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 5
Model Training Approaches

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
7. Present Results

Repeat as needed:
Data Science is an iterative process

The Build Fellowship

Open Avenues FOUNDATION

From Features to Models

- Last week we explored a number of featurization methods

- 3 possible feature spaces to leverage for modeling
  - Opportunity to expand further

- Assignment 4 - already built a simple model

- What comes next?

  1. Review modeling options
  2. Relate models to feature options
  3. Determine a robust framework for training models

Model Types

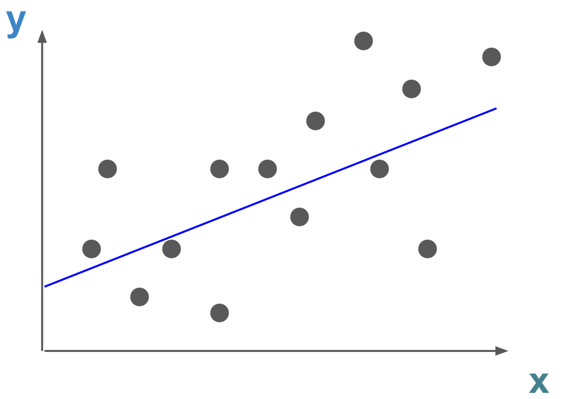
Linear Models

- Binary S vs R targets
- Good starting point

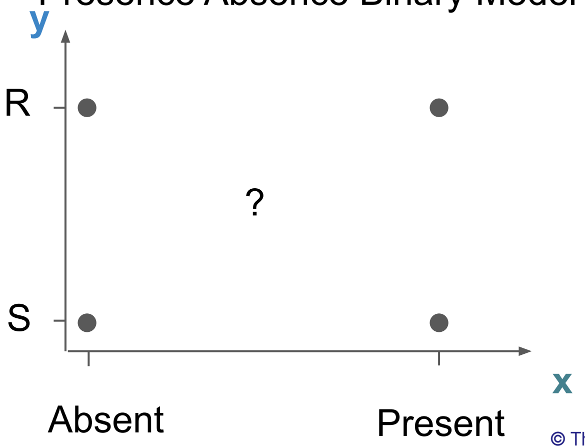
Examples:
- Regression
  - Linear
  - Logistic
- SVM

Assumptions:
- Linear relationships
- Is this expected from our data?
  - Presence/Absence
  - Kmers
  - Sequences

Continuous Linear Model



Presence Absence Binary Model



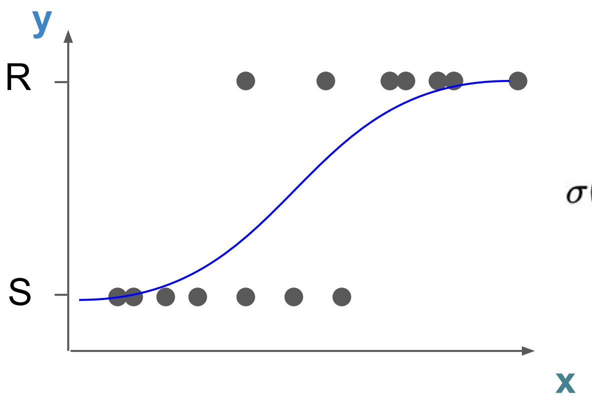
Logistic Regression

- Simple Linear regression won't work for us
- Target is binary & Features = binary or counts

Overview:
- Logistic regression = GLM
- Link function
- Mapping from continuous to bounded
- Sigmoid function

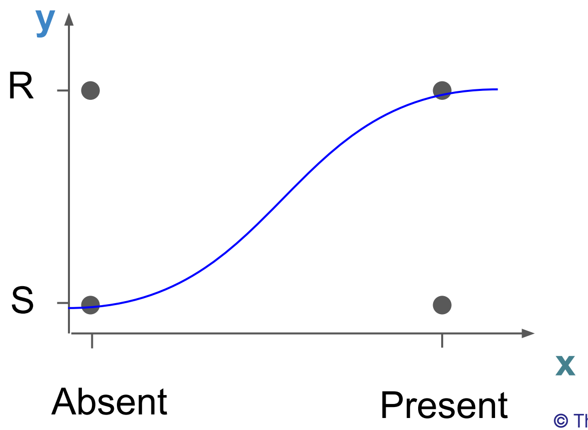
Python:
- Sklearn.linear_model
- [Documentation]

Logistic Model



$$\sigma(x) = \frac{1}{1+e^{-x}}$$

Presence Absence Binary Model



y = *"predicted probability"*

The Build Fellowship

OpenAvenues FOUNDATION

Tree Based Models

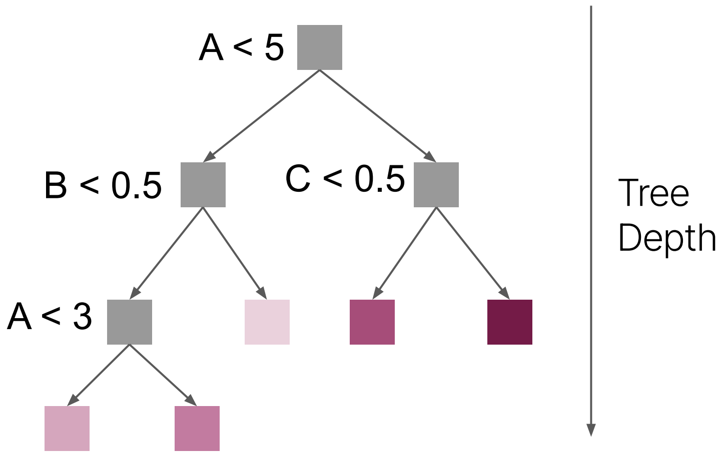
- Decision Trees
- Random Forest (Ensemble)

Decision Trees:
- Non-linear modeling approach
- Based off stacking binary decisions
- Natural fit for our presence/absence features
- Decision trees rarely used alone

Random Forest:
- Ensemble = combine multiple models
- Leverages Bagging (Bootstrap aggregating)

Python:
- Sklearn.ensemble
- Documentation

Decision Tree



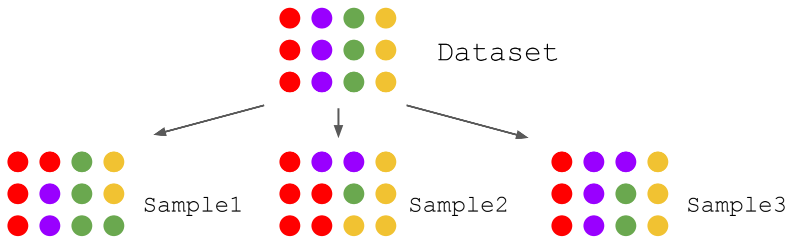
Process:
1. Decide on which feature to split (gain)
2. Determine a splitting threshold
3. Continue splitting until stop criteria (e.g. maximum depth)

Inference:
- New sample traverses tree
- Ends up at a single “leaf node”
- Take average value from leaf

The Build Fellowship

OpenAvenues FOUNDATION

Bagging = Repeatedly sample with replacement



Tree Based Models

- Decision Trees
- Random Forest (Ensemble)

Decision Trees:
- Non-linear modeling approach
- Based off stacking binary decisions
- Natural fit for our presence/absence features
- Decision trees rarely used alone

Random Forest:
- Ensemble = combine multiple models
- Leverages Bagging (Bootstrap aggregating)

Python:
- Sklearn.ensemble
- Documentation

Tree Based Models

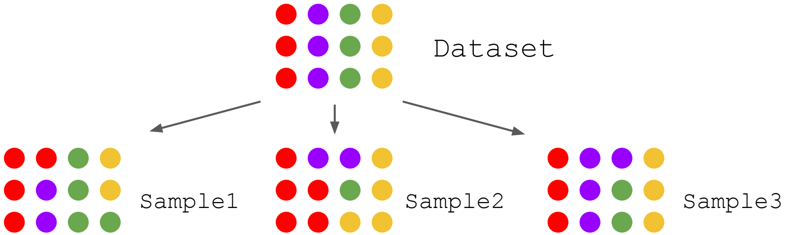
- Decision Trees
- Random Forest (Ensemble)

Decision Trees:
- Non-linear modeling approach
- Based off stacking binary decisions
- Natural fit for our presence/absence features
- Decision trees rarely used alone

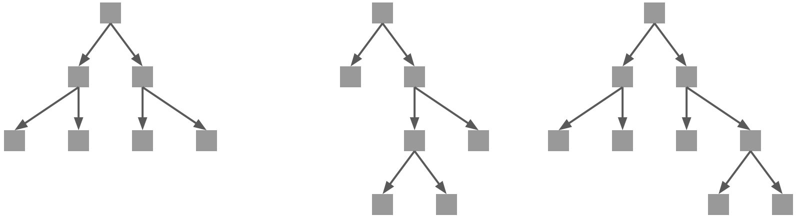
Random Forest:
- Ensemble = combine multiple models
- Leverages Bagging (Bootstrap aggregating)

Python:
- Sklearn.ensemble
- [Documentation]

Bagging = Repeatedly sample with replacement



Random Forest = Build New Tree per Bag



Tree Based Models

- Decision Trees
- Random Forest (Ensemble)

Decision Trees:
- Non-linear modeling approach
- Based off stacking binary decisions
- Natural fit for our presence/absence features
- Decision trees rarely used alone

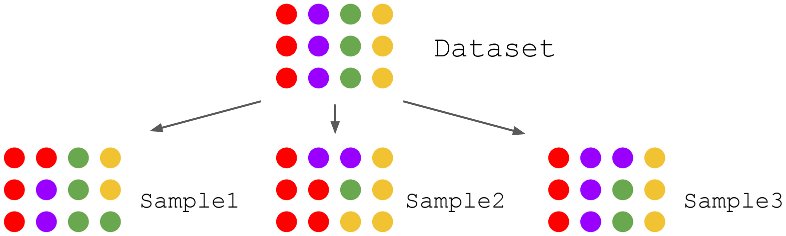
Random Forest:
- Ensemble = combine multiple models
- Leverages Bagging (Bootstrap aggregating)

Python:
- Sklearn.ensemble
- [Documentation]

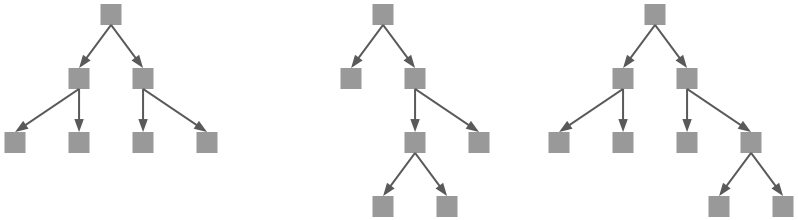
The Build Fellowship

OpenAvenues FOUNDATION

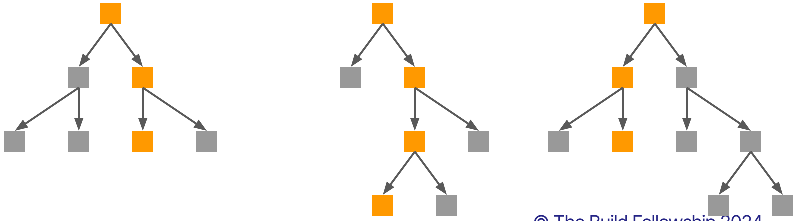
Bagging = Repeatedly sample with replacement



Random Forest = Build New Tree per Bag



Prediction = Each Sample passes through each Tree



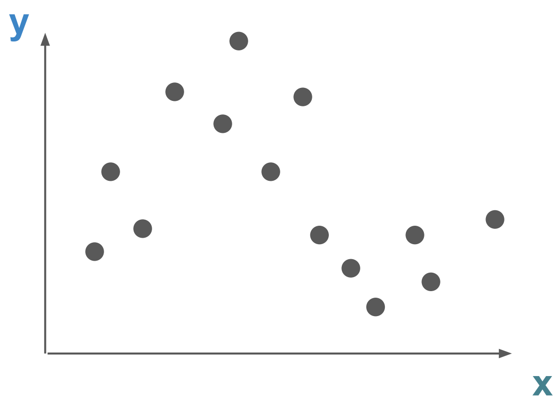
Output = Average across trees

Gradient Boosting

- Ensemble method
- Can use any underlying model
  - Common to use decision trees

Overview:
- Iteratively fit models on residuals of the previous
- Learn from the mistakes of the last model
- Leveraging lots of simple models

Options:
- Sklearn Gradient Boosting
- LightGBM
- XGBoost

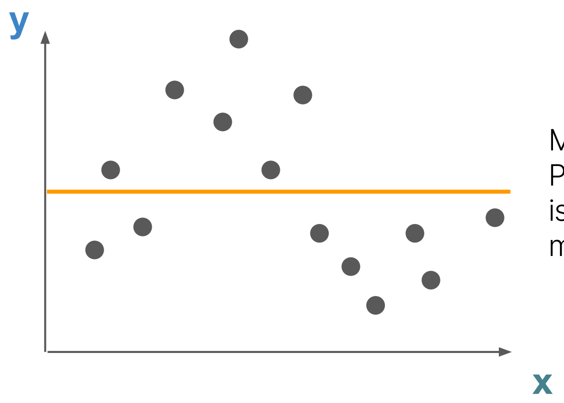


Gradient Boosting

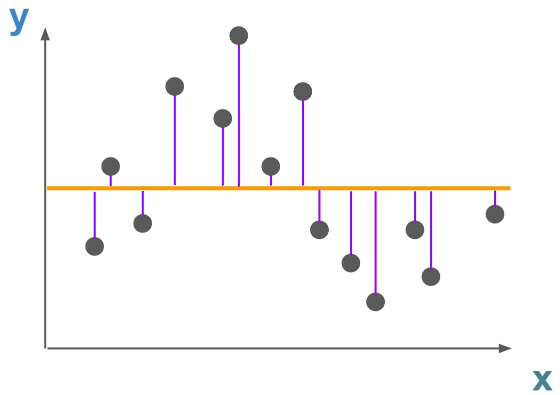
- Ensemble method
- Can use any underlying model
    - Common to use decision trees

Overview:
- Iteratively fit models on residuals of the previous
- Learn from the mistakes of the last model
- Leveraging lots of simple models

Options:
- [Sklearn Gradient Boosting]
- [LightGBM]
- [XGBoost]



Mean Prediction is base model



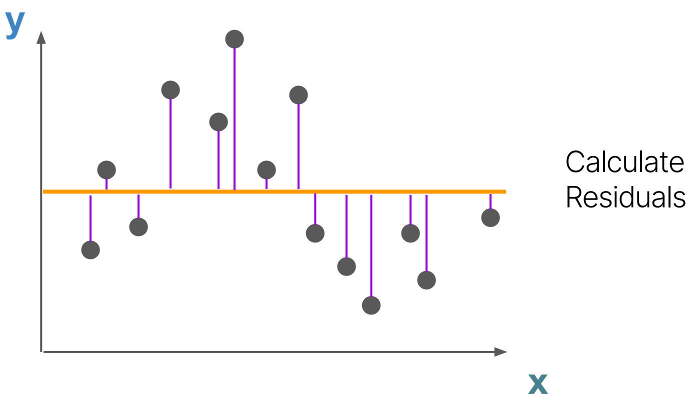
Calculate Residuals

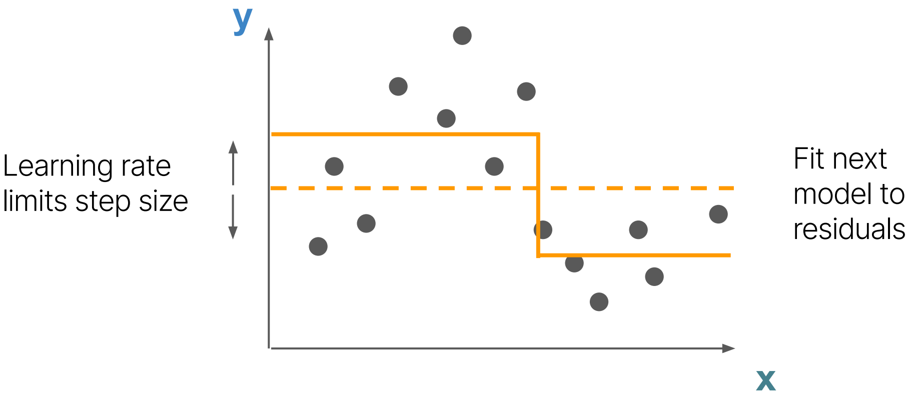
Gradient Boosting

- Ensemble method
- Can use any underlying model
  - Common to use decision trees

Overview:
- Iteratively fit models on residuals of the previous
- Learn from the mistakes of the last model
- Leveraging lots of simple models

Options:
- Sklearn Gradient Boosting
- LightGBM
- XGBoost





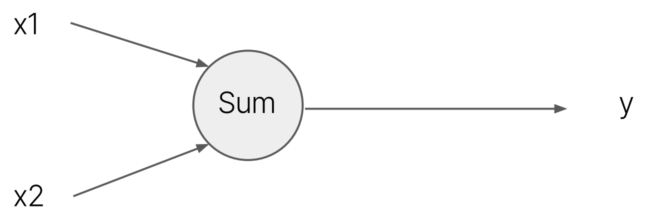
Neural Networks

- Many different architectures
  - Convolutional Neural Network (CNN)
  - Recurrent Neural Network (RNN)
- Tabular & Sequence data (e.g. audio/image)

Overview:
- Very powerful and flexible
- Expensive to train & can overfit easily
- Network of simple nonlinear functions

Options:
- Tensorflow
- Keras (higher level interface)
- Pytorch

Single Neuron



$$y = x1 + x2$$

The Build Fellowship OpenAvenues FOUNDATION

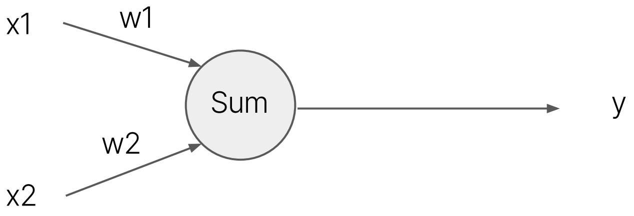
Neural Networks

- Many different architectures
  - Convolutional Neural Network (CNN)
  - Recurrent Neural Network (RNN)
- Tabular & Sequence data (e.g. audio/image)

Overview:
- Very powerful and flexible
- Expensive to train & can overfit easily
- Network of simple nonlinear functions

Options:
- [Tensorflow]
- [Keras] (higher level interface)
- [Pytorch]

Single Neuron



$$y = w1x1 + w2x2$$

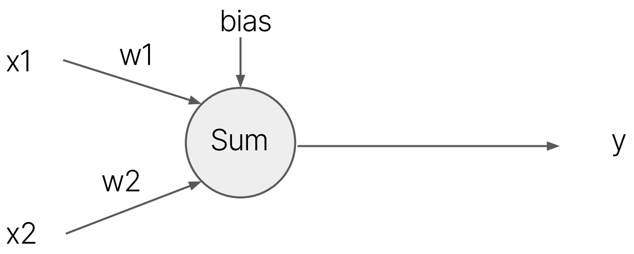
Neural Networks

- Many different architectures
  - Convolutional Neural Network (CNN)
  - Recurrent Neural Network (RNN)
- Tabular & Sequence data (e.g. audio/image)

Overview:
- Very powerful and flexible
- Expensive to train & can overfit easily
- Network of simple nonlinear functions

Options:
- Tensorflow
- Keras (higher level interface)
- Pytorch

Single Neuron



$$y = w1x1 + w2x2 + b$$

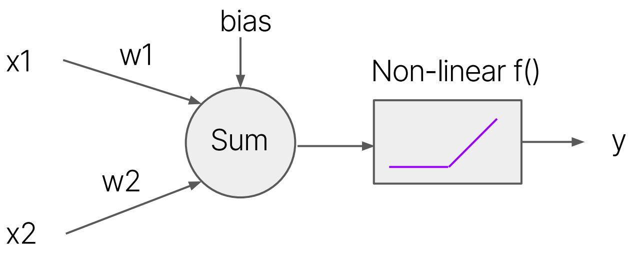
Neural Networks

- Many different architectures
  - Convolutional Neural Network (CNN)
  - Recurrent Neural Network (RNN)
- Tabular & Sequence data (e.g. audio/image)

Overview:
- Very powerful and flexible
- Expensive to train & can overfit easily
- Network of simple nonlinear functions

Options:
- Tensorflow
- Keras (higher level interface)
- Pytorch

Single Neuron



$$y = f(w1x1 + w2x2 + b)$$

Neural Networks

- Many different architectures
  - Convolutional Neural Network (CNN)
  - Recurrent Neural Network (RNN)
- Tabular & Sequence data (e.g. audio/image)

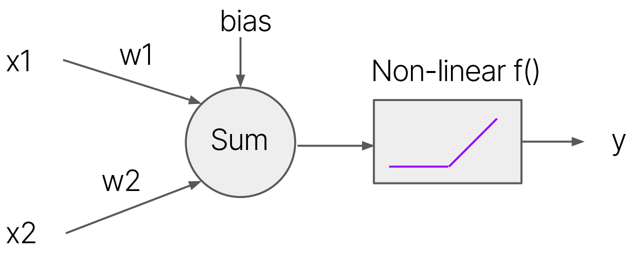
Overview:

- Very powerful and flexible
- Expensive to train & can overfit easily
- Network of simple nonlinear functions

Options:

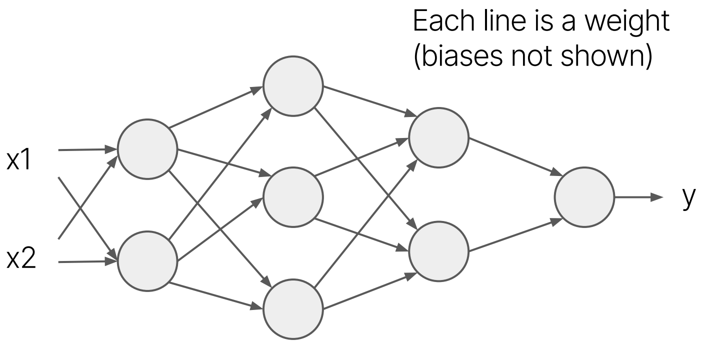
- [Tensorflow]
- [Keras] (higher level interface)
- [Pytorch]

Single Neuron



$$y = f(\ w1x1 + w2x2 + b\ )$$

Neural Network



$$y = f(\ f() + f() + \dots\ )$$

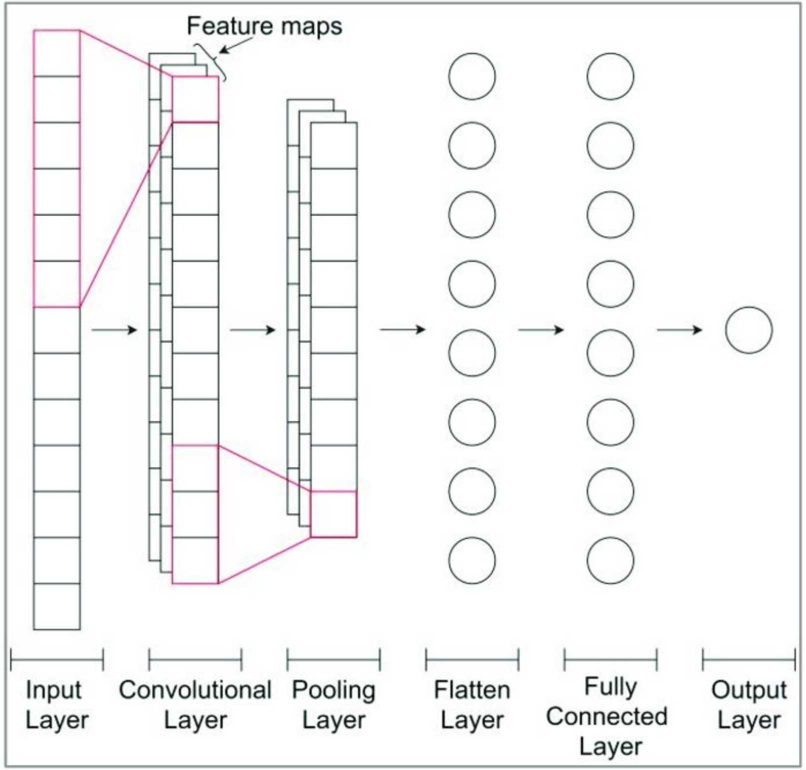
Neural Networks

- Many different architectures
  - Convolutional Neural Network (CNN)
  - Recurrent Neural Network (RNN)
- Tabular & Sequence data (e.g. audio/image)

Overview:
- Very powerful and flexible
- Expensive to train & can overfit easily
- Network of simple nonlinear functions

Options:
- Tensorflow
- Keras (higher level interface)
- Pytorch

Convolutional (1D) Neural Network



[1]

QUIZ TIME !?

Which model would you try first and why?

a) Convolutional neural network
b) Decision tree
c) Logistic regression
d) Random forest
e) Gradient boosted trees

Model Training

Data Splitting - Cross validation

Cross validation is a fundamental concept for training ML models:

- Training a single model is risky
- Wish to train many models to study uncertainty

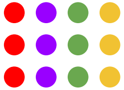
Standard splitting method: **K-fold**

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

Each samples appears in validate exactly once

Note: separate from having a completely held out test set

Dataset



Data Splitting - Cross validation

Cross validation is a fundamental concept for training ML models:

- Training a single model is risky
- Wish to train many models to study uncertainty

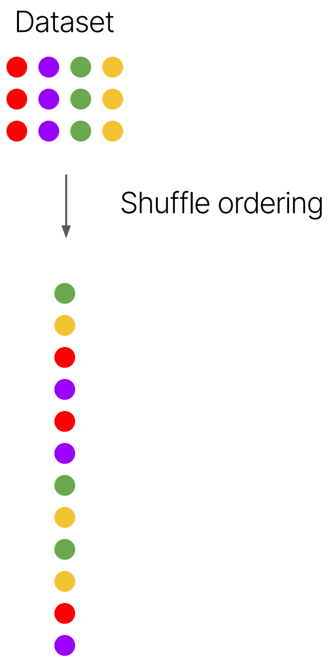
Standard splitting method: **K-fold**

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

Each samples appears in validate exactly once

Note: separate from having a completely held out test set



Data Splitting - Cross validation

Cross validation is a fundamental concept for training ML models:

- Training a single model is risky
- Wish to train many models to study uncertainty

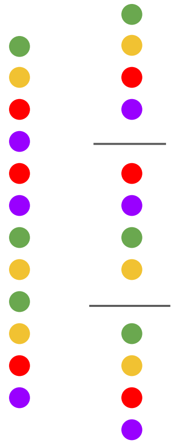
Standard splitting method: **K-fold**

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

Each samples appears in validate exactly once

Note: separate from having a completely held out test set

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



Data Splitting - Cross validation

Cross validation is a fundamental concept for training ML models:

- Training a single model is risky
- Wish to train many models to study uncertainty

Standard splitting method: **K-fold**

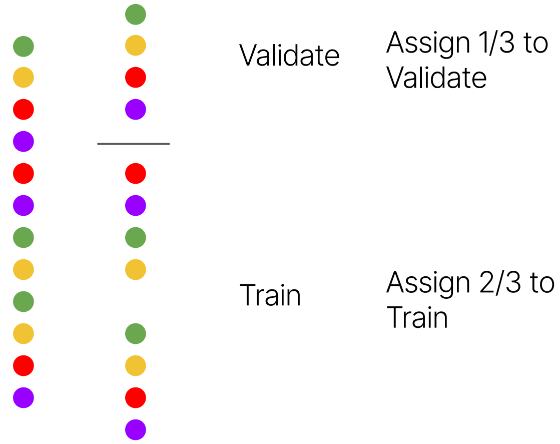
Process:

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

Each samples appears in validate exactly once

Note: separate from having a completely held out test set

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



Data Splitting - Cross validation

Cross validation is a fundamental concept for training ML models:

- Training a single model is risky
- Wish to train many models to study uncertainty

Standard splitting method: **K-fold**

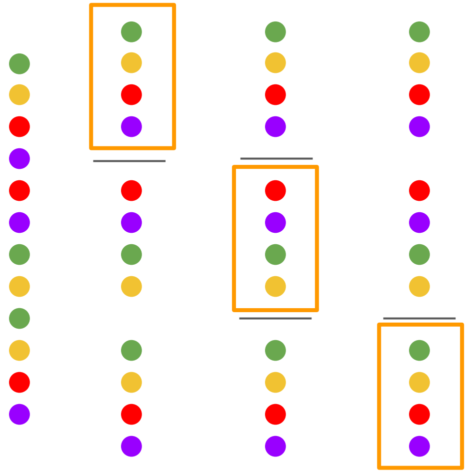
Process:

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

Each samples appears in validate exactly once

Note: separate from having a completely held out test set

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



Repeat 3 times to make 3 datasets
(Orange box = Validate)

The Build Fellowship

OpenAvenues FOUNDATION

Final Project

Final Project - Overview

Challenge: **Build the best model you can to Predict Cefepime S vs R in Escherichia coli**

Requirements

- Use any feature sets
- Try at least two models
- Use CV
- Tune model to optimize performance
- Evaluate and discuss model performance

Future weeks will cover tuning and evaluation in more detail

Final Project - Evaluation

Challenge: **Build the best model you can to Predict Cefepime S vs R in Escherichia coli**

Evaluation

- Not being scored based on model performance
- Should be able to get good (>90% accuracy) performance using simple models + features

1. Clear and well documented code
2. Well specified and trained model
3. Discussion and presentation of results

Submission:

- Results Notebook (can include .py file as desired)
- Summary slides (5-10 slide max, 5 minute run through)

Final Project - Evaluation

Challenge: **Build the best model you can to Predict Cefepime S vs R in Escherichia coli**

- https://www.kaggle.com/t/e4838697a45d4d1dac9dfca2fe16d687
- Join using the link above!
- This should be very lightweight and happy to answer any questions about using the page

Workshop 5
Model Building

The Build Fellowship

OpenAvenues FOUNDATION

References

[1]:https://www.researchgate.net/figure/One-dimensional-convolutional-neural-network-1D-CNN-architecture-for-the-timeseries_fig2_348502722