大数据分析实践

Machine Learning I

Qiong Zeng (曾琼)

qiong.zn@sdu.edu.cn

Research *is creative and systematic work undertaken to increase the stock of knowledge.*

Does chocolate help with weight loss?

Naïve Approach

When all you have is a hammer…

Many people learn one algorithm (*e.g.*, regression) and always use it

Won’t lead to the best results!

Naïve Approach

When all you have is a hammer…

Many people learn one algorithm (*e.g.*, regression) and always use it

Different approaches fit different data better or worse

Naïve Approach

When all you have is a hammer…

Many people learn one algorithm (*e.g.*, regression) and always use it

Others aimlessly wander

- Machine learning takes too long for this to work well!
- Also may not be comprehensive

Naïve Approach

When all you have is a hammer…

Many people learn one algorithm (*e.g.*, regression) and always use it

Solution: PROCESS KNOWLEDGE

- *Know the capabilities and limitations of algorithms*
- *Generate hypotheses from the data*
- *Conduct careful error analysis when things go wrong*

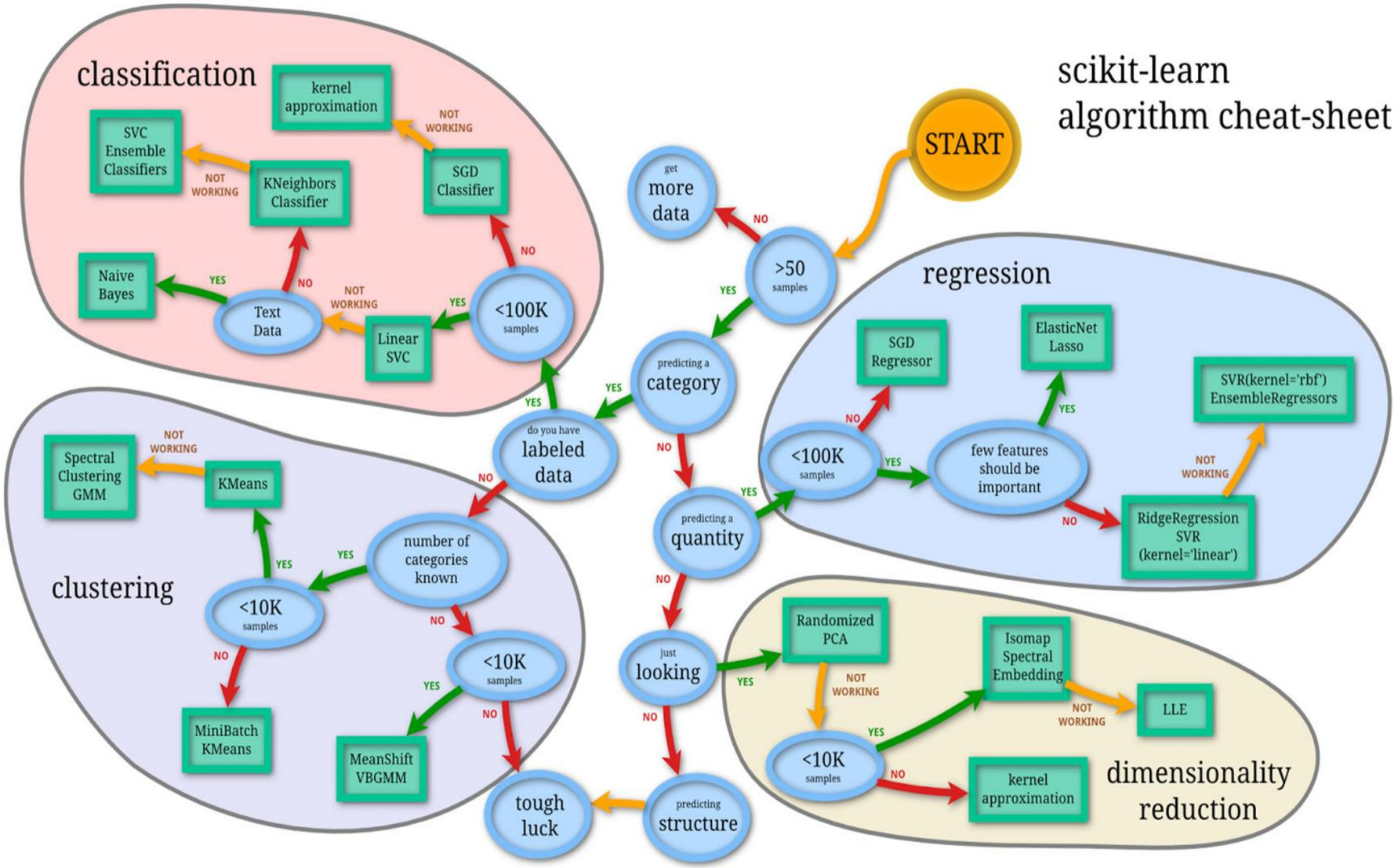
“Just enough to be dangerous” …

ML department devoted to this subject!
Practical start

- (Very) basic concepts
- A start on what questions to ask

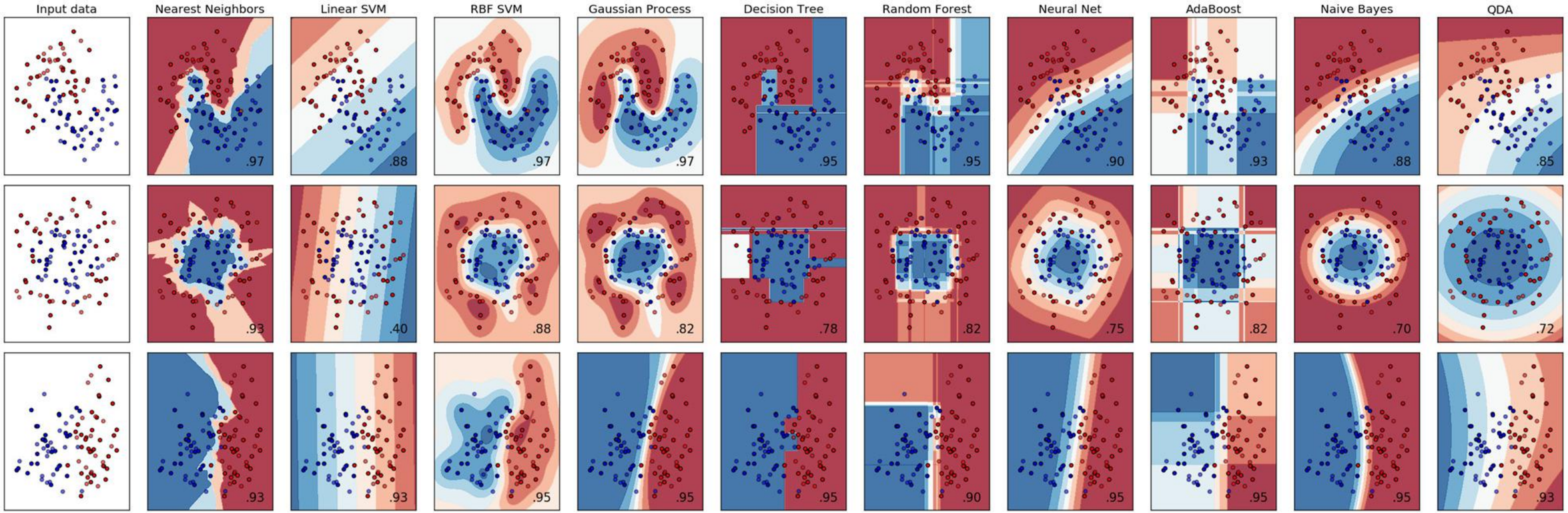
The goal in this class is not to teach ML, but to understand what it does and how to select the right tool for your data!

Selecting algorithms



http://www.quora.com/What-are-the-advantages-of-different-classification-algorithms

Selecting algorithms



http://www.quora.com/What-are-the-advantages-of-different-classification-algorithms

What is machine learning?

Automatically or *semi-automatically*

- Inducing concepts (*i.e.*, rules) from data
- Finding patterns in data
- Explaining data
- Making predictions

Two main approaches

- Supervised learning (we have lots of examples of what should be predicted)
- Unsupervised learning (e.g. clustering into groups and inferring what they are about)
- Can combine these (semi-supervised)
- Can learn over time or train up front

Example uses of ML

- Recognizing what appliance is being used from whole-house energy (“NILM”)
- Recognizing activities
- Predicting sewer overflows
- Figuring out which set of documents an email is similar to (could know about sets as ‘spam’ ‘not spam’ or just care about better search results)
- Recommending movies

Today's Focus

- ***Supervised* learning (we have lots of examples of what should be predicted)**
- *Unsupervised* learning (*e.g.* clustering into groups and inferring what they are about)
- Can combine these (*semi-supervised)* Can *learn over time* or ***train up front***

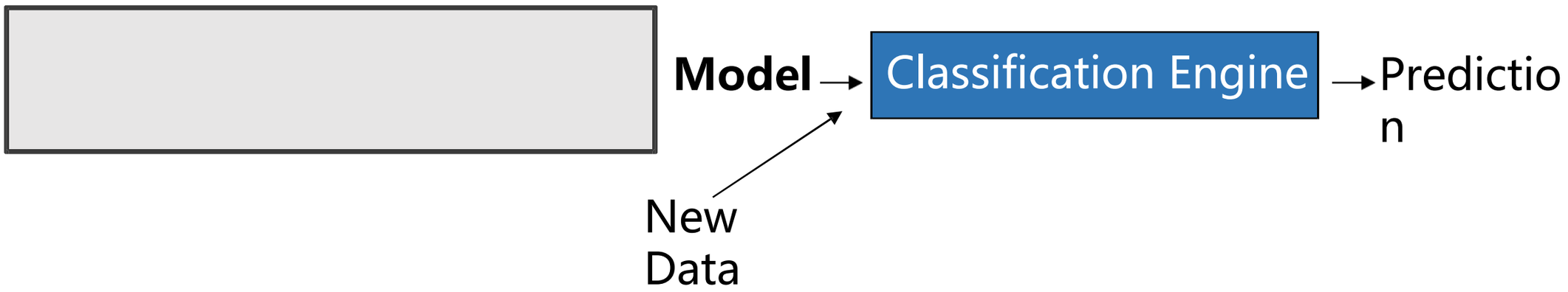
Typical Supervised Learning Flow

Steps 1： Training

Typical Supervised Learning Flow

Steps 1： Training

Steps 2： Prediction



Example: School in Session?

GIVEN “features” such as

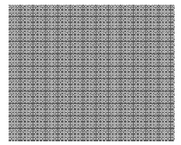
- Steeler’s Game Ad?
- Weather
- …

Example: School in Session?

GIVEN "features" such as

- Temperature
- Weather Date
- …

Class "A"



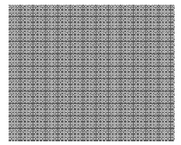
PREDICT School in Session (yes or no)

Example: School in Session?

GIVEN "features" such as

- Temperature
- Weather Date
- …

Class "B"



PREDICT School in Session (yes or no)

Example: School in Session?

GIVEN “features” such as

- Temperature
- Weather Date
- …

PREDICT School in Session (yes or no)

Typical approach: “Classification”

Classification

- Take a set of observed features
  F = <f1, f2, f3,…>
  (a feature vector)

- Use this to estimate which class it belongs to (a prediction)

Training data: Multiple Examples

outlook	temperature	humidity	windy	play
sunny	hot	high	FALSE	no
sunny	hot	high	TRUE	no
overcast	hot	high	FALSE	yes
rainy	mild	high	FALSE	yes
rainy	cool	normal	FALSE	yes
rainy	cool	normal	TRUE	no
overcast	cool	normal	TRUE	yes
sunny	mild	high	FALSE	no
sunny	cool	normal	FALSE	yes
rainy	mild	normal	FALSE	yes
sunny	mild	normal	TRUE	yes
overcast	mild	high	TRUE	yes
overcast	hot	normal	FALSE	yes
rainy	mild	high	TRUE	no

Including Features Features

outlook	temperature	humidity	windy	play
sunny	hot	high	FALSE	no
sunny	hot	high	TRUE	no
overcast	hot	high	FALSE	yes
rainy	mild	high	FALSE	yes
rainy	cool	normal	FALSE	yes
rainy	cool	normal	TRUE	no
overcast	cool	normal	TRUE	yes
sunny	mild	high	FALSE	no
sunny	cool	normal	FALSE	yes
rainy	mild	normal	FALSE	yes
sunny	mild	normal	TRUE	yes
overcast	mild	high	TRUE	yes
overcast	hot	normal	FALSE	yes
rainy	mild	high	TRUE	no

And Labels

outlook	temperature	humidity	windy	Class: play
sunny	hot	high	FALSE	no
sunny	hot	high	TRUE	no
overcast	hot	high	FALSE	yes
rainy	mild	high	FALSE	yes
rainy	cool	normal	FALSE	yes
rainy	cool	normal	TRUE	no
overcast	cool	normal	TRUE	yes
sunny	mild	high	FALSE	no
sunny	cool	normal	FALSE	yes
rainy	mild	normal	FALSE	yes
sunny	mild	normal	TRUE	yes
overcast	mild	high	TRUE	yes
overcast	hot	normal	FALSE	yes
rainy	mild	high	TRUE	no

How does classification work?

outlook	temperature	humidity	windy	play
sunny	hot	high	FALSE	no
sunny	hot	high	TRUE	no
overcast	hot	high	FALSE	yes
rainy	mild	normal	FALSE	yes
sunny	mild	normal	TRUE	yes
overcast	mild	high	TRUE	yes
overcast	hot	normal	FALSE	yes
rainy	mild	high	TRUE	no

Two simple algorithms

- 0R – Predict the majority class
- 1R – Use the most predictive single feature

What will the prediction be?

Model

Outlook: Sunny -> No
Overcast -> Yes Rainy-> Yes

New Data

outlook	temperature	humidity	windy	play
rainy	cool	high	FALSE	?

Yes

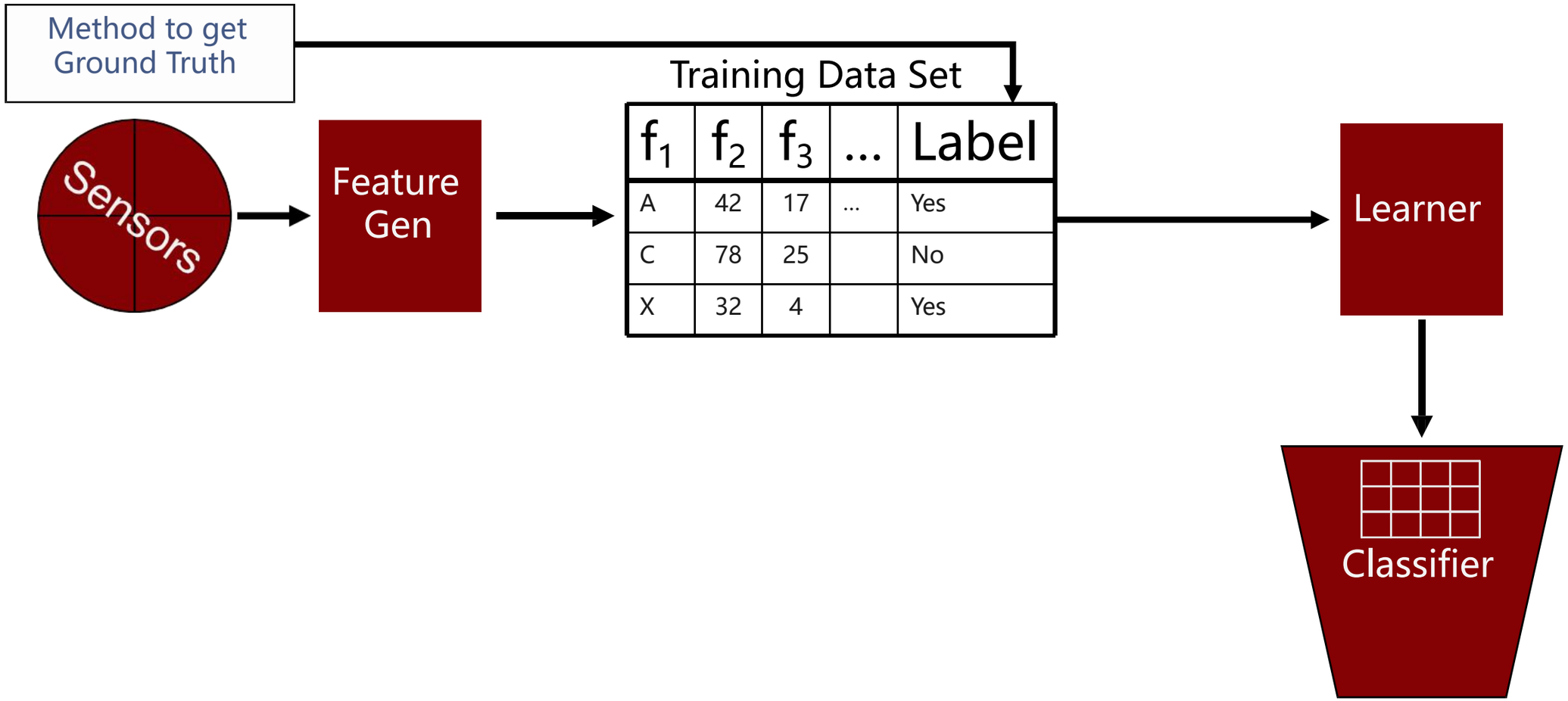
0R & 1R Classification

Problem: Brittle and error prone

Instead, ML classifiers use *statistical models* to learn from past data

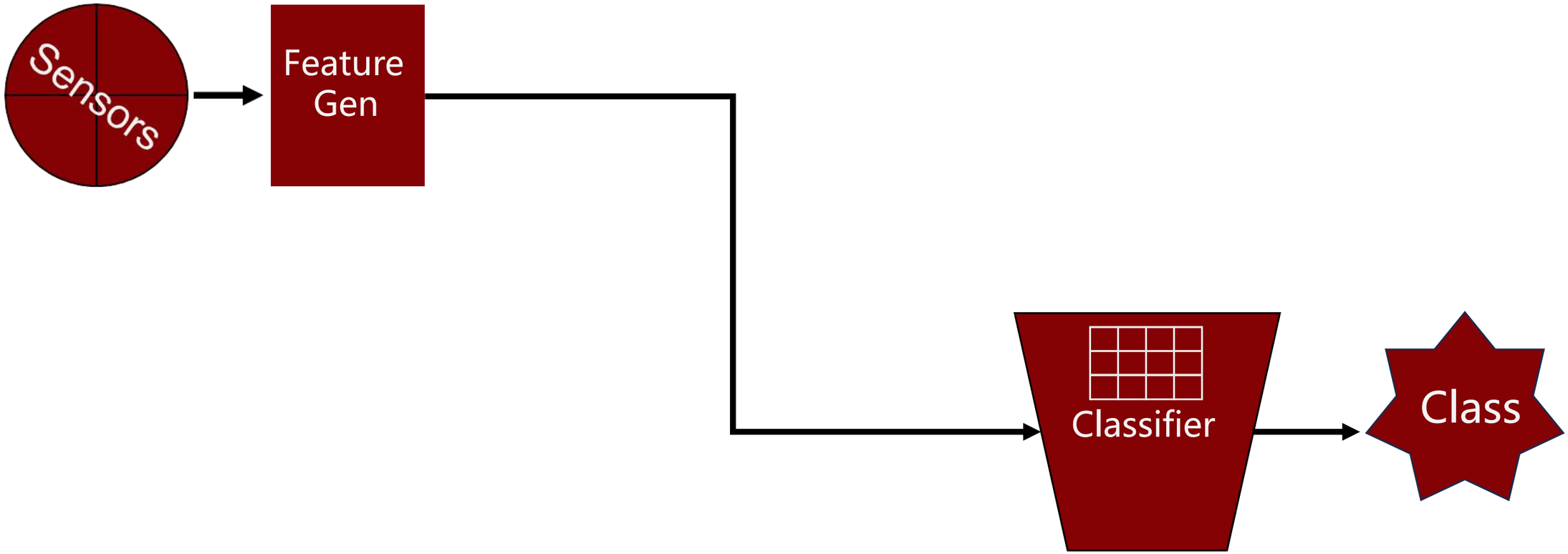
Learned Classifiers

Training Time

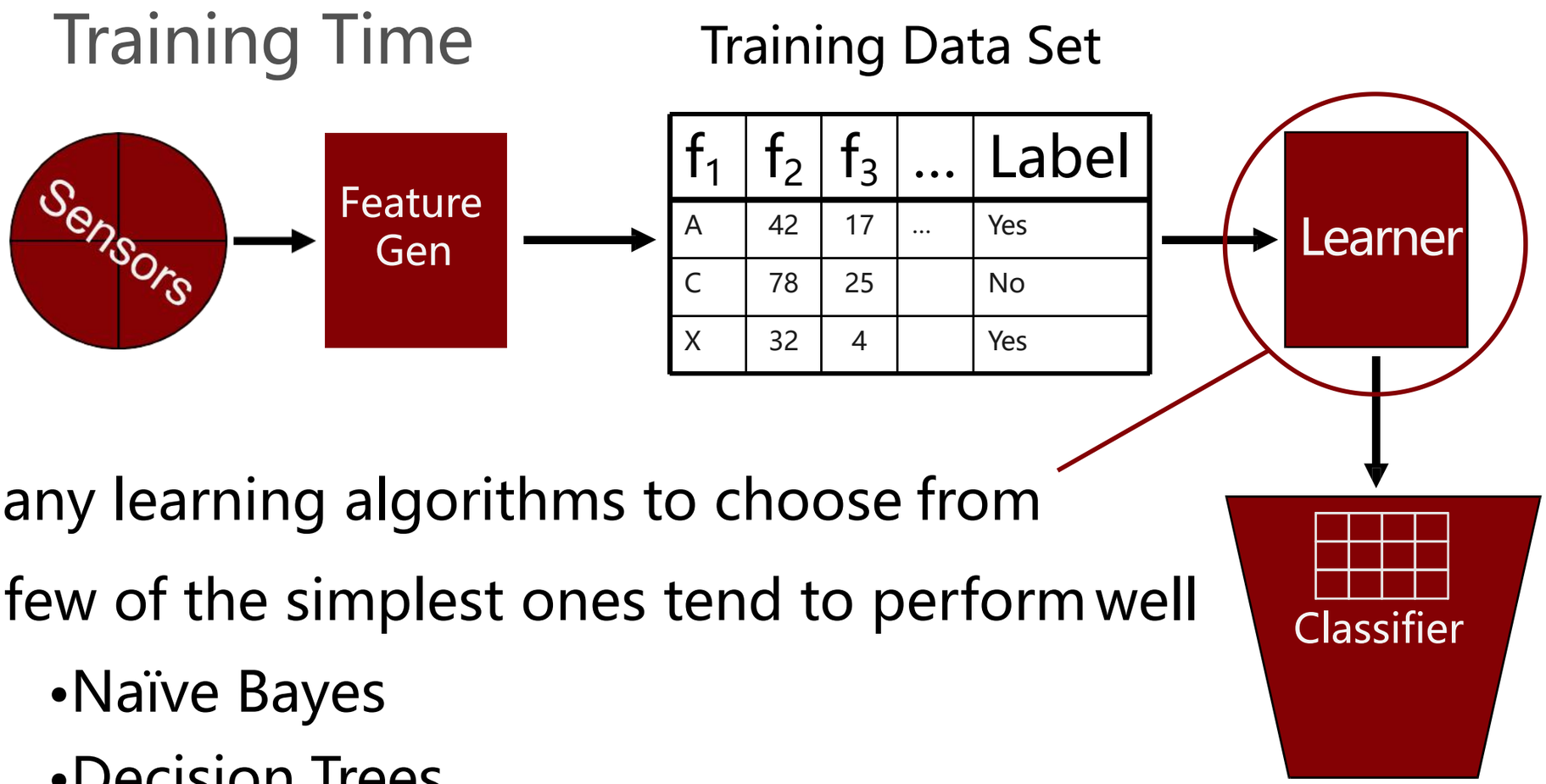


Learned Classifiers

Run/Classify Time



Learned Classifiers

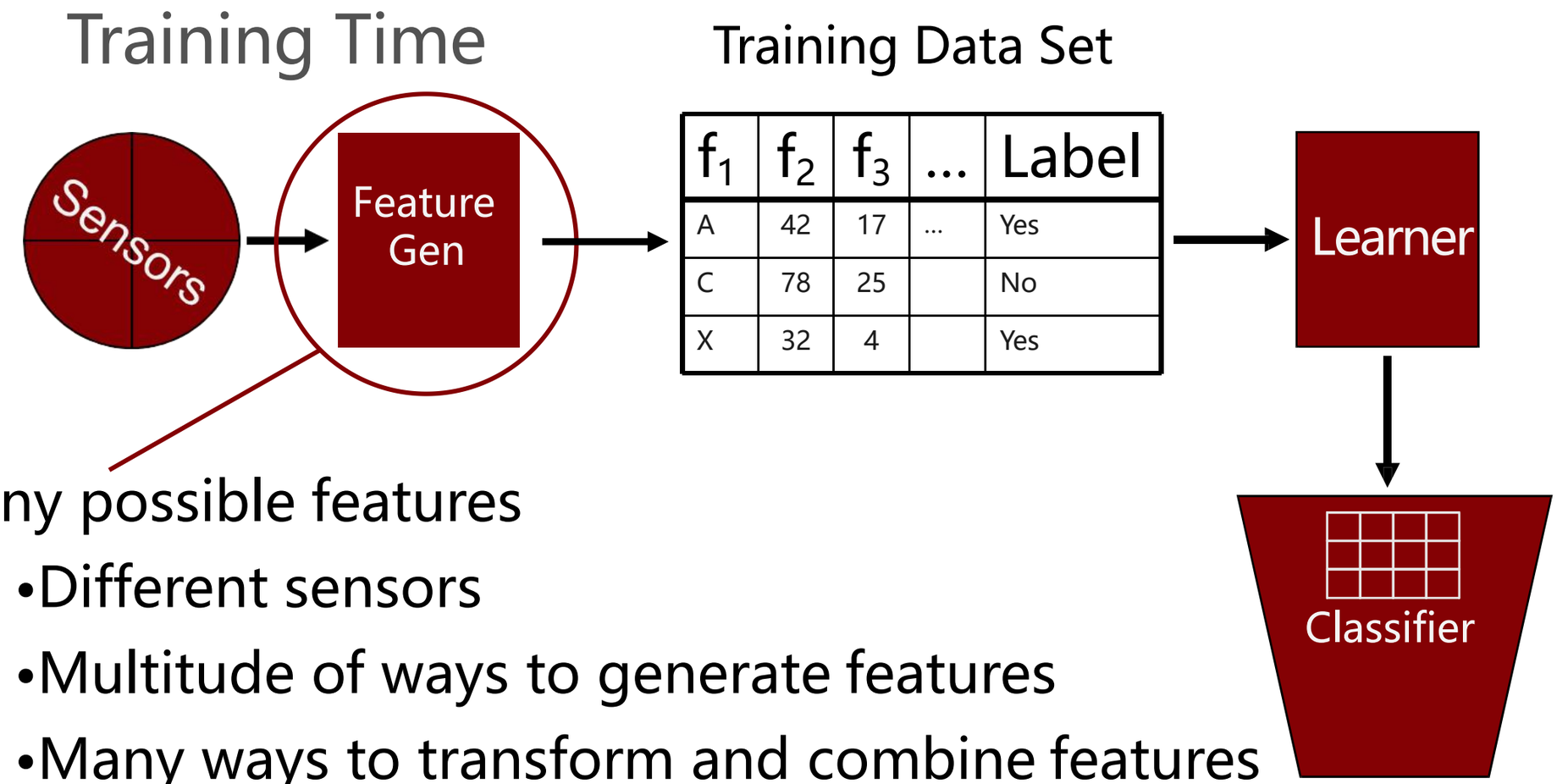


Many learning algorithms to choose from

A few of the simplest ones tend to perform well

- Naïve Bayes
- Decision Trees
- Regression-based Models (for continuous)

Learned Classifiers



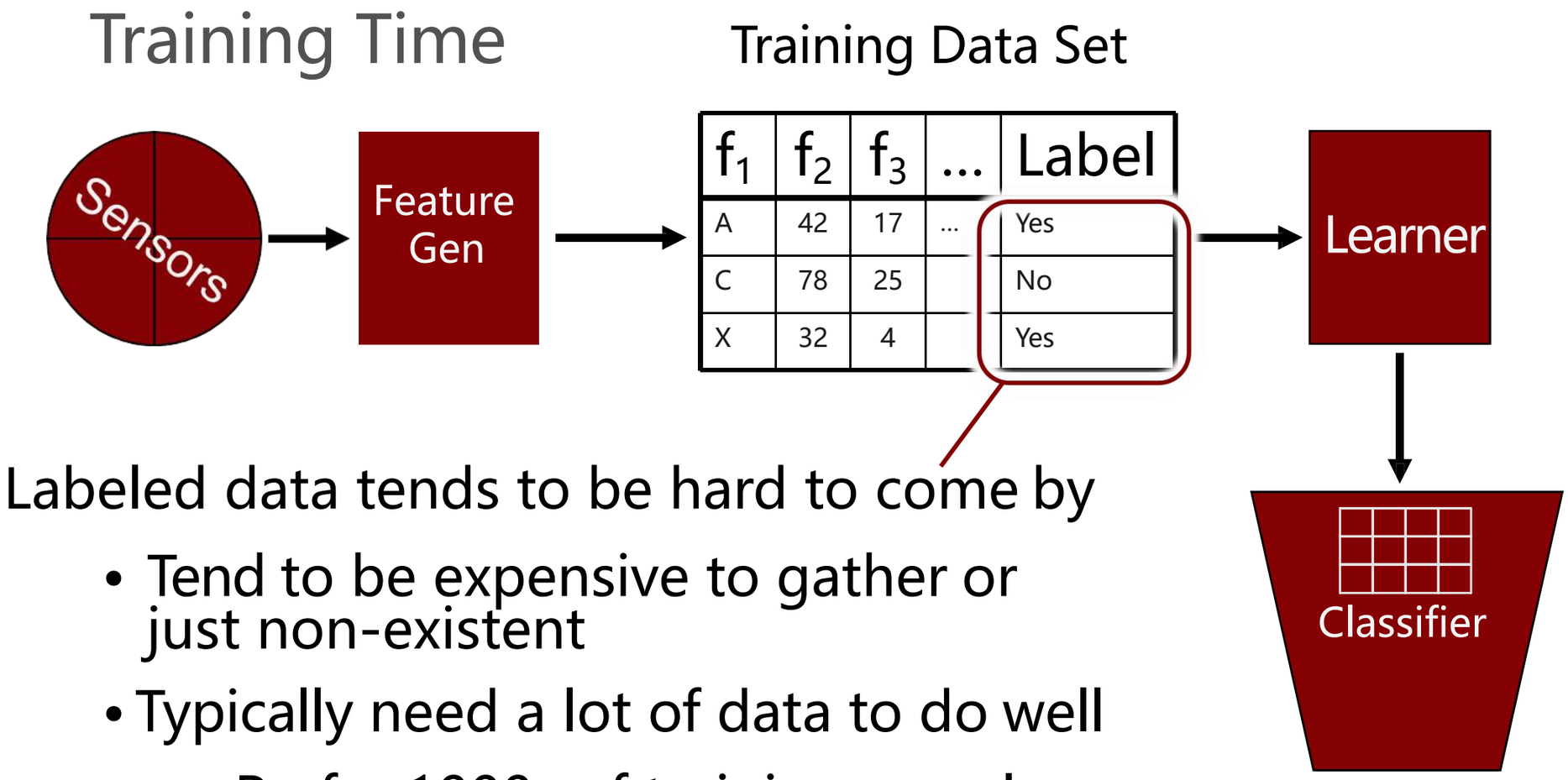
Many possible features

- Different sensors
- Multitude of ways to generate features
- Many ways to transform and combine features

Some much more predictive / useful than others

➔ "Feature selection" tends to be very important

Learned Classifiers



Labeled data tends to be hard to come by

- Tend to be expensive to gather or just non-existent
- Typically need a lot of data to do well
  - Prefer 1000s of training samples

How can I tell if my classifier is any good?

Common measure: Accuracy

X% of the time, the recognizer was right

How can I tell if my classifier is any good?

Common measure: Accuracy

X% of the time, the recognizer was right

But our data actually looks something like:

	True Class : Sleeping	True Class: Biking
Predicted Class: Sleeping	90	15
Predicted Class: Biking	12	10

How can I tell if my classifier is any good?

Accuracy: (90+10)/(90+15+12+10) = 79%

What about all the other information here?

	True Class : Sleeping	True Class: Biking
Predicted Class: Sleeping	90 (True Positive)	15
Predicted Class: Biking	12	10 (True Negative)

Alternative takes on Accuracy

$$\text{Precision} = TP / (TP + FP) = 90/(90+15) = 86\%$$

$$\text{Recall} = TP / (TP + FN) = 90/(90+12) = 88\%$$

$$\text{F-Score} = 2 * \frac{(\text{Precision} * \text{Recall})}{(\text{Precision} + \text{Recall})} = 87\%$$

	True Class : Sleeping	True Class: Biking
Predicted Class: Sleeping	90 (True Positive)	15 (False positive)
Predicted Class: Biking	12 (False positive)	10 (True Negative)

Kappa: Accounting for Skew

$$\text{Kappa} = \frac{(\text{observed acc.} - \text{expected acc.})}{(1 - \text{expected acc.})}$$

	True Class : Sleeping	True Class: Biking
Predicted Class: Sleeping	90 (True Positive)	15 (False positive)
Predicted Class: Biking	12 (False positive)	10 (True Negative)

Kappa: Accounting for Skew

$$\text{Kappa} = \frac{(\text{observed acc.} - \text{expected acc.})}{(1 - \text{expected acc.})}$$

Observed acc. = .79

	True Class : Sleeping	True Class: Biking
Predicted Class: Sleeping	90 (True Positive)	15 (False positive)
Predicted Class: Biking	12 (False positive)	10 (True Negative)

Kappa: Accounting for Skew

$$\text{Kappa} = \frac{(\text{observed acc.} - \text{expected acc.})}{(1 - \text{expected acc.})}$$

Expected acc. E[activity]

need to compare ground truth to predictions

	True Class : Sleeping	True Class: Biking
Predicted Class: Sleeping	90 (True Positive)	15 (False positive)
Predicted Class: Biking	12 (False positive)	10 (True Negative)

Kappa: Accounting for Skew

$$\text{Kappa} = \frac{(\text{observed acc.} - \text{expected acc.})}{(1 - \text{expected acc.})}$$

$$\text{Expected acc. E[sleep]} = \frac{F_{obs}(s) * F_{pred}(s)}{N}$$

Kappa: Accounting for Skew

$$\text{Kappa} = \frac{(\text{observed acc.} - \text{expected acc.})}{(1 - \text{expected acc.})}$$

$$\text{Expected acc. E[sleep]} = \frac{F_{obs}(s) * F_{pred}(s)}{N}$$

Sum of all observed instances in the s (sleep) class = 90 + 12 = 102

Kappa: Accounting for Skew

$$\text{Kappa} = \frac{(\text{observed acc.} - \text{expected acc.})}{(1 - \text{expected acc.})}$$

$$\text{Expected acc. E[sleep]} = \frac{F_{obs}(s) * F_{pred}(s)}{N}$$

Sum of all observed instances in the s (sleep) class = 90 + 12 = 102

Sum of all predicted instances in the s (sleep) class = 90 + 15 = 105

Kappa: Accounting for Skew

$$\text{Kappa} = \frac{(\text{observed acc.} - \text{expected acc.})}{(1 - \text{expected acc.})}$$

$$\text{Expected acc. E[sleep]} = \frac{102 * 105}{127} = 84.33$$

Kappa: Accounting for Skew

$$\text{Kappa} = \frac{(\text{observed acc.} - \text{expected acc.})}{(1 - \text{expected acc.})}$$

$$\text{Expected acc. E[activity]} = \frac{\text{E[sleeping]} + \text{E[Biking]}}{N}$$

$$= (84.33 + 4.33) / 127$$
$$= .70$$

Kappa: Accounting for Skew

$$\text{Kappa} = \frac{(.79 - .70)}{(1 - .70)} = .3\ (\textit{terrible!})$$

Observed acc. = .79

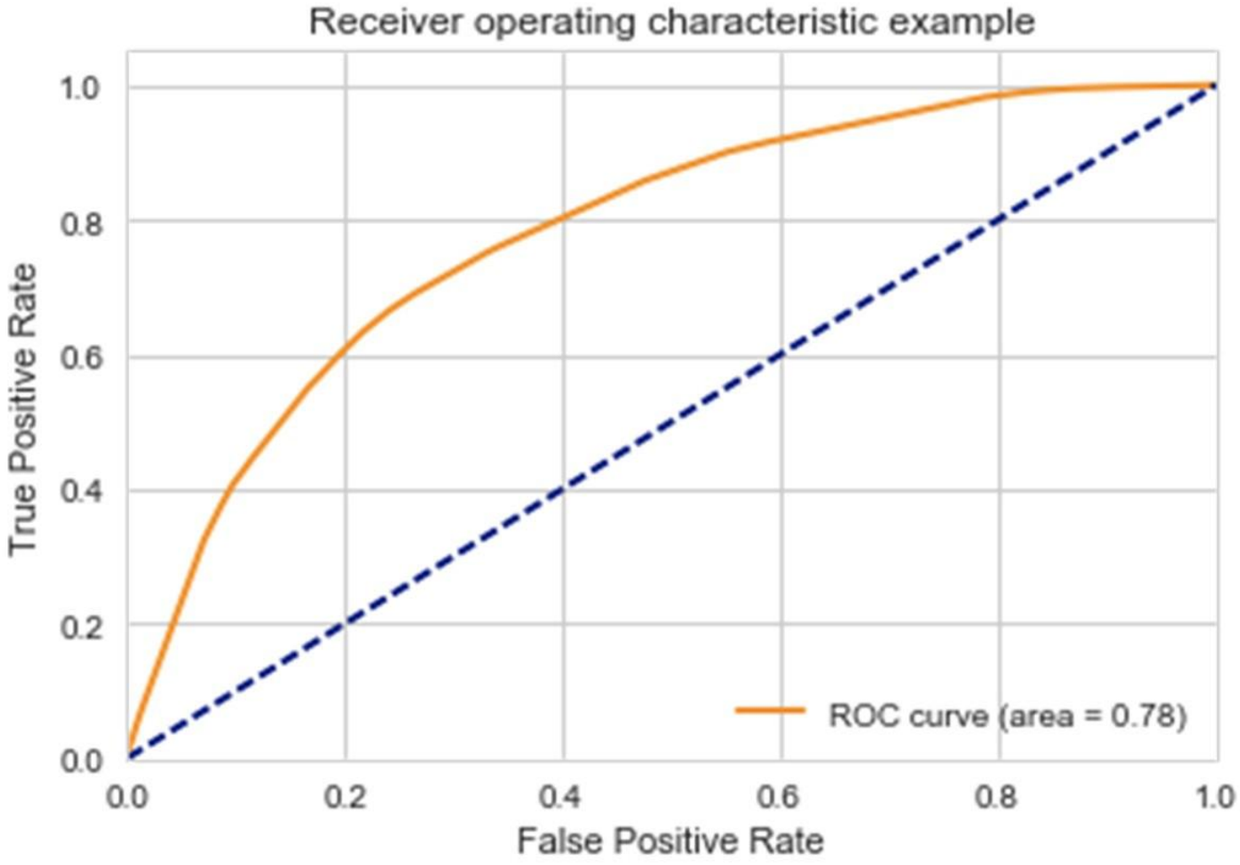
Expected acc. E[activity] = .70

	True Class : Sleeping	True Class: Biking
Predicted Class: Sleeping	90 (True Positive)	15 (False positive)
Predicted Class: Biking	12 (False positive)	10 (True Negative)

ROC Curve

Impact of probabilities for choosing classes

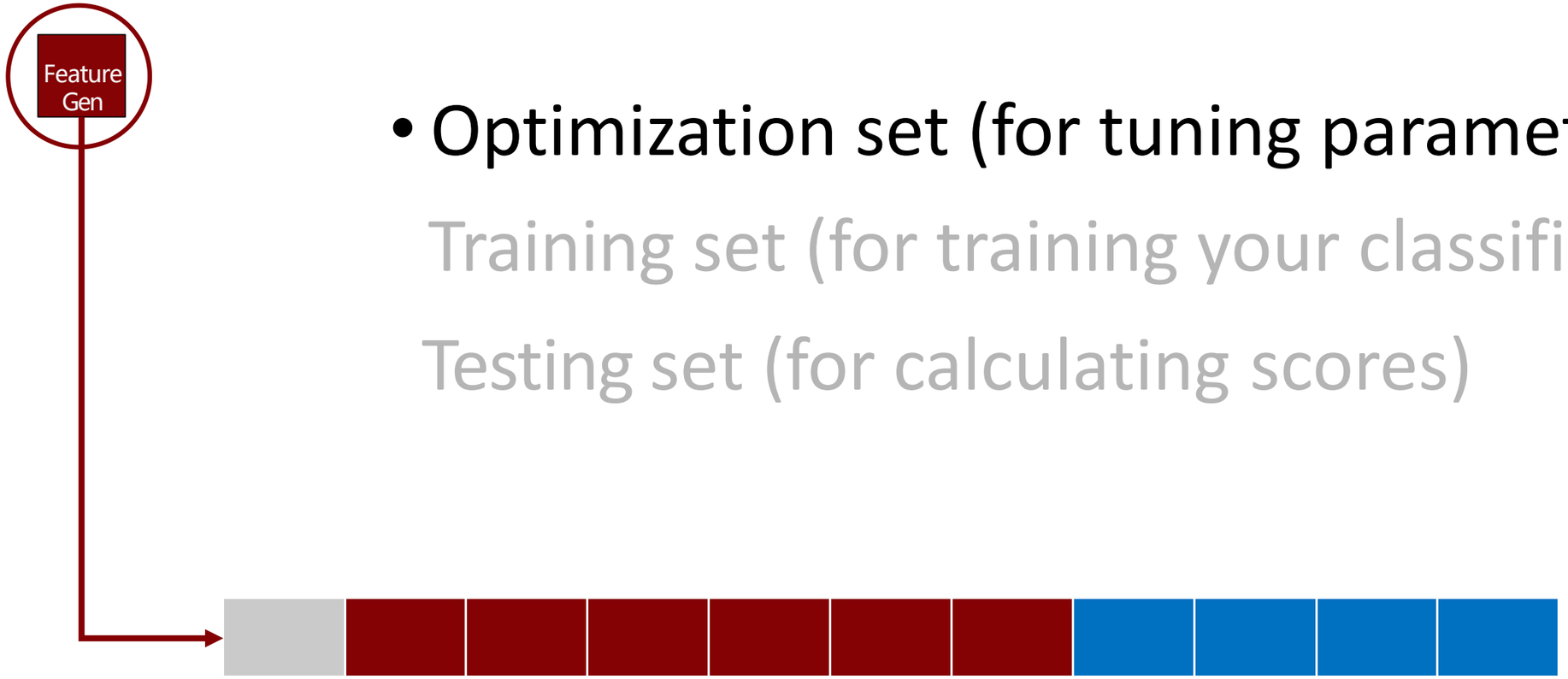
Fogarty, James, Ryan S. Baker, and E. Hudson. "Case studies in the use of ROC curve analysis for sensor-based estimates in human computer interaction." *GI 2005*.



51

0/25/2018

Best Approach



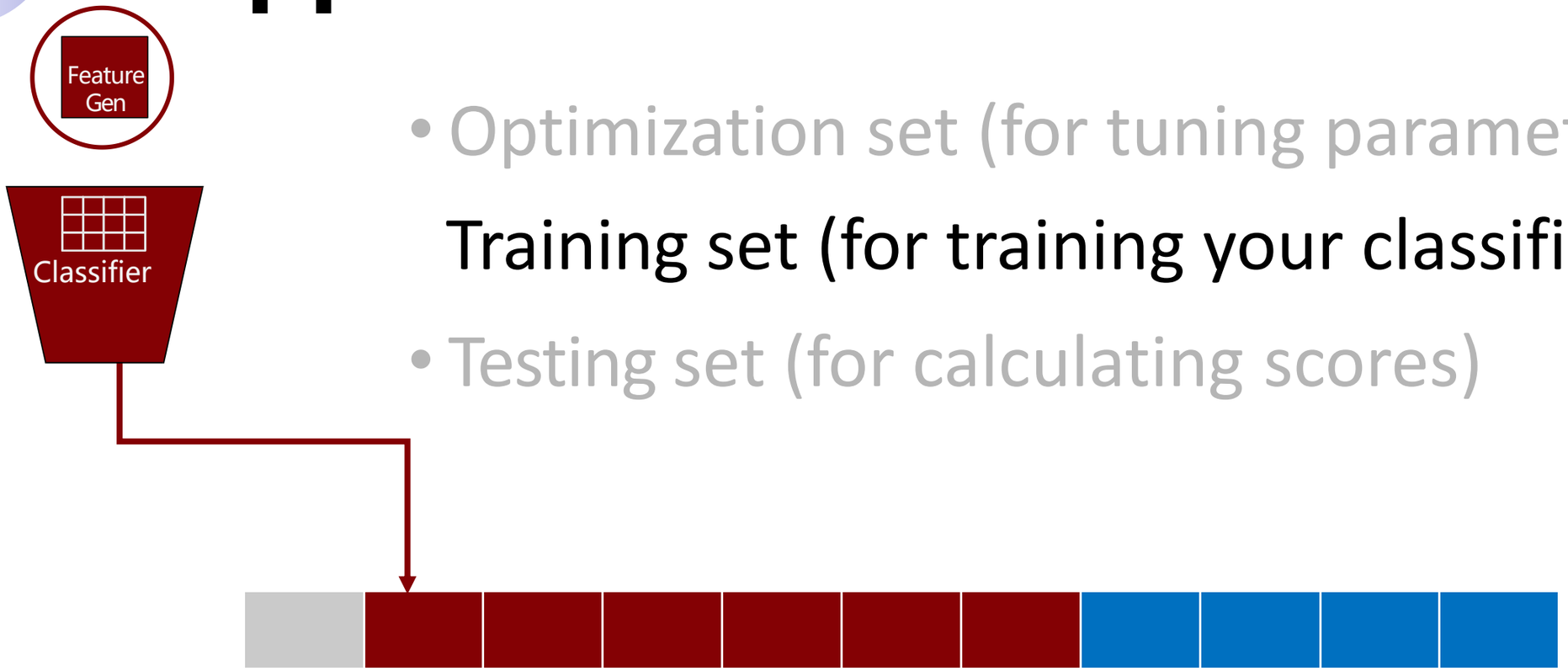
- Optimization set (for tuning parameters; *etc.*)

Training set (for training your classifier)

Testing set (for calculating scores)

Read your data. Explore it. Try things out (feature sets, algthm parameters). But do it *all* on the optimization set.

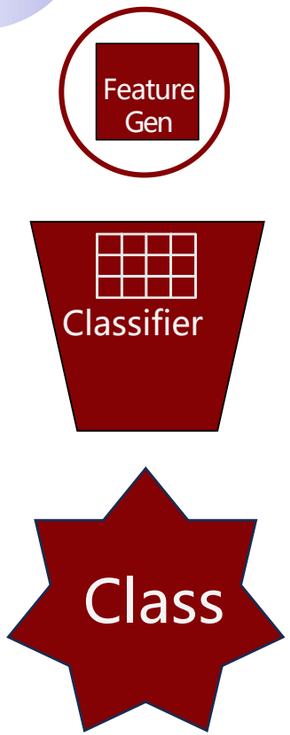
Best Approach



- Optimization set (for tuning parameters; *etc.*)
- Training set (for training your classifier)
- Testing set (for calculating scores)

Use this for training. Produces a classifier.

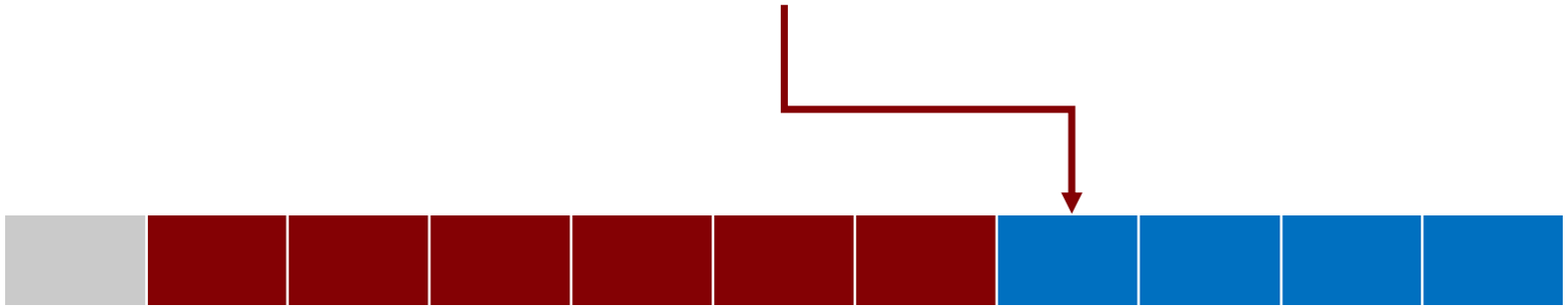
Best Approach



- Optimization set (for tuning parameters; *etc.*)

  Training set (for training your classifier)

- Testing set (for calculating scores)



Eventually you will run the classifier on ‘real world’ (unlabeled) data. But to know how well it works, you need to run it on data.

outlook	temperature	humidity	windy	play
sunny	hot	high	FALSE	no
sunny	hot	high	TRUE	no
overcast	hot	high	FALSE	yes
rainy	mild	high	FALSE	yes
rainy	cool	normal	FALSE	yes
rainy	cool	normal	TRUE	no
overcast	cool	normal	TRUE	yes
sunny	mild	high	FALSE	no
sunny	cool	normal	FALSE	yes
rainy	mild	normal	FALSE	yes
sunny	mild	normal	TRUE	yes
overcast	mild	high	TRUE	yes
overcast	hot	normal	FALSE	yes
rainy	mild	high	TRUE	no

outlook	temperature	humidity	windy	play
sunny	hot	high	FALSE	no
sunny	hot	high	TRUE	no
overcast	hot	high	FALSE	yes
rainy	mild	high	FALSE	yes
rainy	cool	normal	FALSE	yes
rainy	cool	normal	TRUE	no
overcast	cool	normal	TRUE	yes
sunny	mild	high	FALSE	no
sunny	cool	normal	FALSE	yes
rainy	mild	normal	FALSE	yes
sunny	mild	normal	TRUE	yes
overcast	mild	high	TRUE	yes
overcast	hot	normal	FALSE	yes
rainy	mild	high	TRUE	no

Train

Test

If Outlook = sunny, no
else if Outlook = overcast, yes
else if Outlook = rainy and Windy = TRUE, no else yes

Performance on training data?

outlook	temperature	humidity	windy	play
sunny	hot	high	FALSE	no
sunny	hot	high	TRUE	no
overcast	hot	high	FALSE	yes
rainy	mild	high	FALSE	yes
rainy	cool	normal	FALSE	yes
rainy	cool	normal	TRUE	no
overcast	cool	normal	TRUE	yes
sunny	mild	high	FALSE	no
sunny	cool	normal	FALSE	yes
rainy	mild	normal	FALSE	yes
sunny	mild	normal	TRUE	yes
overcast	mild	high	TRUE	yes
overcast	hot	normal	FALSE	yes
rainy	mild	high	TRUE	no

If Outlook = sunny, no
else if Outlook = overcast, yes
else if Outlook = rainy and Windy = TRUE, no else yes

Performance on testing data?

outlook	temperature	humidity	windy	play
sunny	hot	high	FALSE	no
sunny	hot	high	TRUE	no
overcast	hot	high	FALSE	yes
rainy	mild	high	FALSE	yes
rainy	cool	normal	FALSE	yes
rainy	cool	normal	TRUE	no
overcast	cool	normal	TRUE	yes
sunny	mild	high	FALSE	no
sunny	cool	normal	FALSE	yes
rainy	mild	normal	FALSE	yes
sunny	mild	normal	TRUE	yes
overcast	mild	high	TRUE	yes
overcast	hot	normal	FALSE	yes
rainy	mild	high	TRUE	no

If Outlook = sunny, no
else if Outlook = overcast, yes
else if Outlook = rainy and Windy = TRUE, no else yes

outlook	temperature	humidity	windy	play
sunny	hot	high	FALSE	no
rainy	mild	high	TRUE	no

IMPORTANT!

If you evaluate the performance of your rule on the same data you trained on, you won't get an accurate estimate of how well it will do on new data.

Best Approach

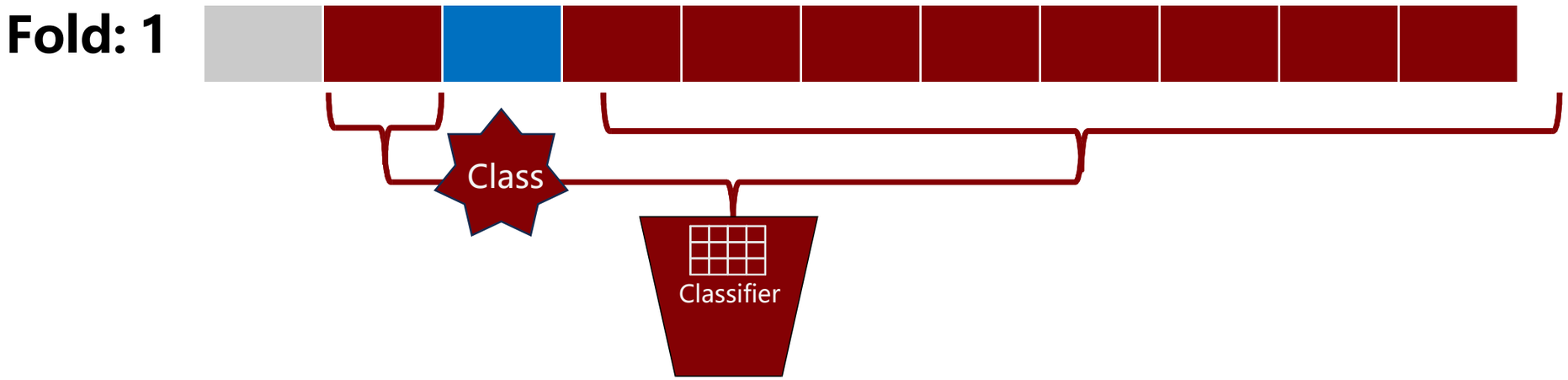
- Optimization set (for tuning parameters; *etc.*) Training set (for training your classifier) Testing set (for calculating scores)

- **In practice: Not enough labeled data!**
- *Never* give up on the optimization set, just make it smaller

Cross Validation Helps for Small Data Sets

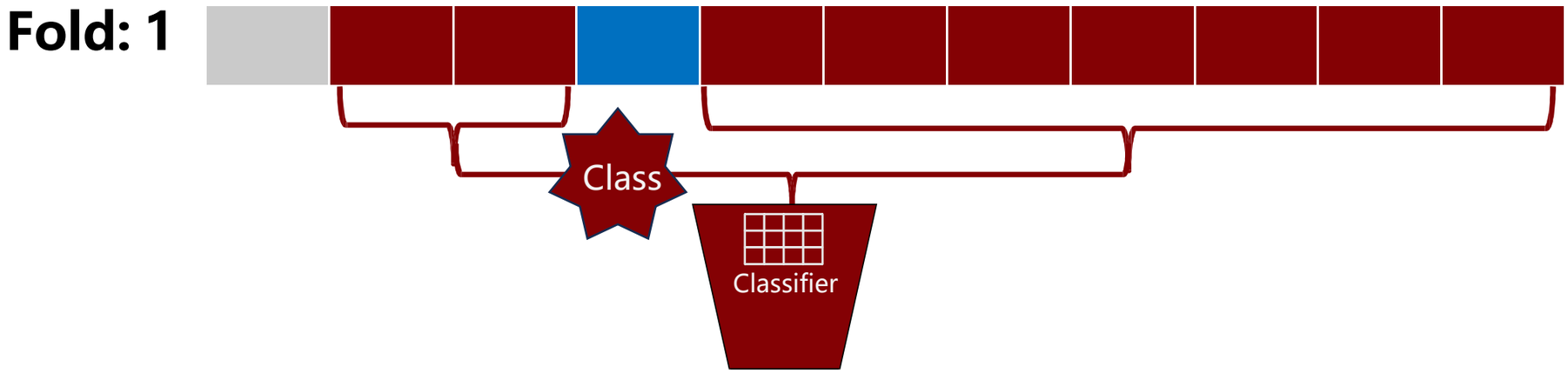
- Let's say your data has attributes A, B, and C
- You want to train a rule to predict D
- First train on 2, 3, 4, 5, 6, 7, 8, 9, 10
- and apply trained model to 1
- The results is Accuracy1

Cross Validation Helps for Small Data Sets



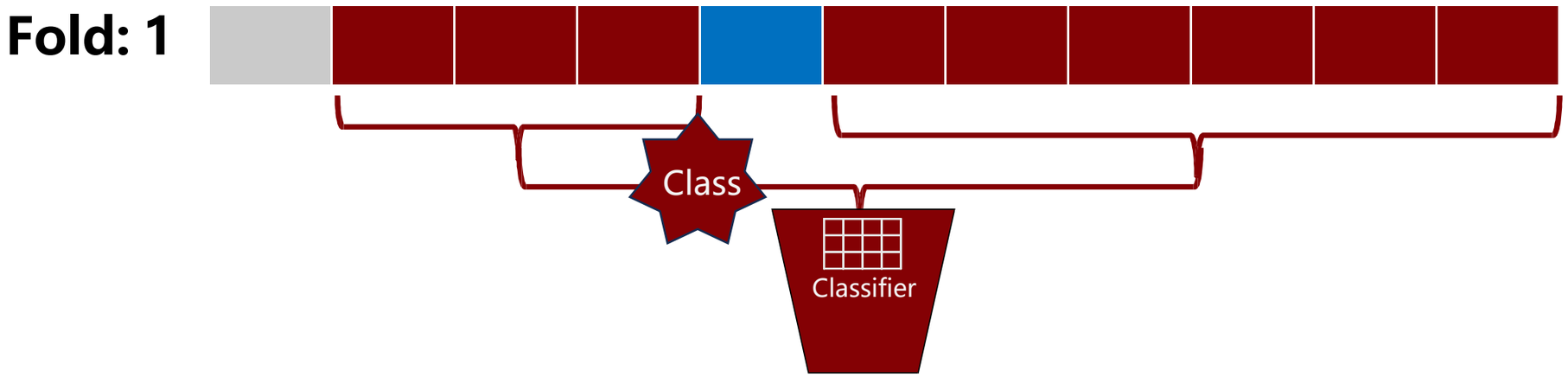
- Let's say your data has attributes A, B, and C
- You want to train a rule to predict D
- First train on 1, 3, 4, 5, 6, 7, 8, 9, 10
- and apply trained model to 2
- The results is Accuracy2

Cross Validation Helps for Small Data Sets



- Let's say your data has attributes A, B, and C
- You want to train a rule to predict D
- First train on 1, 2, 4, 5, 6, 7, 8, 9, 10
- and apply trained model to 3
- The results is Accuracy3

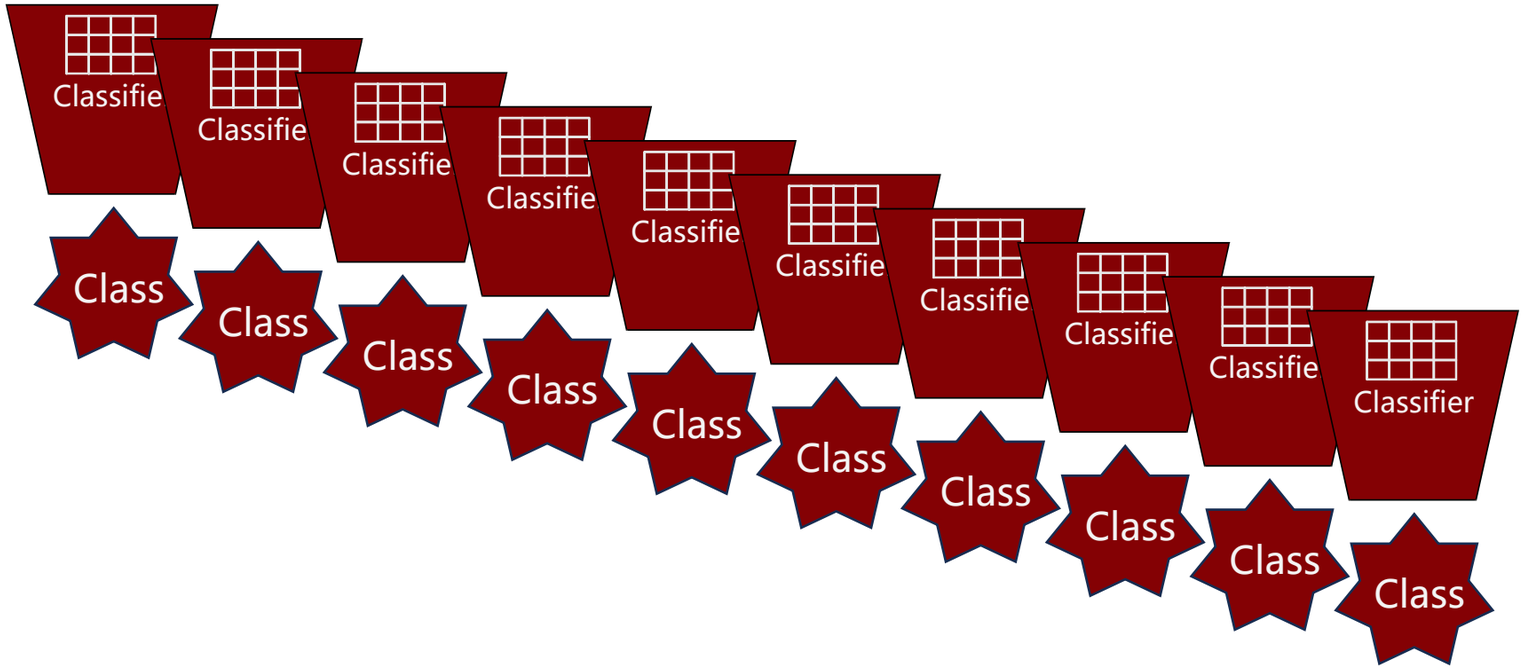
Cross Validation Helps for Small Data Sets



- Let's say your data has attributes A, B, and C
- You want to train a rule to predict D
- First train on 1, 2, 3, 5, 6, 7, 8, 9, 10
- and apply trained model to 4
- The results is Accuracy4

Cross Validation Helps for Small Data Sets

- Continue for all folds
- *Average* the resulting accuracies



Cross Validation

- Typically done with 10 subsamples (ten-fold cross validation)
- Provides an estimate of performance on an independent sample
- Makes the most of your data – large portion used for training
- But many iterations you are using insights from your testing data in building your model → need that optimization set

Partitioning your data

- "Stratified" → randomly choose partitions but in a way guaranteeing that each class is represented in approximately the right proportion

Partitioning your data

- "Stratified" → randomly choose partitions but in a way guaranteeing that each class is represented in approximately the right proportion
- "Per Person" → randomly choose one or more people to hold out (less standard, but may be more accurate)

Especially good if you want to know how your classifier would perform if a new user comes along

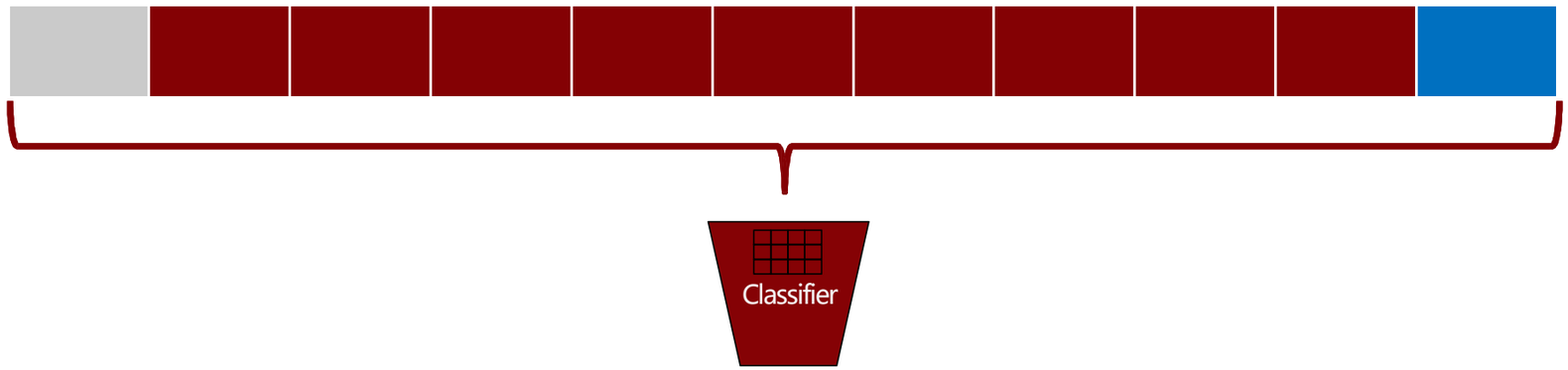
Do we have to do all of the folds?

- Yes!
- The test set on each fold is too small to give you an accurate estimate of performance alone
- Variation across folds
- Evaluation over part of the data is likely to be misleading

To make your classifier

If satisfied with the performance estimate we get using cross-validation

Build the model with the **WHOLE SET**



Don’t use cross-validation to build the model

To make your classifier

If you are not satisfied with the performance you get, then you should try to determine what went wrong, and then evaluate a different model that compensates.

How to compare two classifiers?

- How do we know that we picked the classifier that actually performs the best and not happen by chance?

- We don’t! But we can test (with some confidence) that the difference we are seeing did not happen by chance.

A quick intro to NHST

- Null Hypothesis Significance Testing
- Many disciplines banned NHST due to issues with p-hacking!

A quick intro to NHST

Usual setup:

- State Null Hypothesis (assume no difference)
- E.g., h0: "classifiers c1 and c2 have the same accuracy"
- Run an NHST test (e.g., t-test comparing the difference in variance between the two sets of accuracies)
- **Be careful!** Such tests make a lot of assumptions!
- The test returns a result (including a p-value)
- If p-value less than some alpha (e.g., $p < 0.05$), we **REJECT** the null hypothesis, which means that c1 and c2 have different accuracies.
- If $p >= 0.05$, we can't say anything.

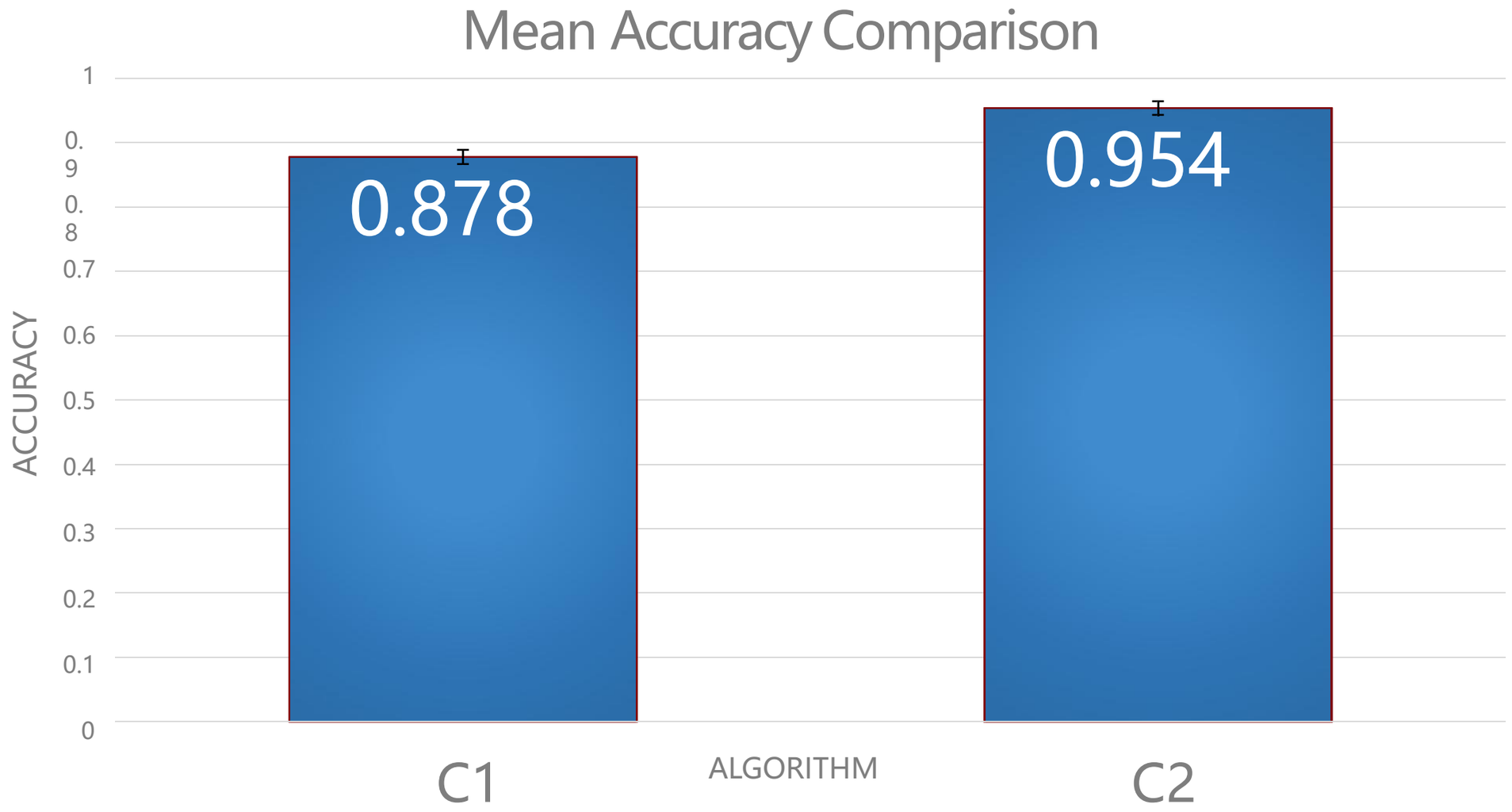
An example

How to compare two classifiers?

Fold	C1 Accuracy	C2 Accuracy
1	.89	.96
2	.88	.92
3	.91	.97
4	.87	.96
5	.88	.96
6	.86	.95
7	.90	.98
8	.85	.93
9	.86	.95
10	.88	.96

An example

How to compare two classifiers?



An example

How to compare two classifiers?

t-Test: Paired Two Sample for Means

	Variable 1	Variable 2
Mean	0.878	0.954
Variance	0.000351111	0.000315556
Observations	10	10
df	9	
t Stat	-15.23389079	
P(T<=t) two-tail	9.86181E-08	

An example

How to compare two classifiers?

Suppose $p > 0.05$. What then?

Sources of Bias

Training and testing on same data (you know too much!)

Optimizing and training on same data (same effect)

Feature Selection

Earlier indicated that feature selection is typically important to success

Very typical feature selection method: use information gain to "score" your features

- Use the top N features ranked by information gain

Wrapper based feature selection

Combinatoric search / optimization through possible feature subsets using classifier accuracy as the objective function

(this can be “wrapped” around any type of classifier)

Wrapper based feature selection

Combinatoric search / optimization through possible feature subsets using classifier accuracy as the objective function (this can be "wrapped" around any type of classifier)

Example:

- Start with an empty set of selected features
- Train a new classifier adding each candidate new feature
- Keep the feature which produces the classifier with the best accuracy
- Repeat until things stop improving

Wrapper based feature selection

This is an expensive process

- Building and testing a classifier is "inside the loop" here so typically want a fast learner (e.g., naïve Bayes, decision trees)

What would happen if you do feature selection over your whole set of data prior to cross validation?

Wrapper based feature selection

This is an expensive process

- Building and testing a classifier is "inside the loop" here so typically want a fast learner (e.g., naïve Bayes, decision trees)

Will overfit if you do it on *entire data set*
(have to do on *each fold*)

Finding Features

The same problem occurs if you design new features based on observations over your whole set of data.

- Can you explain why?

Finding Features

The same problem occurs if you design new features based on observations over your whole set of data.

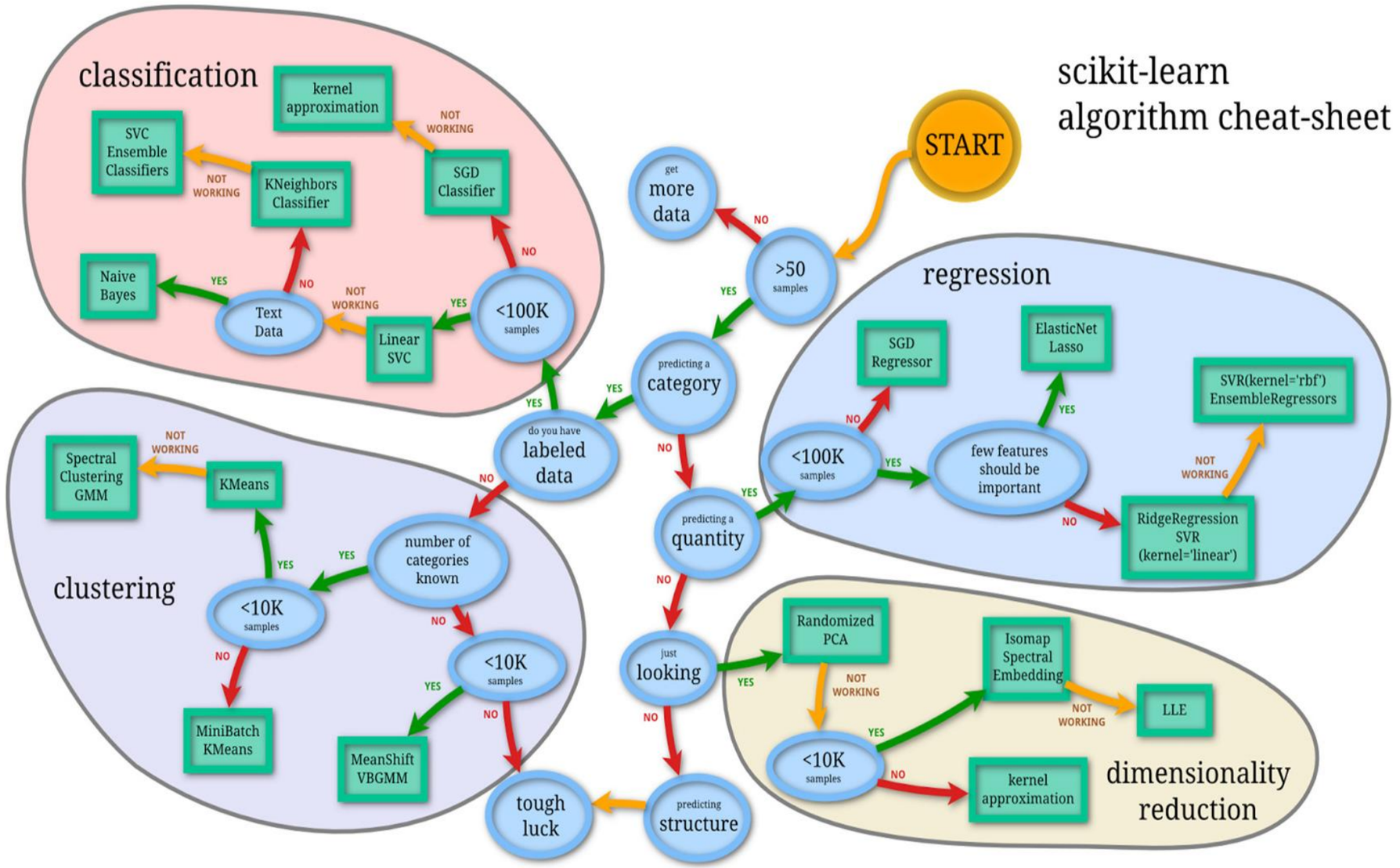
- At training time the features will perform better than they should because they were design based on the training data
- At testing time, you will have used ‘omniscience’ to build features

Finding Features

The same problem occurs if you design new features based on observations over your whole set of data.

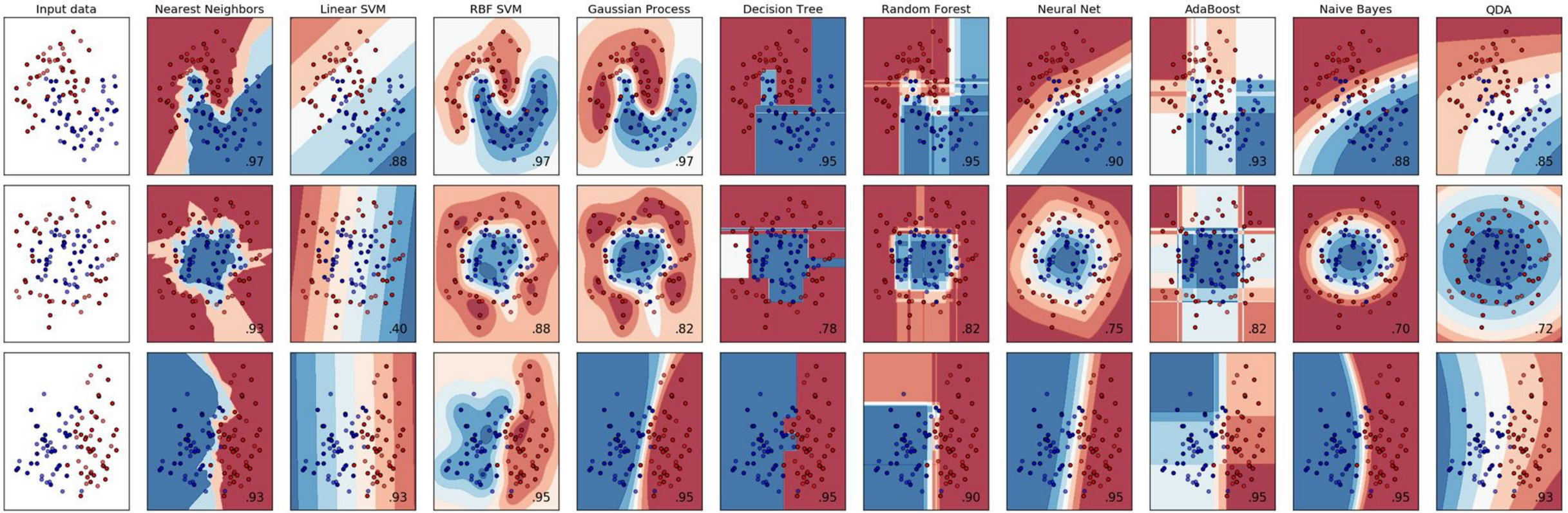
-> Yet another reason for the optimization set!

Selecting algorithms



http://www.quora.com/What-are-the-advantages-of-different-classification-algorithms

Selecting algorithms



http://www.quora.com/What-are-the-advantages-of-different-classification-algorithms

Summary

Automatically or *semi-automatically*

- Inducing concepts (*i.e.*, rules) from data
- Finding patterns in data
- Explaining data
- Making predictions

Supervised ML depends on *labels* and *features*

Labels typically expensive to acquire Features key to success

Metrics for assessing results

Accuracy [limited value]

$$\text{Precision} = TP / (TP + FP) = 90/(90+15) = 86\%$$

$$\text{Recall} = TP / (TP + FN) = 90/(90+12) = 88\%$$

$$\text{F-Score} = 2 * \frac{(\text{Precision} * \text{Recall})}{(\text{Precision} + \text{Recall})} = 87\%$$

$$\text{Kappa} = \frac{(\text{observed acc.} - \text{expected acc.})}{(1 - \text{expected acc.})}$$

Best Practices

Divide your data

Optimization set (for tuning parameters; *etc.*) Training set (for training your classifier) Testing set (for calculating scores)

Estimate performance on training set (10 fold cross)

Test set is for reporting

All data is for producing a classifier to use in the real world

Read your data. Explore it. Try things out (feature sets, algthm parameters). But do it *all* on the optimization set.

Thank You