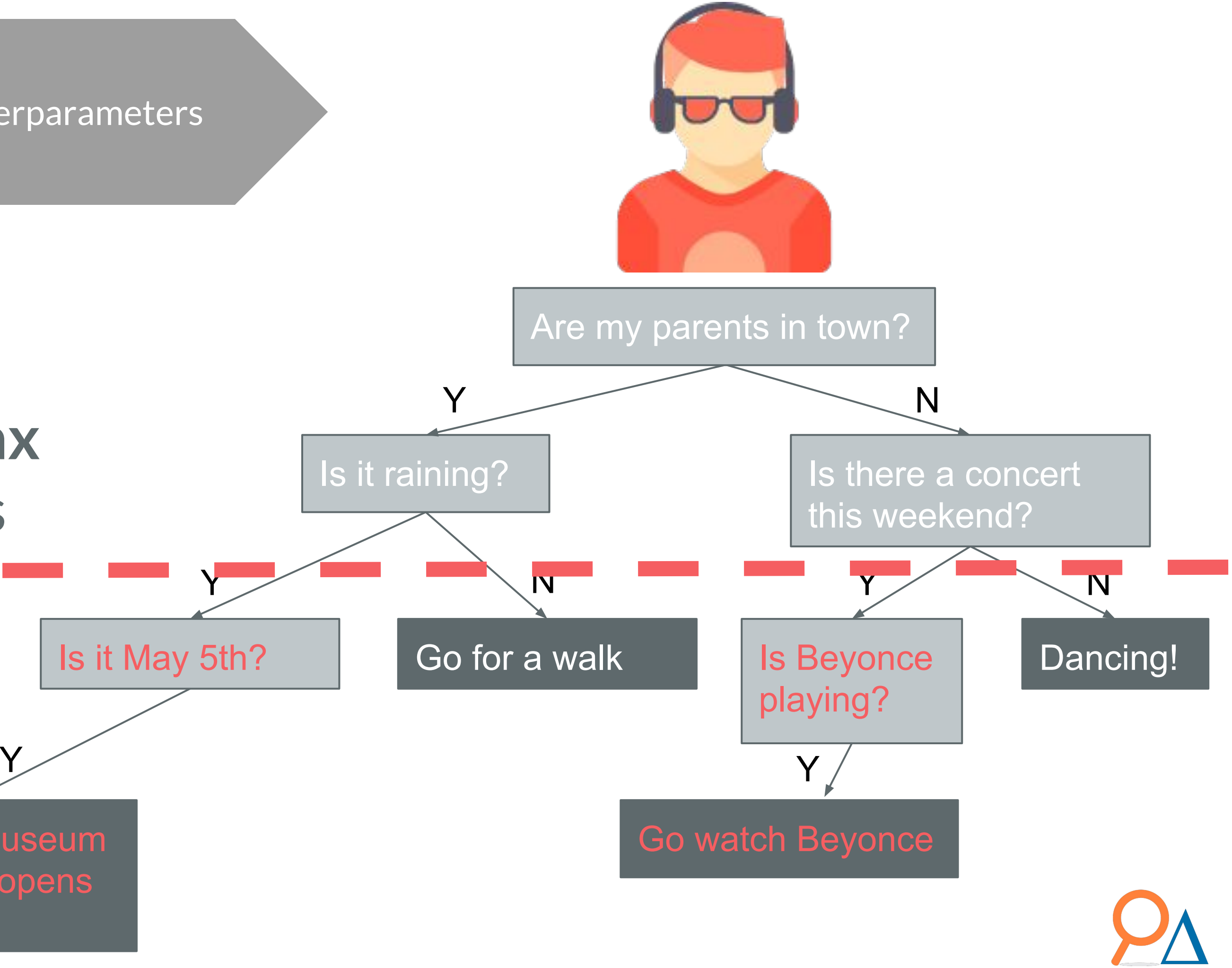


Maximum depth reached, no more splitting

Learning Methodology

Hyperparameters

Here we set the hyperparameter **max depth** to 2 for Sam's decision tree.

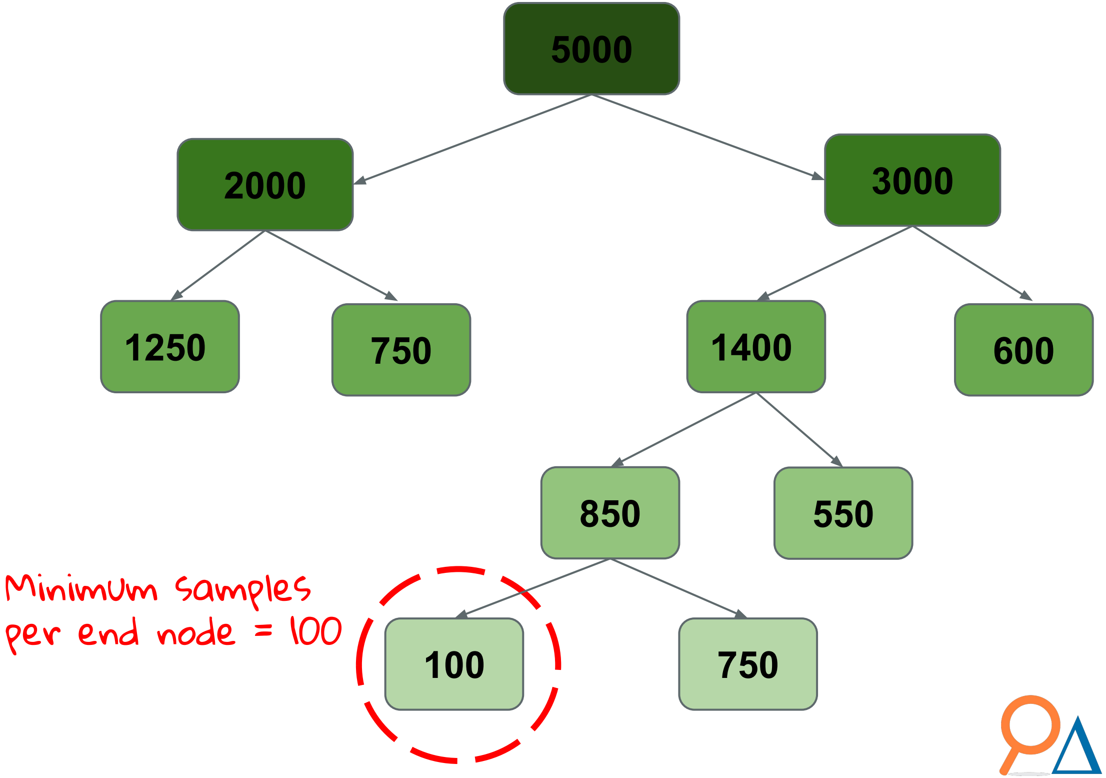


Hyperparameters

Minimum observations per lead

Minimum observations per end node

This hyperparameter establishes a minimum number of observations in an end node, which reduces the possibility of modelling data noise.



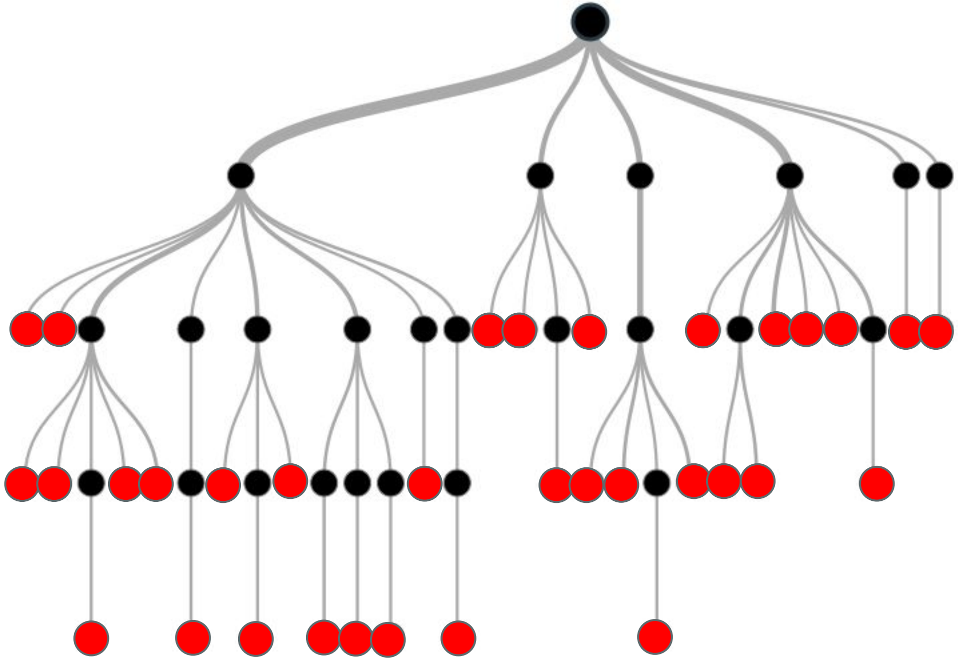
Source

Hyperparameters

Maximum number of terminal nodes

Maximum number of terminal nodes limits the number of branches in our tree.

The maximum terminal nodes limits the number of branches in our tree. Again, this limits overfitting and reduces the possibility of modelling noise.



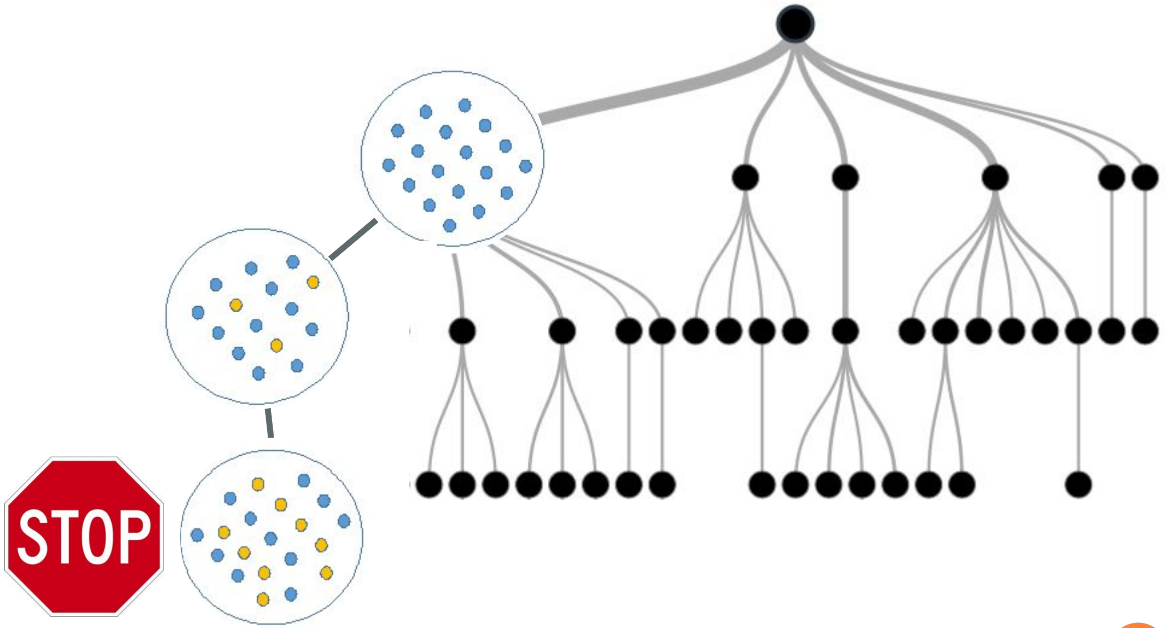
= terminal nodes

Hyperparameters

Minimum impurity

Minimum impurity

- Recall the Information Gain cost function
- The model iterates until it's reached a certain level of impurity



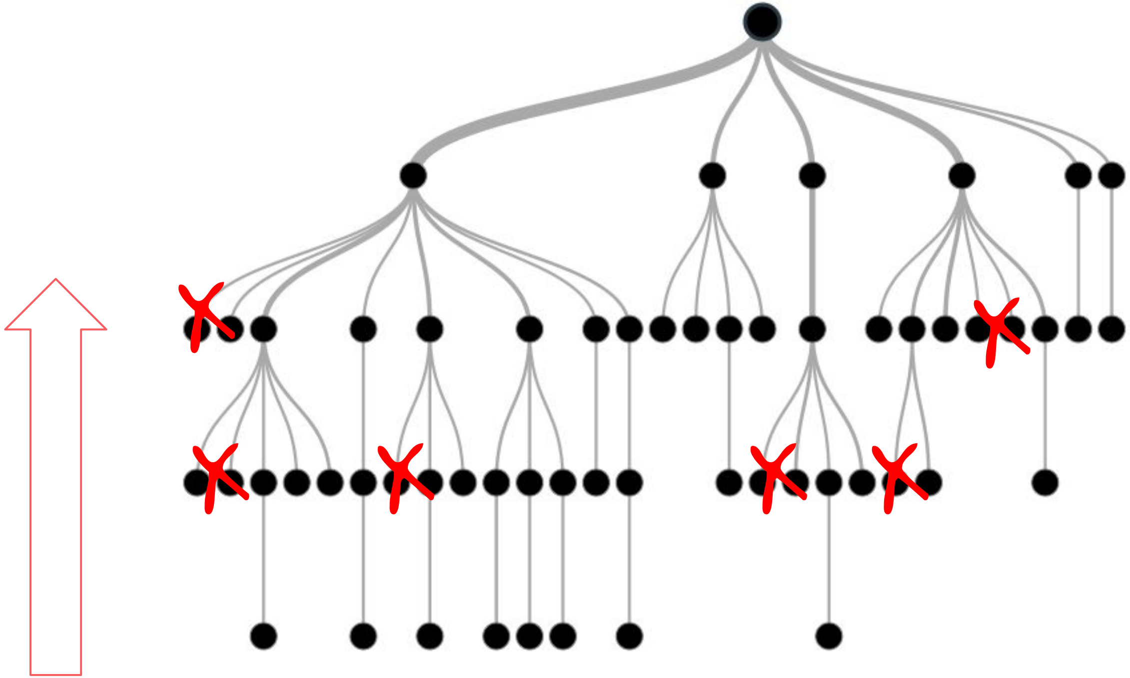
Which hyperparameter should I use?

No objectively best hyperparameter for each model.
Use trial and error - see how each changes your results!

Learning Methodology

Optimization Process

Bottom-up pruning



Selectively merges some terminal nodes together

Bottom-up pruning

Let's see how a single merge works:

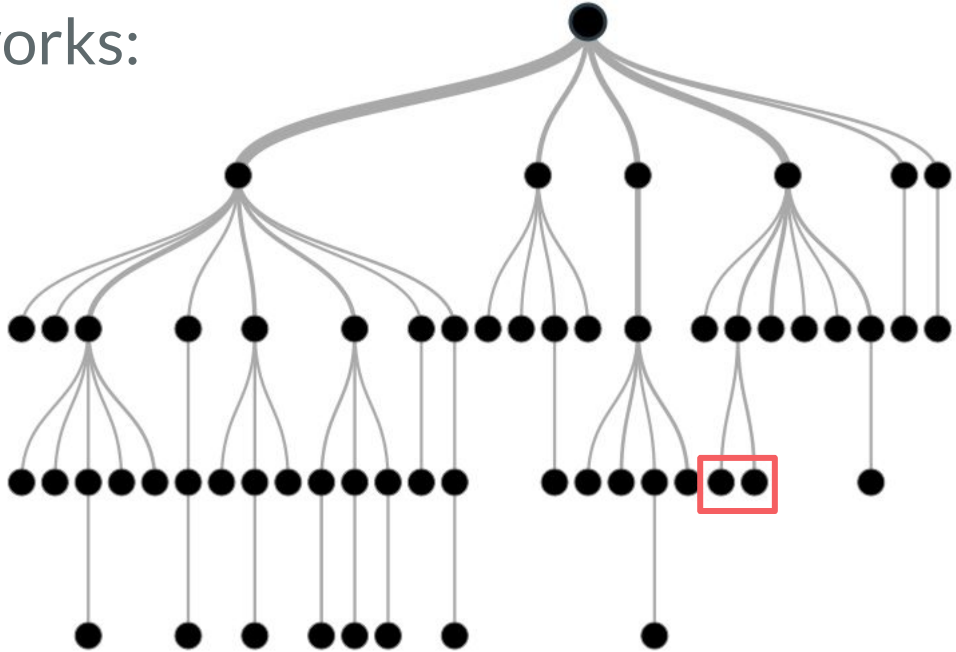
The model randomly selects the merge marked in the tree.

The model calculates:

Error of tree without merge = a
Error of tree with merge = b

If b < a, it chooses to merge!

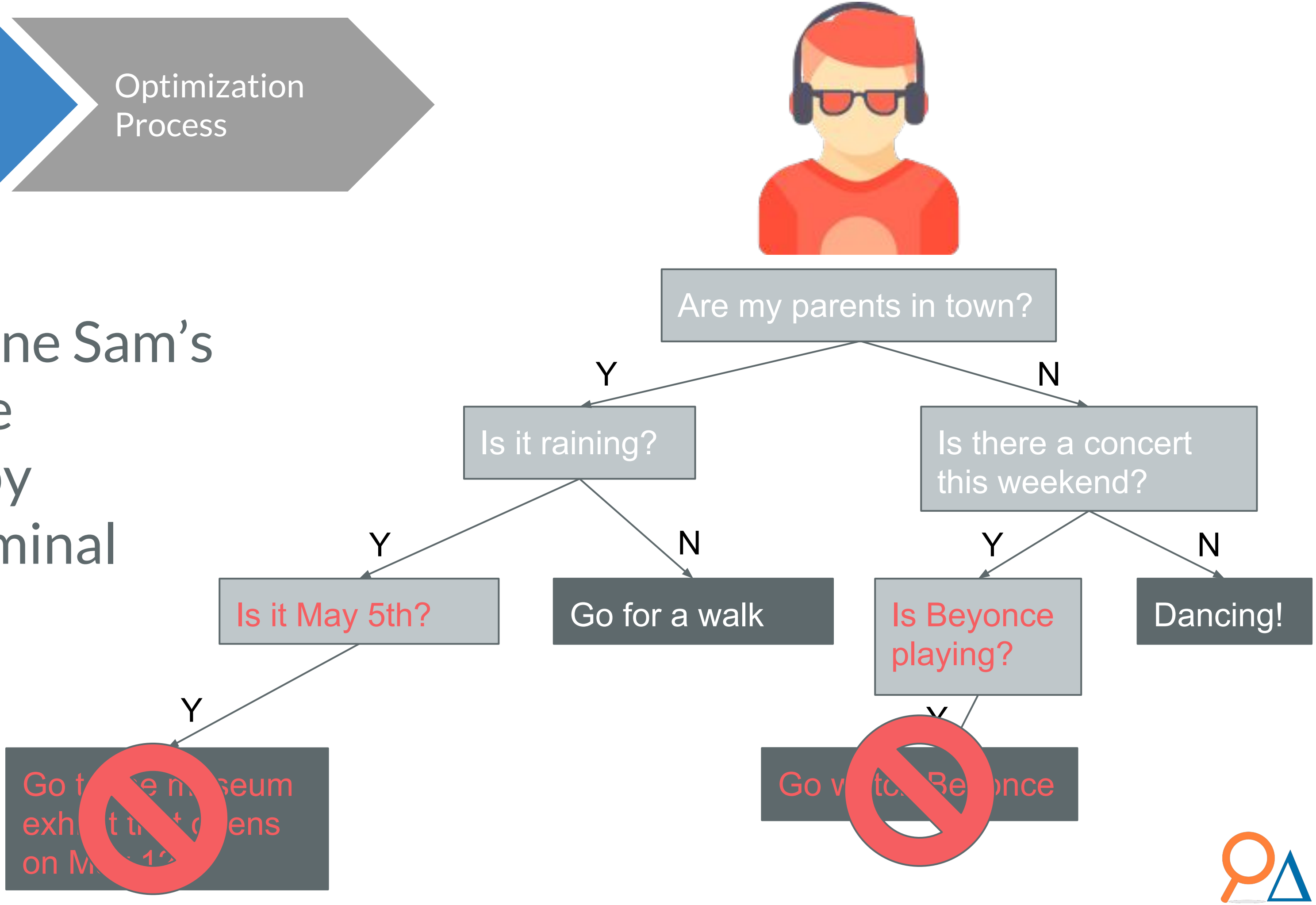
… And repeat



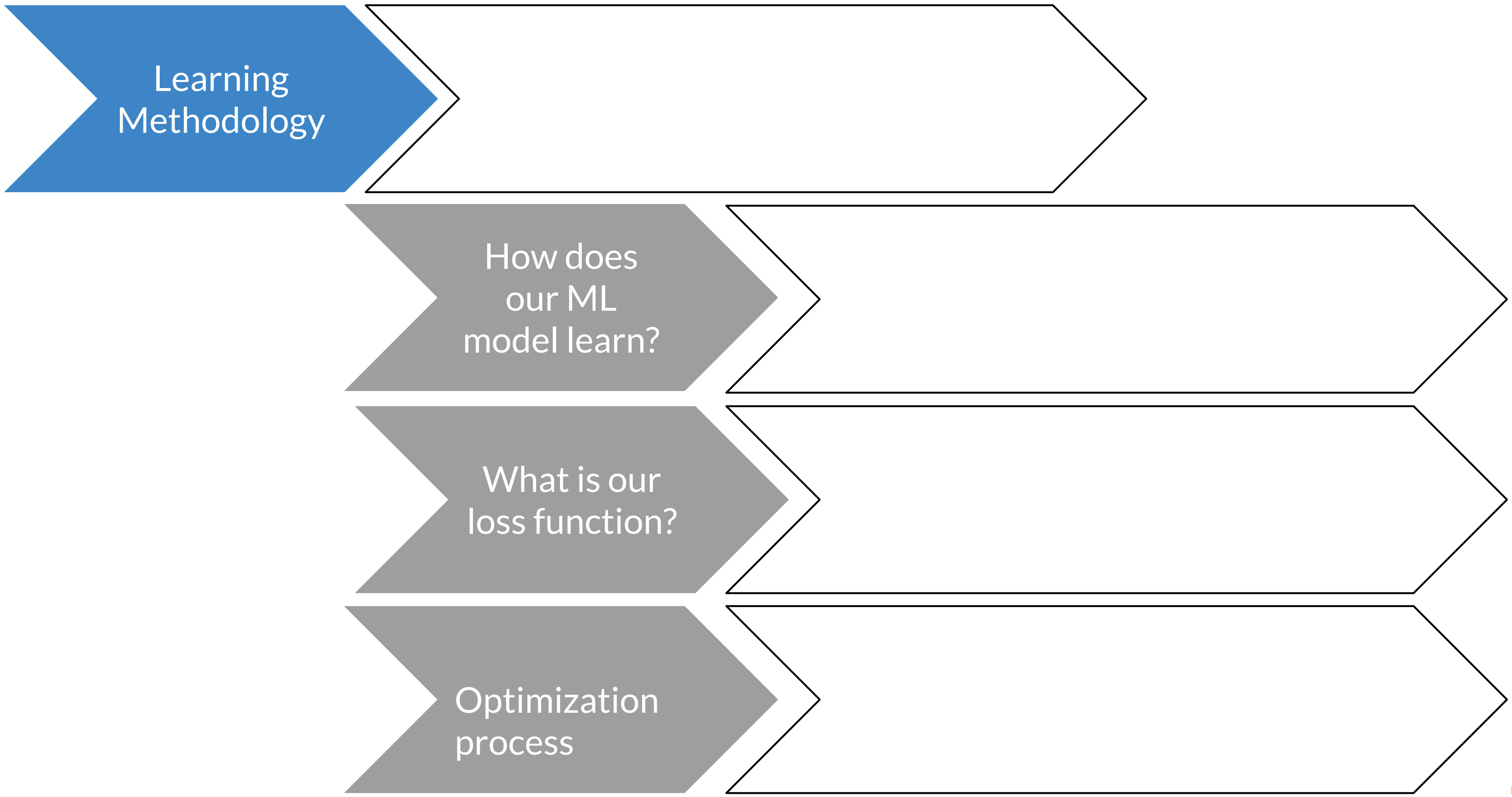
Learning Methodology

Optimization Process

Here we prune Sam's decision tree bottom-up by merging terminal nodes.



Learning Methodology



Learning Methodology

Learning Methodology	Supervised Learning
How does our ML model learn?	Iteration! Evaluating different splits, deciding which one's best
What is our loss function?	Gini Index, Information gain, RMSE
Optimization process	Pruning

End of theory

Checklist:

- ✓ Decision trees
  - ✓ Intuition
  - ✓ The “best” split
  - ✓ Model performance
  - ✓ Model optimization (pruning)

Advanced resources

Want to take this further? Here are some resources we recommend:

- Textbooks
  - An Introduction to Statistical Learning with Applications in R (James, Witten, Hastie and Tibshirani): Chapter 8
  - The Elements of Statistical Learning: Data Mining, Inference, and Prediction (Hastie, Tibshirani, Friedman): Chapter 9
- Online resources
  - Analytics Vidhya's guide to understanding tree-based methods