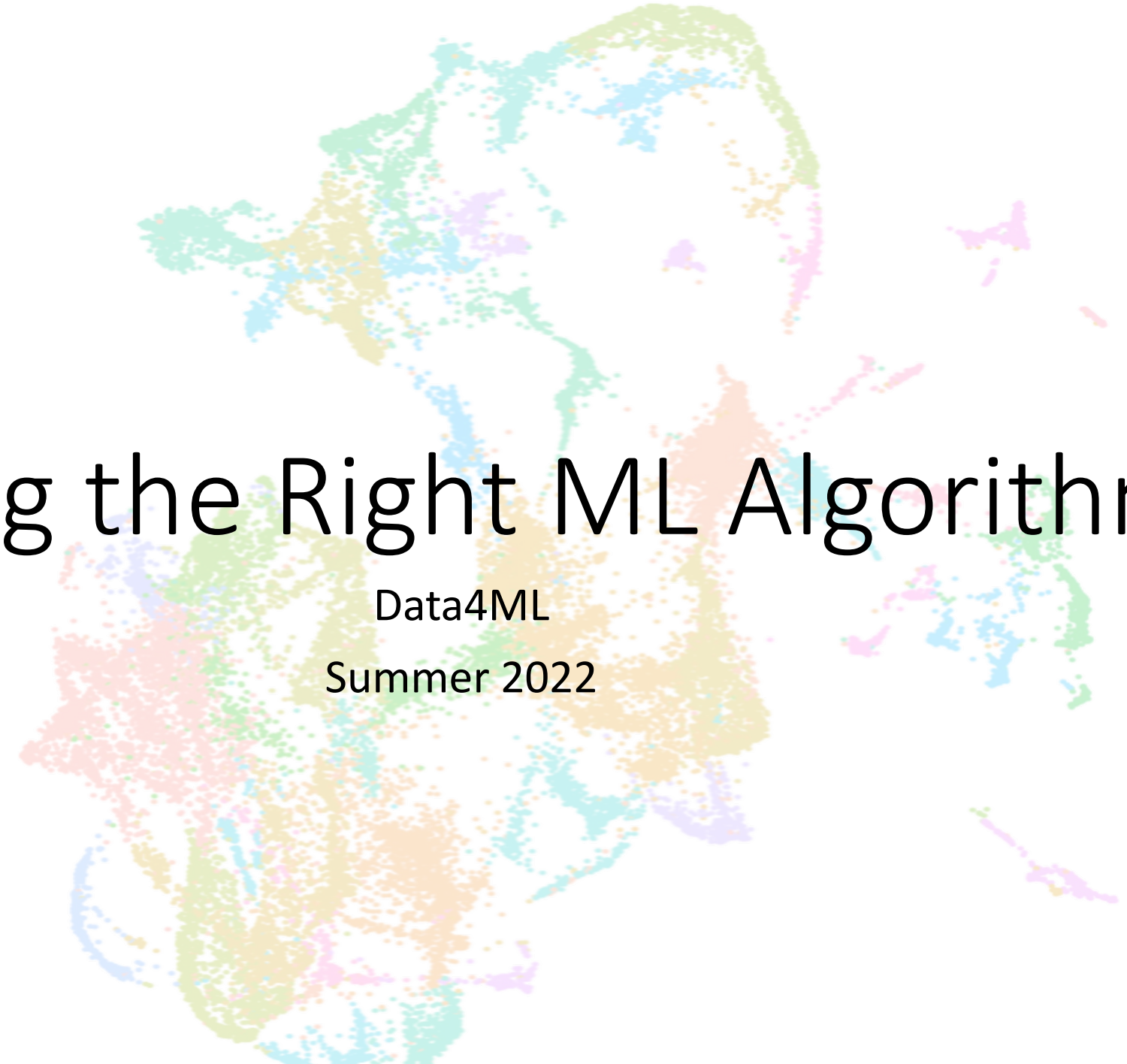


Picking the Right ML Algorithm

Data4ML

Summer 2022

It’s still all about data

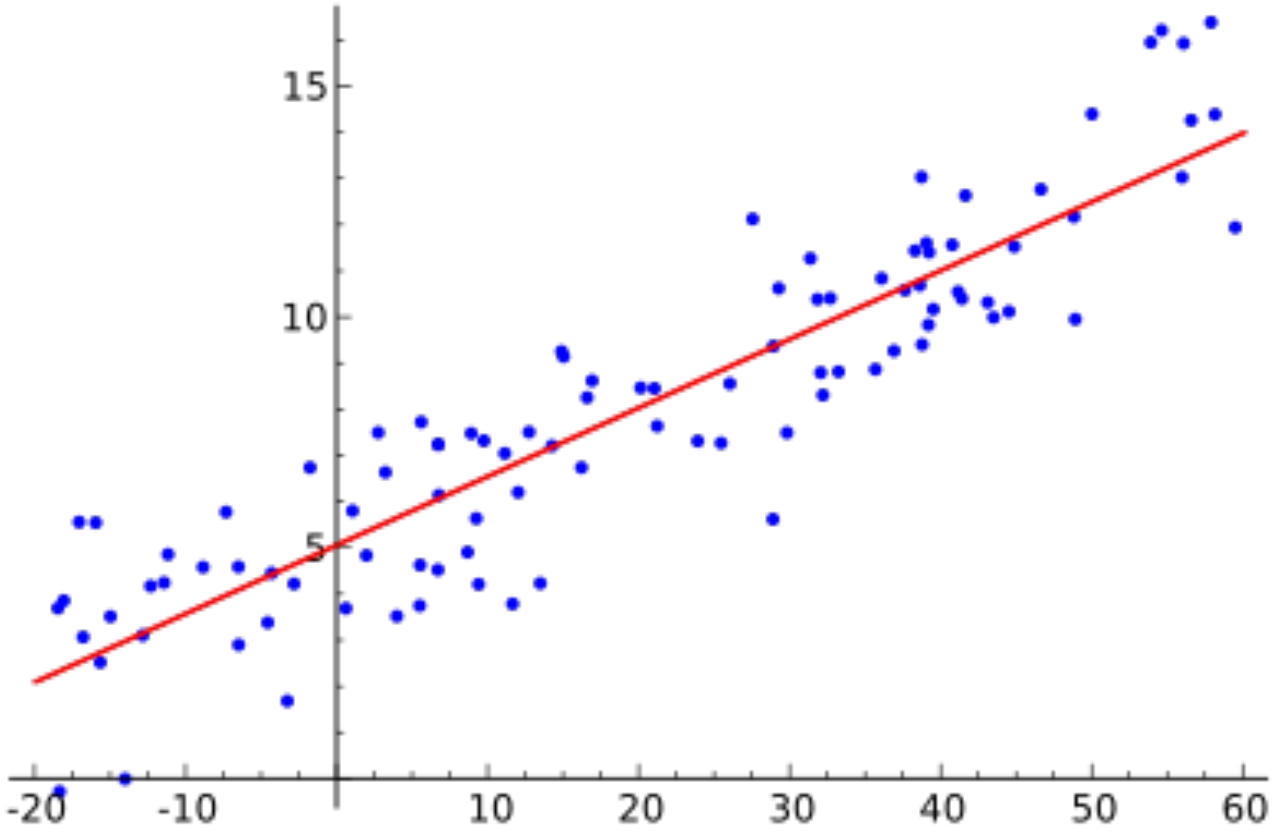
- There are lots of ML algorithms. What’s makes them different is how they perform on different datasets

- What makes an algorithm ‘good’
  - Low bias, high accuracy
  - Able handle your input features (size and type)
  - How easy it is in interpret
  - How fast it learns

Structured Data - Methods

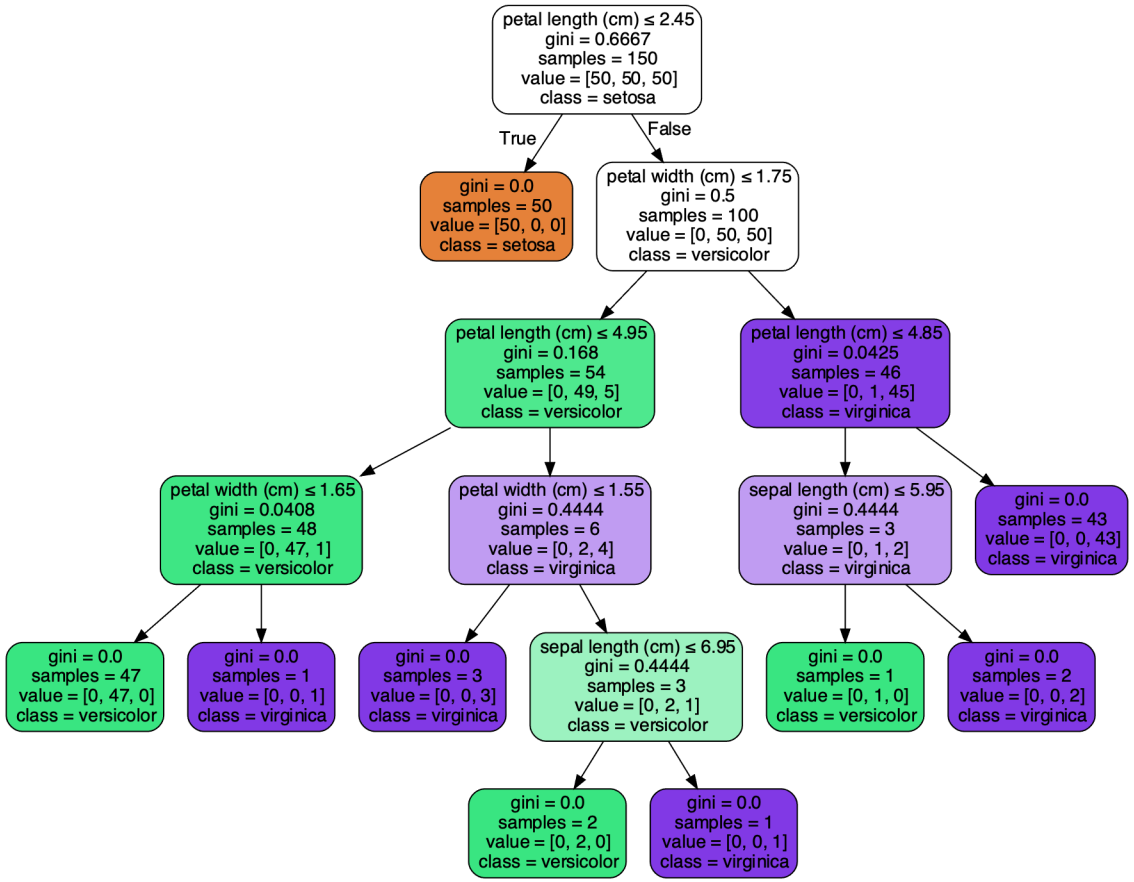
Model Parade – Linear Regression

- Linear Regression
- Assume the data can be represented by a line

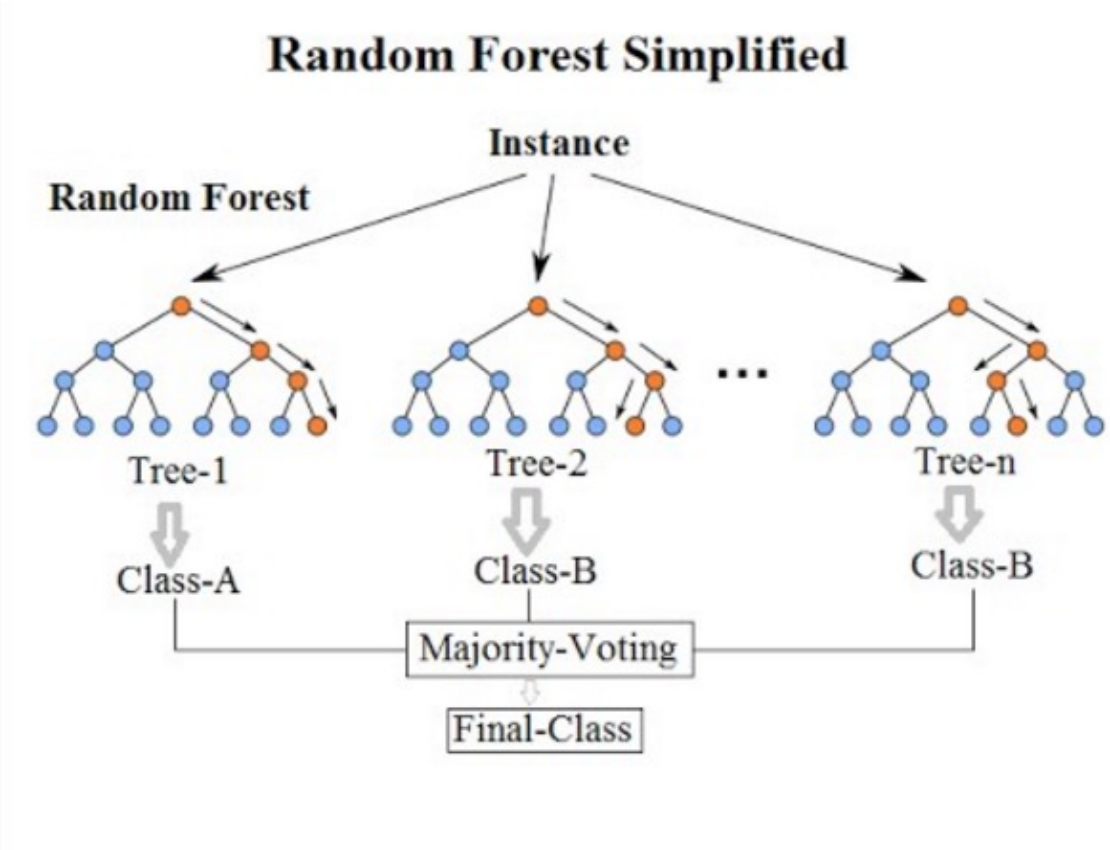


Model Parade – Random Forest

Decision Tree



Decision Forest



Model Parade – Random Forest

- Each node is split to minimize the gini index (p*(1-p)) based on a simple cut on a variable.
- Variables to cut on at each node are selected at random from all the variables normally three are used
- Each tree uses a subset of the data
- Robust to overfitting due to random sub-sampling
- Good error estimates from Out of Bag samples (those not selected)
- **Importance** can be used as an interpretation

Model Parade – Boosted Decision Tree

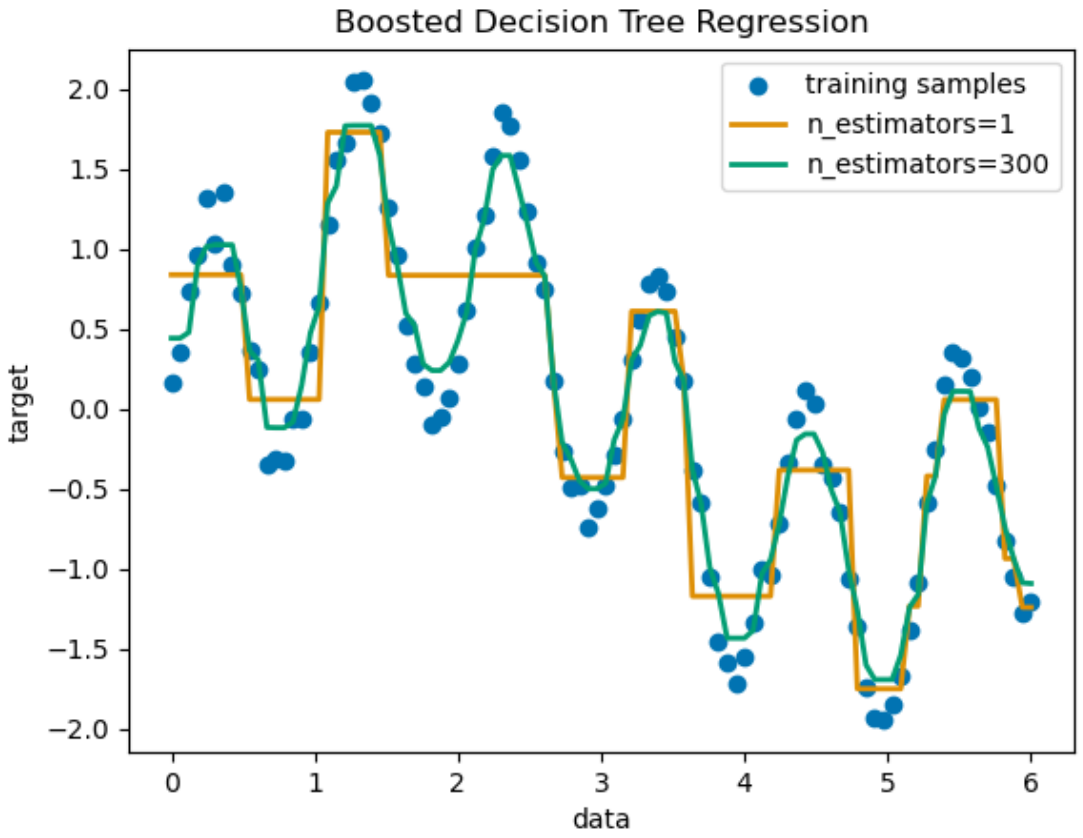
- Same tree as a random forest, but focus is on accuracy
- Data is not randomly sub-sampled
- Trees in the forest are trained iteratively with data miss-classified by the previous tree ‘boosted’ so subsequent trees pay more attention to mistakes
- **Accuracy is better than a random forest on large datasets**
- **Overfitting should be checked for**

A Quick Note - Classification vs. Regression

- Most or all ML techniques can be used for both:
  - Classification – targets are discrete labels
  - Regression – targets are continuous outputs

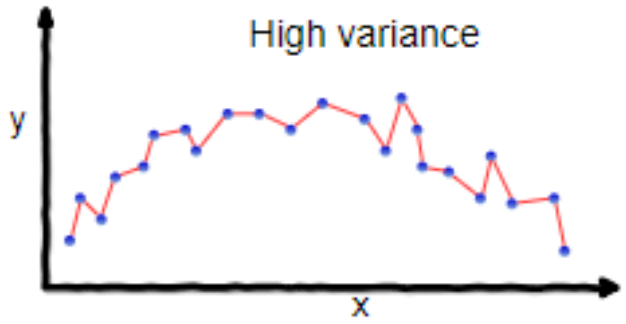
- There are generally some modification that need to be make
  - Linear regression becomes Logistic regression
  - A Decision Tree becomes a Regression tree

- Sometimes you'll need to run a different function or specify in the arguments. Sometimes the packages will figure this out for you

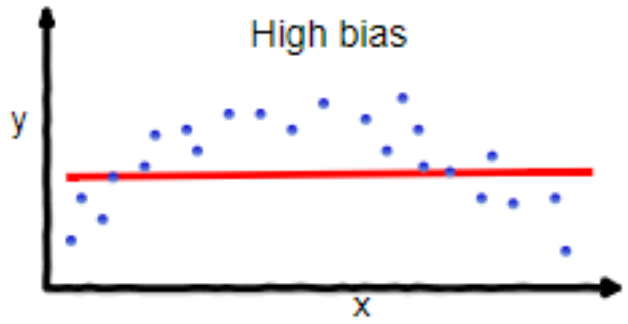


https://scikit-learn.org/stable/auto_examples/ensemble/plot_adaboost_regression.html

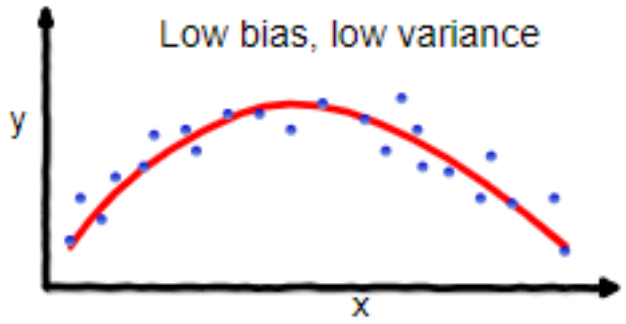
Bias Variance Tradeoff



overfitting



underfitting

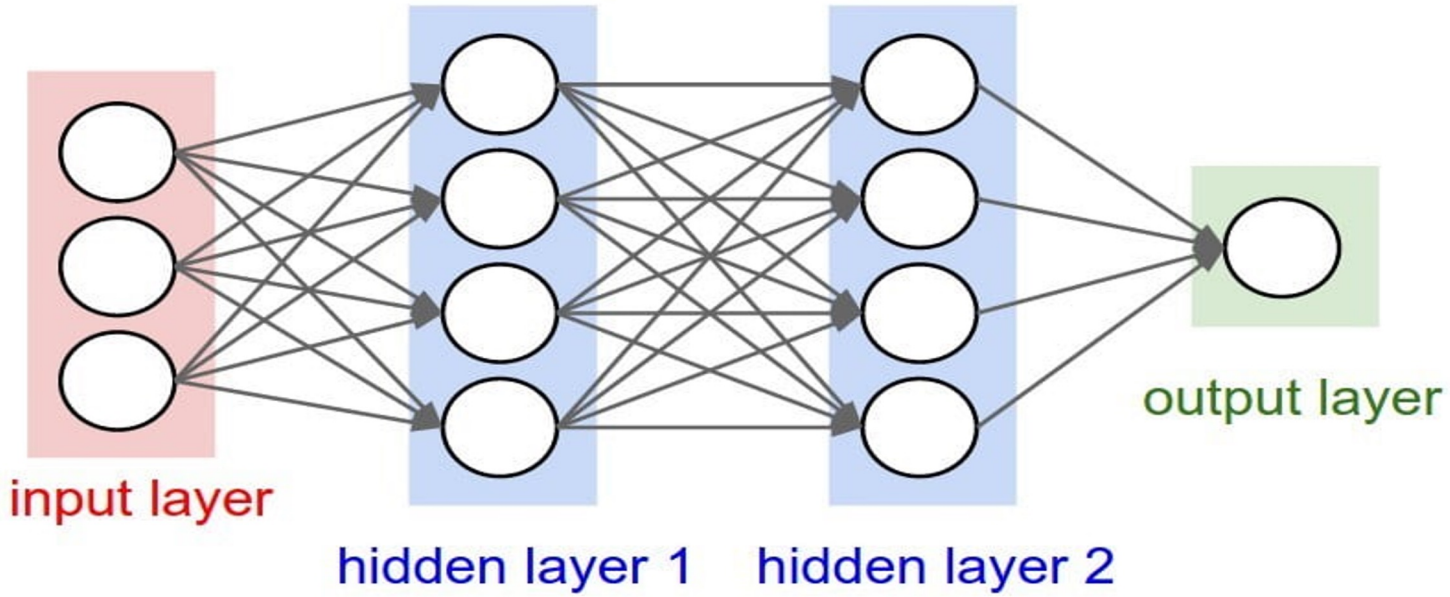


Good balance

https://towardsdatascience.com/understanding-the-bias-variance-tradeoff-165e6942b229

Neural Networks

- A bit tricker to work with than decision tree
  - Need to be careful about data normalization
- Training may not actually be better than a boosted decision tree
- Why use them?
  - Offers tremendous flexibility
  - Can use any differential loss function
  - Has excellent performance if tuned



Support Vector Machines

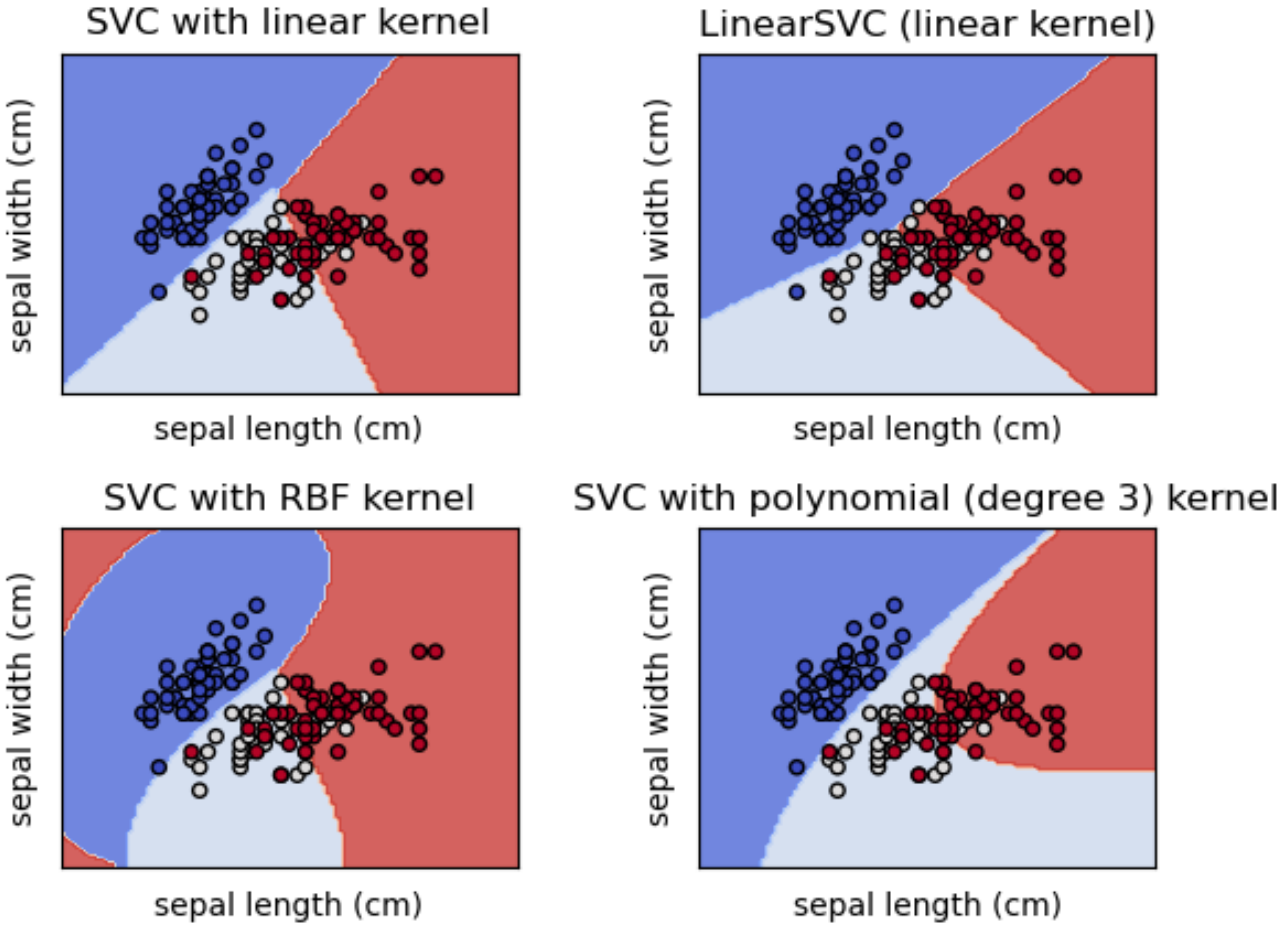
The advantages of support vector machines are:

- Effective in high dimensional spaces.
- Still effective in cases where number of dimensions is greater than the number of samples.

The disadvantages of support vector machines include:

- If the number of features is much greater than the number of samples, avoid over-fitting in choosing Kernel functions and regularization term is crucial.

- SVMs do not directly provide probability estimates, these are calculated using an expensive five-fold cross-validation (see Scores and probabilities, below).



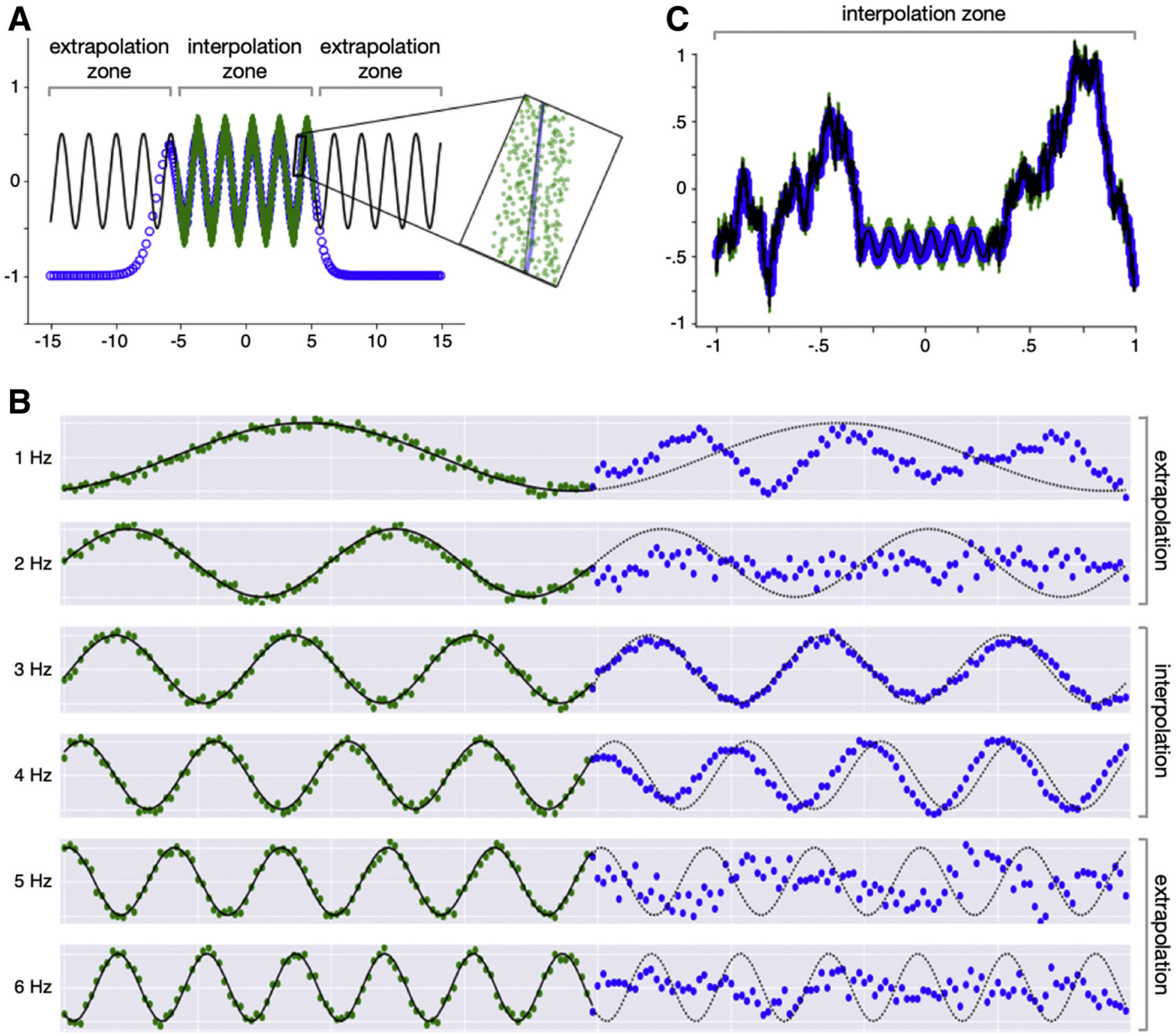
https://scikit-learn.org/stable/modules/svm.html

Assumptions ’buy’ Data

- Simple models like linear regression, assume your data can be fit to a line
  - You need comparatively little data to do this
  - But it your dataset isn’t linear you can end up with a high bias
- If you use complicated models without a functional form assumption
  - You have to worry about overfitting
  - May need more data for good performance
- In both cases extrapolating beyond your dataset relies on assumptions
  - In the linear case you assume your line continues
  - In the

Extrapolation

- ML does a lot of neat things, but it isn't magic
- Unless specified treat ML algorithms as universal function approximators
  - i.e. 'Linear' regression is not a universal function approximator
- If you're collecting data make sure you collect 'representative' samples



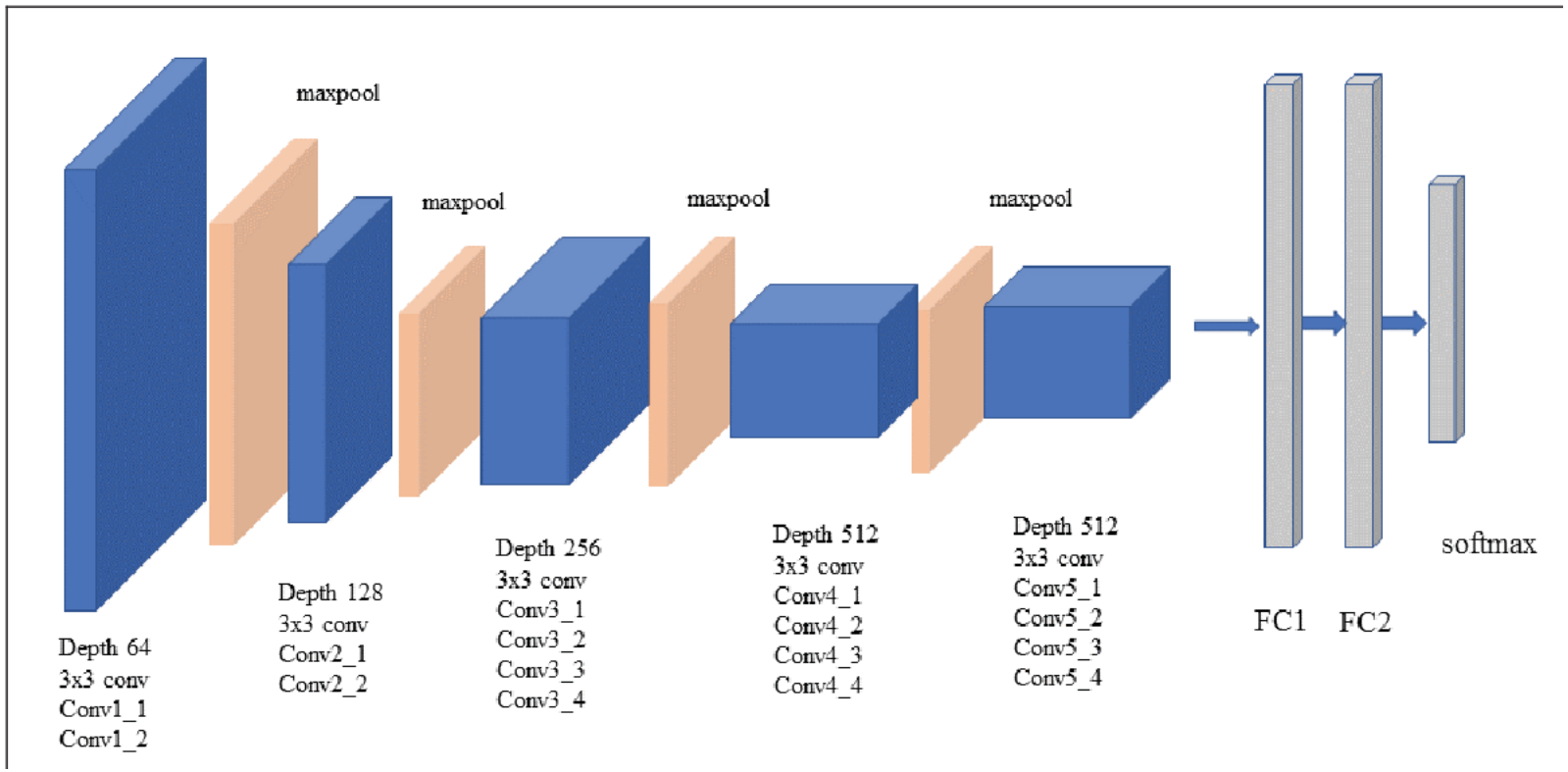
Unstructured Data

- Unstructured Data is generally used to refer to anything that doesn’t easily fit into our normal rows are examples columns are variables data model
- Images, text, sounds etc.
- Two common ways of dealing with this data
  - Feature engineering: Writing methods to extract ‘structured data’
    - i.e. A Bag of words from text
  - **Deep Learning**

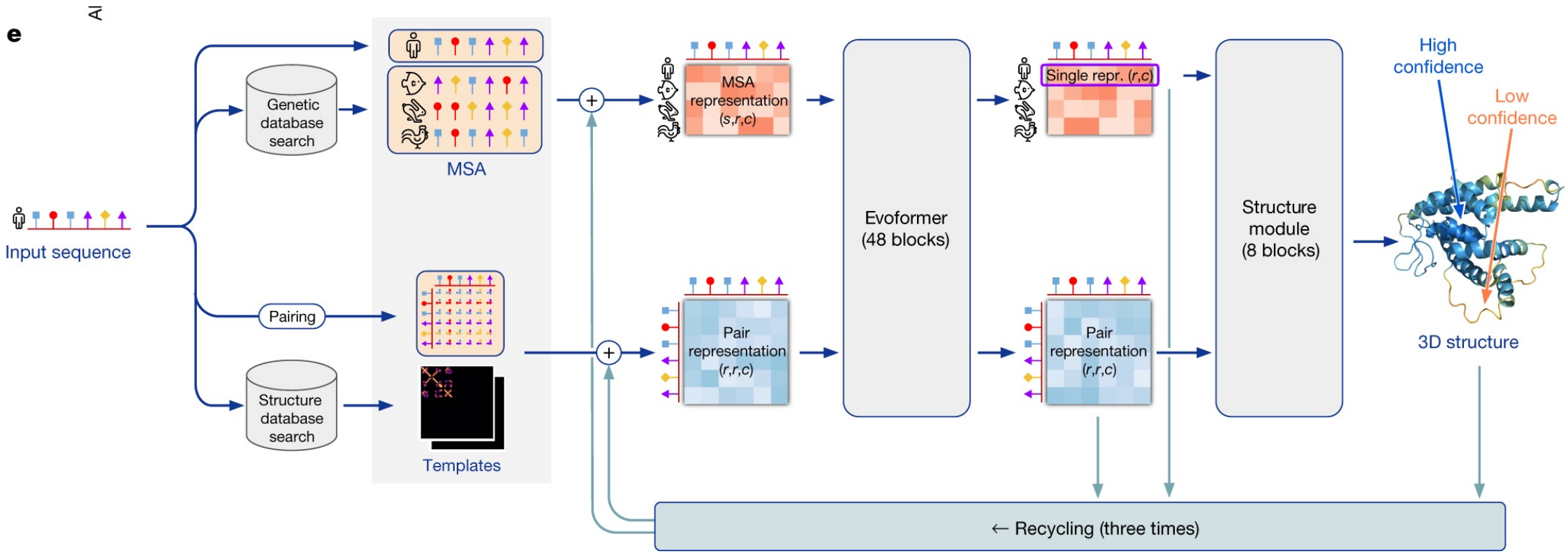
Deep Learning

- Unlike the above algorithms Deep Learning doesn't neatly fit into an R packages
- It’s better to think of it as a box of Lego blocks
- Deep Neural networks are made up of layers
- Layers exist for text, images, structured data, etc.
- Allows for building and training custom models
- This models are often made available

Example VGG 19 (Image Classification)



Example Alpha Fold (Protein Structure Prediction)



Evaluating Algorithms

- I’m a big proponent of of evaluating models based on downstream goals
  - Are the predictions useful for my understanding
  - Is the model saving me time
  - Etc.
- However, there is a lot of formal methods for evaluating the quality of predictions
  - Accuracy
  - Precision
  - Recall
  - F1
  - It’s worth asking yourself how good these need to be to meet your goal.

The Problem with Accuracy

- $Accuracy = \frac{Correct\ Classifications}{All\ Predictions}$
- Common metric, but you need to be careful about class imbalances

	Prediction True	Prediction False
Class True	90	0
Class False	10	0

- The above case guesses True for everything
- It’s 90% accurate but isn’t very useful

Other Metrics

	Prediction True	Prediction False
Class True	True Positive	False Negative
Class False	False Positive	True Negative

	Prediction A	Prediction B
Class A	90	0
Class B	10	0

$$Precision = \frac{T_p}{T_p + F_p} = 90/100 = 0.9$$

$$Recall/Sensitivity = \frac{T_p}{(T_p + F_n)} = 90/(90) = 1.0$$

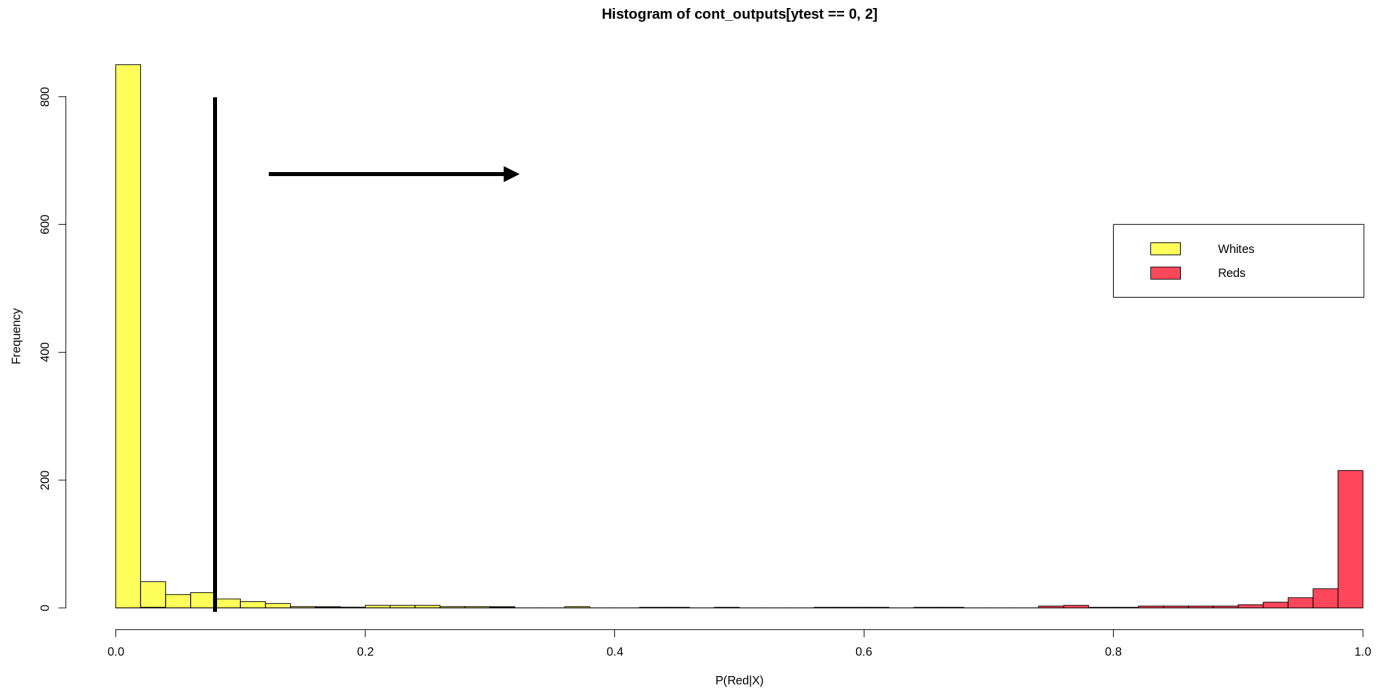
$$\boldsymbol{Specificity = \frac{T_n}{Tn + Fp} = \frac{0}{0 + 10} = 0}$$

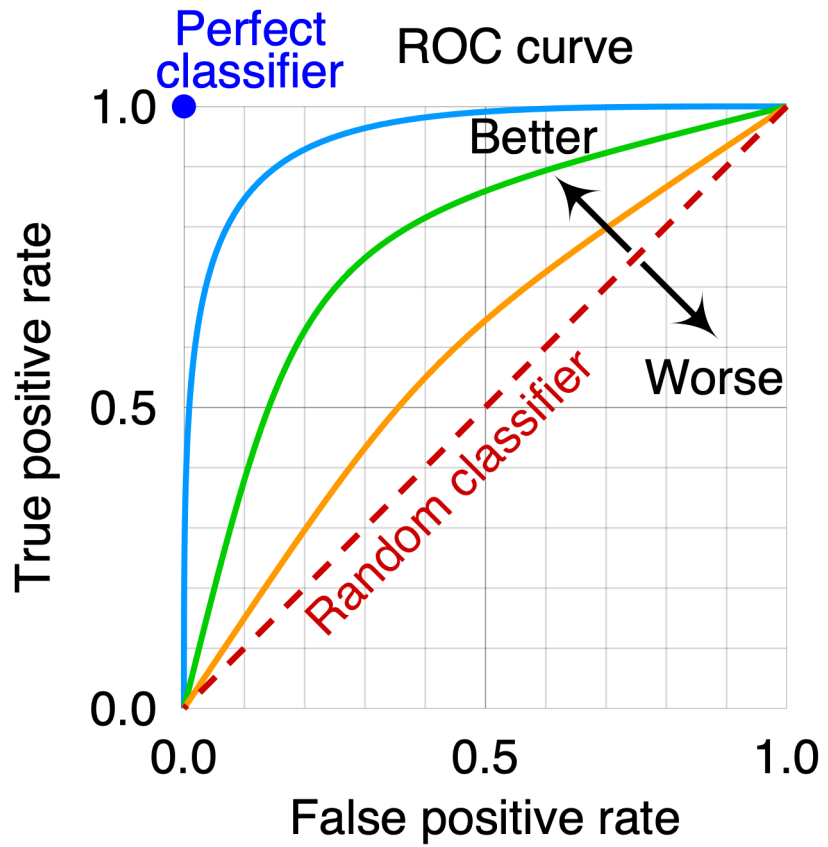
$$F_1 = 2 * \frac{Precision * Recall}{(Precision + Recall)} = 0.95$$

- Lots of metrics that go by lots of names
  - In this case specificity is a warning sign the model is broken/useless, but the others look okay, there are examples that go the other way as well
- Recommendation make sure you check the performance on each of your classes and ask if it's good enough for your goals

ROC Curves

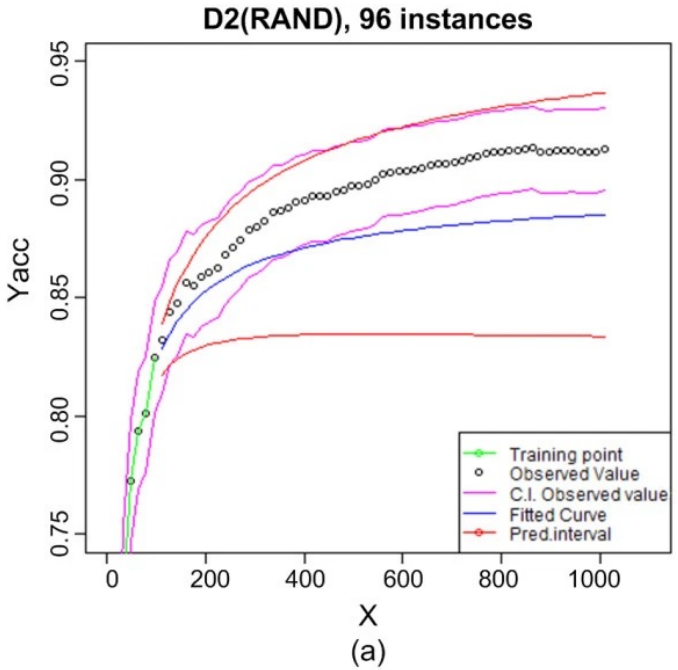
- It can be a bit more complicated as we often don't just know the decision we can get a probability
- By default most tools assume >0.5 is True, but it isn't required
- You can give this Receiver Operator Characteristic
- **AUC = Area under Curve with 1 being the best**

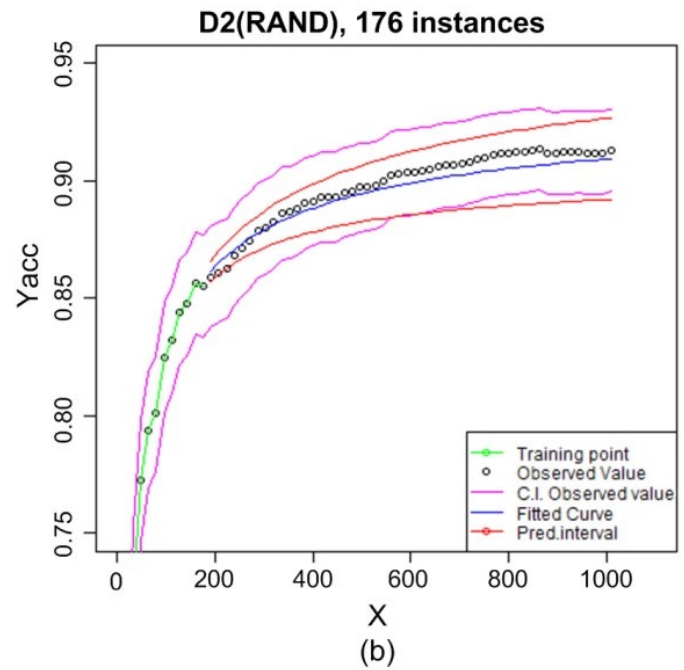


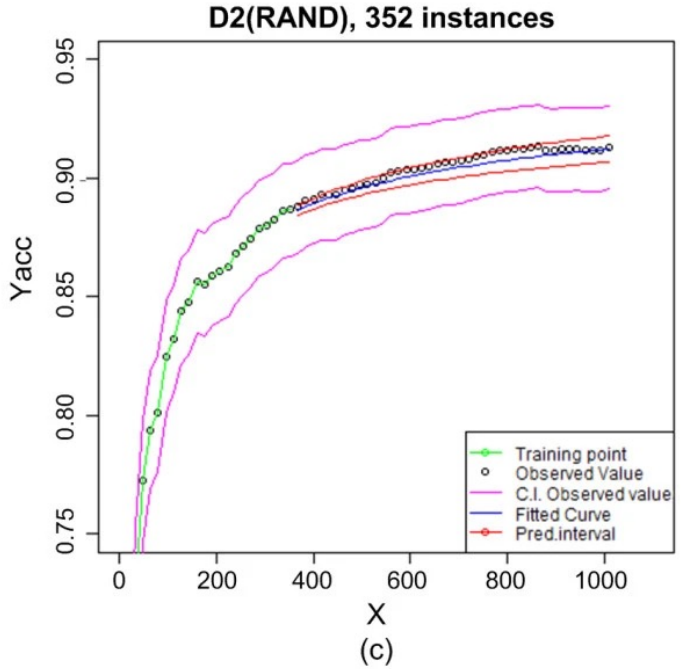


Learning Curves

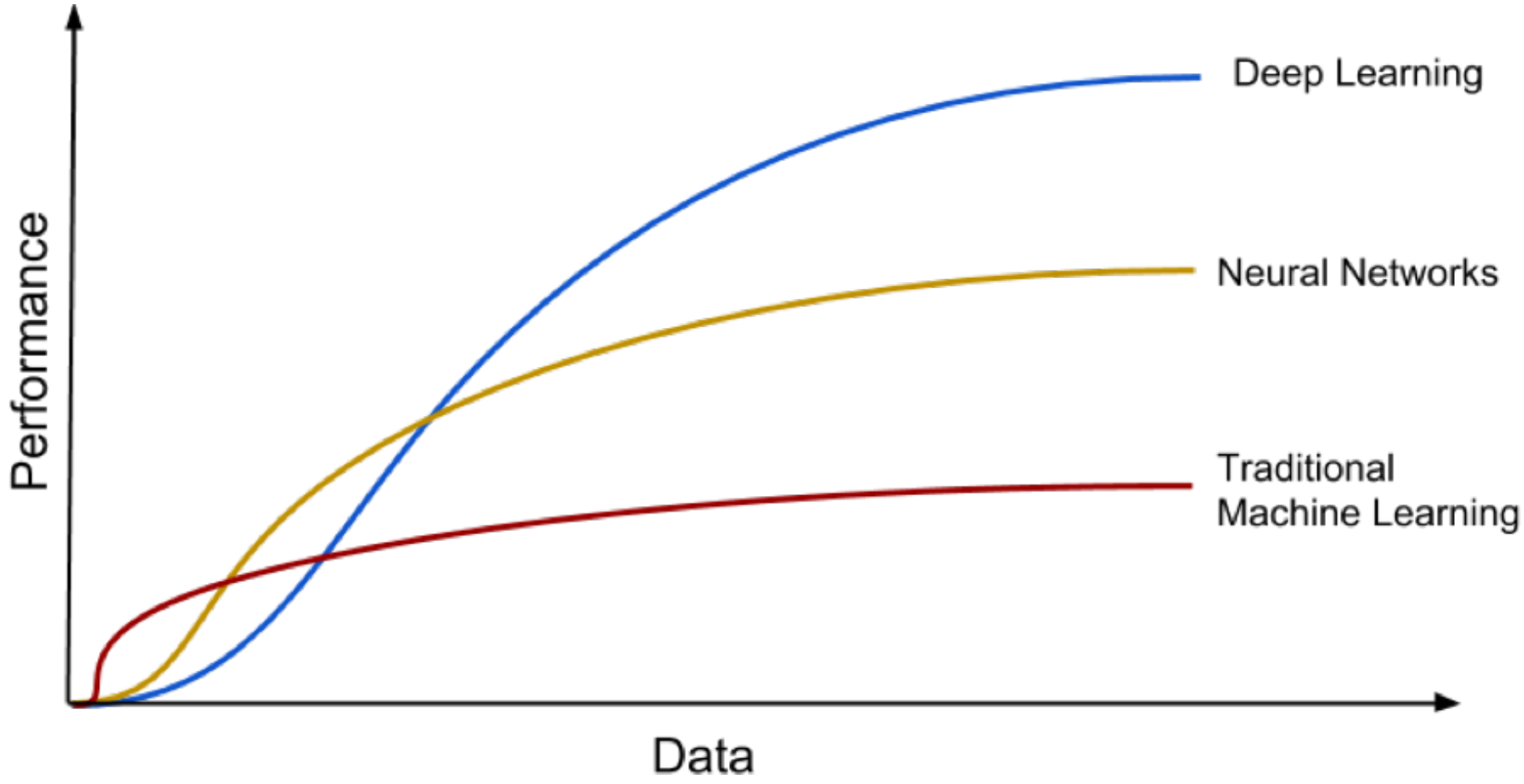
- **Predicting sample size required for classification performance** https://bmcmedinformdecismak.biomedcentral.com/articles/10.1186/1472-6947-12-8
- You'll need some data to estimate how much more you'll need to reach a given accuracy
- ML algorithms improve with data until they reach a saturation point
- Can be useful to train models early in data taking to get an idea of how much more you'll need to reach a certain goal







Models have different learning curves



Conclusions

- Lots of available algorithms and lots of ways to evaluate them
- Things to consider:
  - The kind of data you have
    - Structured vs. Unstructured
  - How much of it you have

- **Suggestion:** Always keep your broader goals in mind when evaluating your models