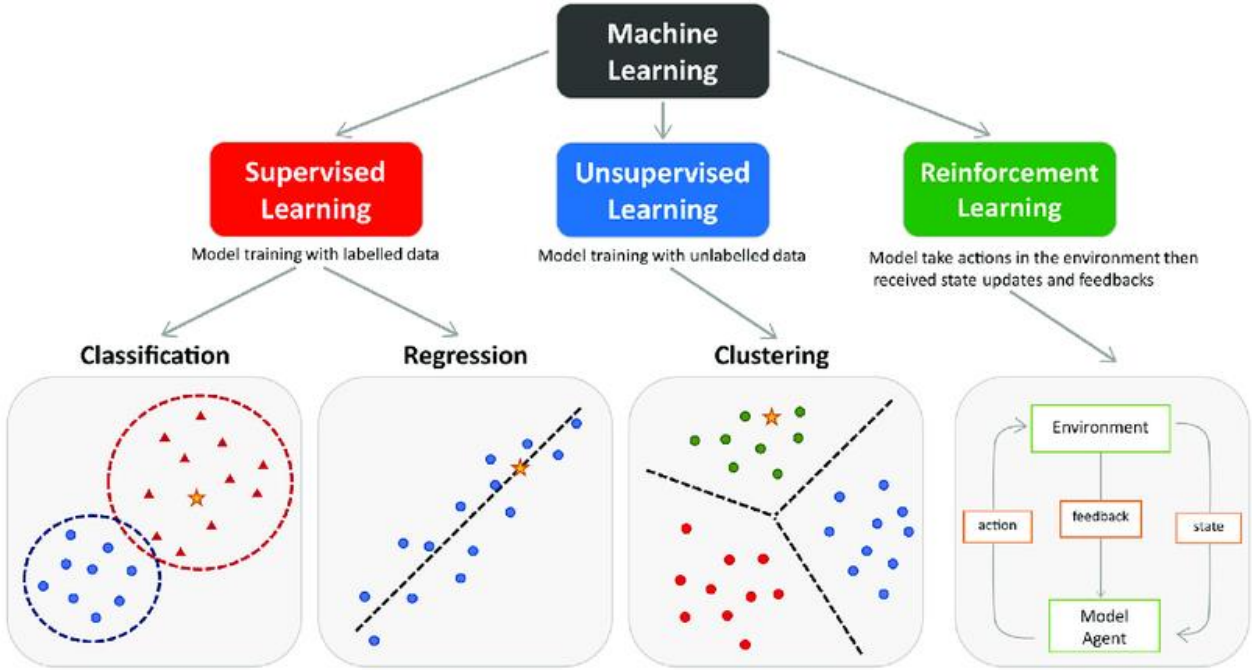
ECG Event Classification using Machine Learning

By Shane McNicholas

Introduction

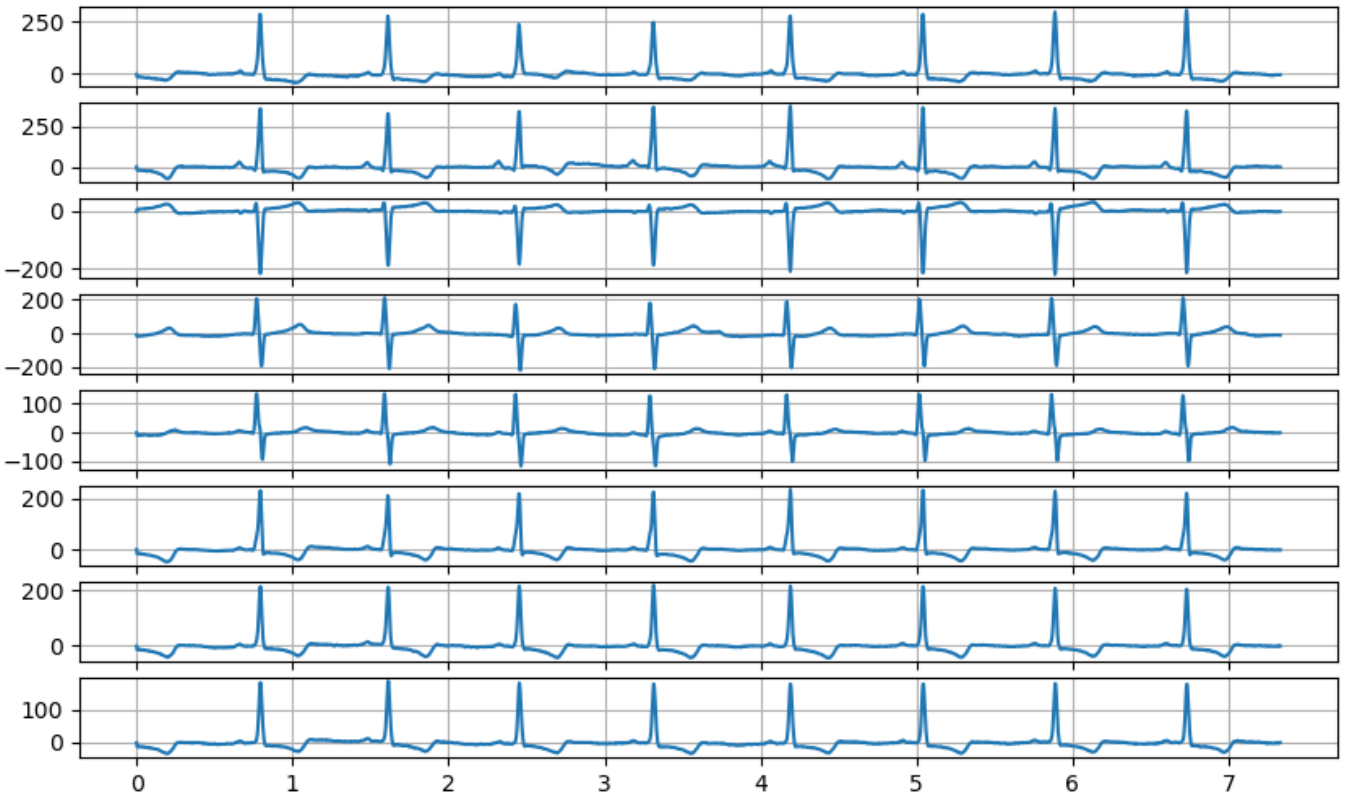
This project aims to:

- Develop 2 machine learning models for multi-label classification of ECG data.
- Briefly explore and evaluate different neural network and non-neural network architectures.
- Identify and exploit key features of the data.
- Train -> Evaluate -> Tune
- Understand the different metrics of scoring and how they individually relate to the model's performance



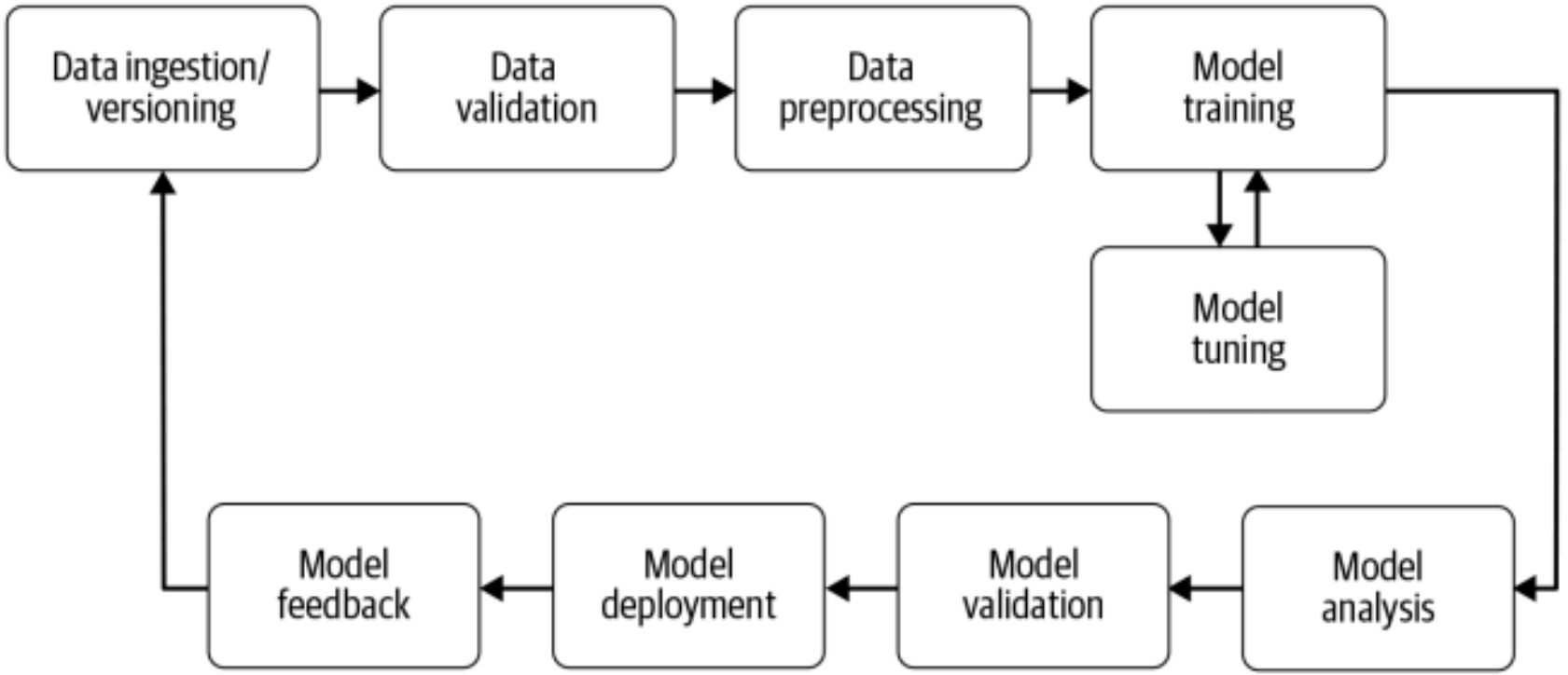
Electrocardiogram (ECG)

- Electrocardiograms are recordings of the heart's electrical activity over time.
- Voltage signals taken with electrodes placed strategically on a patient's body.
- 8-channel data sampled at 300Hz.
- Each ECG recording is split into 7.33s segments, resulting in 2200 samples per recording.
- 6 labels; 1dAVb, RBBB, LBBB, SB, AF, ST.
    - Each sample can have multiple labels



Methods

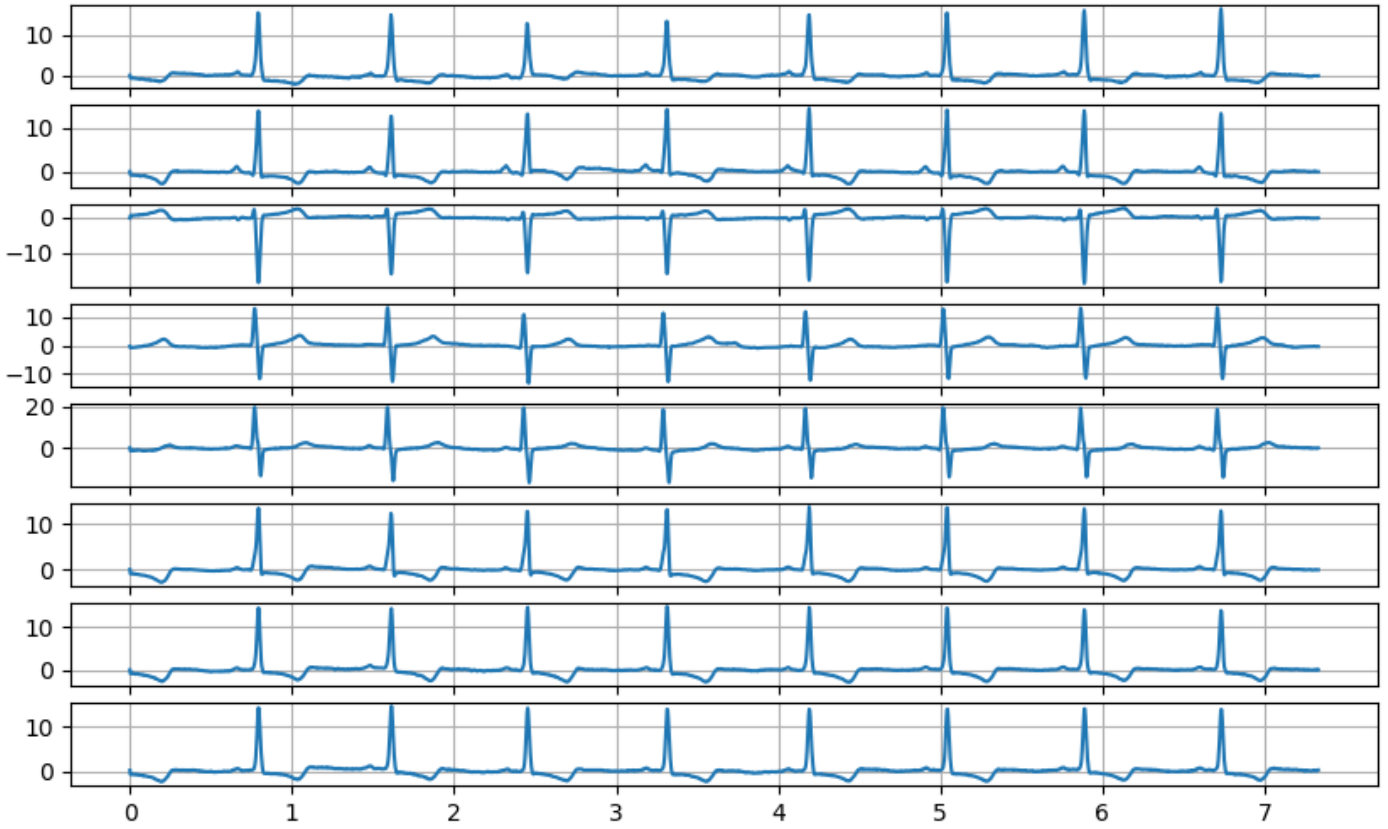
- Two models, one neural network and one non-neural network
  - Random Forest classifier (non-neural network)
  - Multilayer Perceptron (neural network)
- Data augmentation
  - Limit training samples?
- Preprocessing
  - Normalization?
  - Filtering?
  - Reshaping?
  - Feature extraction?
- Parameter tuning
- Postprocessing
  - Thresholds
- Software design



Preprocessing - Normalization

Robust Scaling

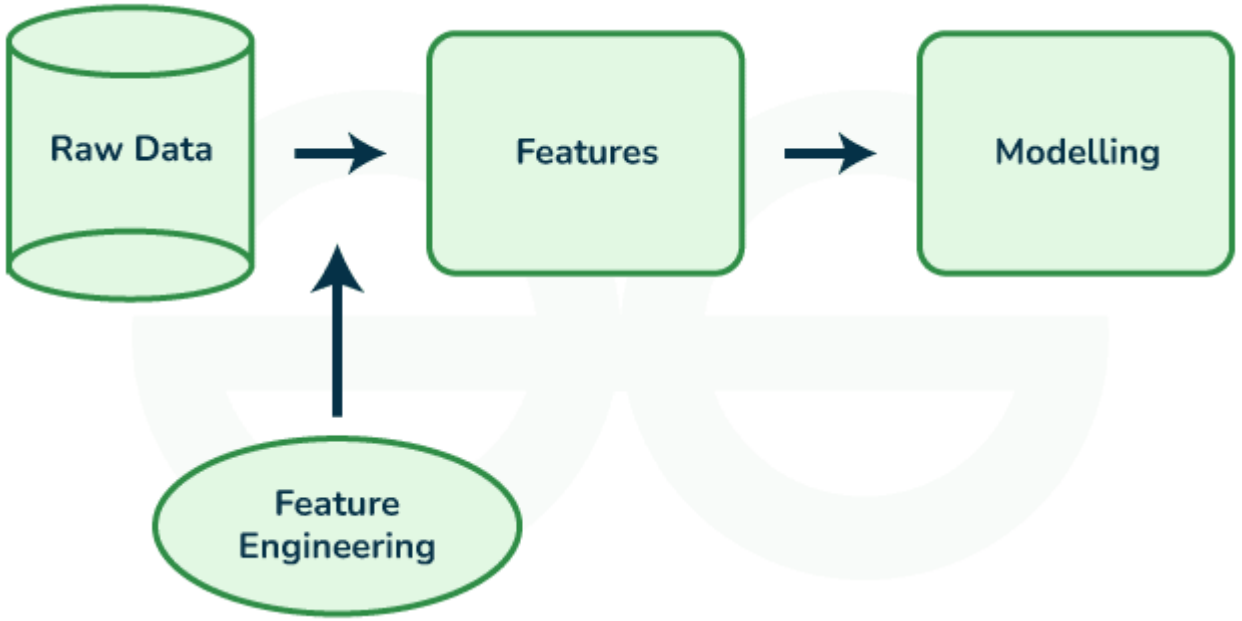
- “Center to the median and component wise scale according to the interquartile range.”
- Insensitive to outliers
- Preserves signal shape
- Non-parametric
- Removes the need for filtering



Preprocessing

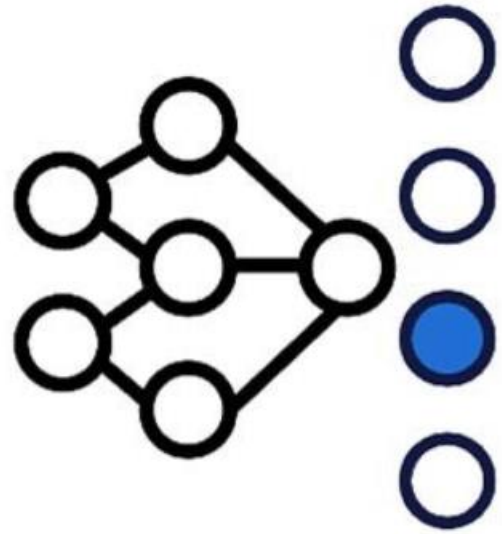
Feature Engineering

- Started with raw and normalized channel data.
- Stacked each channel on top of another.
  - CH11,CH12,CH13,...,CH81,CH82,...
- Hand picked feature extraction for RF.
  - Mean, std, max amplitude, min amplitude, peak-to-peak, total power, mean power
- Used normalized, stacked data for MLP.

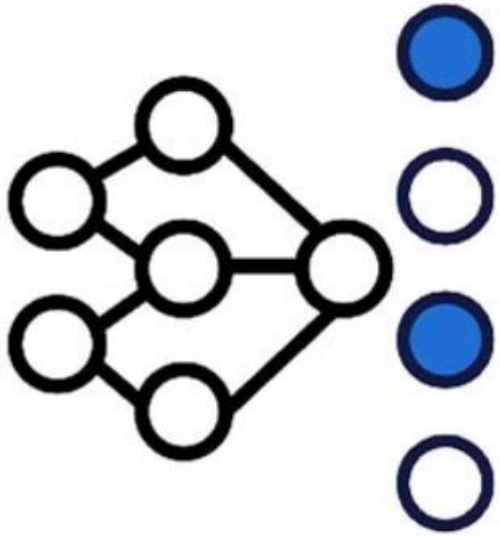


Multi-Output Classification

- Each sample can have multiple labels.
- Some models *cannot* predict multiple labels per sample
  - MLPs can, RFs cannot
- An alternative approach must be used for multi-output classification
  - A binary classifier for each of the given estimator is trained for each label in the data.
  - Each individual model makes a binary classification for a corresponding label in the data.
  - sklearn.multioutput.MultiOutputClassifier



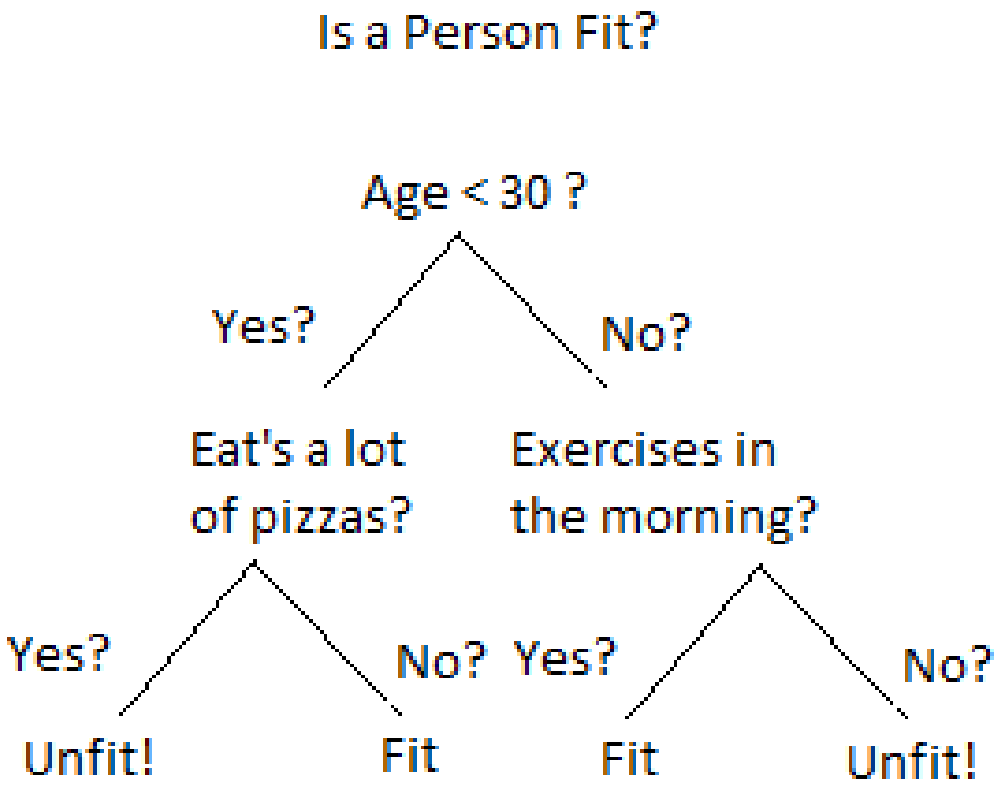
Multi-Class



Multi-Label

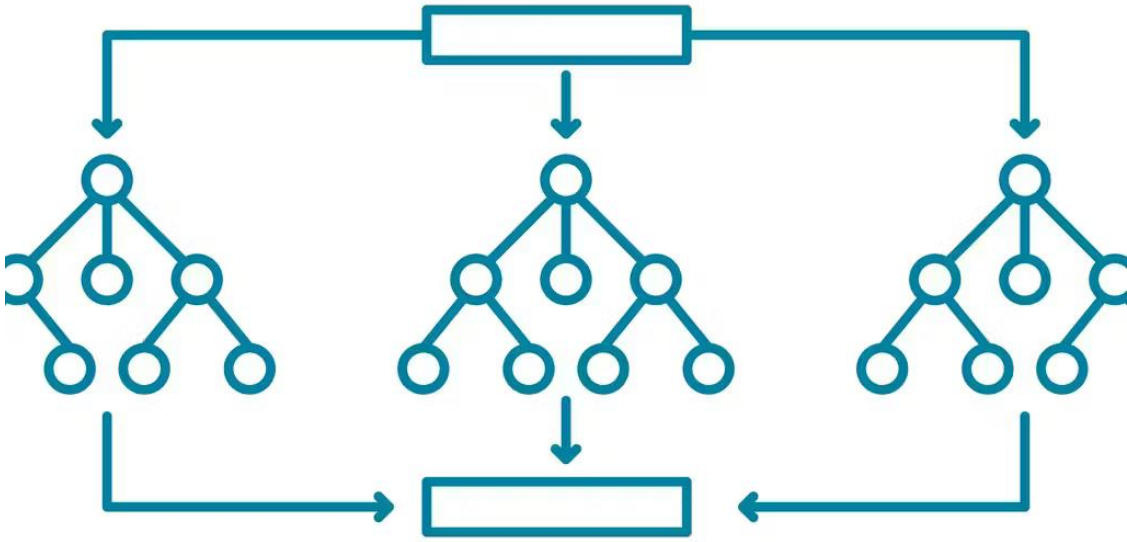
Random Forest Classifier - Overview

- A decision-tree based classifier
  - A series of if statements that a sample passes through that leads to a classification.
- Generates multiple decision trees on sub-samples of the dataset.
  - Averages the outputs of every tree to improve accuracy and limit overfitting.
- Relies heavily on inputs features.
  - Nodes of the trees are based on these features.
- Not inherently multi-label.
  - An entire RF is fitted for each label in the data.
- Easily parallelizable.
  - Each decision tree in the forest can be ran in parallel.



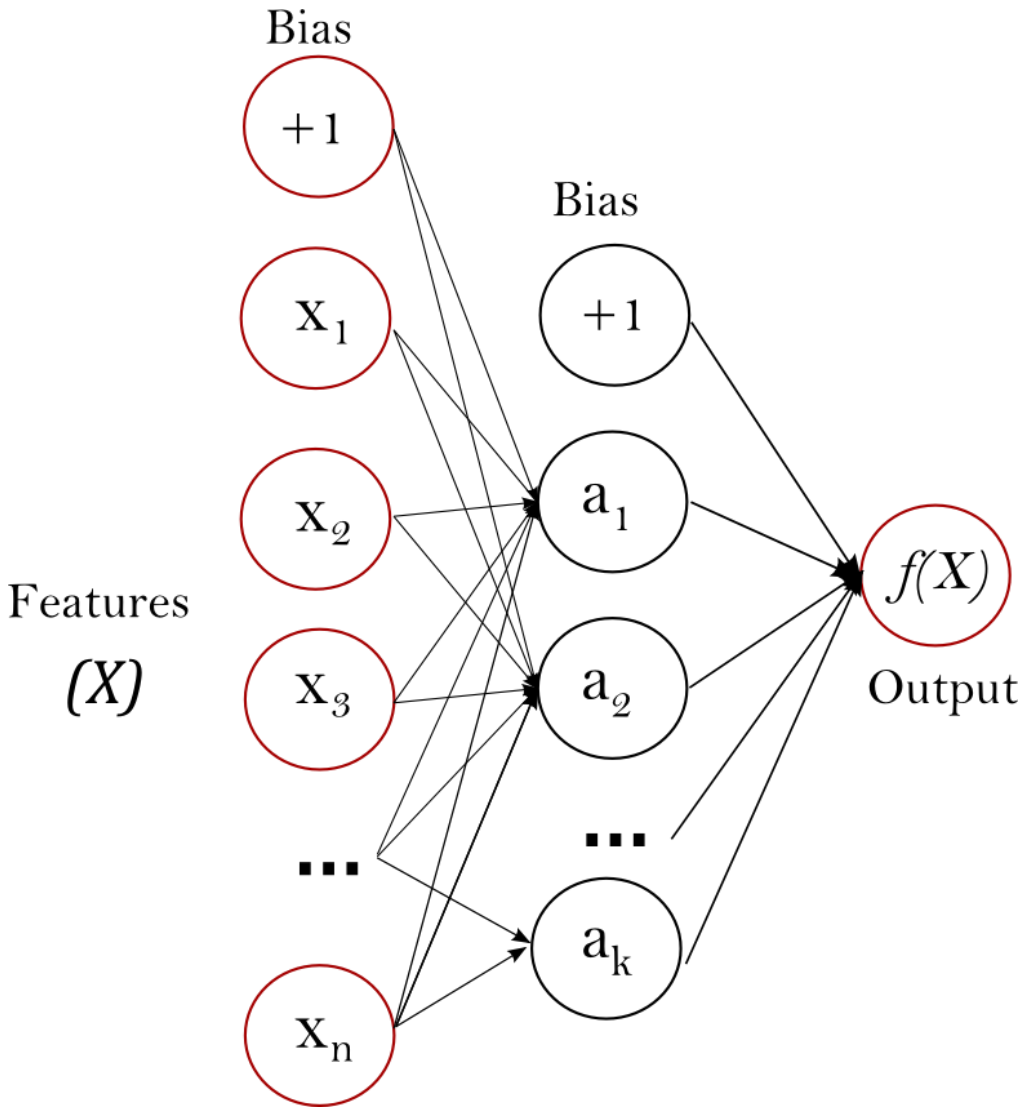
Random Forest Classifier - Hyperparameters

- Criterion: The function to measure the quality of a split (dividing a node into child nodes)
  - Gini: Measures how often a randomly chosen element from a set would be incorrectly labeled if it was randomly labeled according to the distribution of labels in the subset.
- N estimators: The number of trees in the forest.
  - The default of 100 is reasonable for most datasets.
- Class weights: Apply a weight to each class for output.
  - Could correlate weights to distributions of labels.



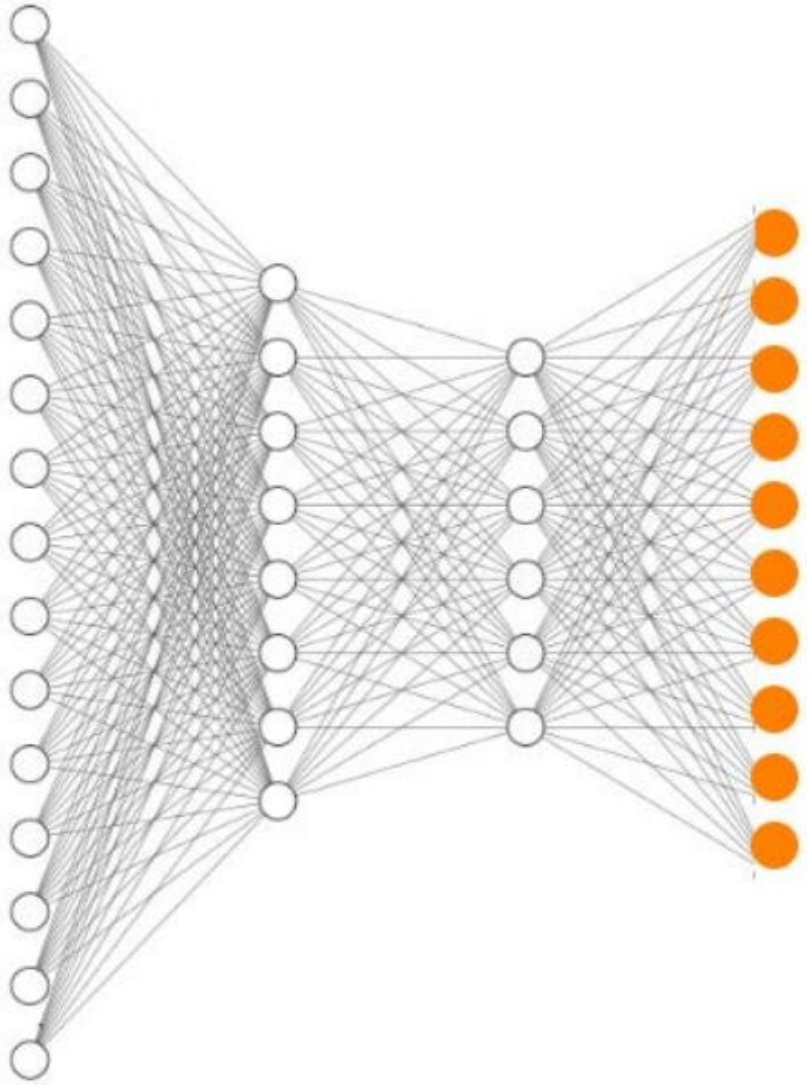
Multi-layer Perceptron – Overview

- A supervised, neural-network model that can learn non-linear function approximators for either classification or regression.
- Input layer values are passed to one or more hidden layers where the previous layer's values are transformed using a weighted summation.
- Trains using gradient descent, where the gradient descents are calculated using backpropagation.
- Classifies by minimizing a cross-entropy loss function.
  - Natively supports multi-output classification.
- Depends heavily on a set of tunable hyperparameters.
  - Hidden layers, epochs, alpha, solver, etc…



Multilayer Perceptron - Hyperparameters

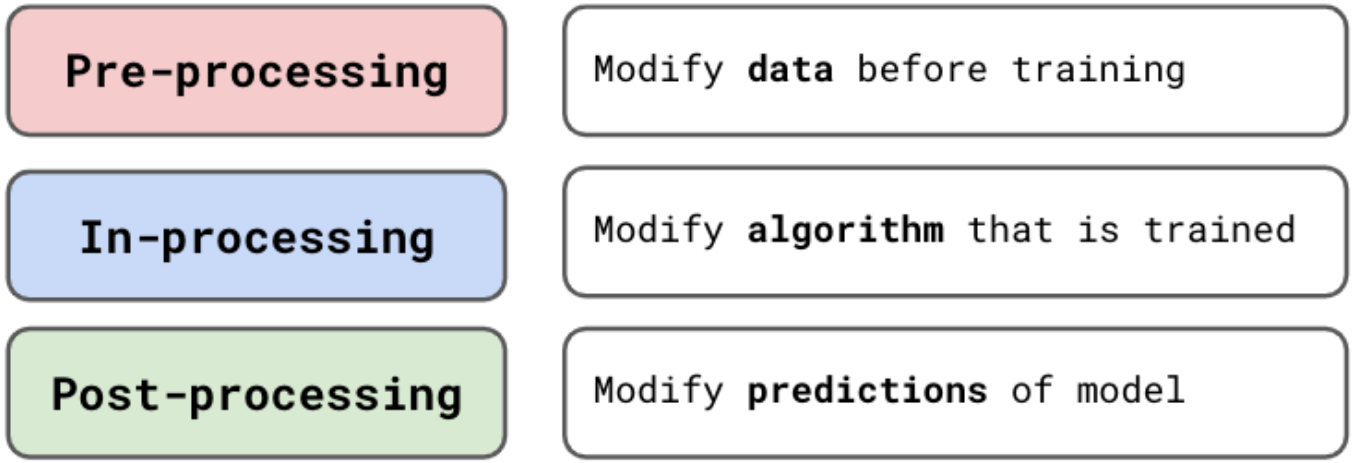
- Activation function: Enables complex modeling.
  - Rectified Linear Unit (ReLU): returns f(x) = max(0, x)
- Solver: The algorithm used for optimizing hidden-layer weights.
  - Adaptive Moment Estimation (ADAM): “a stochastic gradient-descent based optimizer”
- Hidden Layer Architecture: The number of neurons per each hidden layer.
  - (100, 50, 25, 10) – Results did not change much for different architectures.
  - Input layer of 17600
  - Output layer of 6



Postprocessing

Thresholding

- A simple postprocessing technique to classify based on label probabilities
- Compute label probabilities for a given sample, classify as 1 if greater than or equal to threshold, 0 if less than.
- Allows greater control over a decoder.
  - SKLearn's default threshold is 0.5
- Different thresholds for each model
  - MLP: 0.55
  - RF: 0.15



Results

	Random Forest Classifier							
	Training				Development			
	accuracy	precision	recall	F1	accuracy	precision	recall	F1
1dAVB	0.9111	0.5025	0.1211	0.1951	0.8986	0.1728	0.0397	0.0646
RBBB	0.8096	0.4797	0.8768	0.6201	0.7924	0.4553	0.8417	0.5909
LBBB	0.9468	0.6480	0.8837	0.7477	0.9347	0.5958	0.8449	0.6988
SB	0.8632	0.3785	0.7670	0.5069	0.8493	0.3390	0.6963	0.4560
AF	0.7938	0.2749	0.5954	0.3762	0.7726	0.2300	0.5019	0.3155
ST	0.9110	0.4997	0.1207	0.1945	0.8976	0.1502	0.0346	0.0562
Macro	**0.8726**	**0.4639**	**0.5608**	**0.4401**	**0.8576**	**0.3239**	**0.4932**	**0.3637**

	Multi-Layer Perceptron							
	Training				Development			
	accuracy	precision	recall	F1	accuracy	precision	recall	F1
1dAVB	0.9934	0.9632	0.9629	0.9630	0.8594	0.1716	0.1554	0.1631
RBBB	0.9858	0.9776	0.9417	0.9593	0.9206	0.7912	0.7532	0.7718
LBBB	0.9906	0.9472	0.9479	0.9475	0.9563	0.7501	0.7673	0.7586
SB	0.9883	0.9773	0.8936	0.9336	0.9533	0.7715	0.6885	0.7276
AF	0.9820	0.9453	0.8780	0.9104	0.8651	0.3315	0.2874	0.3079
ST	0.9934	0.9632	0.9629	0.9630	0.8594	0.1716	0.1554	0.1631
Macro	**0.9889**	**0.9623**	**0.9312**	**0.9461**	**0.9023**	**0.4979**	**0.4679**	**0.4820**

Challenges and Final Remarks

- Understanding the nature of the data
  - What do the labels mean?
  - What are the discerning factors of each label?
  - What are the best features to extract from the data?
  - Filtering and normalizing the data?
- Time
  - Developing infrastructure
  - I/O reads
  - Models trained faster than planned

If I were to do it again:

- Go further into deep learning than MLP
- Try an image-based approach
- Experiment more with RF parameters
- Experiment with models before developing infrastructure (notebooks)