Ensemble Algorithms

Delta Analytics builds technical capacity around the world.

This course content is being actively developed by Delta Analytics, a 501(c)3 Bay Area nonprofit that aims to empower communities to leverage their data for good.

Please reach out with any questions or feedback to inquiry@deltanalytics.org.

Find out more about our mission here.

Module 6:
Ensemble approaches

Module Checklist:

- ❑ Ensemble approaches
  - ❑ Bootstrap
  - ❑ Bagging
  - ❑ Random forest
  - ❑ Boosting

Where are we?

Question → Exploratory Analysis → Modeling Phase → Validation Phase

<u>What we've done:</u>
Exploratory Analysis
Linear Regression (our first model)
Decision Trees (our second model)

Building intuition

<u>Up next:</u>
Even more models!

Expanding our toolkit

Source: Udacity - [Model Building and Validation]

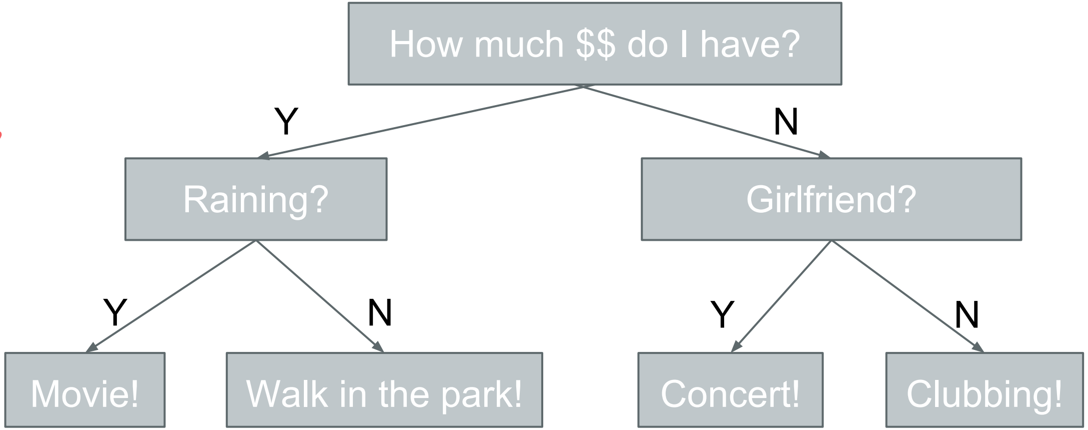
Decision Tree Task

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.

Recap: 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**.



Source: Friedman, Hastie, and Tibshirani. The elements of statistical learning. Vol. 1. Springer, Berlin: Springer series in statistics, 2001.

Decision Tree Task

Recap: You are Sam's weekend planner. What should she do this weekend?

You’ve run a decision tree for Sam, and now you’ve got a model. But does it work well?

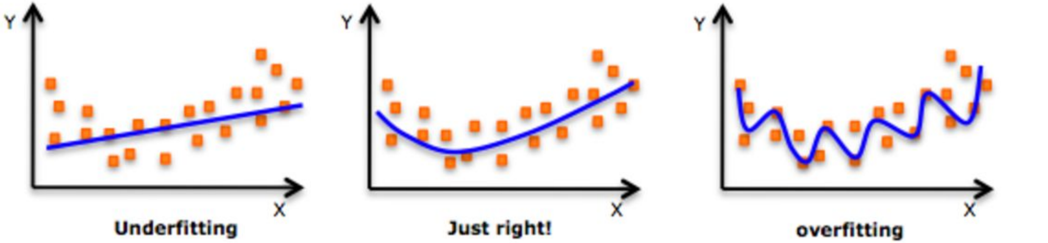
As always, we use our test data to check our model, ***before we*** tell him what to do with this weekend.

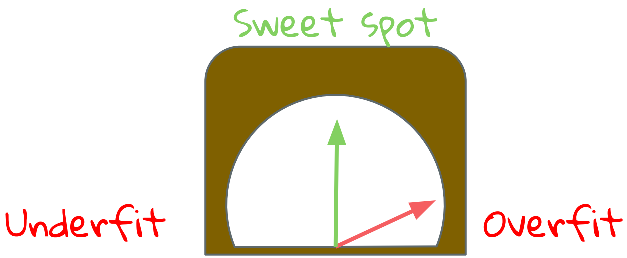
Performance

Ability to generalize to unseen data

Our most important goal is to build a model that will generalize well to unseen data.

Recall our discussion of over and underfitting in previous modules:



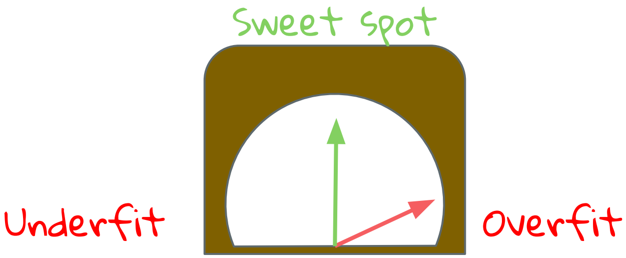


Our goal in evaluating performance is to find a sweet spot between overfitting and underfitting.

Performance

How do we measure underfitting/ overfitting?

Figuring out if you are overfitting or underfitting involves knowing how to compare you train to to your test results.

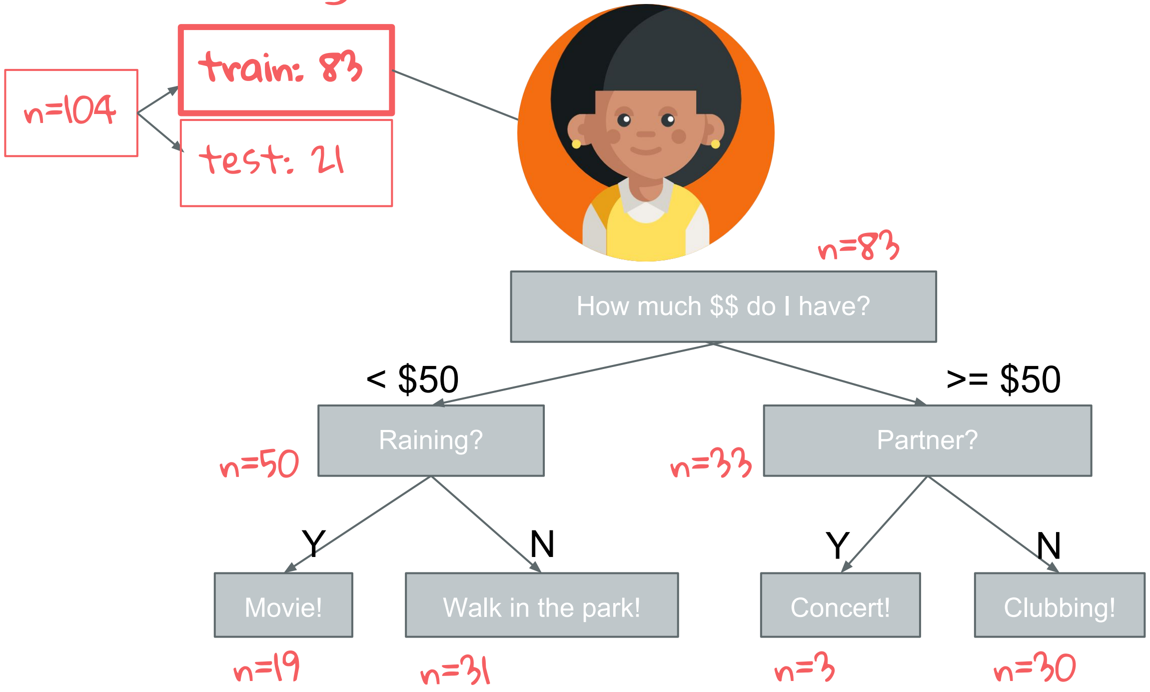


Training R2	Relationship	Test R2	Condition
high	>	low	Overfitting
high	~	high	Sweet spot
low	~	low	Underfitting
low	<	high	**never happens**

Performance

Model evaluation

Let's see how this works in practice. Firstly, we train our model f(x) using training data.

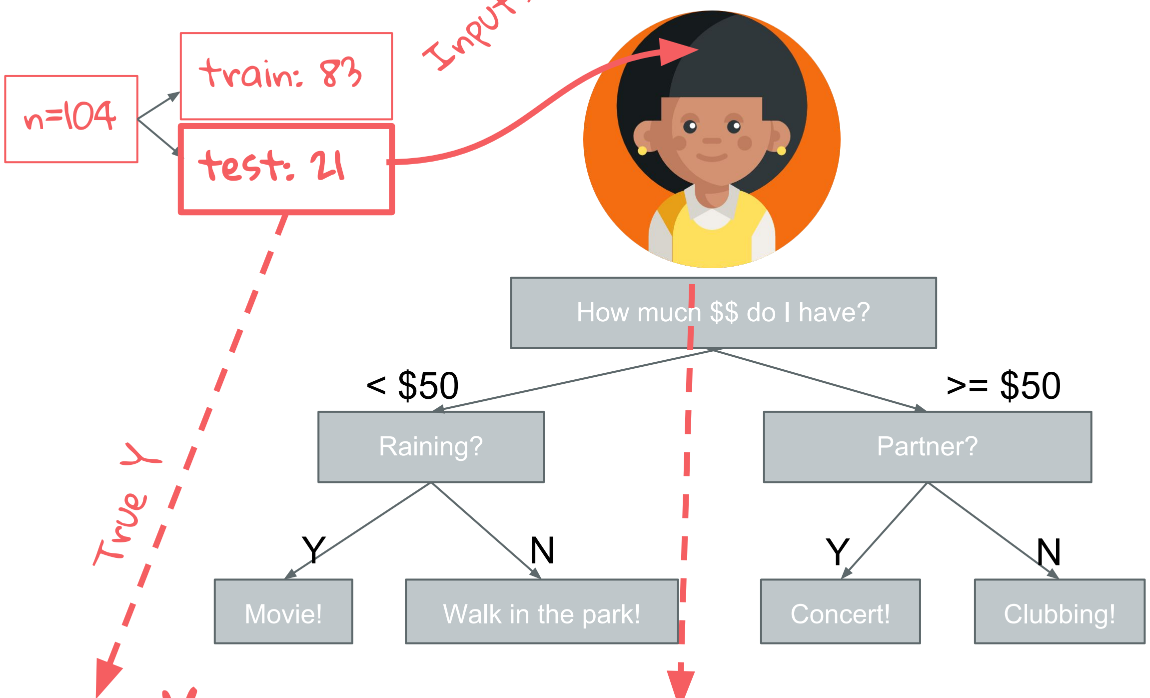


1. **Split data into train/test**
2. **Run model on train data**
3. Test model on test data

Model error = Train True Y - Train Y*

Performance → Model evaluation

Now, we use our f(x) developed using training data to score unseen test data.



1. Split data into train/test
2. Run model on train data
3. **Test model on test data**

Remember generalization error?
Review Module 4!

Test True Y

Output: Test Y*

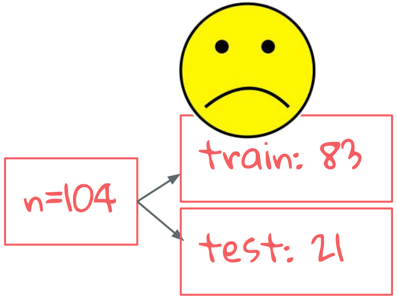
Generalization error = Test True Y - Test Y*

There are some shortcomings associated with the hold out method as a way to do model evaluation.

The holdout set method is great - it lets us test our model on unseen data, the most important metric for any model.

However, one potential problem arises:

What if our test dataset, even though it was picked randomly, is unrepresentative of the data?

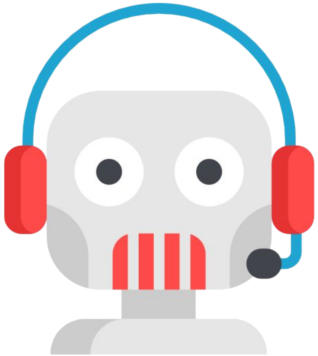


E.g. We managed to pick the 21 weekends in Sam's dataset where he had just broken up with his girlfriend, or failed a test, or fought with his friend, and ended up staying home. Then our test set would say that our model is awful and didn't predict Y* accurately.

We can do better...

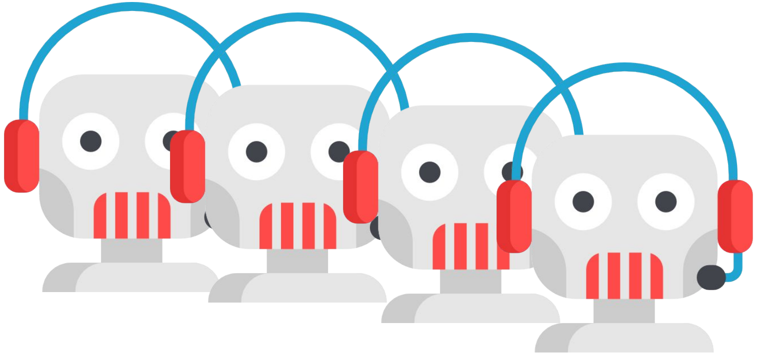
A powerful way to overcome any issues with a biased single holdout is to run the model many times.

If a single run of your model is one expert opining on the data, an **ensemble approach** gathers a *crowd of experts*.



Our Model

VS.



Our Model + model friends

Source: Fortmann-Roe, Accurately Measuring Model Prediction Error.
http://scott.fortmann-roe.com/docs/MeasuringError.html

Ensemble approaches

1. Bootstrapping
2. Bagging
3. Random forests

Central concept: teamwork!

Ensemble Models: model cheat sheet

Bootstrapping

- Method of repeated sampling with replacement

Bagging

- Bootstrap aggregation, or taking the average of the predicted Y*s from bootstrapped samples
- Random forest is a bagging method
- We are able to calculate out-of-bag error instead of using test/train set

Boosting

- Iterative - each tree learns from the tree that was run last.
- The algorithm weights each training example by how incorrectly it was classified.

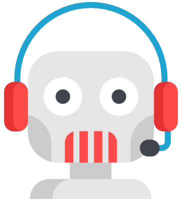
Bootstrapping | Bagging | Random Forest | Boosting

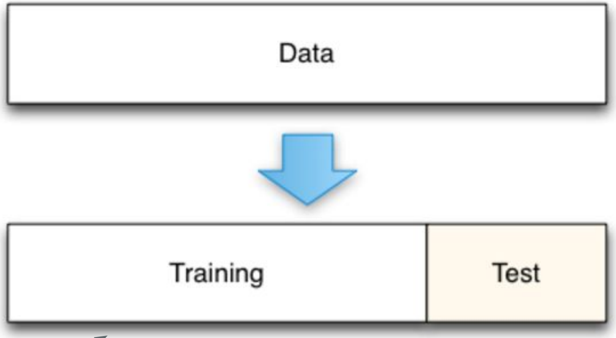
Bootstrapping, bagging, random forests and boosting all leverage a crowd of experts.

Bootstrapping

Instead of only using one holdout, we repeatedly construct different holdouts from the dataset.

Bootstrapping is a ***resampling method*** that takes random samples with replacement from whole dataset.





Example of a single holdout split. Bootstrapping repeats this many, many times. We set the number of holdouts as a hyperparameter.

n=104 → train: 83
n=104 → test: 21

Bootstrapping | Bagging | Random Forest | Boosting

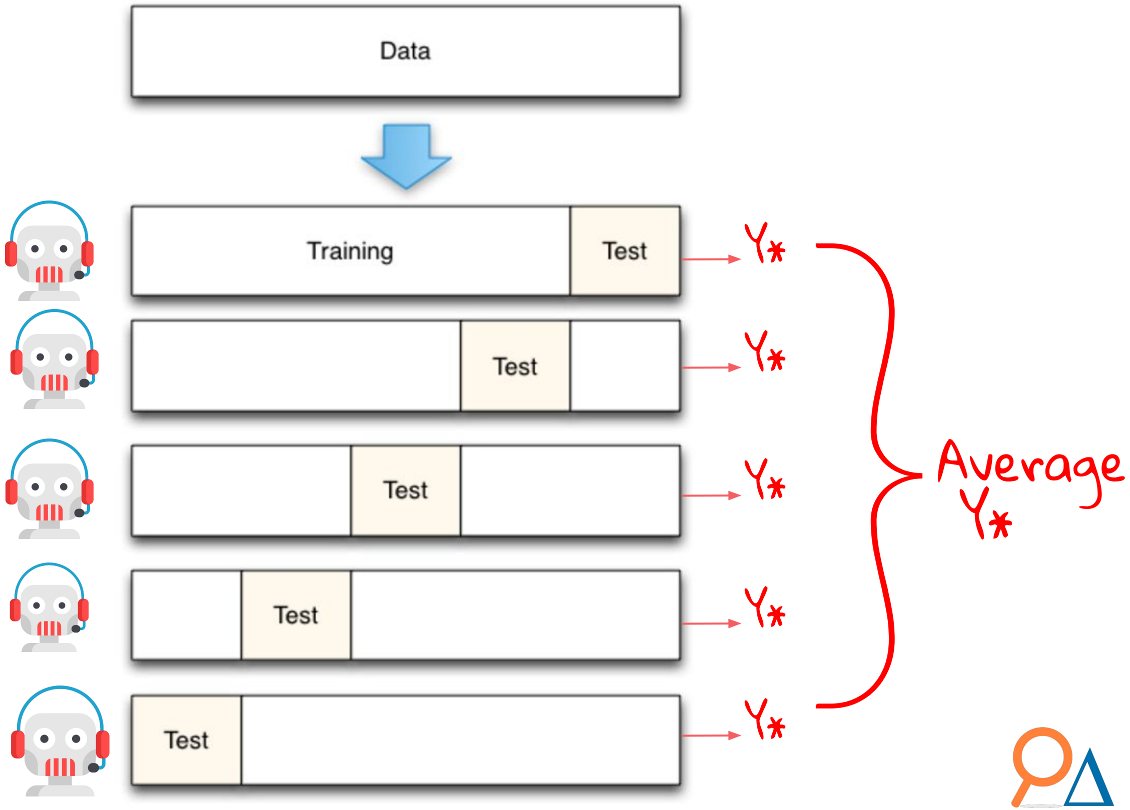
Bagging is an implementation of bootstrapping: it involves taking the average of the random samples drawn by bootstrapping.

Bagging

Bagging improves upon a single holdout by taking the average predicted Y* of boosted random samples.

We train multiple models on random subsets of the datasets and average the predictions.

By averaging the predictions, any chance of **unrepresentative training sets** is reduced.



Bagging

vs.

Normal holdout

n=104

train: 83 → Y_*

train: 83 → Y_*

train: 83 → Y_*

train: 83 → Y_*

train: 83 → Y_*

Average Y_*

n=104

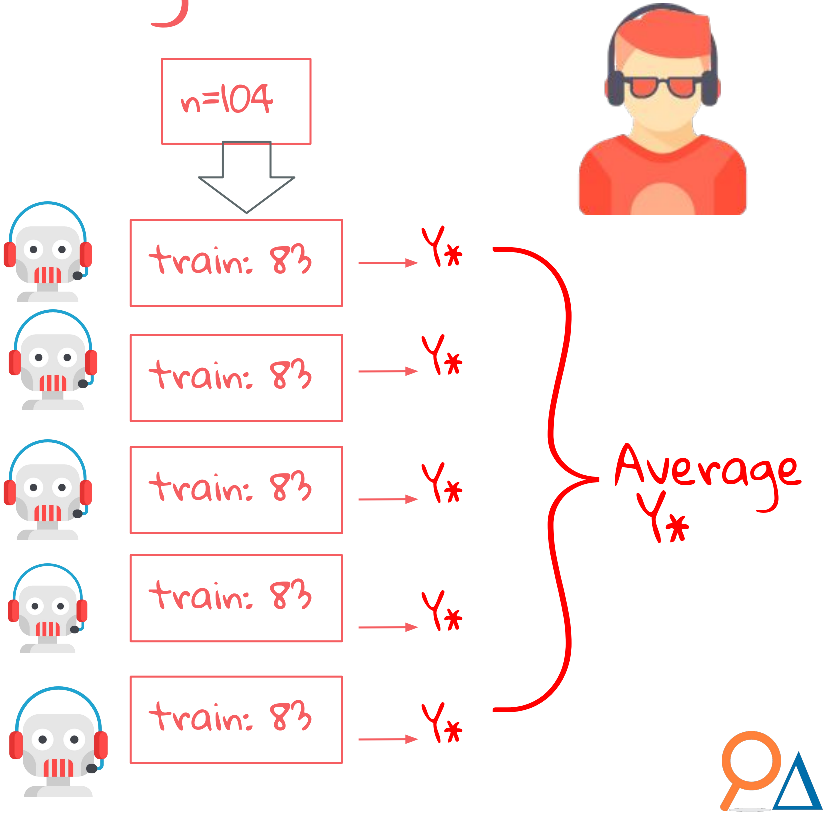
train: 83 → Y_*

Which do you think does a better job of estimating true Y?

Bagging

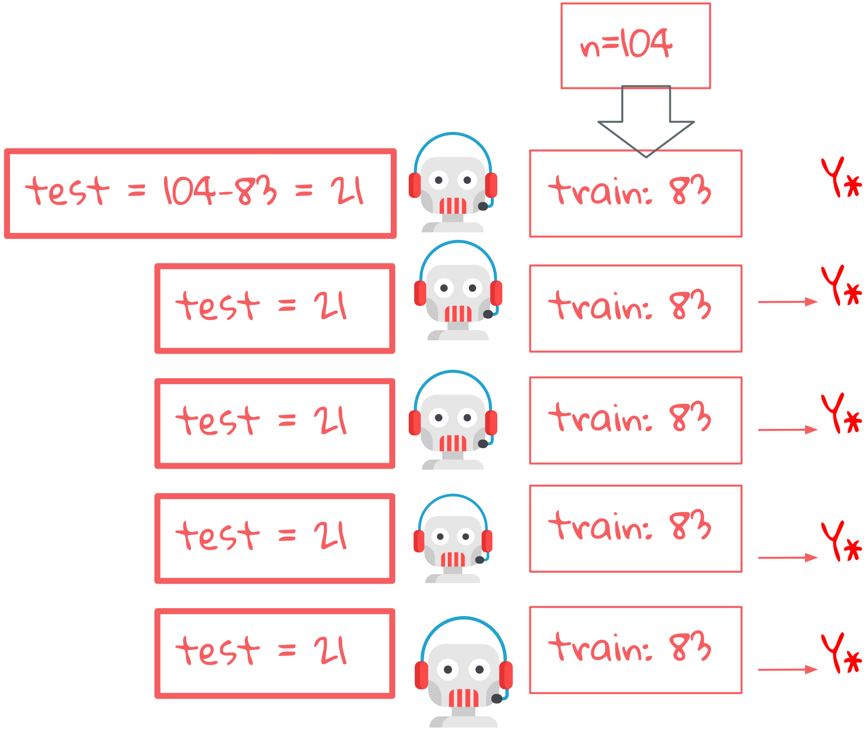
Bagging tends to always outperform a single holdout.

In Sam's case, we still have the problem unrepresentative train dataset. However, now that we're taking different train sets and averaging them, ***the chance of an unrepresentative training set over-influencing the Y* is reduced.***



Bagging | Out-of-bag score

Out-of-Bag Score



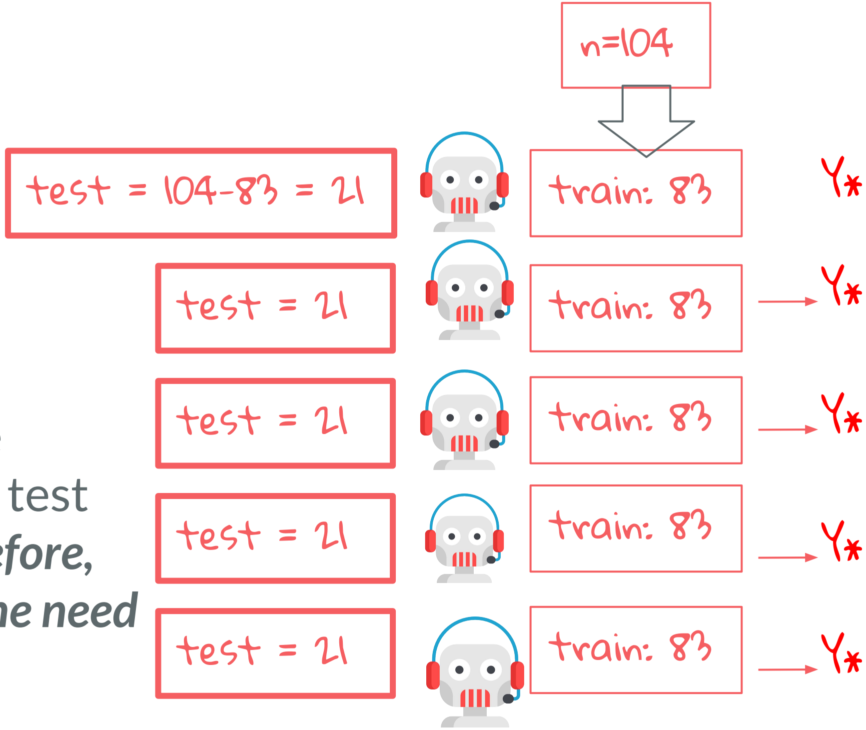
Another amazing benefit of using bagging algorithms is the **out-of-bag score**.

The out-of-bag score is the error rate of observations **not used** in each decision tree.

Source:
https://www.quora.com/What-is-the-out-of-bag-error-in-Random-Forests

Out-of-Bag Score

The out-of-bag score is the error rate of observations **not used** in each decision tree.



Why it matters:
There is empirical evidence to show that the out-of-bag estimate is as accurate as using a test set of the same size as the training set. ***Therefore, using the out-of-bag error estimate removes the need for a set-aside test set.***

Source: Breiman, 1996

Out-of-Bag Score

Out-of-bag score can be calculated for any bootstrap aggregation method, including:

- Random forest
- Bagging
- Boosting

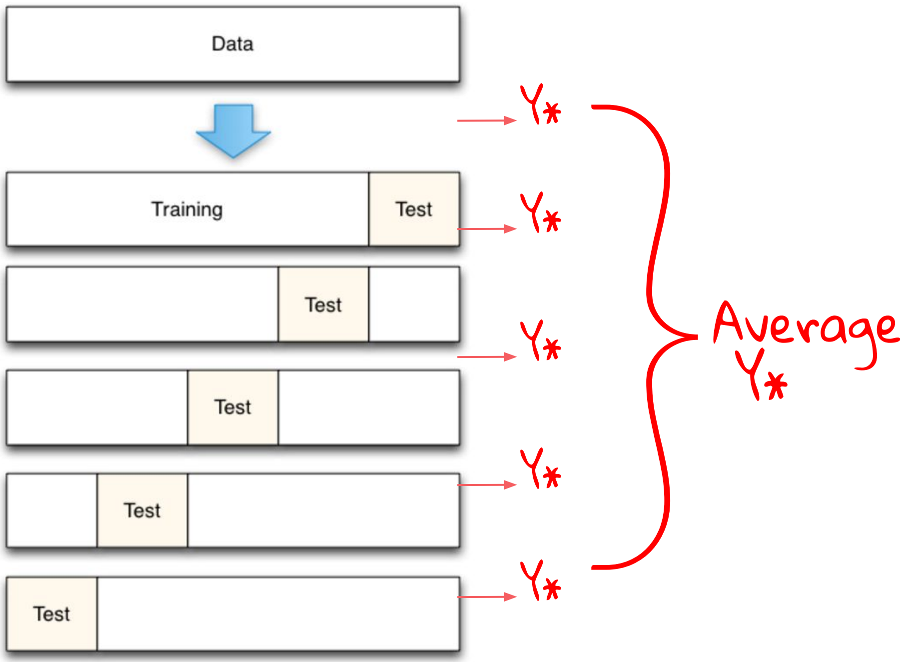
Is bagging perfect? What are some potential tradeoffs?

Bagging

There are a few key limitations to bagging.

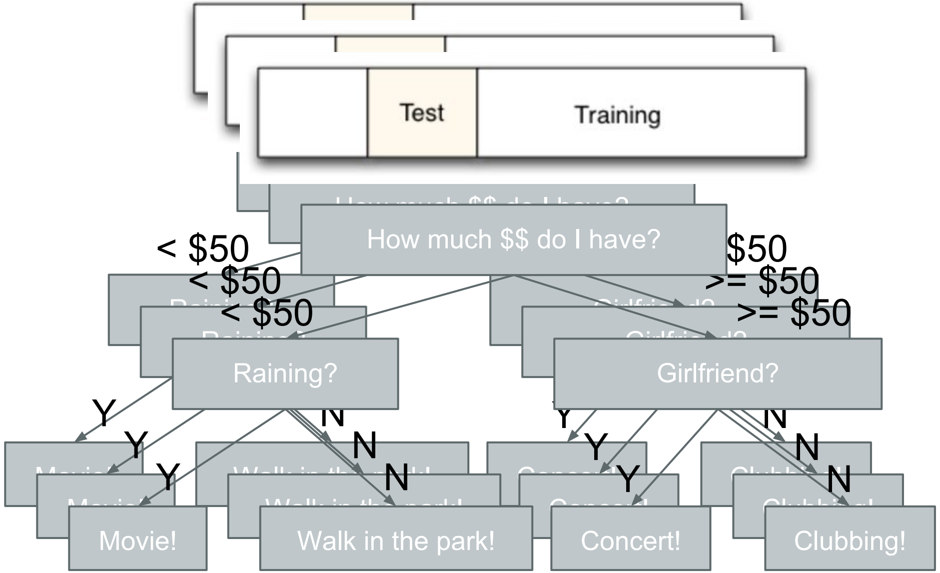
One key trade off is that training and assessing the performance of every additional holdout **costs us computational power and time**.

The computational cost is driven by the data sample size and number of holdouts.



Bagging

A key limitation of bagging is that it may yield correlated (or very similar) trees.

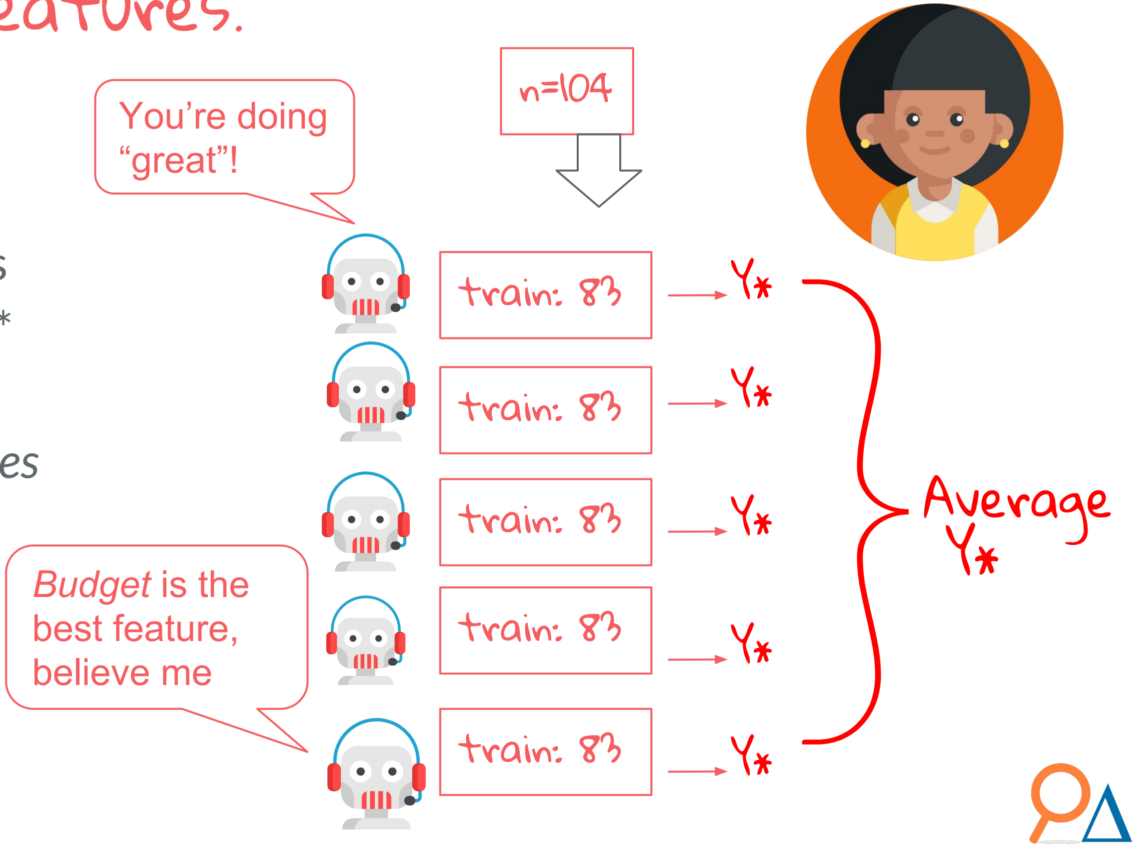


Subsets of the same data may split on the same features and result in very similar predictions.

Bagging

Correlated trees may give us false confidence since they repeatedly yield the same features.

Many identical trees becomes an **echo chamber** of overfitted trees that repeatedly yields a similar Y* value, and repeatedly yields the same important features. *This gives us false confidence in our results.*



We can do better ....

Bootstrapping | Bagging | Random Forest | Boosting

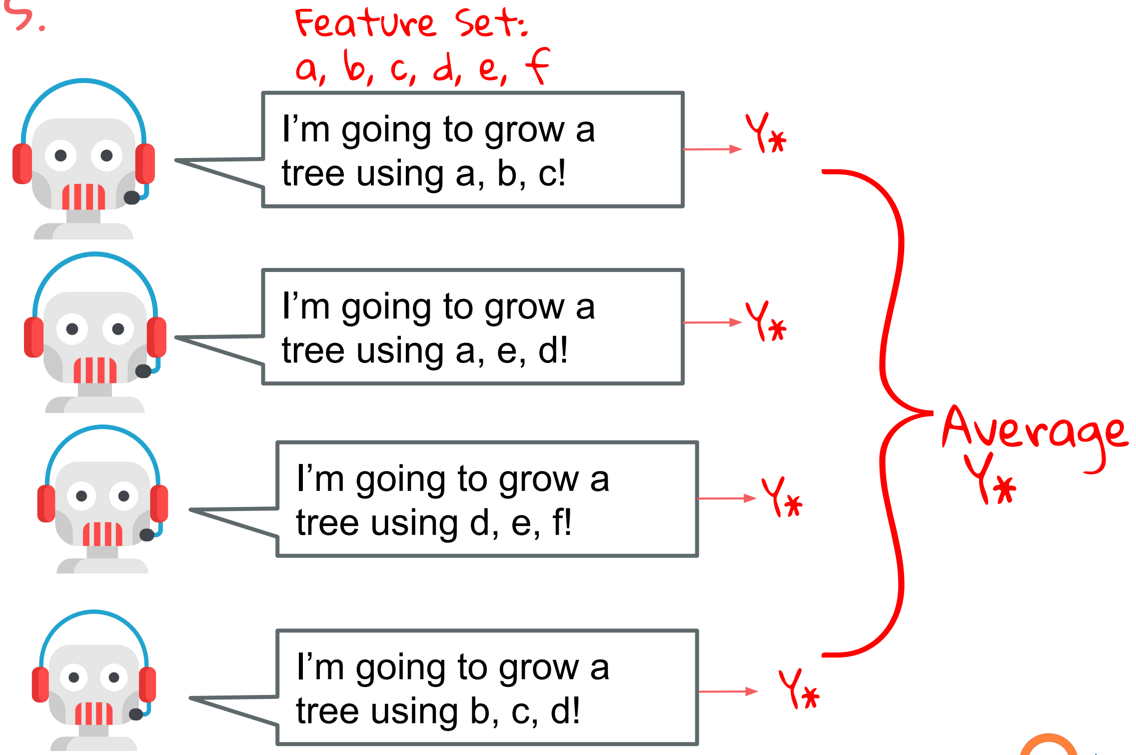
Random forest improves on bagging's tendency to result in correlated trees.

Random Forest

Random forest improves upon bagging by only considering a random subset of features.

Random forest is an implementation of bagging. It improves on bagging by ***de-correlating trees***.

At every split, it only considers a ***random subset*** of the features.



Source: https://dimensionless.in/introduction-to-random-forest/

Performance

Improving on bagging

Random forest adjusts overfit models

Set:
a, b, c, d, e, f

Here, we are still using a subset of the data, but instead of randomly selecting a number of observations, we randomly select some number of features.

Random forest **helps solve the problem of overfitting**.

Note that we can still calculate an accuracy score using OOB

Bootstrapping | Bagging | Random Forest | Boosting

Finally, boosting is a procedure that **iteratively learns** by combining many weak classifiers to produce a powerful committee.

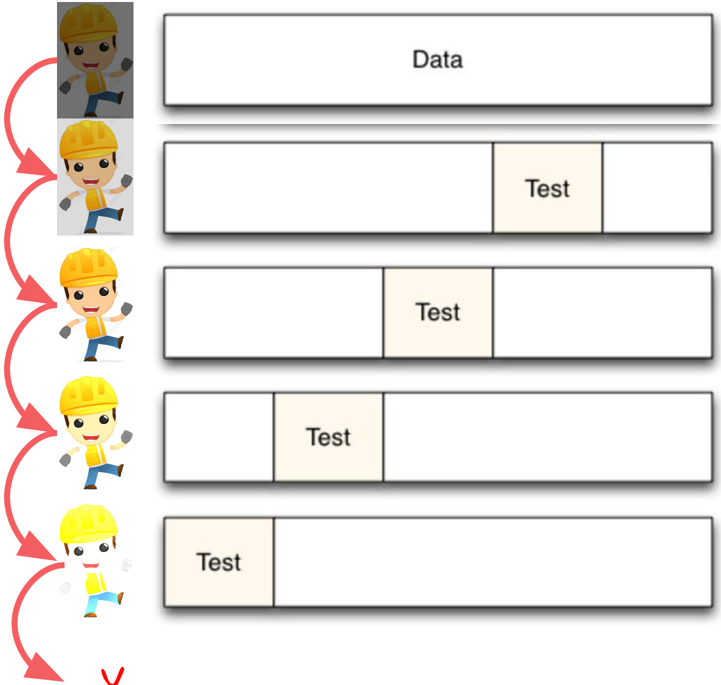
IMPORTANT NOTE: Boosting is one of the most powerful learning ideas introduced in the last 20 years. It sounds similar to but is **fundamentally different from bagging** and other committee-based approaches.

Boosting

Unlike random forest, each tree is not random in boosting

Boosting also creates subsets of training data using bootstrap, but **each tree learns from the previous trees: that is, each tree is not random.**

How does the model learn?



Y_*

Our model gets "brighter"

Source: Carnegie Mellon University, http://www.cs.cmu.edu/~guestrin/Class/10701-S06/Slides/decisiontrees-boosting.pdf

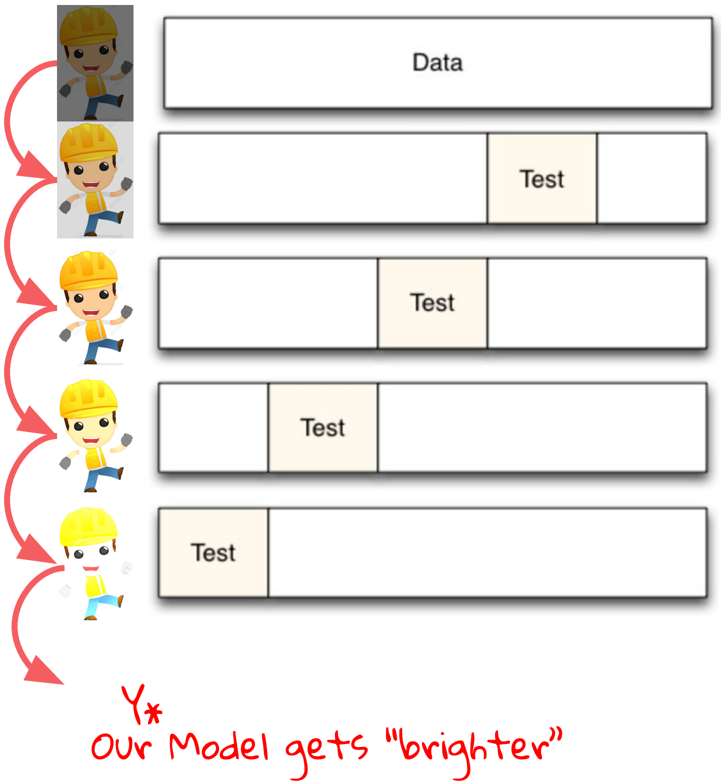
Boosting

Combining weak classifiers = one strong classifier

Boosting uses **many weak classifiers** to make a single strong classifier. A weak classifier is defined as those whose error rates is only slightly better than random guessing.

Boosting sequentially applies weak classification algorithms to repeatedly **modified versions** of the data.

How is the data modified?



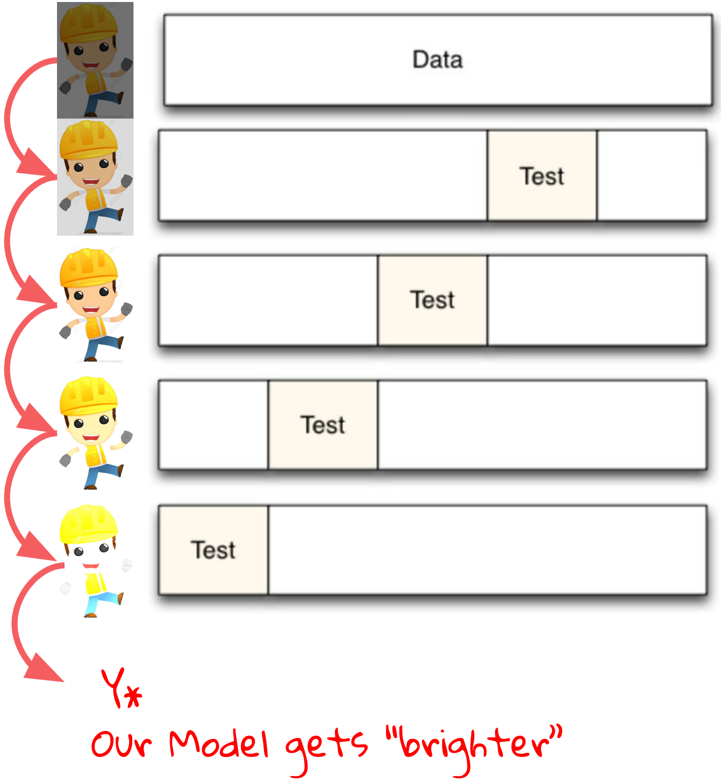
Boosting

Boosting forces the model to focus in on hard-to-classify observations

Each prediction is combined through a **weighted majority vote** to produce the final prediction.

For each iteration, the algorithm weights higher observations that were classified incorrectly. **This forces the algorithm to concentrate on training observations that were classified incorrectly in previous iterations.**

Let's go through each step of the algorithm



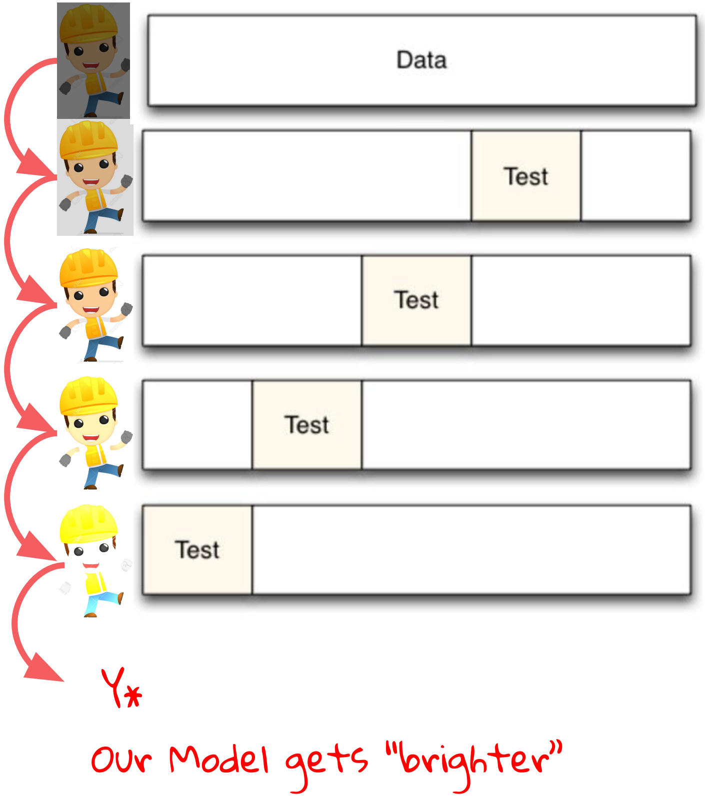
Boosting

Let's go through step by step:

1. Use the whole data set to train a model to produce Y*
2. Evaluate performance (true Y - Y*)
3. Create training set #2 including observations that were ***incorrectly classified***
4. Repeat steps 2-3

Results in low model error, but there is risk of overfitting

Source: https://www.analyticsvidhya.com/blog/2015/09/questions-ensemble-modeling/



Bootstrapping

Bagging

Random Forest

Boosting

We've covered a lot! By now, you have an arsenal of supervised learning algorithms to apply in many situations.

In the next module, we will look at **unsupervised algorithms** and what they can tell us.

End of theory

Module Checklist:

- ✓ Ensemble approaches
  - ✓ Bootstrap
  - ✓ Bagging
  - ✓ Random forest
  - ✓ Boosting

You are on fire! Go straight to the next module here.

Need to slow down and digest? Take a minute to write us an email about what you thought about the course. All feedback small or large welcome!

Email: sara@deltanalytics.org

Congrats! You finished module 6

Find out more about Delta's machine learning for good mission here.