Introduction to Machine Learning

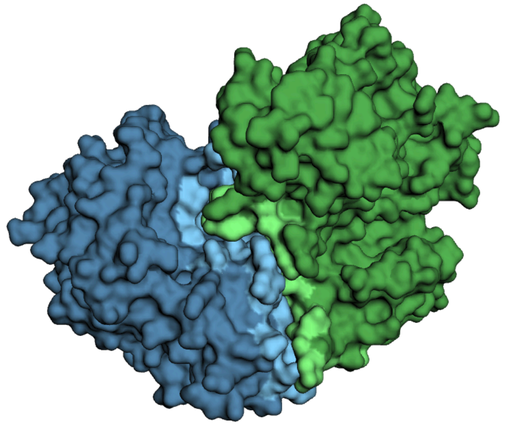
Demystifying Jargon, Application, and Evaluation

Juan Felipe Beltrán, Ph.D.
Cornell University Department of Biomedical Engineering
juanfelipe@cornell.edu
@offbyjuan

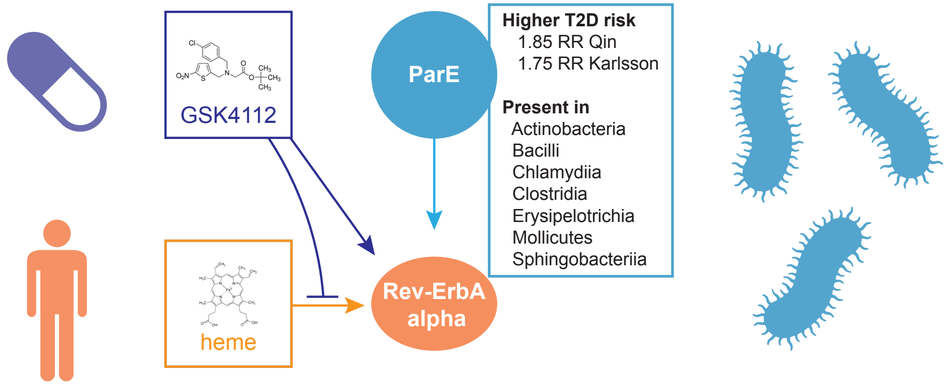
NYU Cardiovascular Research Center

About Me

Can we predict **where** proteins bind?



How do **bacteria** modulate human health?



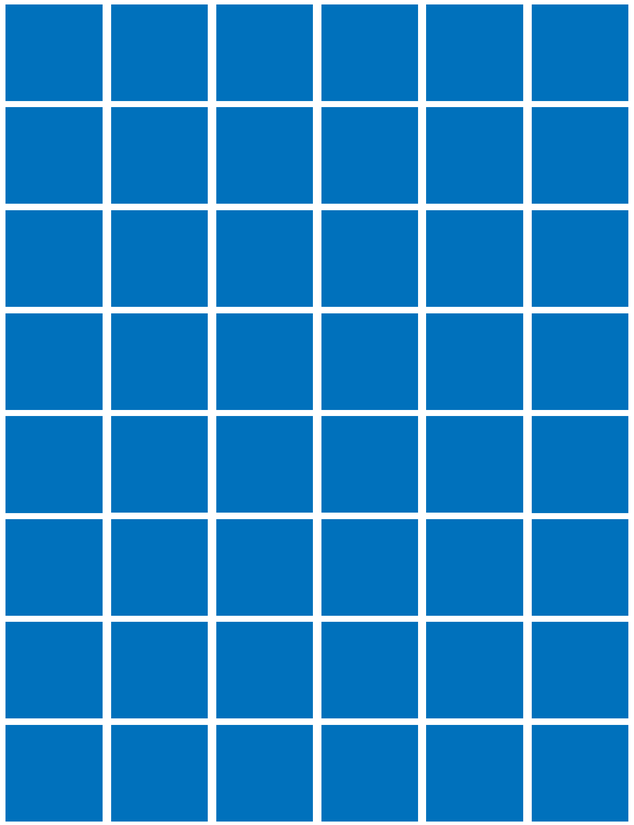

for (var
dynamicSo
return a.length + 1;
} else {

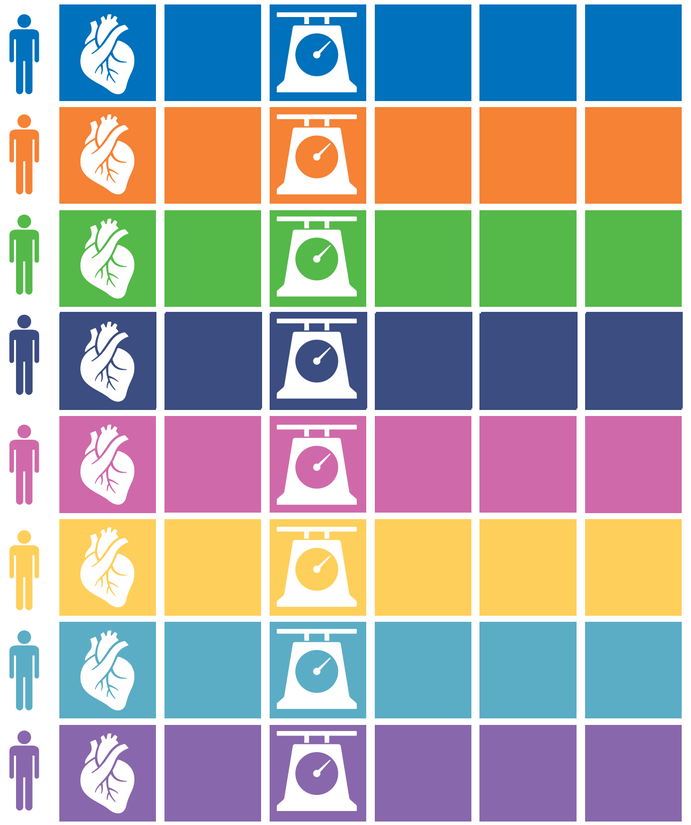
“I have a lot of data…

Can I do machine learning on it?”

Data Classic™

The simplest dataset is a table





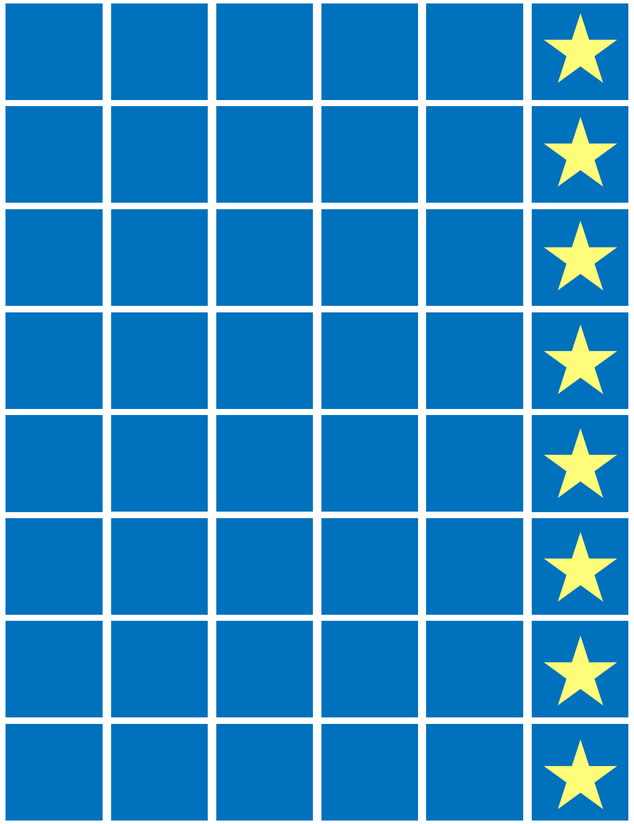
Data Classic™

The simplest dataset is a table

A patient's **feature vector**

Data Classic™

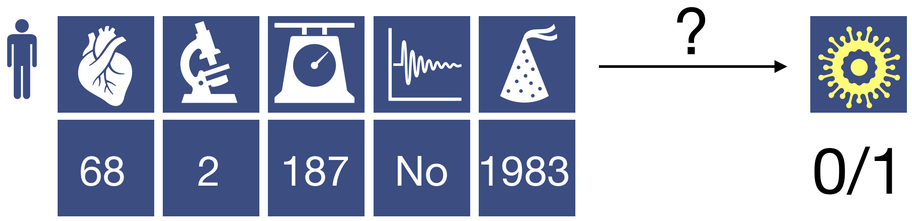
The simplest dataset is a table



Special feature

Label / Outcome / Class

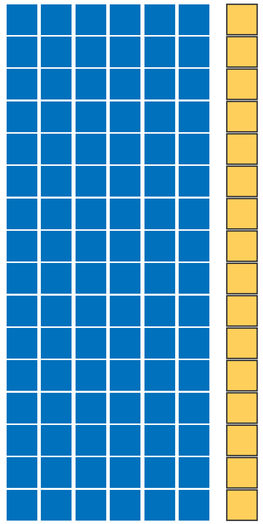
Something you **wish** you could predict based on all the other features.



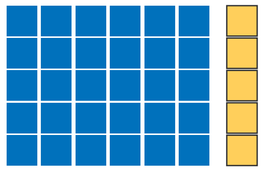
“X”

“y”

A “Supervised” Machine Learning Workflow

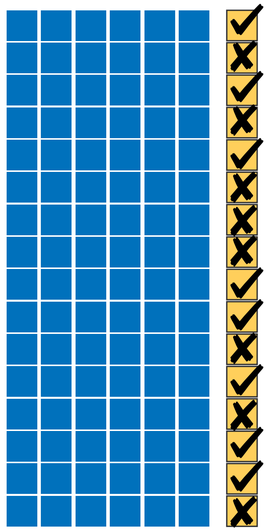


`X_train` `y_train`

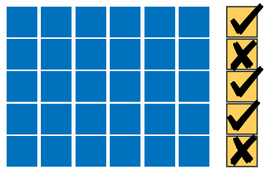


`X_test` `y_test`

A “Supervised” Machine Learning Workflow

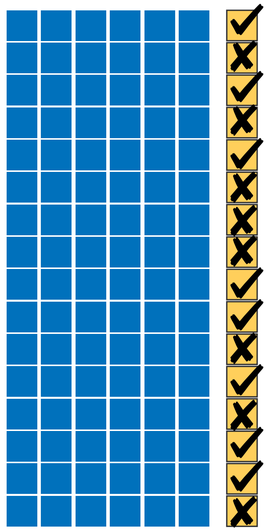


X_train y_train

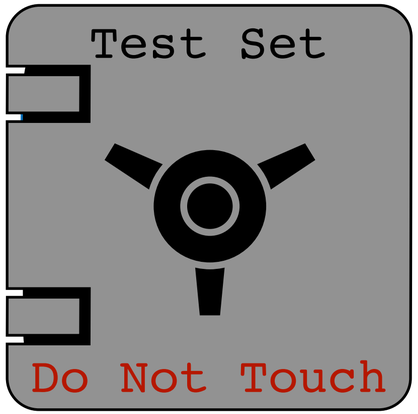


X_test y_test

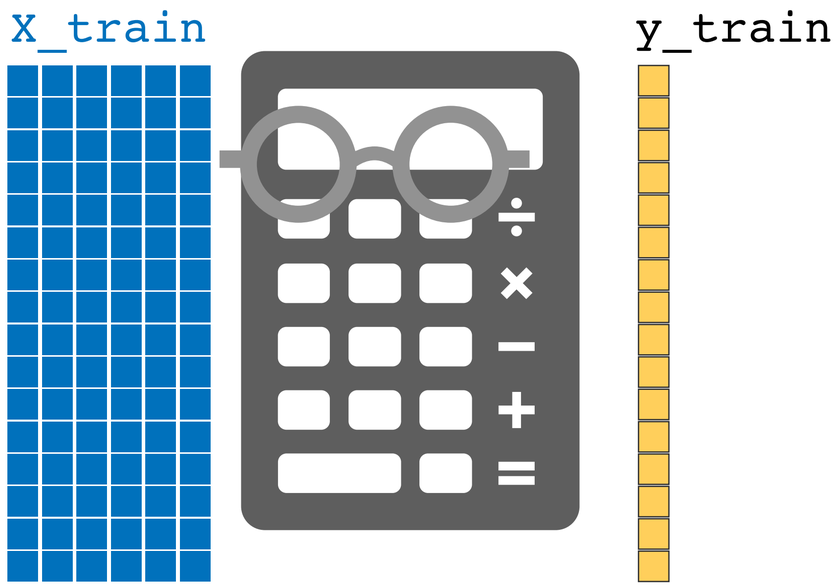
A “Supervised” Machine Learning Workflow

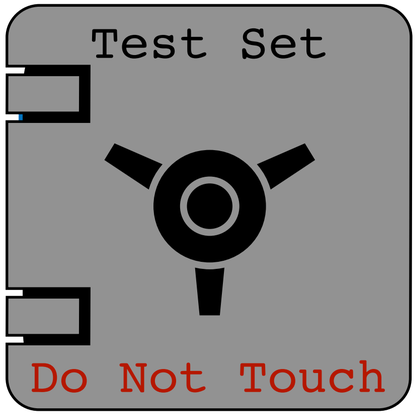


X_train y_train

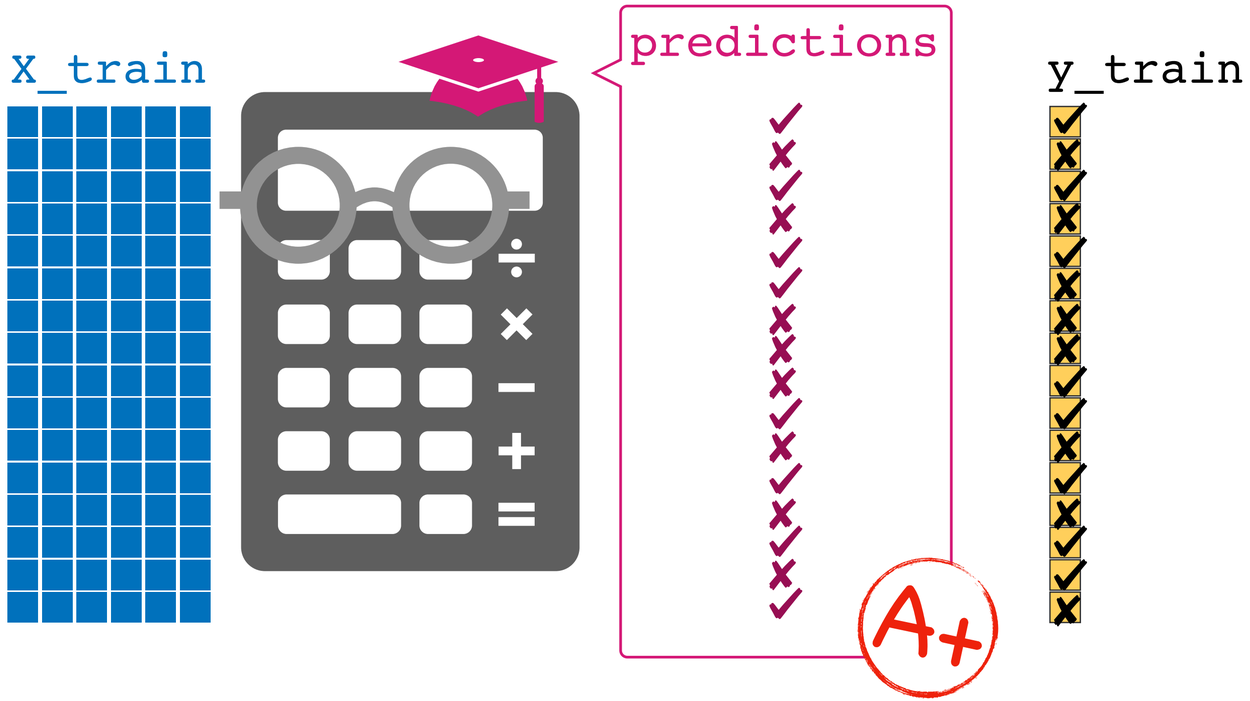


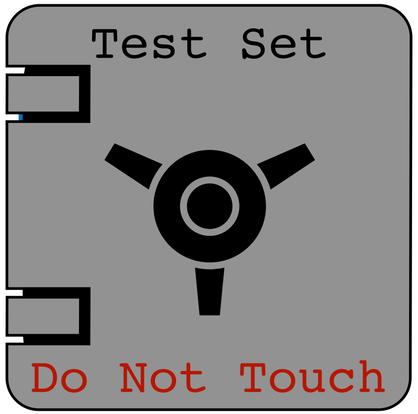
A “Supervised” Machine Learning Workflow



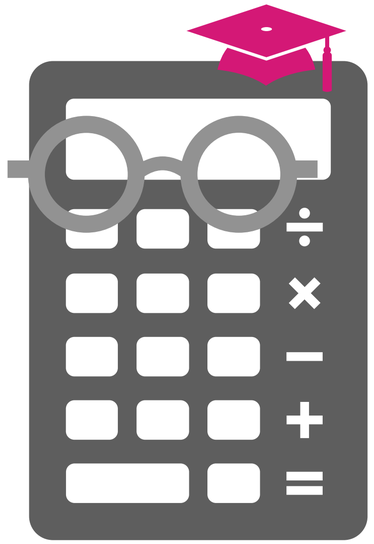


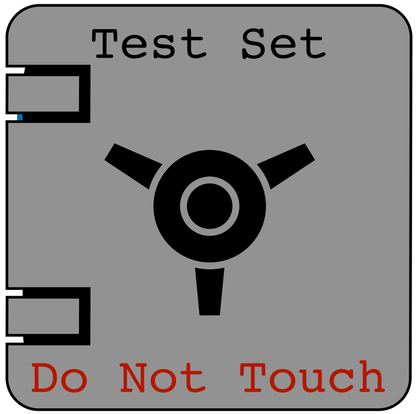
A “Supervised” Machine Learning Workflow



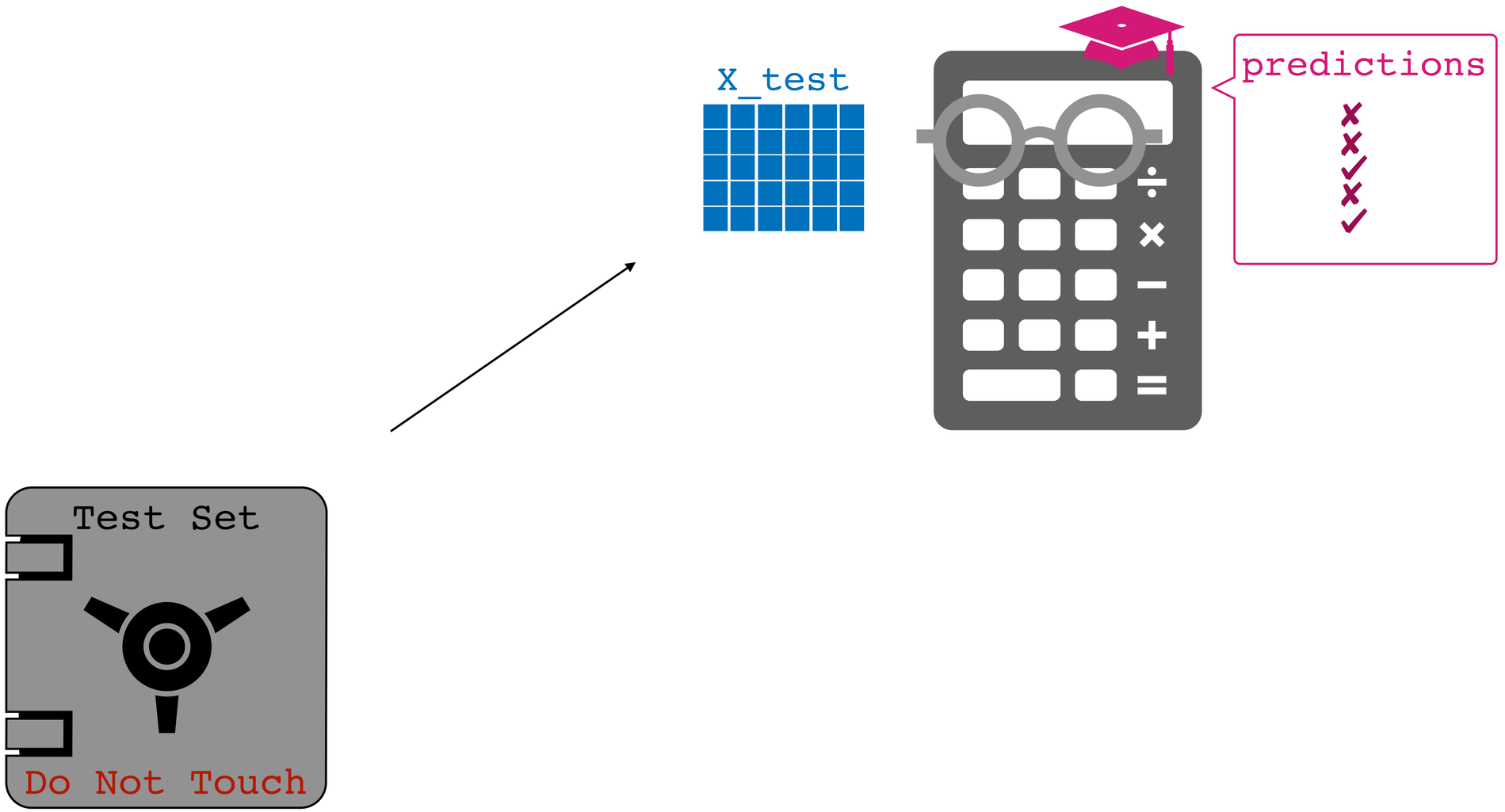


A “Supervised” Machine Learning Workflow

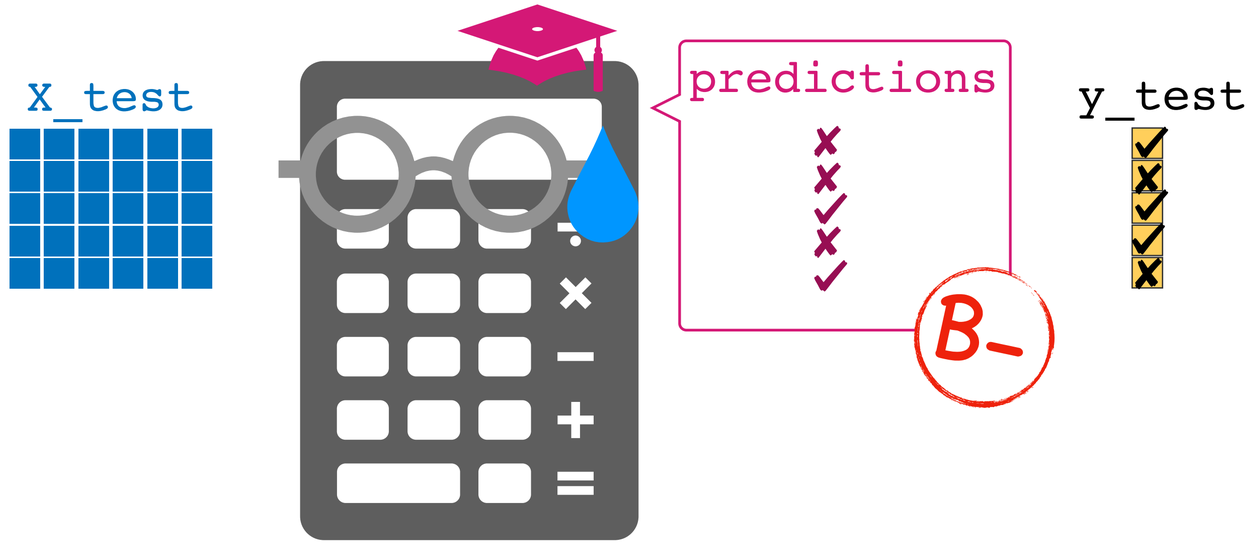


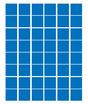


A “Supervised” Machine Learning Workflow

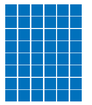


A “Supervised” Machine Learning Workflow

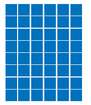




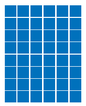
Fitting a **Logistic Regression** to prioritize **drug targets** based on **chemical similarity**



Training a **Random Forest** to predict **interface residues** using **amino acid properties**

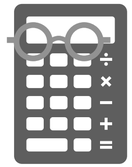


Using a **Neural Network** to diagnose **tumor presence** from **image slides**



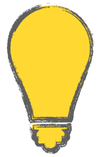
Building a **Decision Tree** to determine **age** from RNAseq **gene abundances**

Making data machine-learnable



Picking a machine learning algorithm

Evaluating classifiers



Applying this to your projects

Feature Engineering

- Split categorical features into multiple columns
- Don’t include identifiers in your dataset
- Normalization is usually required
- As long as you can get it into a table, you’re good!
  (Images, interactions, networks, sequence windows, or all!)

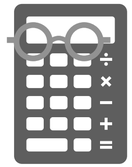
Feature Selection

- Redundant features can make classification harder
- Your expertise comes first: Does this feature make *sense?*
- Bias is everywhere, be vigilant!
- Be careful with missing values in X
- Be extra careful with missing values in y

Formatting

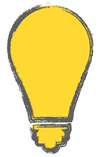
- Make sure your table is just a table
- Comma-separated files (.csv) are **great**
- Write a small README that describes the columns

Making data machine-learnable



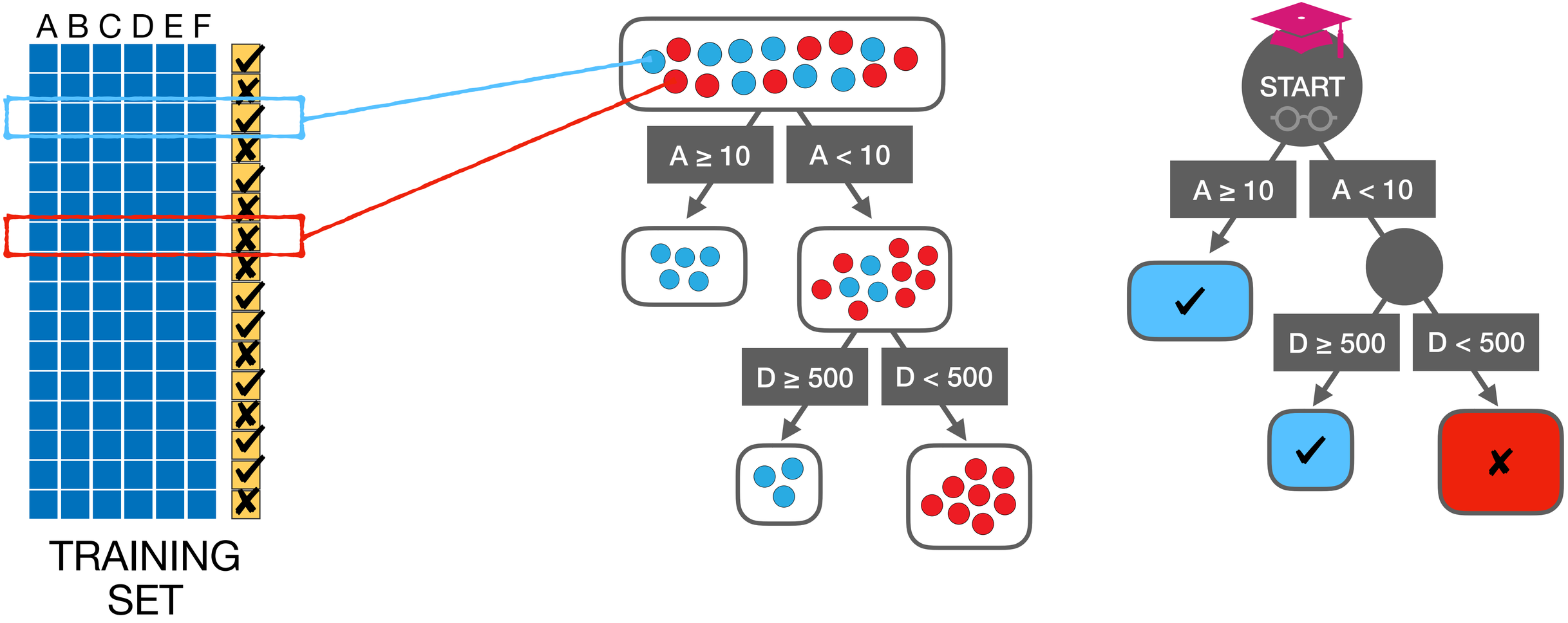
Picking a machine learning algorithm

Evaluating classifiers

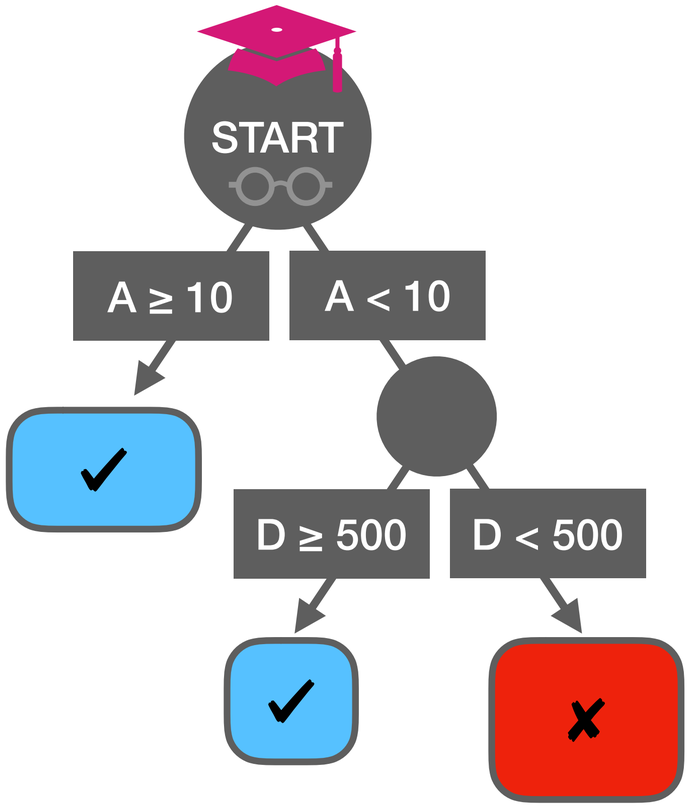


Applying this to your projects

Decision Tree Classifier

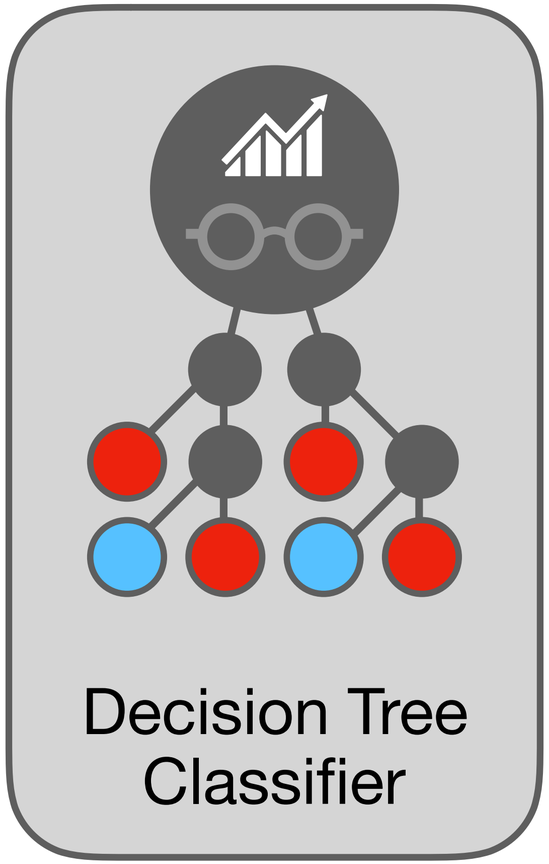


Decision Tree Classifier



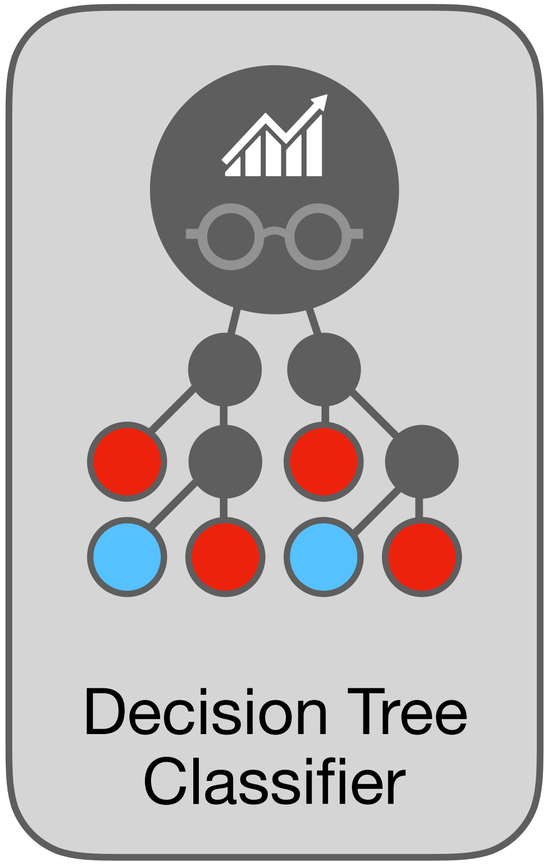
- Always builds the same tree on the same dataset
- Doesn’t care about normalization
- “Greedy”

Decision Tree Classifier



- Always builds the same tree on the same dataset
- Doesn’t care about normalization
- “Greedy”

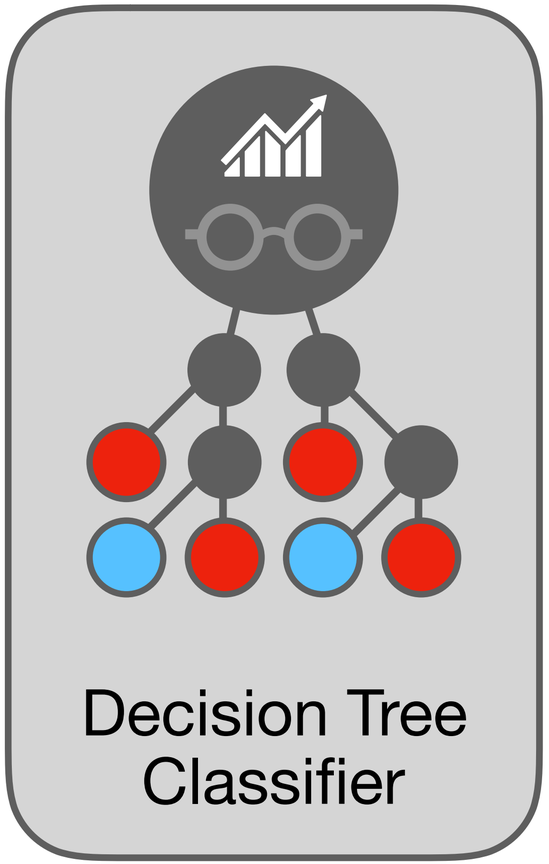
Greediness is bad… How can we relax it?

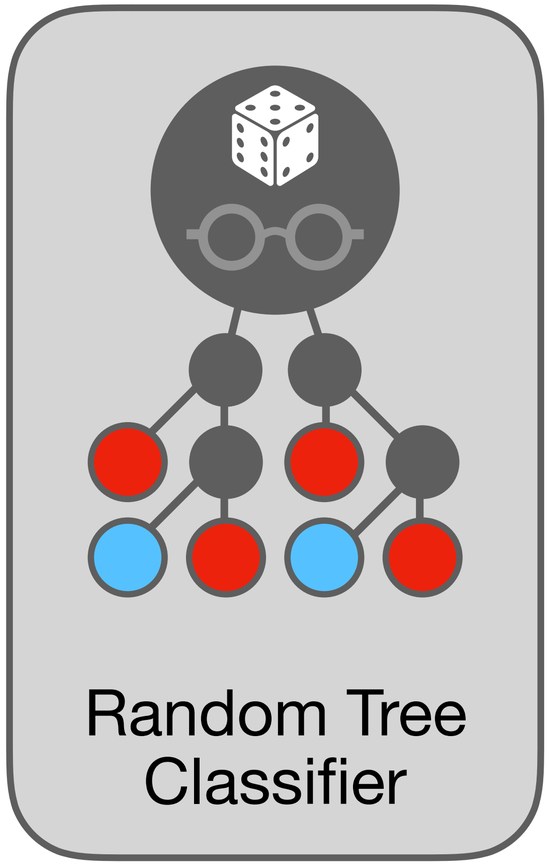


Your patient has a heart condition because their birthday was before March 2nd and their heart rate was between 168 and 172 bpm

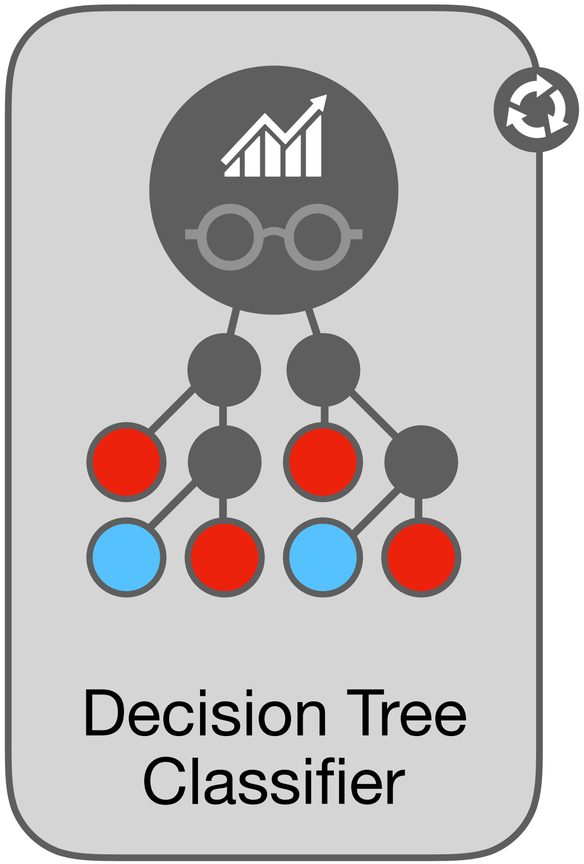
F

Greediness is bad… How can we relax it?

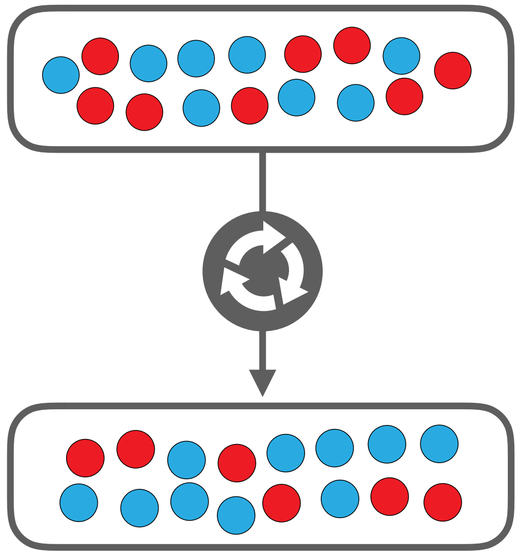




Solution 1: Bootstrapping

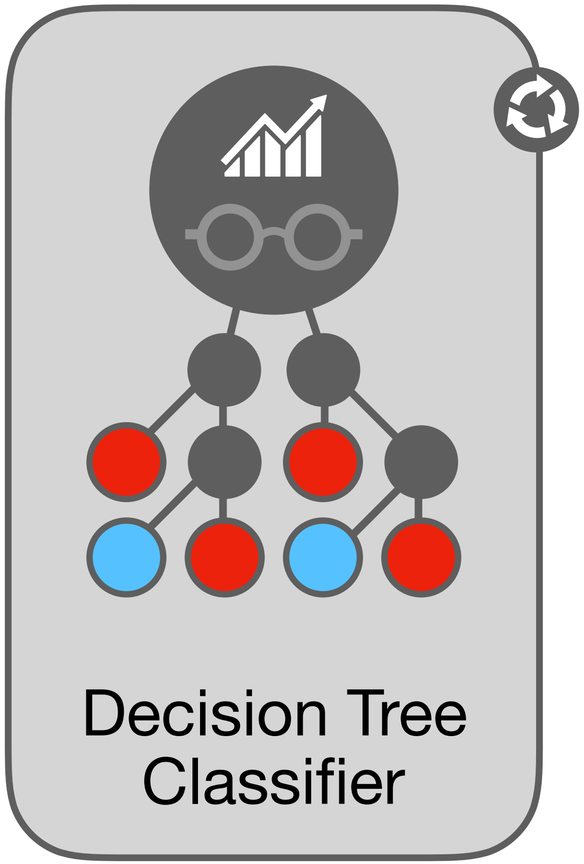


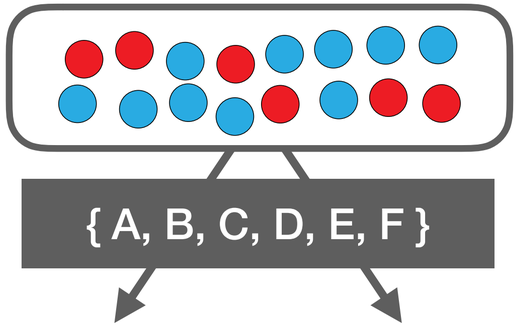
Pick N points at random



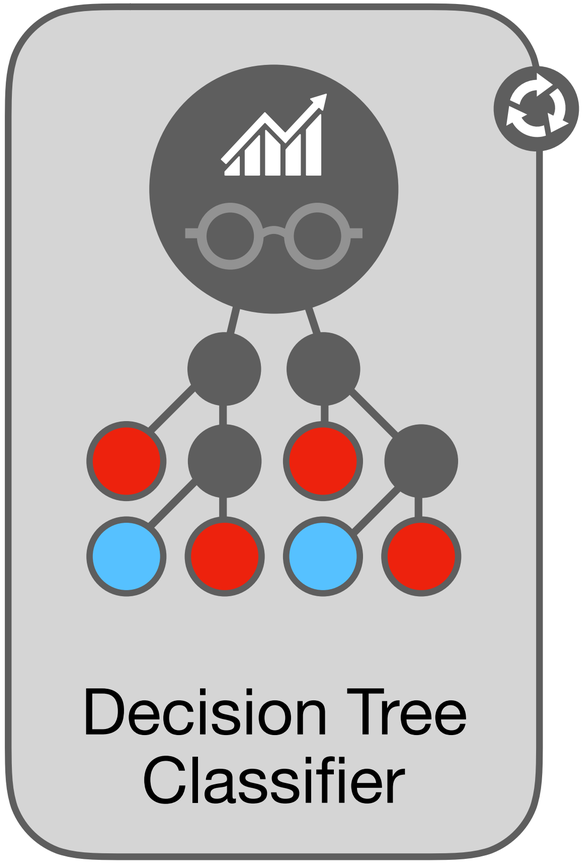
If N is your original size,
you still get a different dataset!

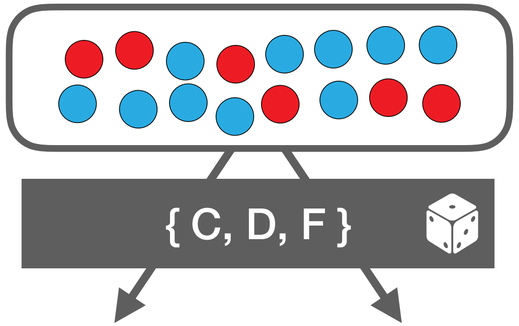
Solution 2: Limit the splits!



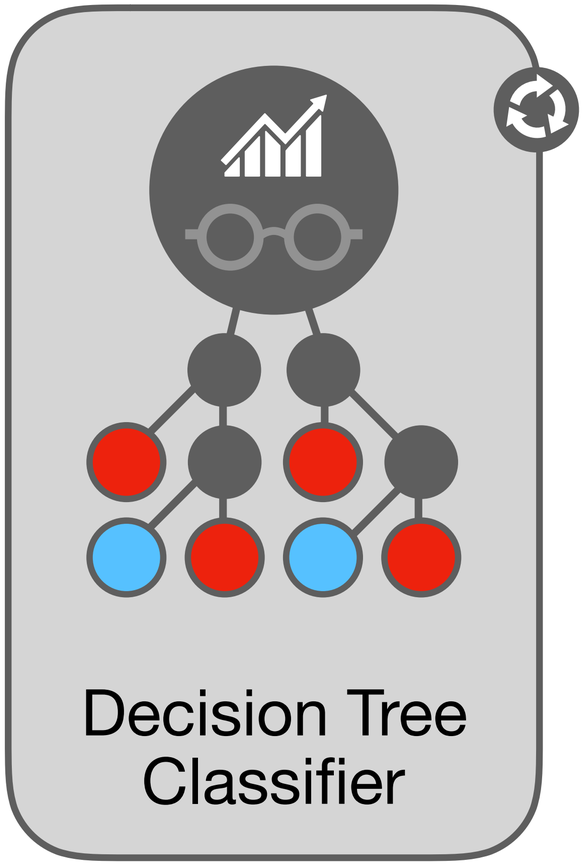


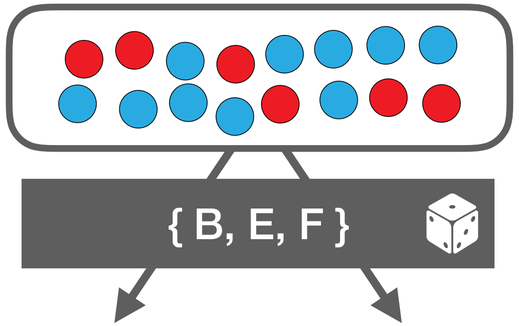
Solution 2: Limit the splits!



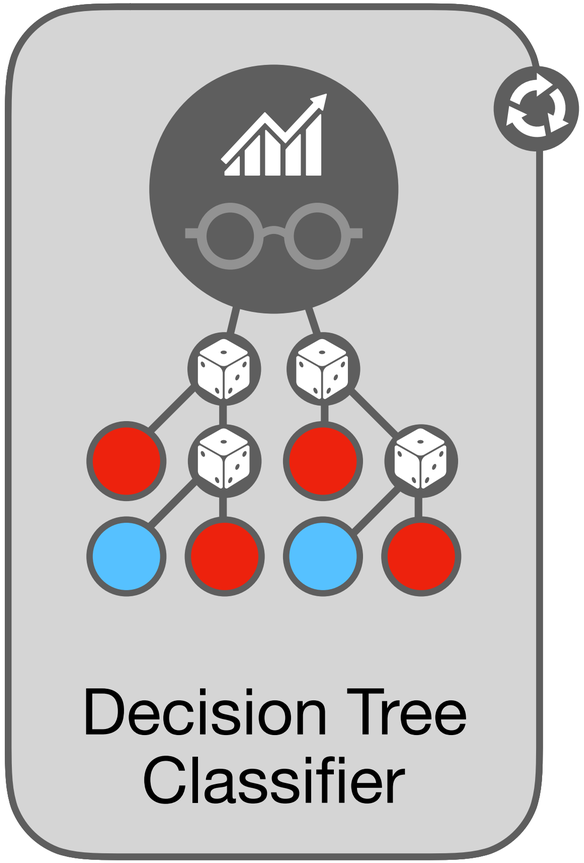


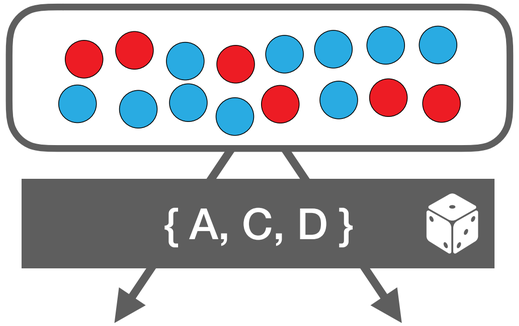
Solution 2: Limit the splits!



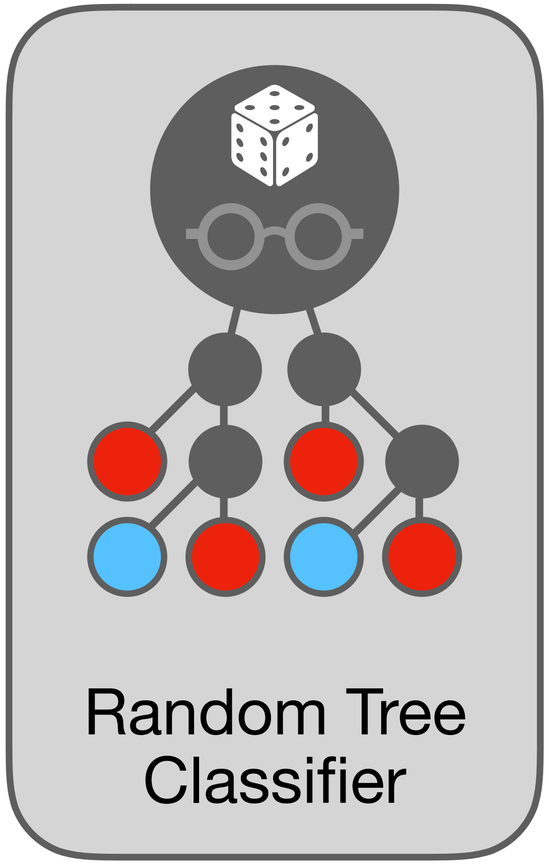


Solution 2: Limit the splits!



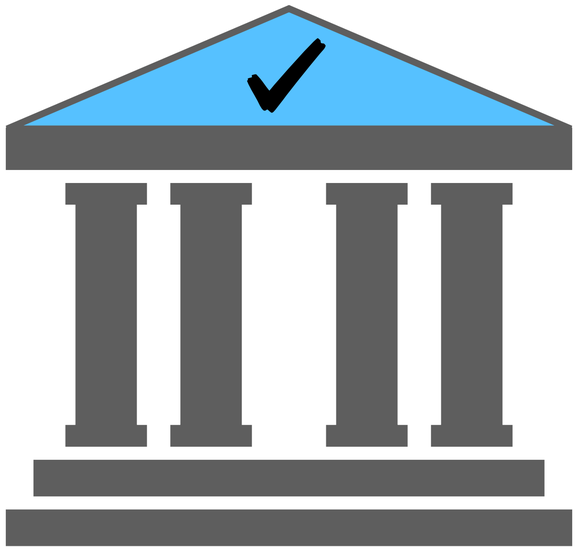


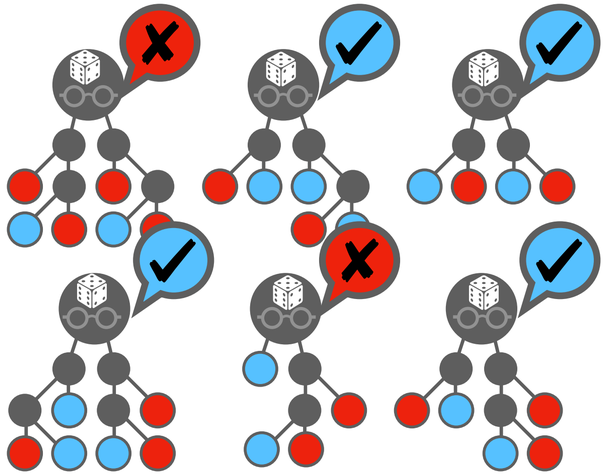
Random Tree Classifier



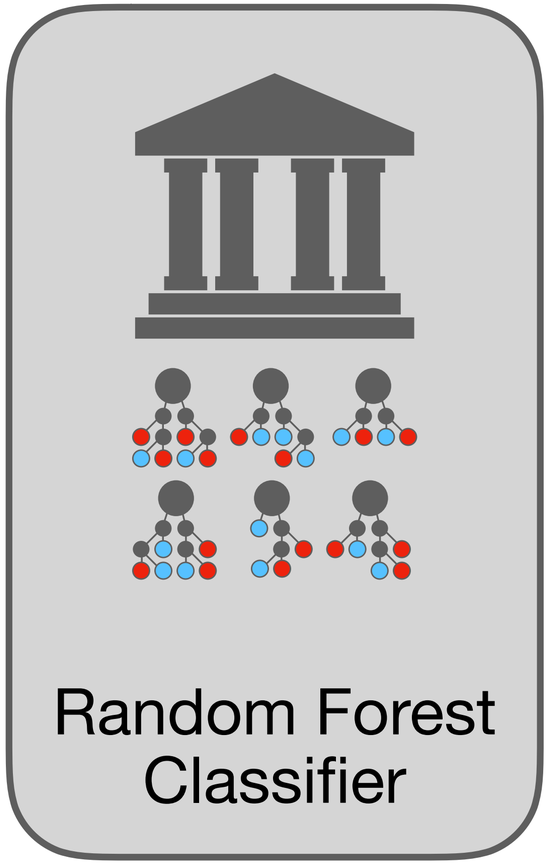
- Can be used just like a Decision Tree
- Builds different trees on the same dataset
- Doesn’t care about feature normalization
- Not a great classifier… Did we go too far?

Random Forest





Random Forest



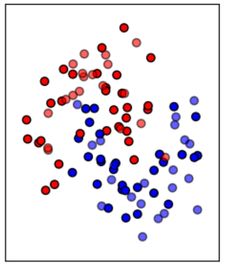
- No “curse of dimensionality”
- Resistant to noise
- Runs very fast
- Can estimate feature importance!
- Current sweetheart of the community

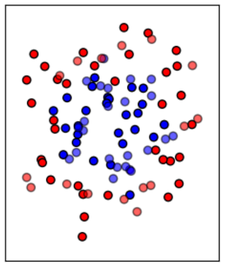
There are many more!

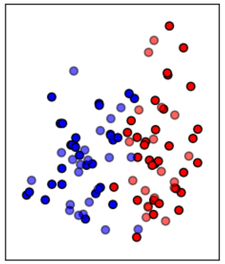
- Regression
- Logistic Regression
- Neural Networks
- Bayesian estimators

Comparisons

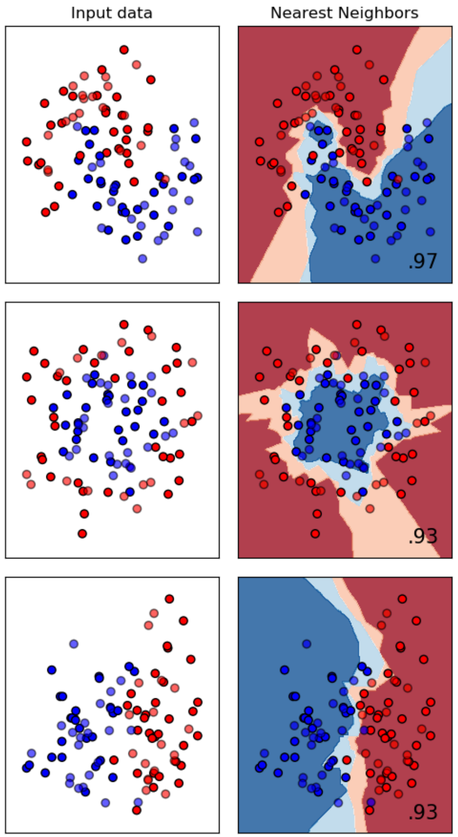
Input data



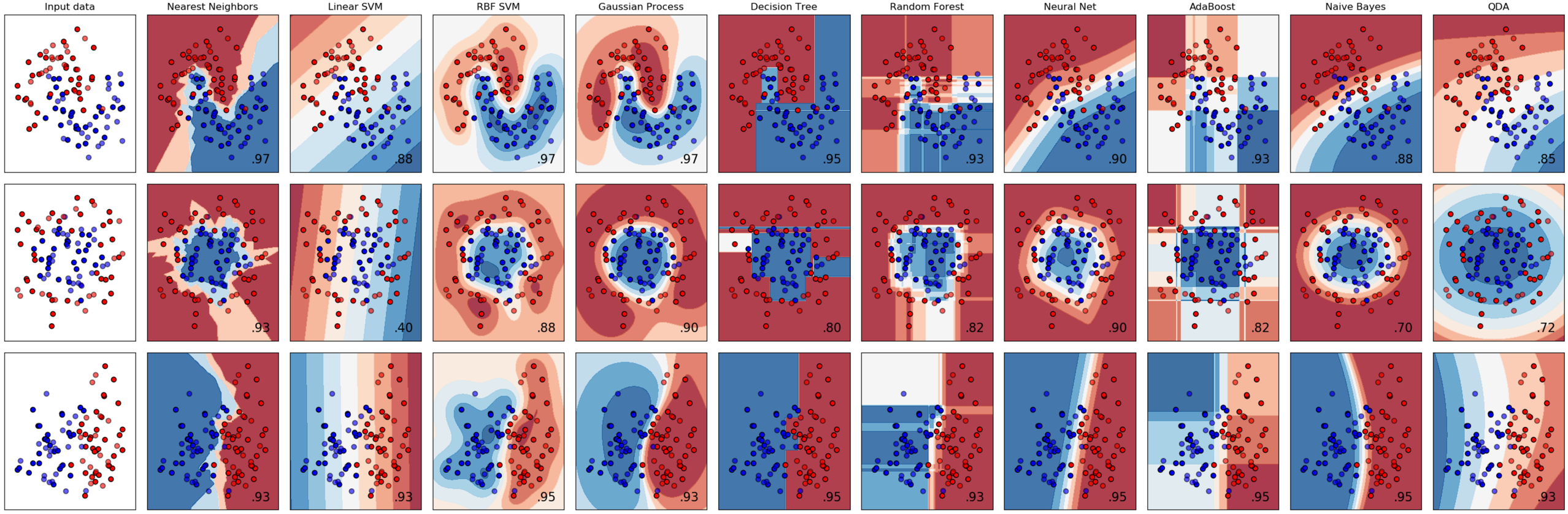




Comparisons



Comparisons

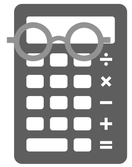


What the code looks like (really)

```
clf = KNeighborsClassifier(3)
clf.fit(X_train, y_train)
clf.predict(X_test)


clf = SVC(kernel="linear", C=0.025)
clf.fit(X_train, y_train)
clf.predict(X_test)


clf = SVC(gamma=2, C=1)
clf.fit(X_train, y_train)
clf.predict(X_test)


clf = GaussianProcessClassifier(1.0 * RBF(1.0))
clf.fit(X_train, y_train)
clf.predict(X_test)


clf = DecisionTreeClassifier(max_depth=5)
clf.fit(X_train, y_train)
clf.predict(X_test)


clf = RandomForestClassifier(max_depth=5, n_estimators=10, max_features=1)
clf.fit(X_train, y_train)
clf.predict(X_test)
```

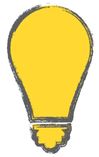
```
clf = MLPClassifier(alpha=1, max_iter=1000)
clf.fit(X_train, y_train)
clf.predict(X_test)


clf = AdaBoostClassifier()
clf.fit(X_train, y_train)
clf.predict(X_test)


clf = GaussianNB()
clf.fit(X_train, y_train)
clf.predict(X_test)


clf = QuadraticDiscriminantAnalysis()
clf.fit(X_train, y_train)
clf.predict(X_test)


clf = RandomForestClassifier()
clf.fit(X_train, y_train)
clf.predict(X_test)
```

Making data machine-learnable



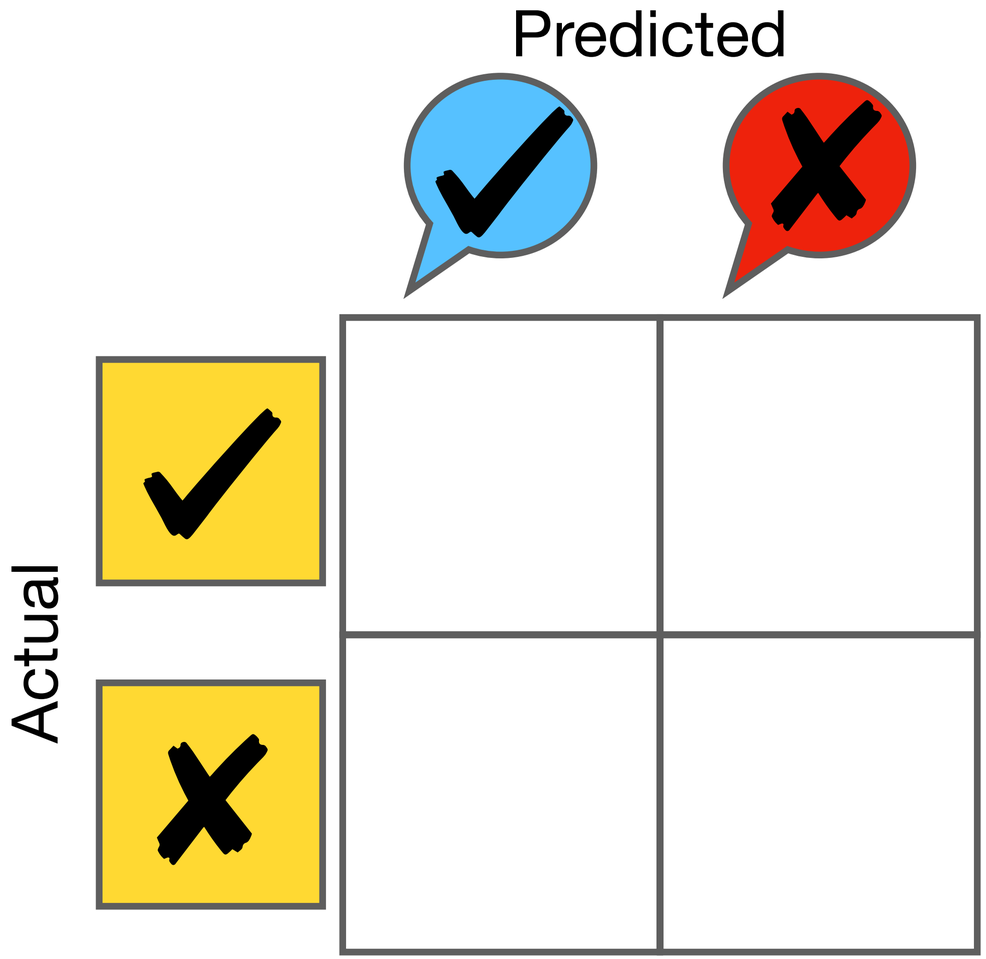
Picking a machine learning algorithm

Evaluating classifiers

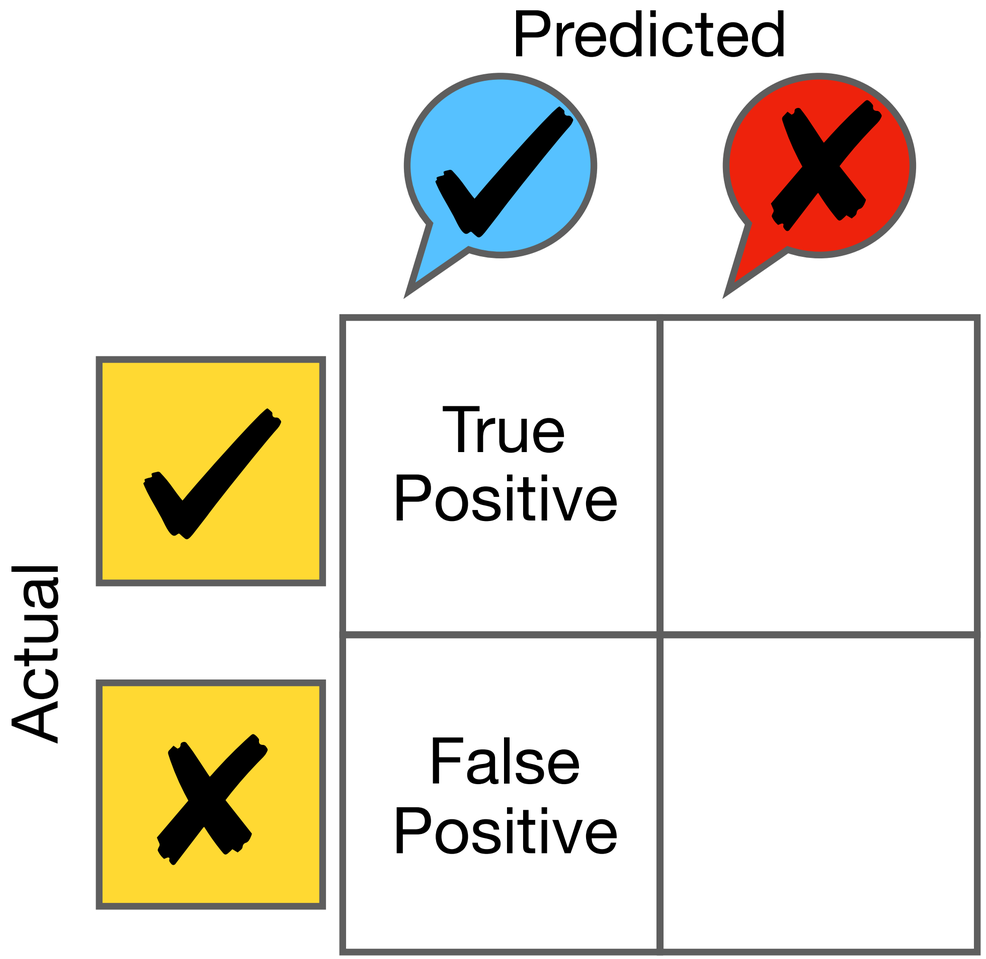


Applying this to your projects

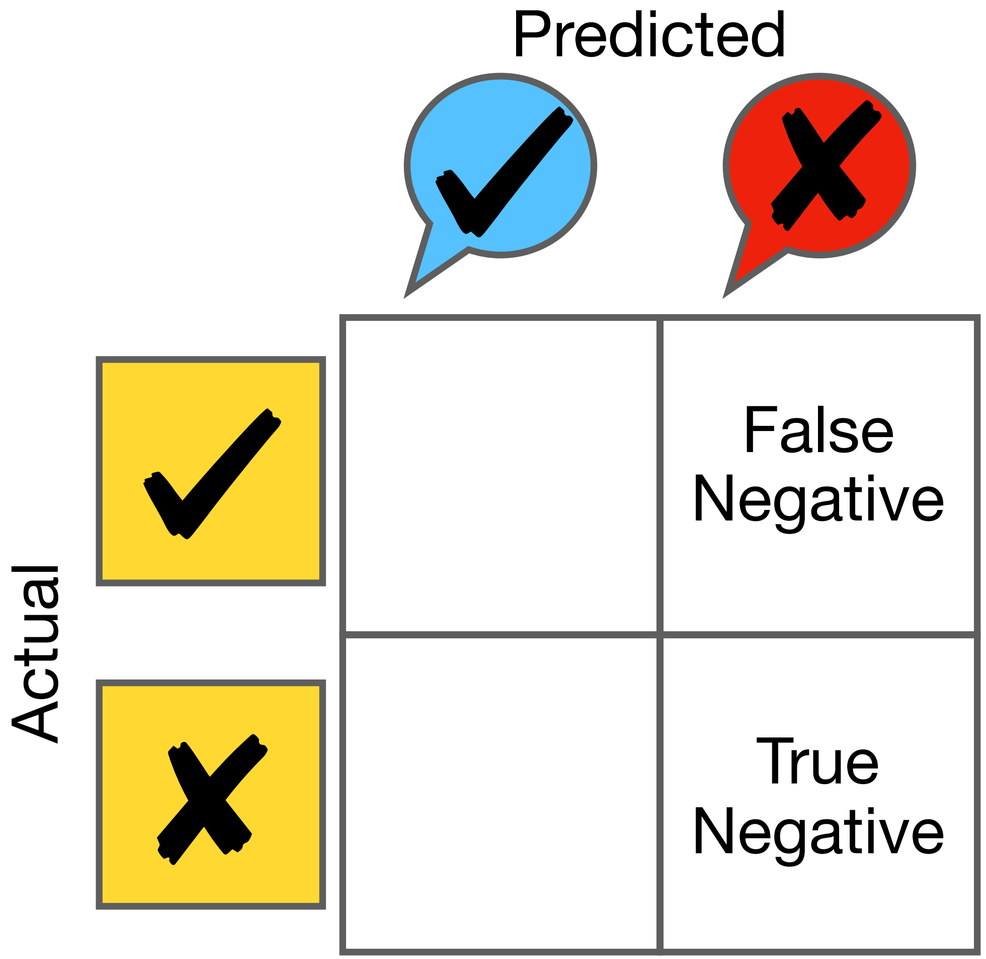
Two kinds of wrong, two kinds of right



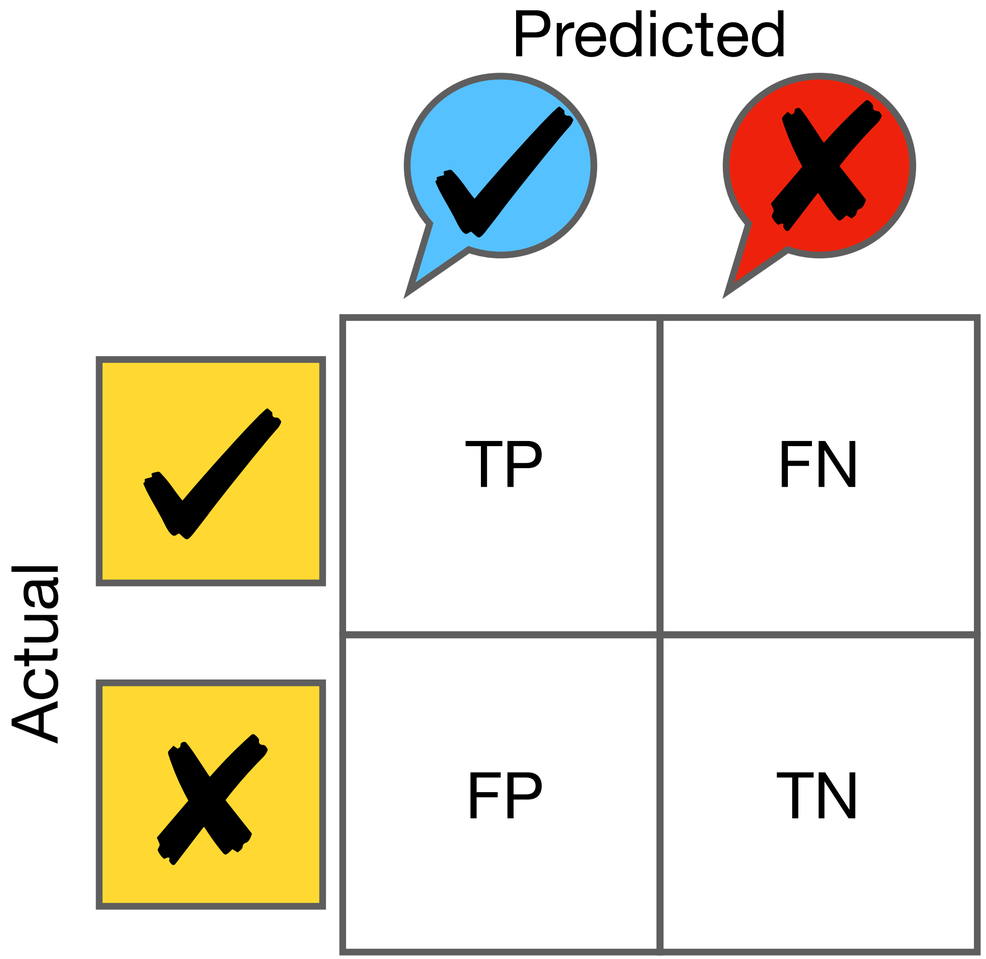
Two kinds of wrong, two kinds of right



Two kinds of wrong, two kinds of right



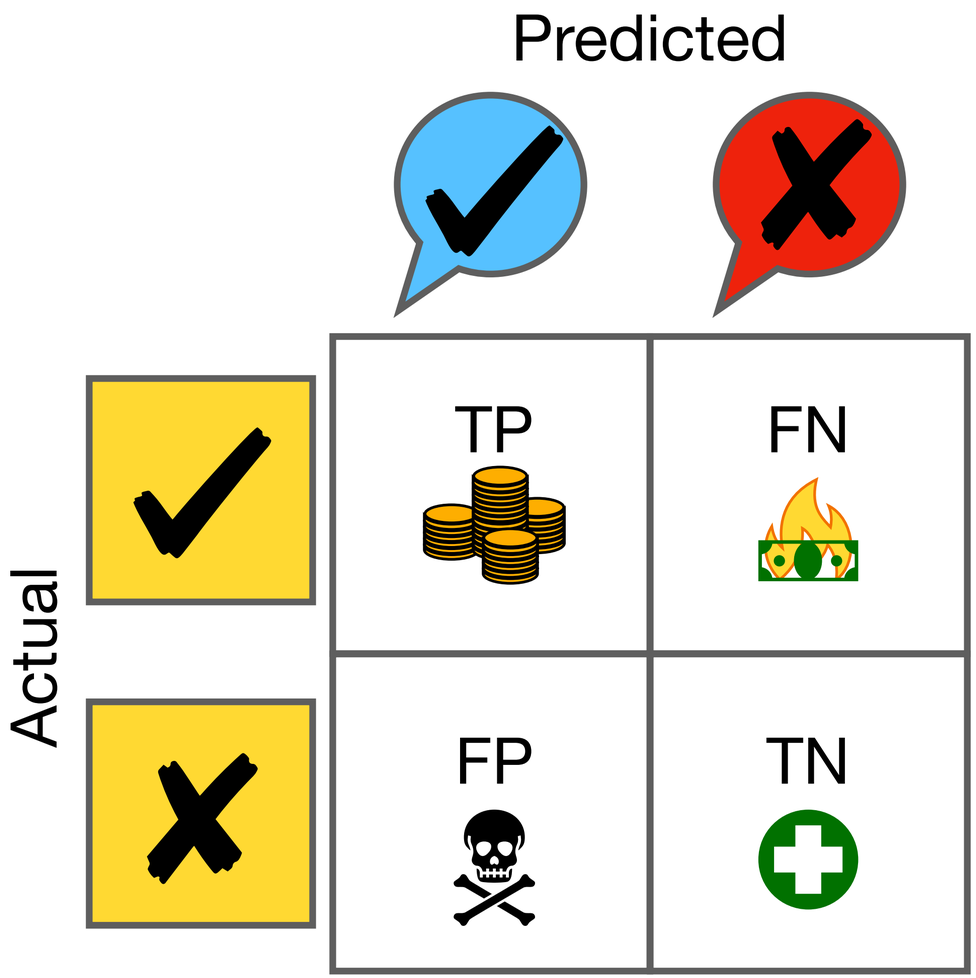
Two kinds of wrong, two kinds of right



Booby-trapped treasure example

✓ Safe, open it

✗ Dangerous, into the fire!



Base Rate

$$\frac{\checkmark}{\checkmark + \times}$$

Precision

$$\frac{TP}{TP + FP}$$

Specificity

$$\frac{TN}{TN + FP}$$

Accuracy

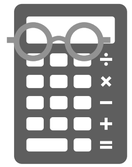
$$\frac{TP + TN}{\checkmark + \times}$$

Recall

$$\frac{TP}{TP + FN}$$

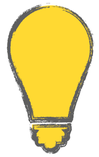
(Sensitivity)

Making data machine-learnable



Picking a machine learning algorithm

Evaluating classifiers



Applying this to your projects

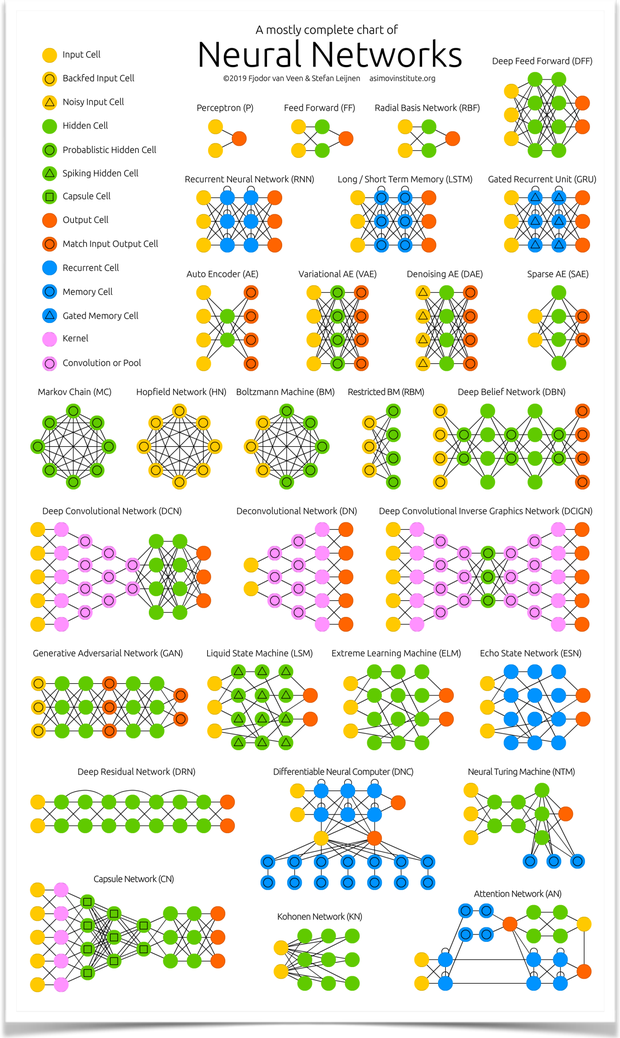
The first questions to ask

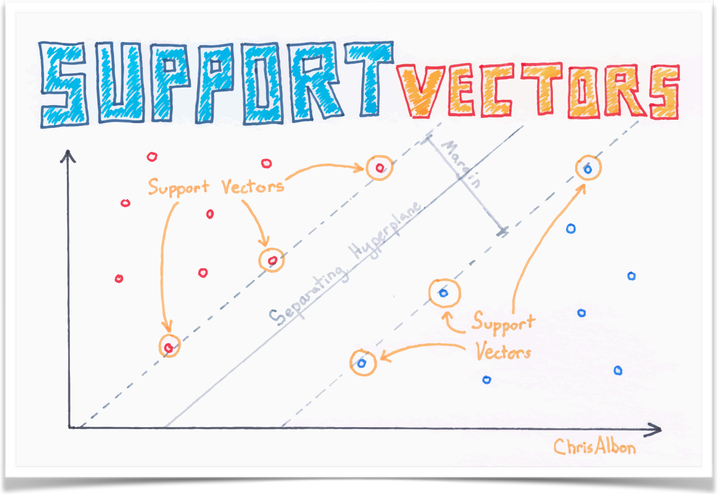
- What do I want to predict?
- What do I think I can use to predict it?
- **Why** do I want to predict this?
- How many samples do I have?
- What is my base rate?
- What would I consider a good prediction?

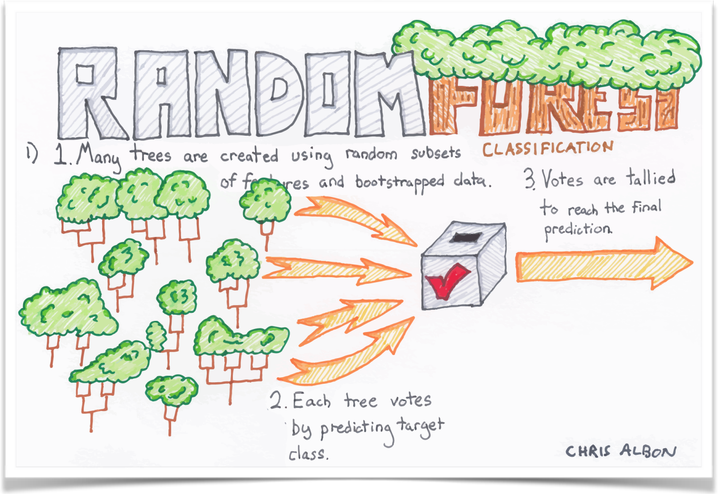
In cases with low data

- Possible pilot for further study
- Leave-one-out cross-validation
- Connect with public datasets
- Can you make a convincing dummy dataset?

There are variations and solutions for (almost) every problem

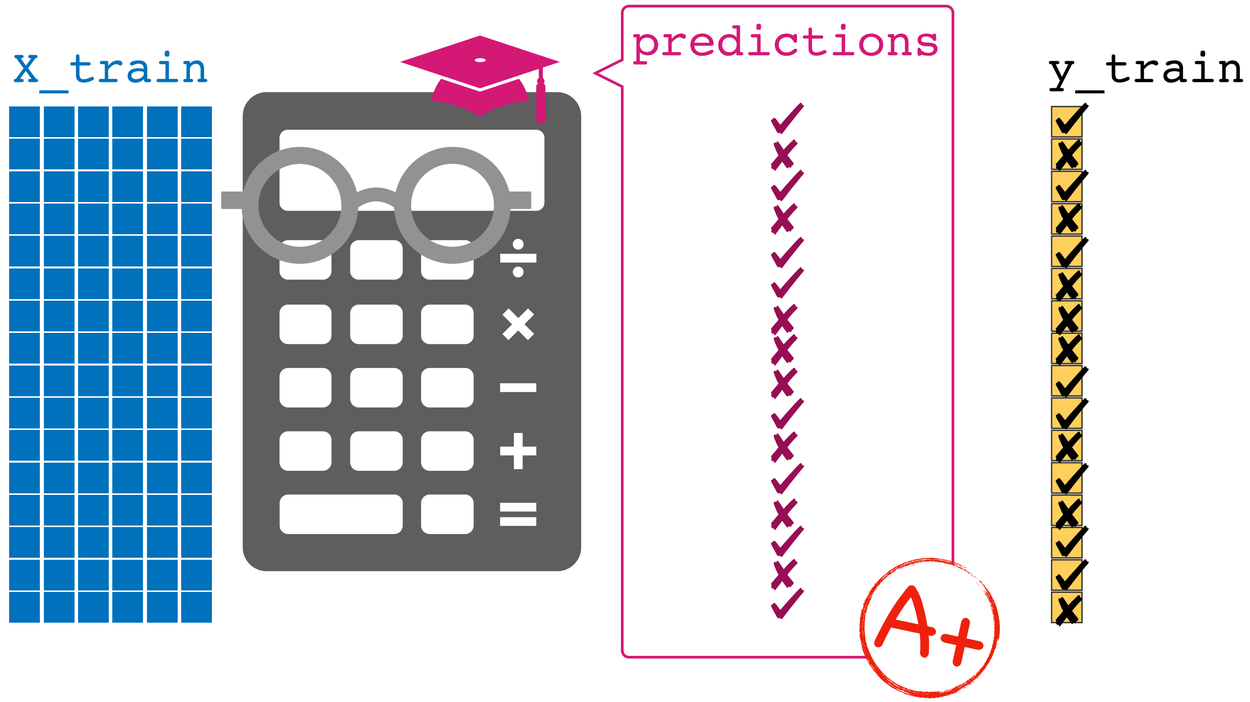






https://machinelearningflashcards.com/

Let’s talk about your data!



Thank you!

Juan Felipe Beltrán, Ph.D.
Cornell University Department of Biomedical Engineering
juanfelipe@cornell.edu
@offbyjuan

NYU Cardiovascular Research Center