The Build Fellowship

BUILDFELLOWSHIP.COM

OpenAvenues

Weekly Updates

- Please provide a quick update on either:
    - Something you did/saw this week that you thought was interesting
    - What you're looking forward to about this week's workshop

(Reminder - please have your cameras on if possible)

The Build Fellowship

Workshop 6 Model Tuning & Hyperparameter Searching

Recap

Sessions Overview

- Workshop 1 – Project Introduction & Setup
- Workshop 2 – Genomic Data (A2 Assignment)
- Workshop 3 – Data Analysis & Visualization (A3 Assignment)
- Workshop 4 – Featurization & Baseline Modeling (A4 Assignment)
- Workshop 5 – Model Training Approaches (Final Assignment Set)
- **Workshop 6 – Model Tuning**
- Workshop 7 – Performance Evaluation (Final Assignment Code/Testing Due)
- Workshop 8 – Results Presentation & Wrap up (Final Presentation Due)

The Data Science Process

1. Understand the Problem
2. Gather Data
3. Explore & Analyze Data
4. Featurize Data
5. Model Building
6. Evaluate and Compare

Repeat as needed:
Data Science is an iterative process

7. Present Results

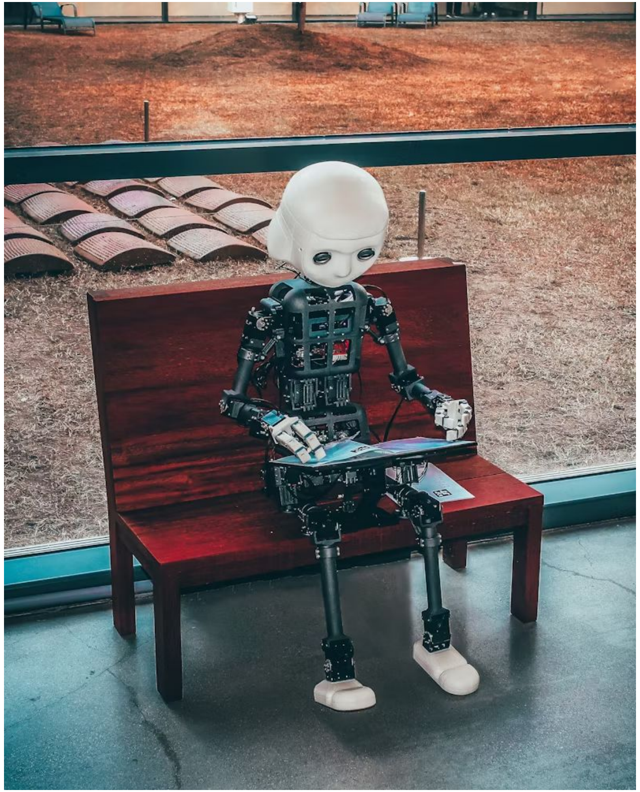
Model Training

- Last week we took a high level tour through different modeling options

- Various models would fit our data structure & features
    - Logistic regression
    - Random Forest
    - Gradient Boosting

- Now working on your final project
    - Can use models above or try your own approach

- What comes next?
    1. Look at common model parameters
    2. Explore hyperparameter tuning
    3. Expand on cross validation (nested)



Searching parameters

Hyperparameter Tuning

What is Model Tuning?

Model performance is significantly impacted by our model building decisions

Default parameters (e.g. last week) may not be a good fit for your data

We wish to optimize model performance:

1. need to search over our parameter space
2. need to compare between parameters
3. need to have a fair comparison between models

There is no one size fits all here - model tuning is an iterative process

Optimized crochet

Monitoring Performance

What are we looking for?

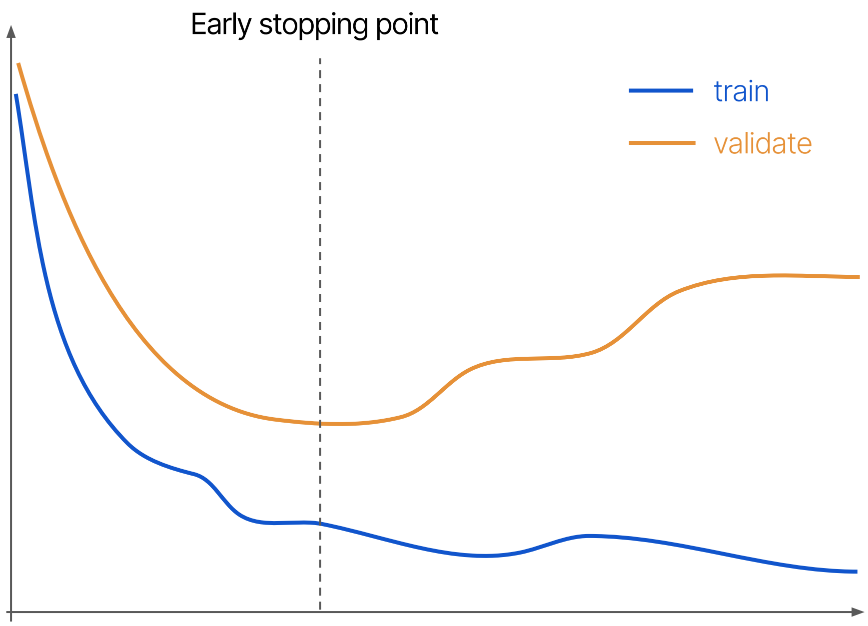
- Minimize validation loss
- Avoid overfitting

In our case:

- Simply compare validation loss between parameter sets

Optional:

- **Early stopping** & other adaptive regularization techniques can help optimize fit during model training



Common Parameters

Which parameters to tune will vary depending on model choice

However many models will share a similar set of parameters:

- Learning rate
- Batch size
- Model complexity (e.g. N trees)
- Regularization (e.g. L1/L2 penalty)

Many dials to turn

The Build Fellowship

Open Avenues FOUNDATION

Parameter Searching Methods

The Build Fellowship

OpenAvenues FOUNDATION

Grid Search

Simplest option - try “all” combinations

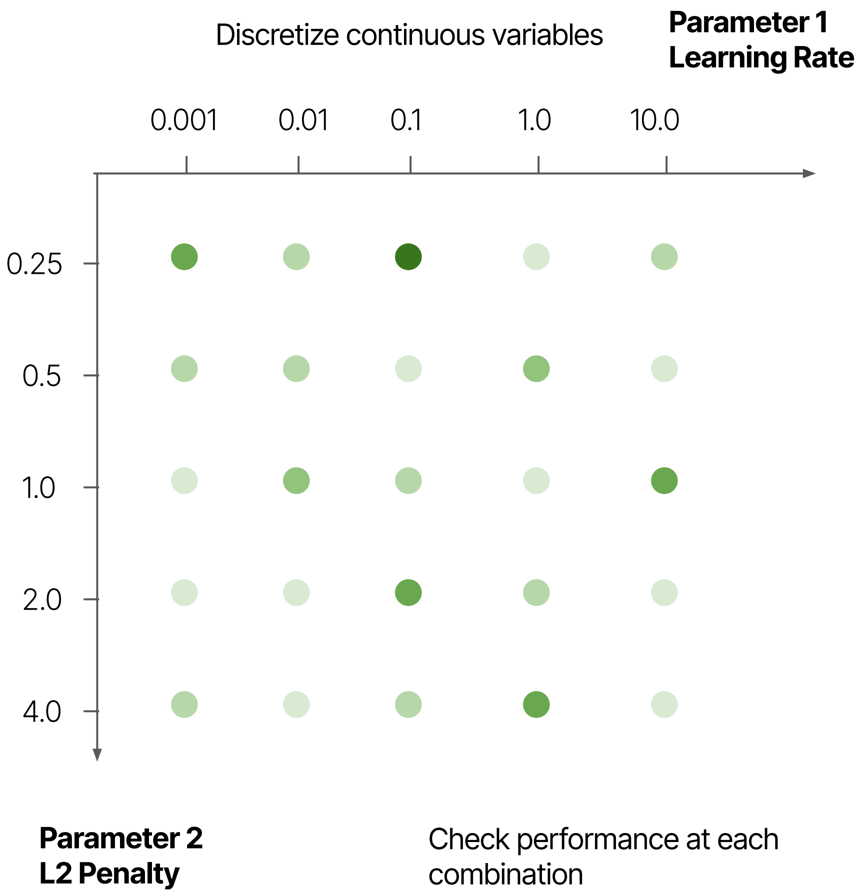
- Categorical variables = try all different options
- Continuous variables = discretize into N options

Brute force search:

- N models = multiply all parameter combinations
- Say 3 parameters, with 5 options each
- N models = 5 * 5 * 5 = 125

Hugely inefficient but covers “full” parameter space

Works better for low parameter numbers



Random Search

Randomly sample from the full parameter space

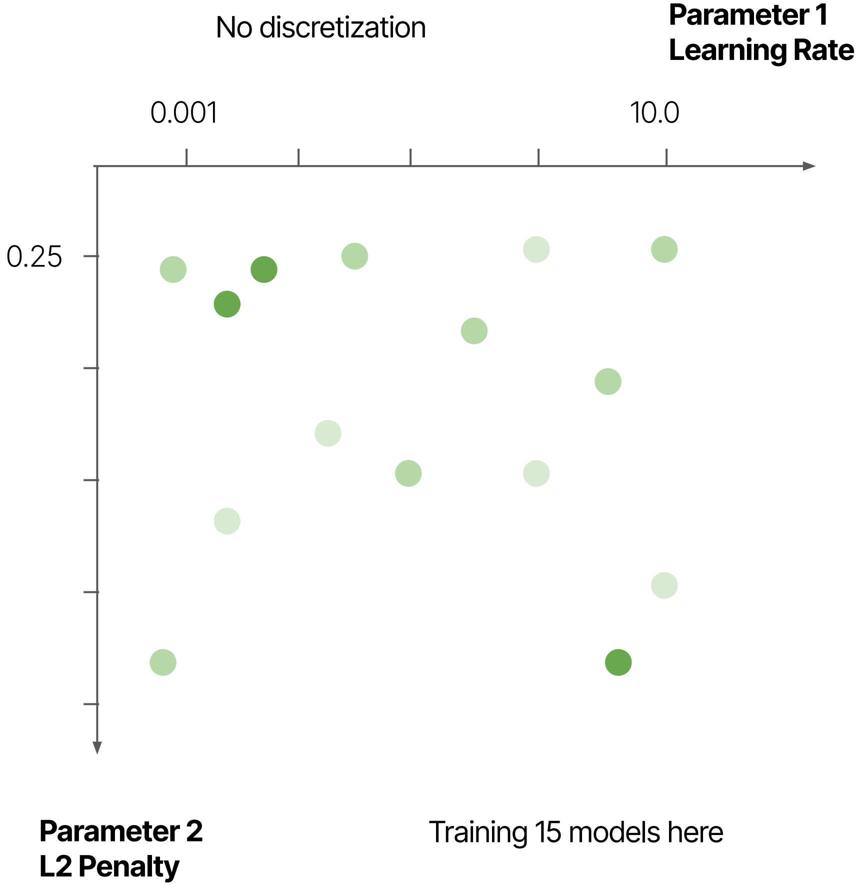
- Sample categories & continuous values
- Ensure N samples is sufficient to "cover" parameter space

Efficient but naive

- Explores completely randomly
- Could miss areas of the feature space with "good" parameters
- Choose N models to train

Works well for small/medium size feature spaces

- Struggles to cover large features spaces effectively



Bayesian Optimization

Use Bayesian statistics to select promising hyperparameters

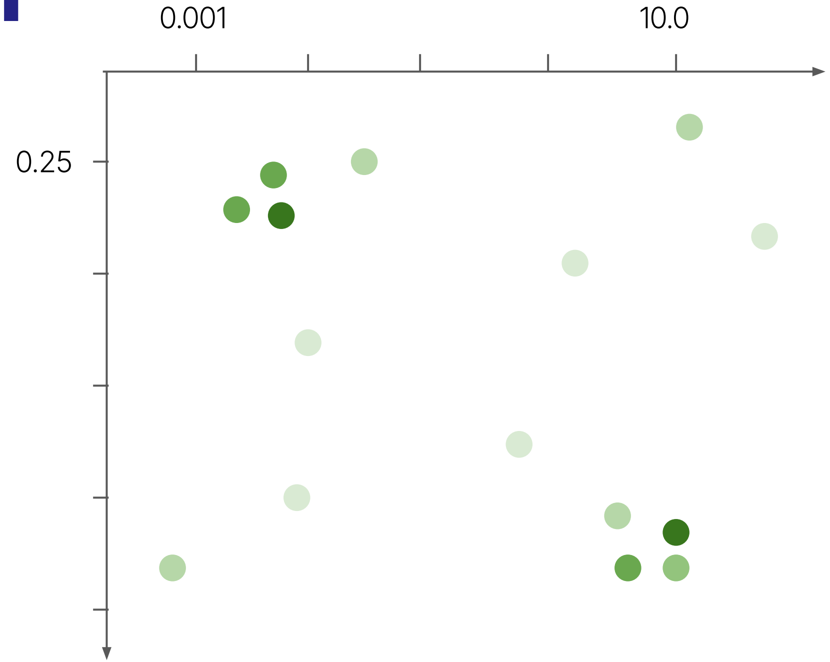
- Sample with a higher probability from areas of the parameter space that provides better models
- Explores space with "directed" randomness
- Converge on "good" parameters

Efficient but complex
- Converges on promising parameters faster
- More time consuming per model
- Gives more parameters "optimize" the optimizer

Great for very large parameter spaces

Will sample more often from areas with high model performance

Parameter 1
Learning Rate



Parameter 2
L2 Penalty

Training 15 models here

Nested Cross Validation

The Build Fellowship

Open Avenues FOUNDATION

QUIZ TIME !?

Why would we do nested cross validation?

a) To tune hyperparameters and evaluate model performance more reliably
b) To allow us to train multiple models in parallel
c) To reduce the computational cost of model training.
d) We're big fans of bird watching

Aims of CV

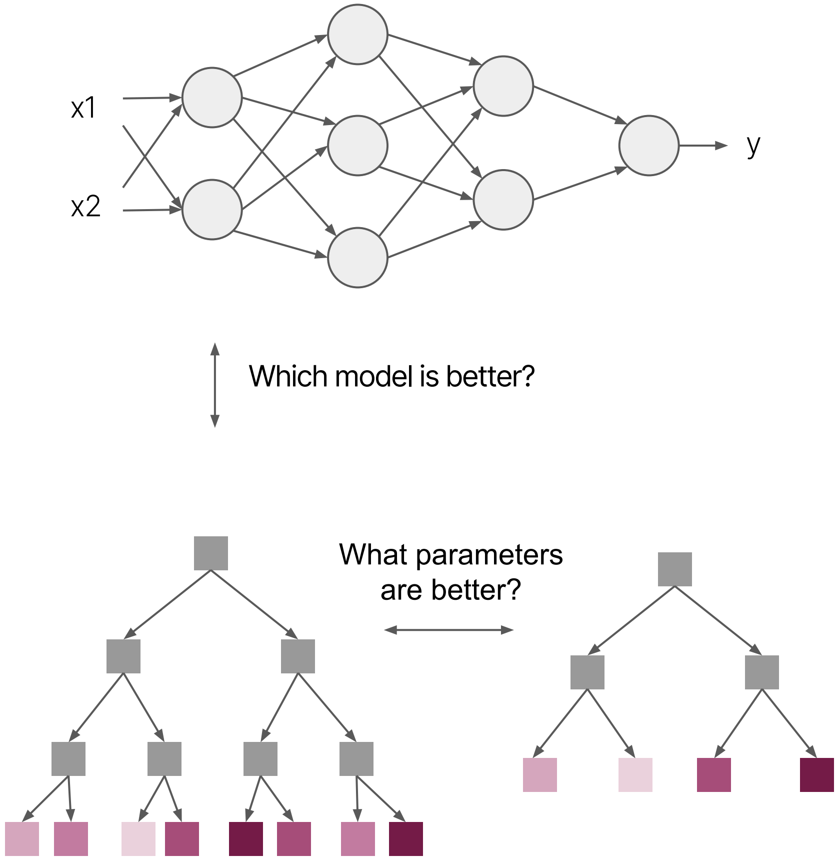
Splitting our data to allow training multiple models

Say we have 5 folds:
- Could we train 5 different models?
- Do we want to train 5 of the same model?

Different outcomes

Ideally we want:

- Train the same model on random splits of data
- Train multiple different parameters to compare
- Each of those models wants a consistent set of validation data to allow us to compare



K-Fold Recap

Process:

1. Shuffle your data
2. Split the data into 1/K chunks
3. Iteratively split 1/K to validate and (K-1)/K to train

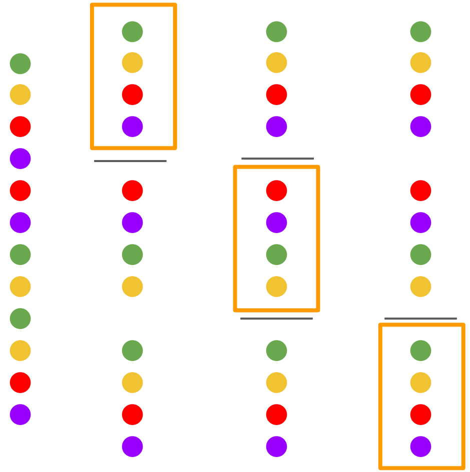
Can we try comparing models here?

- Yes, but: each validation split is different
- Comparing performance across folds may be biased

Shift from thinking about our model as a single object

Our “model” is a full configuration including a optimization method

Say **K = 3**
Split the Data into 3 chunks



Lets Nest

How do we get the best of both worlds?

- “Nest” by splitting each K-fold into a further K
- Average across outer folds to compare models
- Average across inner fold to optimize models

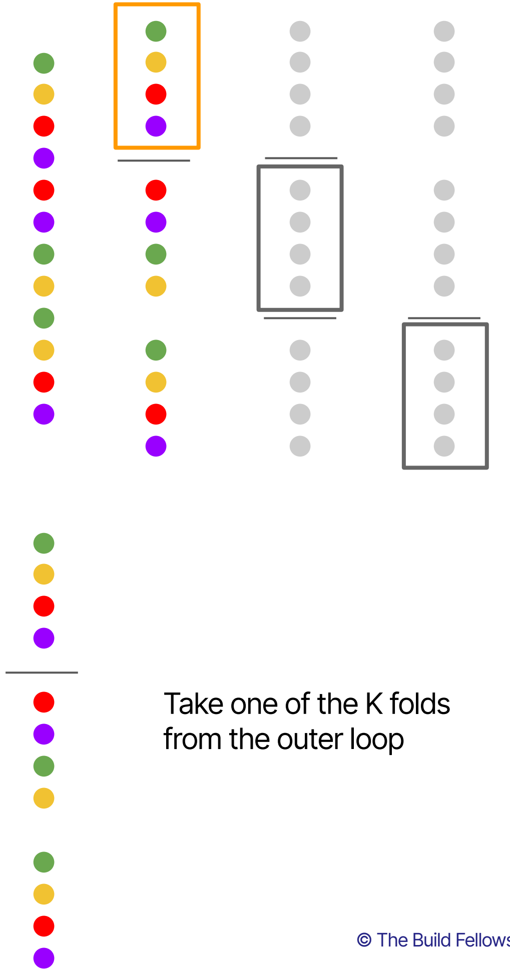
Advantages:

Robust selection +
Avoids data split bias +

Trade offs:

Time & Cost -
Complexity -

Say **Outer K = 3**



Take one of the K folds from the outer loop

The Build Fellowship   OpenAvenues FOUNDATION

Lets Nest

How do we get the best of both worlds?

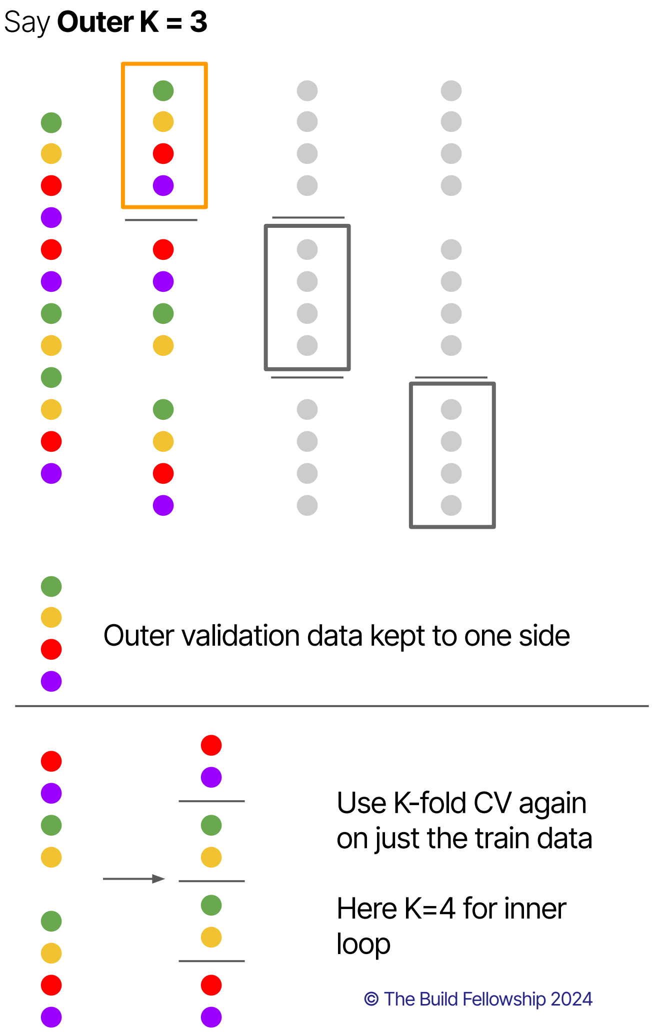
- “Nest” by splitting each K-fold into a further K
- Average across outer folds to compare models
- Average across inner fold to optimize models

Advantages:

Robust selection +
Avoids data split bias +

Trade offs:

Time & Cost -
Complexity -



The Build Fellowship OpenAvenues FOUNDATION

Lets Nest

How do we get the best of both worlds?

- "Nest" by splitting each K-fold into a further K
- Average across outer folds to compare models
- Average across inner fold to optimize models

Advantages:

Robust selection +
Avoids data split bias +

Trade offs:

Time & Cost -
Complexity -

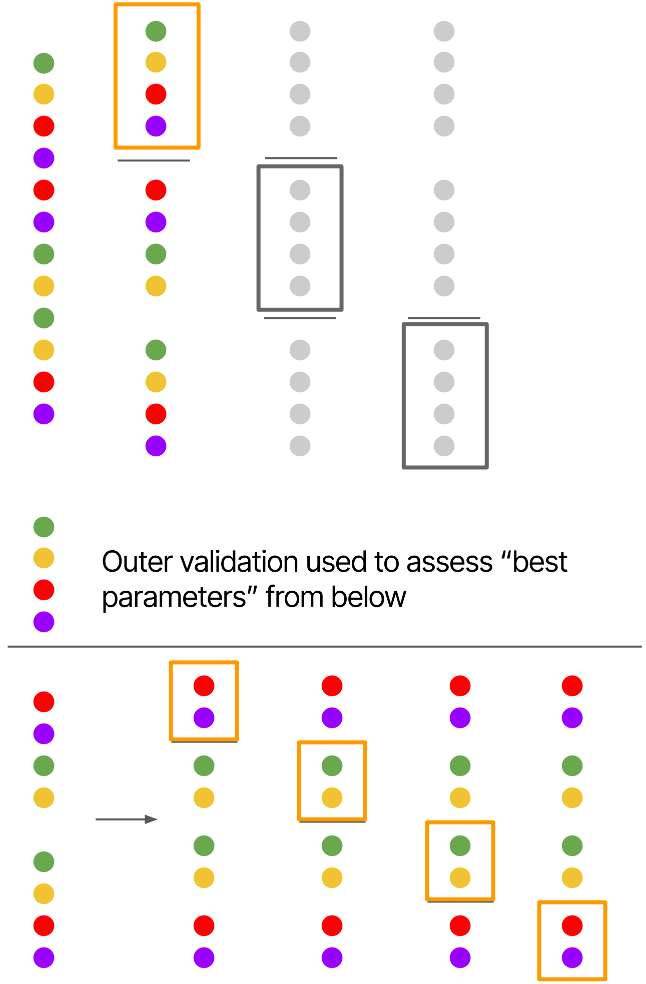
Outer train data split again into K-folds

Each Inner fold used for comparing parameter search

Inner train + inner validate

Repeat for each outer fold to provide fair between model assessment

Say **Outer K = 3**



Workshop 6
Model Tuning

The Build Fellowship

Open Avenues
FOUNDATION