Introduction to Machine learning

Why learning

- Machine learning is programming computers to optimize a performance criterion using example data or past experience
  - Learning general models from a data of particular examples
  - Build a model that is *a good and useful approximation* to the data.

- Data is cheap and abundant (data warehouses, data marts); knowledge is expensive and scarce.

When learning

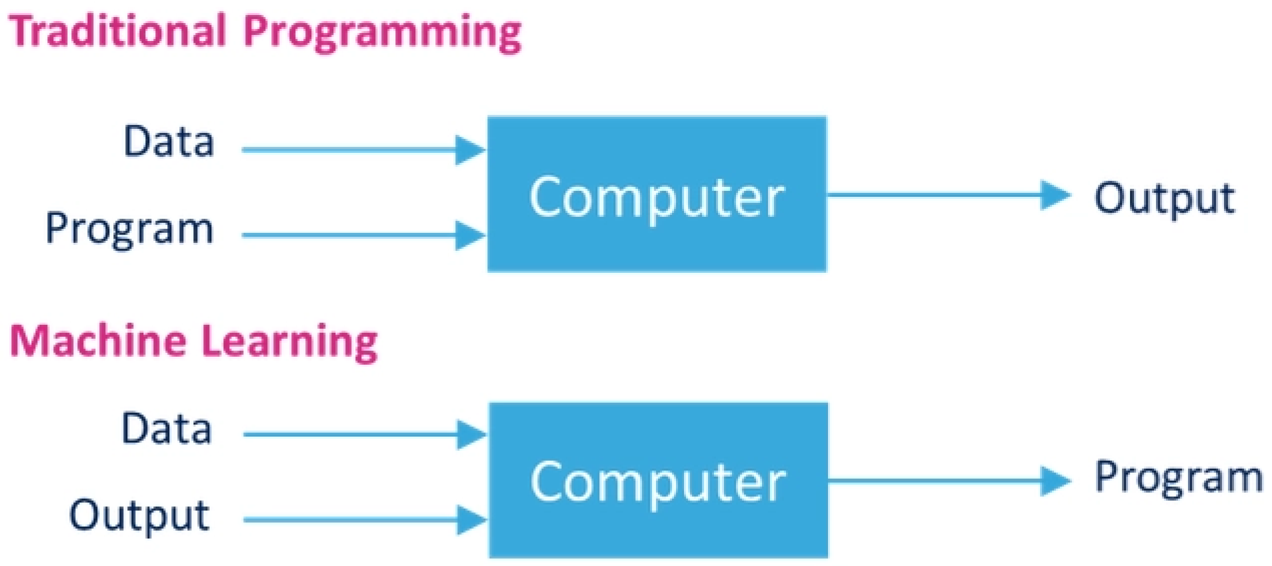
- Machine learning is suitable for problems where:
  - A pattern exists
  - We can not pin it down mathematically
  - We have data on it

- Learning is used when:
  - Human expertise does not exist (navigating on Mars)
  - Humans are unable to explain their expertise (speech recognition)
  - Solution changes in time (routing on a computer network)

How to do the learning

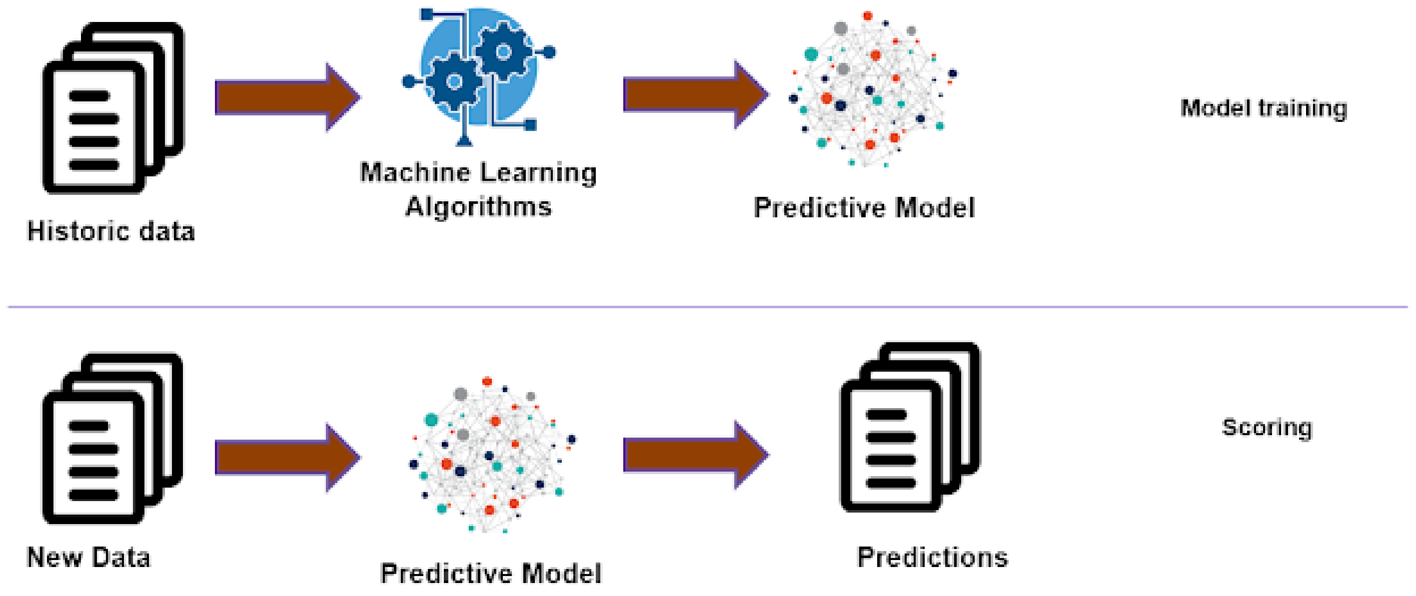
If you don't know how to solve a problem, write it out as an optimization problem

Learning vs Programming



The Learning algorithm is implemented as a program

Learning vs Programming



The Learning algorithm is implemented as a program

Classification

Patient data

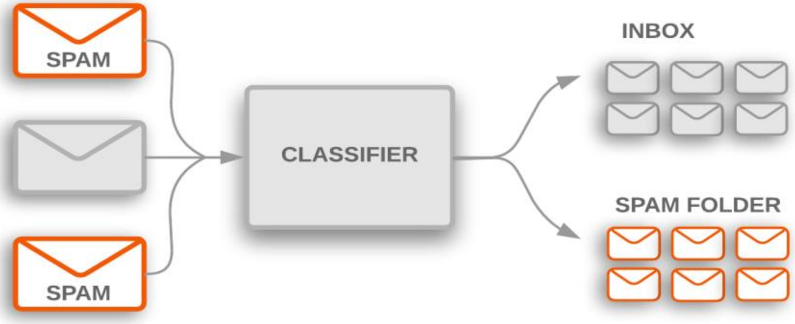
Yes

Is sick?

No

Medical Diagnosis

Text Categorization



Spam detection

Classification assigns data items to target categories or classes

Classification

- The problem of classification is defined as:

- **Given**: A set of training data

  $(x_1, y_1), \dots (x_n, y_n)$ where x_i in $\mathbb{R}^n$ and y_i is the class label

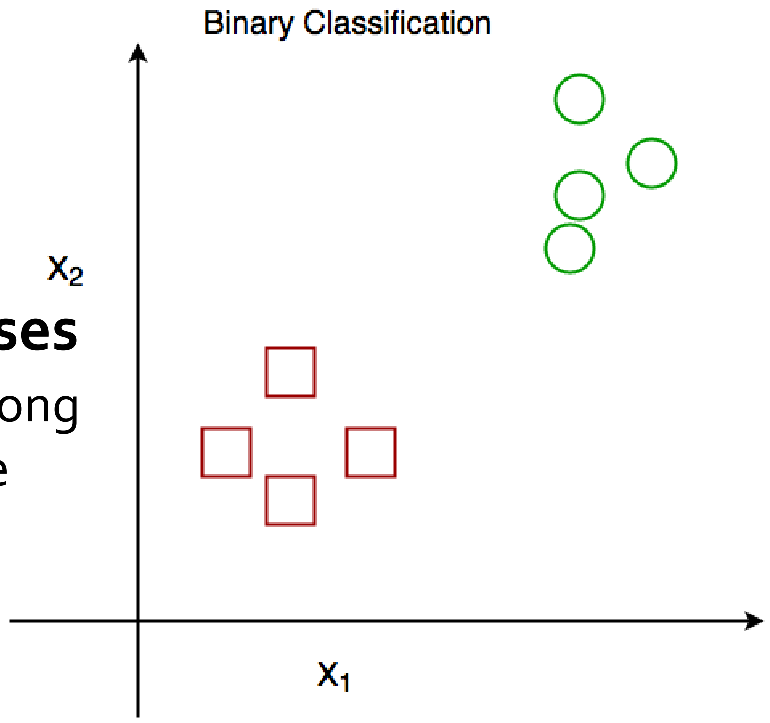
- **Find :** A classification function

  $f\colon \mathbb{R}^n \rightarrow \{c_1, \dots, c_k\}$ which classifies well additional samples $\{x_k\}$

  k is the number of classes

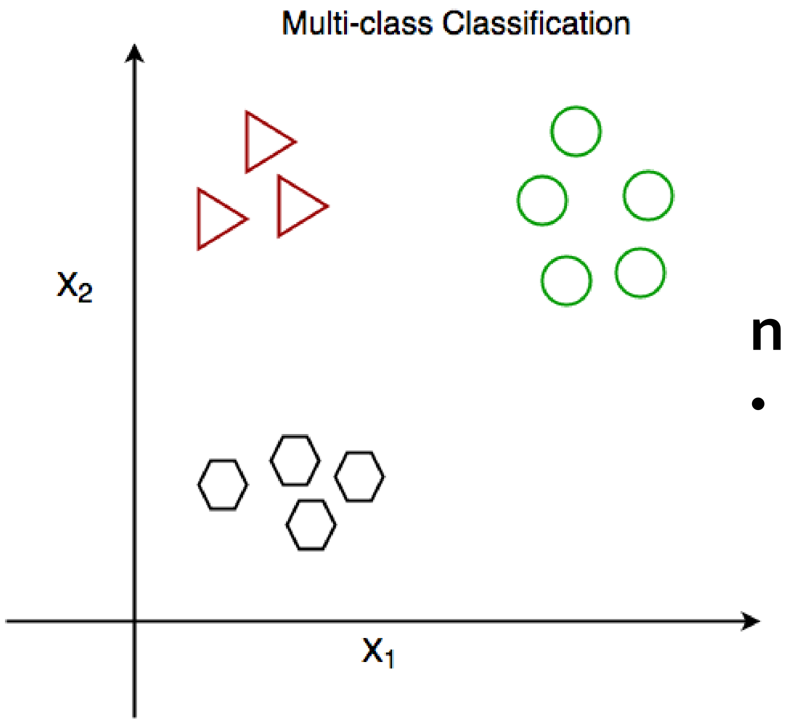
Classification

Binary vs multiclass classification



2 target classes

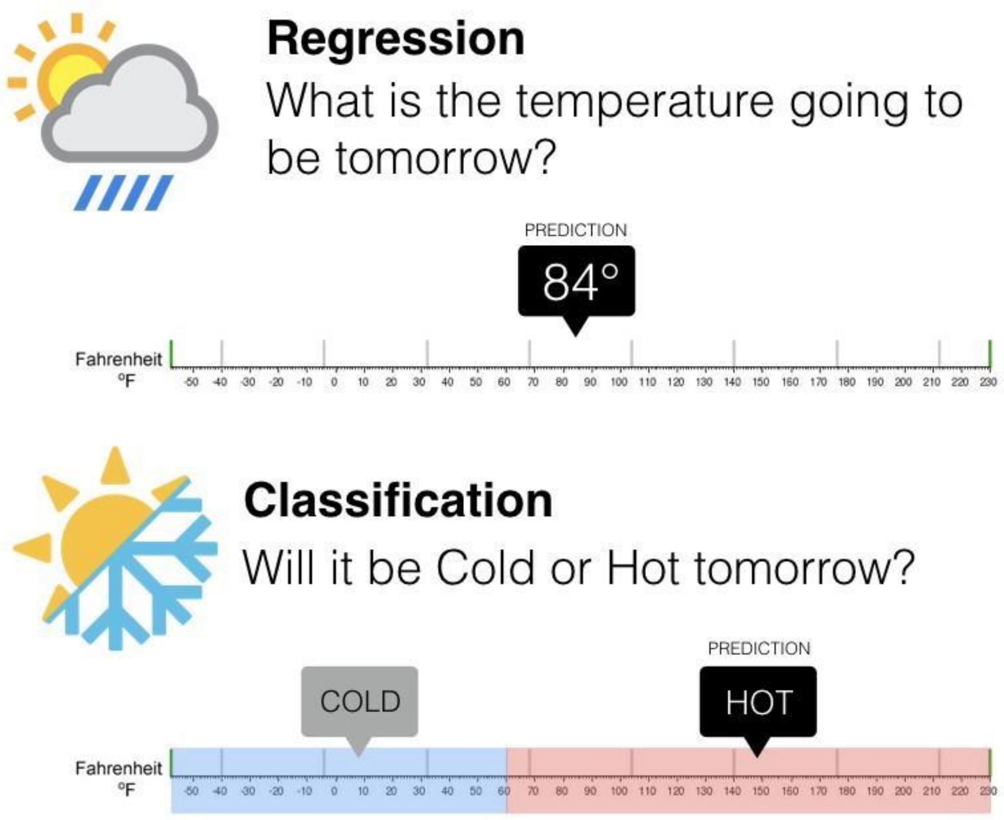
- Samples belong to one of the two classes



n target classes

- Samples belong to one of the n classes

Classification vs regression



Regression is the task of predicting a continuous quantity

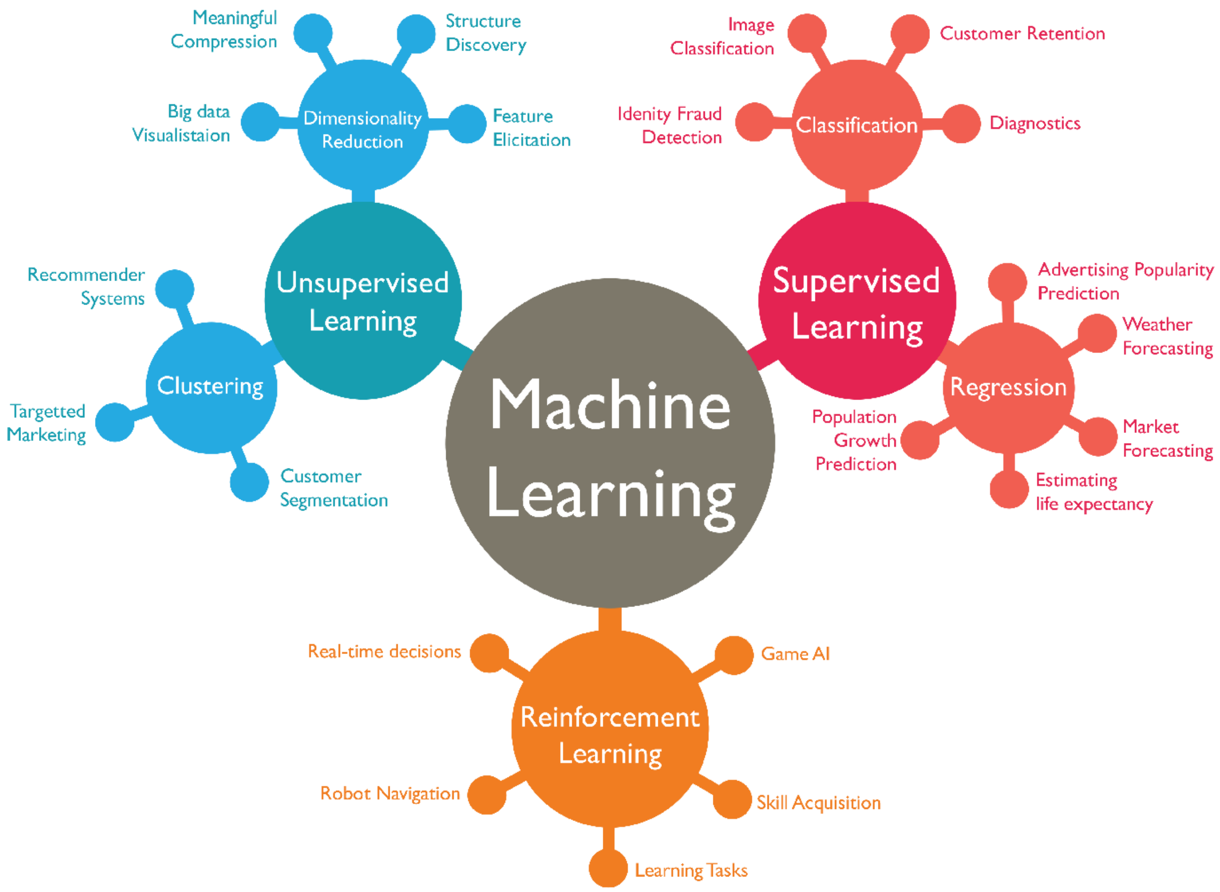
Classification is the task of predicting a discrete class label

But, generally models return numbers (probability, ….)

ML: Classification methods

Approaches to learn classifiers/predictors:

- Linear classifiers: linear Regression, Bayesian classifier
- Support Vector Machines (SVM)
- Decision trees
- Random Forest
- K-Nearest Neighbor
- Neural Networks
- …



What the matter between the different machine learning algorithms?

ML: Main components

- Model representation (hypothesis space, aka learning model):
  - Structure of the functional form of the knowledge to be extracted (Trees, partition, graph,…)

- Search method (learning algorithm):
  - Strategy used to explore the search space and find the optimal or “good” model (backpropagation, local search, divide-and-conquer, greedy search, …)

- Objectif function (cost function):
  - Measure the quality of the model (Gini, Entropy, RMSE, logloss, …)

A toy model: Learning A Decision Stump

“Search and Score”

Supervised Learning

Egg	Milk	Fish	Wheat	Shellfish	Peanuts	…	Sick?
0	0.7	0	0.3	0	0		1
0.3	0.7	0	0.6	0	0.01		1
0	0	0	0.8	0	0		0
0.3	0.7	1.2	0	0.10	0.01		1
0.3	0	1.2	0.3	0.10	0.01		1

- Input for an example (day of the week) is a set of features/attributes (quantities of food).
- Output is a desired class label (whether or not we got sick).
- Our objective:
  - Use data to find a model that outputs the right label based on the features.
    - Whether foods will make you sick (even with new combinations).
  - This framework can be applied any problem where we have input/output examples.

Decision Trees

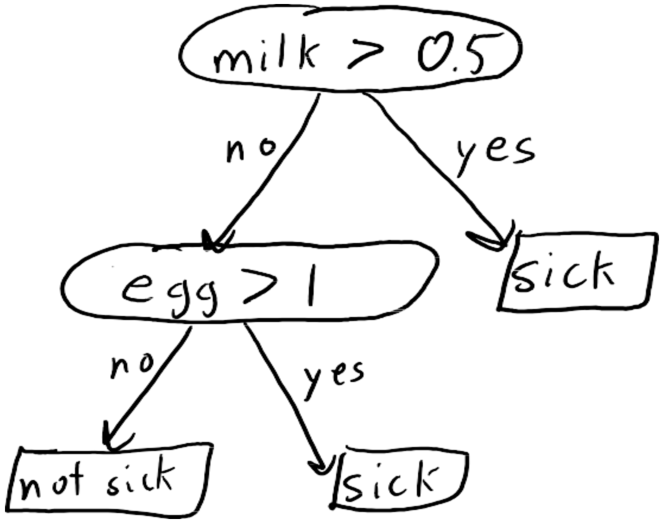
- Decision trees are simple programs consisting of:
  - A nested sequence of "if-else" decisions based on the features (splitting rules).
  - A class label as a return value at the end of each sequence.

- Example decision tree:

```
if (milk > 0.5)
   {
            return 'sick'
   }

   else
   {        if (egg > 1)
                     return 'sick'
            else
                     return 'not
                     sick'
   }
```

Can draw sequences of decisions as a tree:



Decision Trees

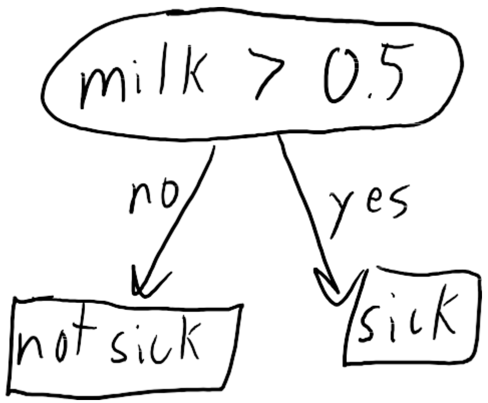
- There are many possible decision trees.
  - We're going to search for one that is good at our supervised learning problem.

- So our input is data and the output will be a program.
  - This is called "training" the supervised learning model.
  - Different than usual input/output specification for writing a program.

- Supervised learning is useful when you have lots of labeled data BUT:
  1. Problem is too complicated to write a program ourselves.
  2. Human expert can't explain why you assign certain labels. OR
  3. We don't have a human expert for the problem.

Decision Stumps

- – Simple decision tree with 1 splitting rule based on thresholding 1 feature.



- How do we find the best "rule" (feature, threshold, and leaf labels)?
  1. Define a 'score' for the rule.
  2. Search for the rule with the best score.

Decision Stumps: Accuracy Score

- Most intuitive score: classification accuracy.
  - "If we use this rule, how many examples do we label correctly?"
- Computing classification accuracy for (egg > 1):
  - Find most common labels if we use this rule:
    - When (egg > 1), we were "sick" 2 times out of 2.
    - When (egg ≤ 1), we were "not sick" 3 times out of 4.
  - Compute accuracy:
    - The accuracy ("score") of the rule (egg > 1) is 5 times out of 6.
- This "score" evaluates quality of a rule.
  - We "learn" a decision stump by finding the rule with the best score.

Milk	Fish	Egg
0.7	0	1
0.7	0	2
0	0	0
0.7	1.2	0
0	1.2	2
0	0	0

Sick?
1
1
0
0
1
0

Learning A Decision Stump: By Hand

- Let's search for the decision stump maximizing classification score:

Milk	Fish	Egg	Sick?
0.7	0	1	1
0.7	0	2	1
0	1.2	0	0
0.7	1.2	0	0
0	1.3	2	1
0	0	0	0

- First we check "baseline rule" of predicting mode (no split): this gets 3/6 accuracy.

- If (milk > 0) predict "sick" (2/3) else predict "not sick" (2/3): 4/6 accuracy
- If (fish > 0) predict "not sick" (2/3) else predict "sick" (2/3): 4/6 accuracy
  If (fish > 1.2) predict "sick" (1/1) else predict "not sick" (3/5): 5/6 accuracy
- If (egg > 0) predict "sick" (3/3) else predict "not sick" (3/3): 6/6 accuracy
  If (egg > 1) predict "sick" (2/2) else predict "not sick" (3/4): 5/6 accuracy

- Highest-scoring rule: (egg > 0) with leaves "sick" and "not sick".
- Notice we only need to test feature thresholds that happen in the data:
  - There is no point in testing the rule (egg > 3), it gets the "baseline" score.
  - There is no point in testing the rule (egg > 0.5), it gets the (egg > 0) score.
  - Also note that we don't need to test "<", since it would give equivalent rules.

Learning A Decision Stump: Training

- What we want to do is :

X =

Egg	Milk	Fish	Wheat	Shellfish	Peanuts
0	0.7	0	0.3	0	0
0.3	0.7	0	0.6	0	0.01
0	0	0	0.8	0	0
0.3	0.7	1.2	0	0.10	0.01
0.3	0	1.2	0.3	0.10	0.01

'd'

'n'

y =

Sick?
1
1
0
1
1

'n'

- Training phase:
  - Use 'X' and 'y' to find a 'model' (like a decision stump).
- Prediction phase:
  - Given an example x_i, use 'model' to predict a label '$\hat{y}_i$' ("sick" or "not sick").
- Training error:
  - Fraction of times our prediction $\hat{y}_i$ does not equal the true y_i label.

Learning A Decision Stump: Cost

- Assume we have:
  - 'n' examples (days that we measured).
  - 'd' features (foods that we measured).
  - 'k' thresholds (>0, >1, >2, …) for each feature.
- Computing the score of one rule costs O(n):
  - We need to go through all 'n' examples to find most common labels.
  - We need to go through all 'n' examples again to compute the accuracy.
- We compute score for up to k*d rules ('k' thresholds for each of 'd' features):
  - So we need to do an O(n) operation k*d times, giving total cost of O(ndk).

Learning A Decision Stump: Cost

- Size of the input data is O(nd):
  - If 'k' is small then the cost is roughly the same cost as loading the data.
    - We should be happy about this, you can learn on any dataset you can load!
  - If 'k' is large then this could be too slow for large datasets.
- Example: if all our features are binary then k=1, just test (feature > 0):
  - Cost of fitting decision stump is O(nd), so we can fit huge datasets.
- Example: if all our features are numerical with unique values then k=n.
  - Cost of fitting decision stump is $O(n^2d)$.
    - We don't like having n^2 because we want to fit datasets where 'n' is large!

Learning A Decision Stump: conclusion

- Decision stumps have only 1 rule based on only 1 feature.
  - Very limited class of models: usually not very accurate for most tasks.

- Decision trees allow sequences of splits based on multiple features.
  - Very general class of models: can get very high accuracy.
  - However, it's computationally infeasible to find the best decision tree.

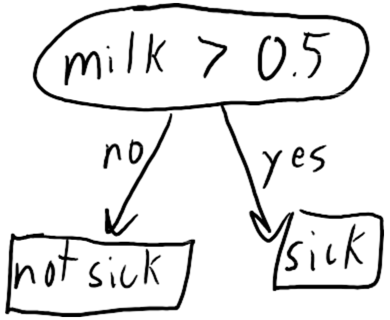
- Most common decision tree learning algorithm in practice:
  - Greedy recursive splitting.

Greedy Recursive Splitting

- Start with the full dataset:

Egg	Milk	...	Sick?
0	0.7		1
1	0.7		1
0	0		0
1	0.6		1
1	0		0
2	0.6		1
0	1		1
2	0		1
0	0.3		0
1	0.6		0
2	0		1

Find the decision stump with the best score:



Split into two smaller datasets based on stump:

Egg	Milk	...	Sick?
0	0		0
1	0		0
2	0		1
0	0.3		0
2	0		1

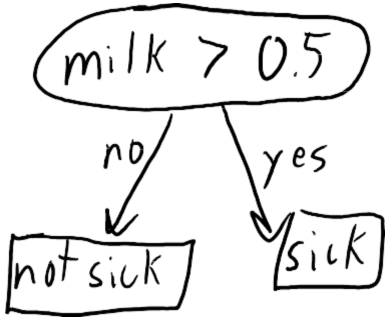
milk ≤ 0.5

Egg	Milk	...	Sick?
0	0.7		1
1	0.7		1
1	0.6		1
2	0.6		1
0	1		1
1	0.6		0

> 0.5

Greedy Recursive Splitting

We now have a decision stump and two datasets:

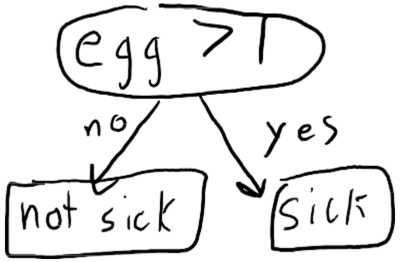


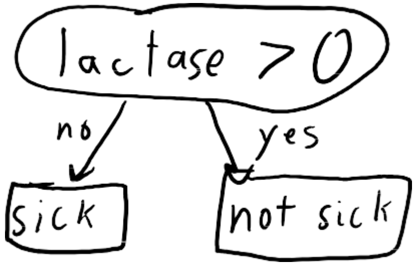
Egg	Milk	...	Sick?
0	0		0
1	0		0
2	0		1
0	0.3		0
2	0		1

Egg	Milk	...	Sick?
0	0.7		1
1	0.7		1
1	0.6		1
2	0.6		1
0	1		1
1	0.6		0

find stump

find stump

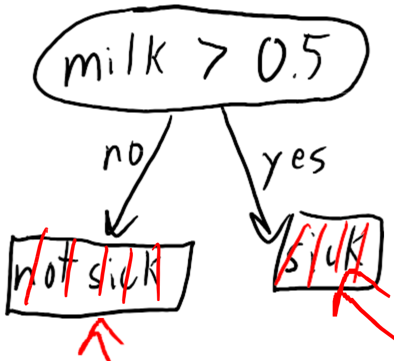




Fit a decision stump to each leaf's data.

Greedy Recursive Splitting

We now have a decision stump and two datasets:



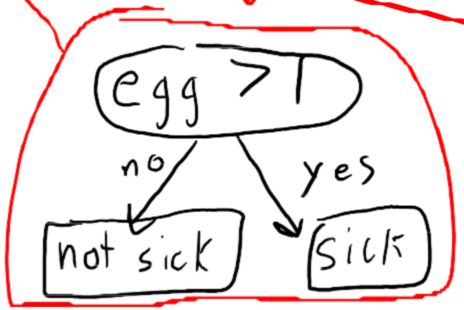
Egg	Milk	...	Sick?
0	0		0
1	0		0
2	0		1
0	0.3		0
2	0		1

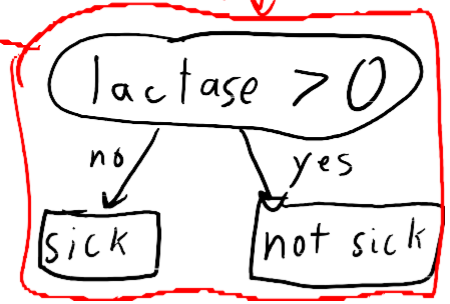
find stump

Egg	Milk	...	Sick?
0	0.7		1
1	0.7		1
1	0.6		1
2	0.6		1
0	1		1
1	0.6		0

find stump

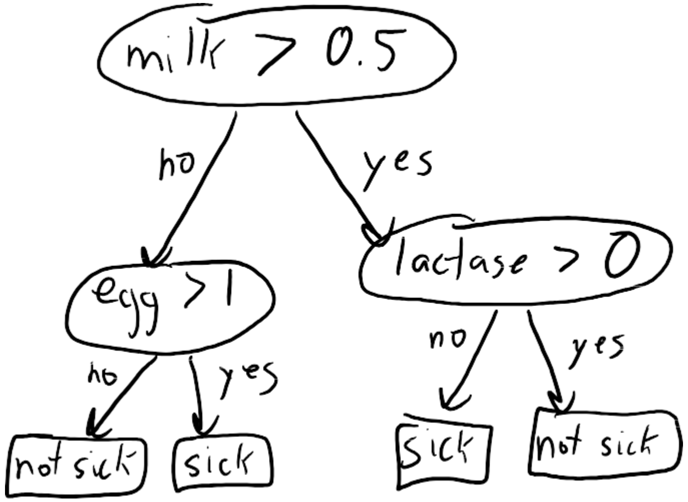
Fit a decision stump to each leaf's data.
Then add these stumps to the tree.





Greedy Recursive Splitting

This gives a "depth 2" decision tree:



It splits the two datasets into four datasets:

milk ≤ 0.5 data

Egg	Milk	...	Sick?
0	0		0
1	0		0
2	0		1
0	0.3		0
2	0		1

milk > 0.5 data

Egg	Milk	...	Sick?
0	0.7		1
1	0.7		1
1	0.6		1
2	0.6		1
0	1		1
1	0.6		0

milk ≤ 0.5, egg ≤ 1

Egg	Milk	...	Sick?
0	0		0
1	0		0
0	0.3		0

Much more accurate!

milk < 0.5, egg > 1

Egg	Milk	...	Sick?
2	0		1
2	0		1

milk > 0.5, lactase ≤ 0

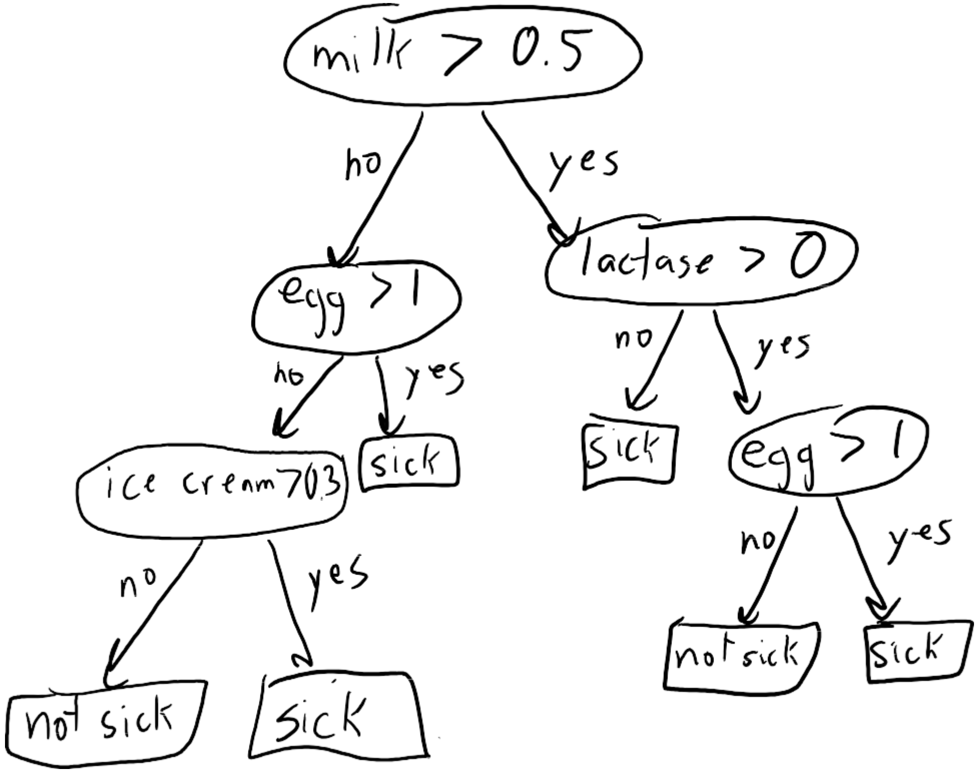
Egg	Milk	...	Sick?
0	0.7		1
1	0.7		1
1	0.6		1
2	0.6		1

milk > 0.5, lactase > 0

Egg	Milk	...	Sick?
1	0.6		0

Greedy Recursive Splitting

We could try to split the four leaves to make a "depth 3" decision tree:



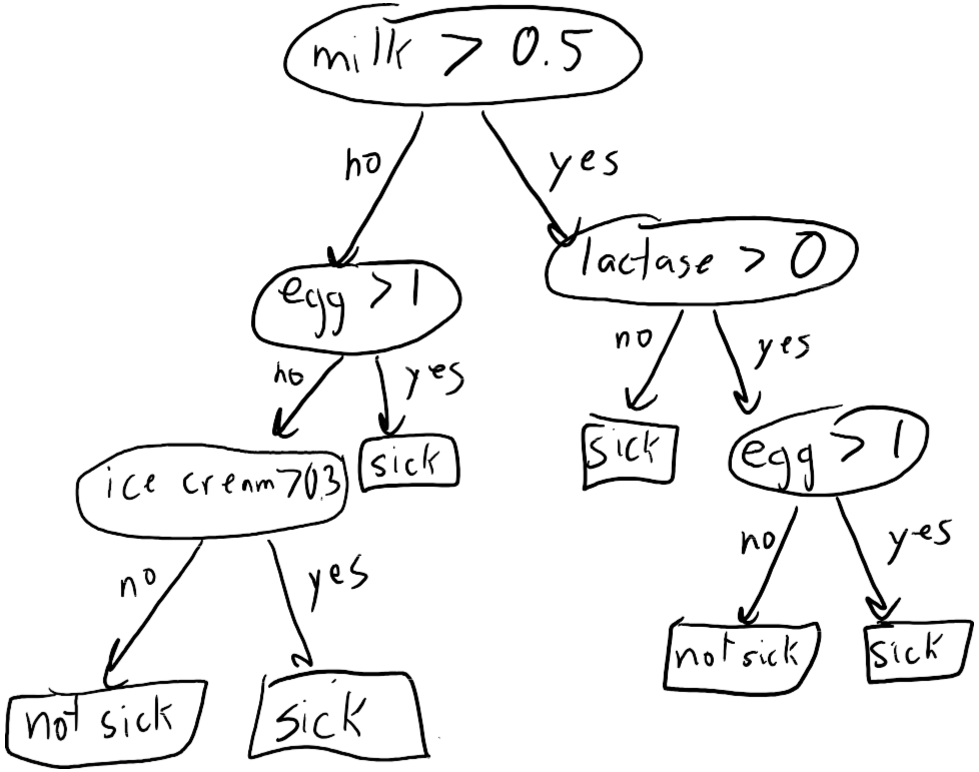
We might continue splitting until:
- The leaves each have only one label.
- We reach a user-defined maximum depth.

- Shouldn't we just use accuracy score?
  - For leafs: yes, just maximize accuracy.
  - For internal nodes: not necessarily.

- Maybe no simple rule like (egg > 0.5) improves accuracy.
  - But this doesn't necessarily mean we should stop!

Greedy Recursive Splitting

We could try to split the four leaves to make a "depth 3" decision tree:



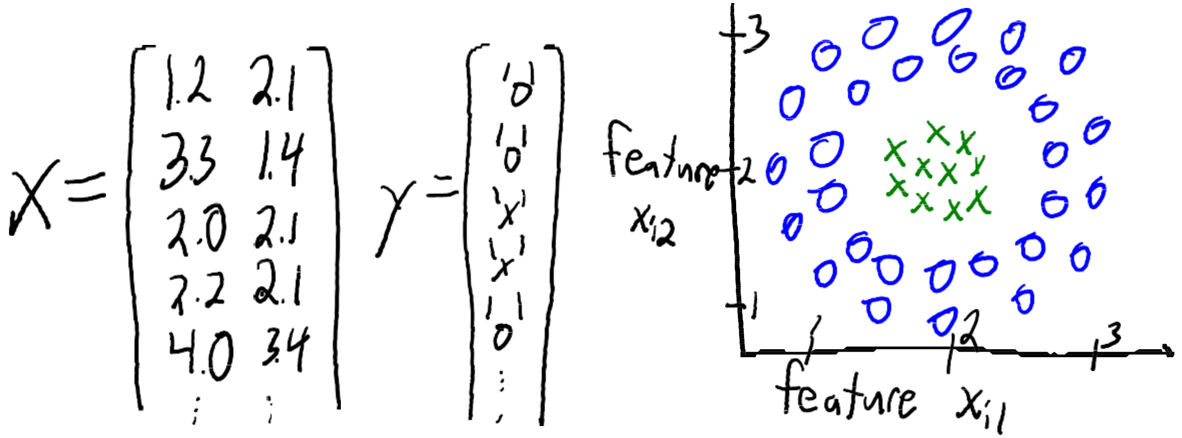
We might continue splitting until:

- The leaves each have only one label.
- We reach a user-defined maximum depth.

- Shouldn't we just use accuracy score?
  - For leafs: yes, just maximize accuracy.
  - For internal nodes: not necessarily.
- Maybe no simple rule like (egg > 0.5) improves accuracy.
  - But this doesn't necessarily mean we should stop!

Greedy Recursive Splitting

- Consider a dataset with 2 features and 2 classes ('x' and 'o').
  - Because there are 2 features, we can draw 'X' as a scatterplot.
    - Colours and shapes denote the class labels 'y'.



- A decision stump would divide space by a horizontal or vertical line.
  - Testing whether $x_{i1} > t$ or whether $x_{i2} > t$.
- On this dataset no horizontal/vertical line improves accuracy.
  - Baseline is 'o', but need to get many 'o' wrong to get one 'x' right.