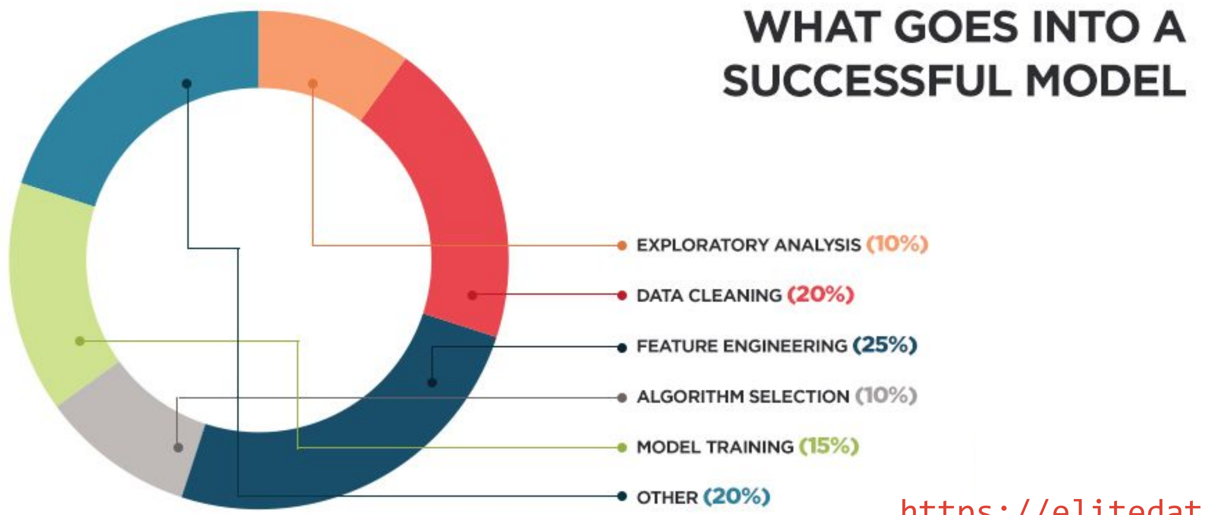
feature engineering

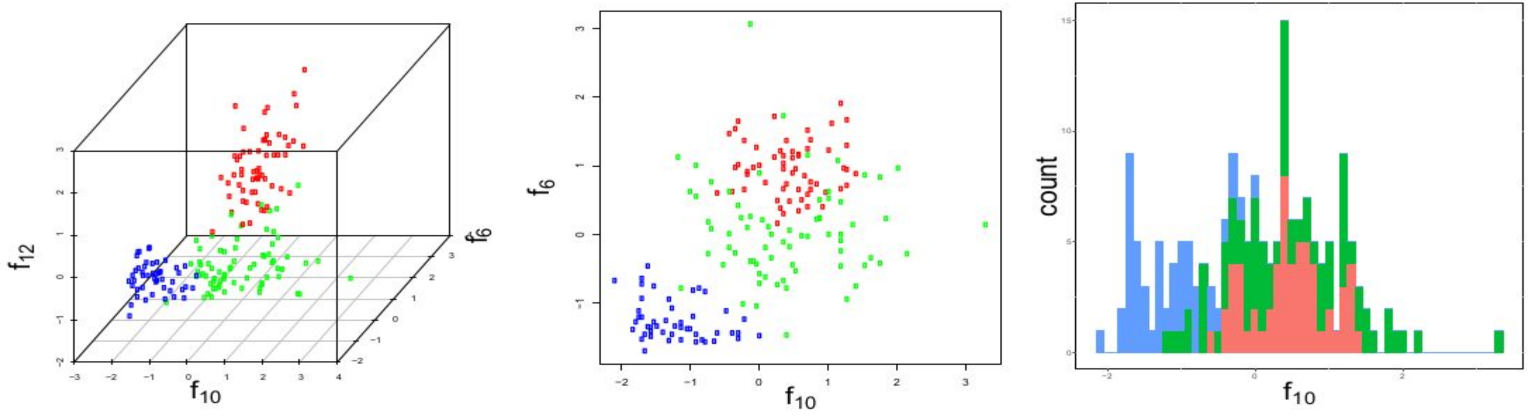
Q: how do you get the most out of your data, for predictive modelling?

Data scientists typically spend a lot of time doing “feature engineering”...



https://elitedatascience.com/feature-engineering

More features = good?



(a) 3 features: f_{10}, f_6, f_{12}. (b) 2 features: f_{10}, f_6. (c) 1 feature: f_{10}.

Figure 1.1: Wine dataset projected across varying numbers of features. Wine contains three classes, 13 features, and 178 instances.

from Andrew Lensen's PhD thesis, VUW, 2019

e.g. plot 3d demo

an imaginary example…

1000 cases, half are ♀ / ♂, 30% have heart disease

frequency of the antecedant (l.h.s), overall

fraction of cases where the consequent (r.h.s.) was true

	Rule	*fr*	ϕ
1	smoking → heart disease	120	0.400
2	sports → ¬ heart disease	400	0.800
3	coffee → ¬ heart disease	240	0.700
4	stress → heart disease	150	0.300
5	pine bark extract → ¬ heart disease	1	1.000
6	female → ¬ heart disease	352	0.704
7	female, stress → heart disease	100	0.385
8	stress, smoking → heart disease	100	0.500
9	smoking, coffee → heart disease	96	0.400
10	smoking, sports → heart disease	20	0.333
11	female, sports → ¬ heart disease	203	0.808

simple dependencies (rules 1–2)

independencies (rules 3–4)

spurious (too few) (rule 5)

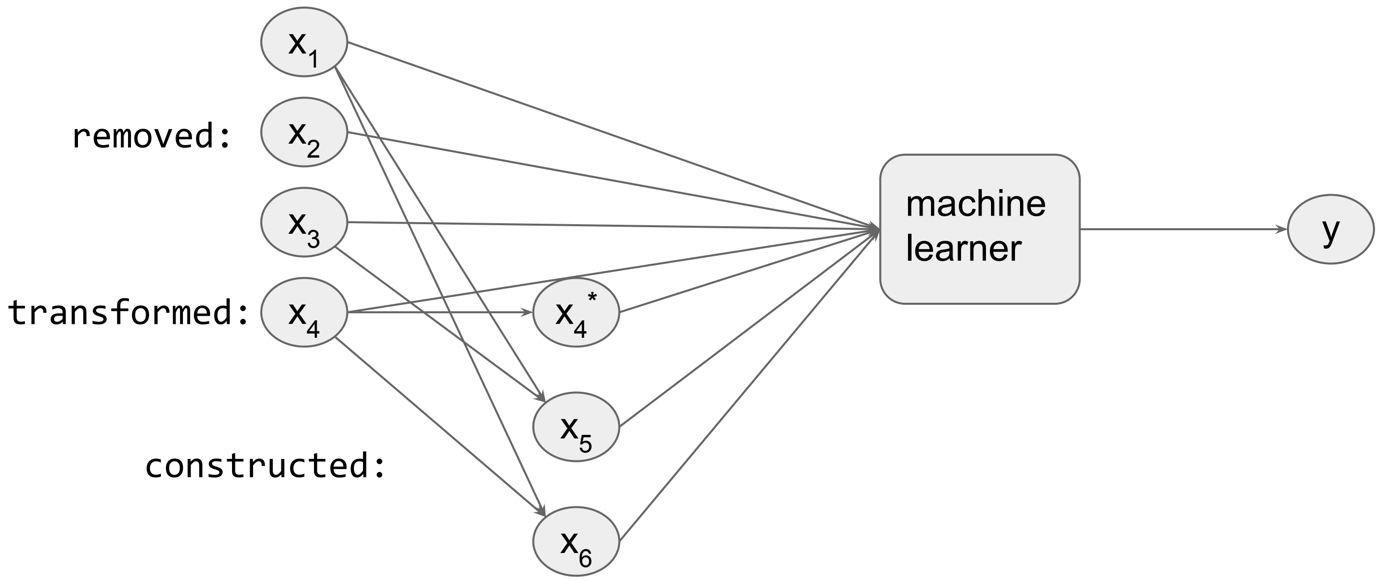
spurious (too *small*) (rule 6)

non-monotonicity! (rule 7)

superfluous variables (rules 10–11)

http://i.giwebb.com/2019/02/25/tutorial-on-statistically-sound-pattern-discovery/

changes to features

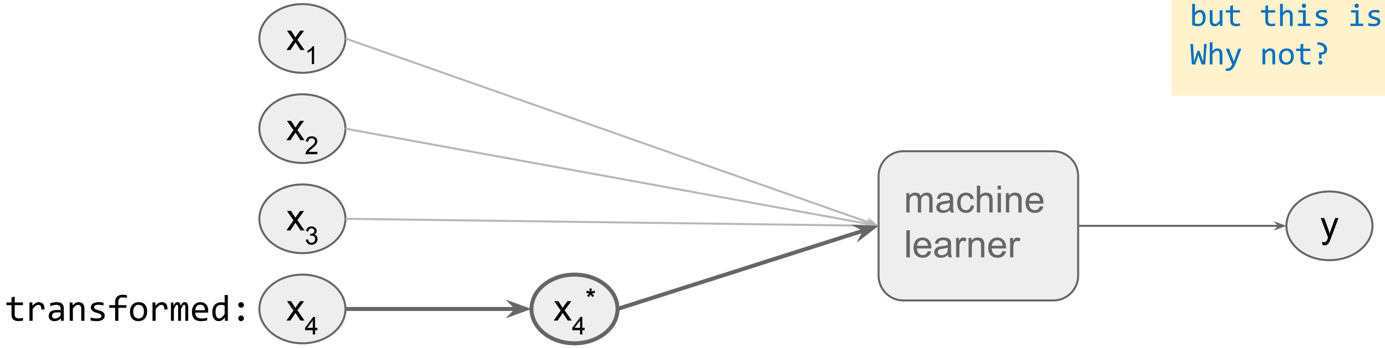


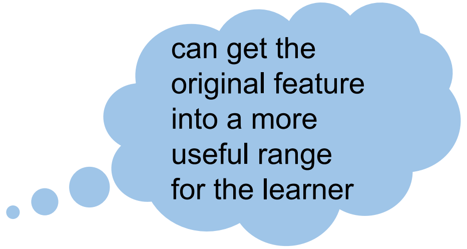
one-to-one

aside: if $x_1 \in$ {dog, cat, ferret} it is tempting to transform into

dog	$\rightarrow$ 1
cat	$\rightarrow$ 2
ferret	$\rightarrow$ 3

but this is *not* a good idea.
Why not?

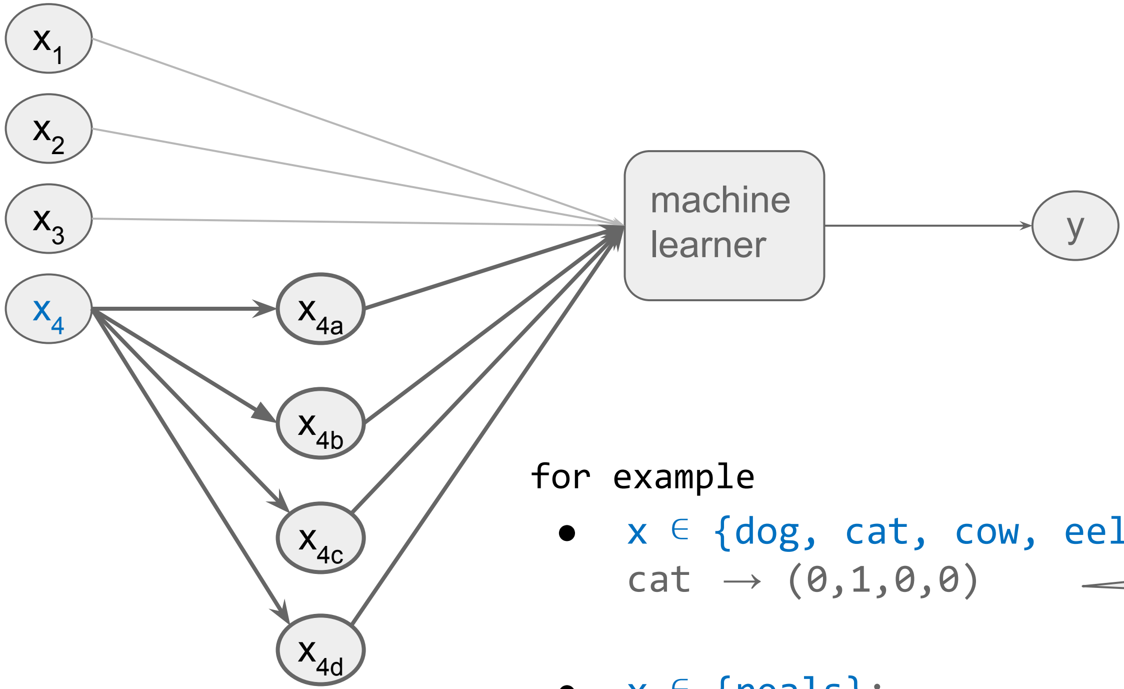



can get the original feature into a more useful range for the learner

for example

- clipping
- normalization
- discretization
- mathematical transformations such as logarithm log(x), reciprocal function 1/x, exponentiation exp(x), etc…

one-to-many



might make it easier / more natural for the machine learner to work with

```
for example
```

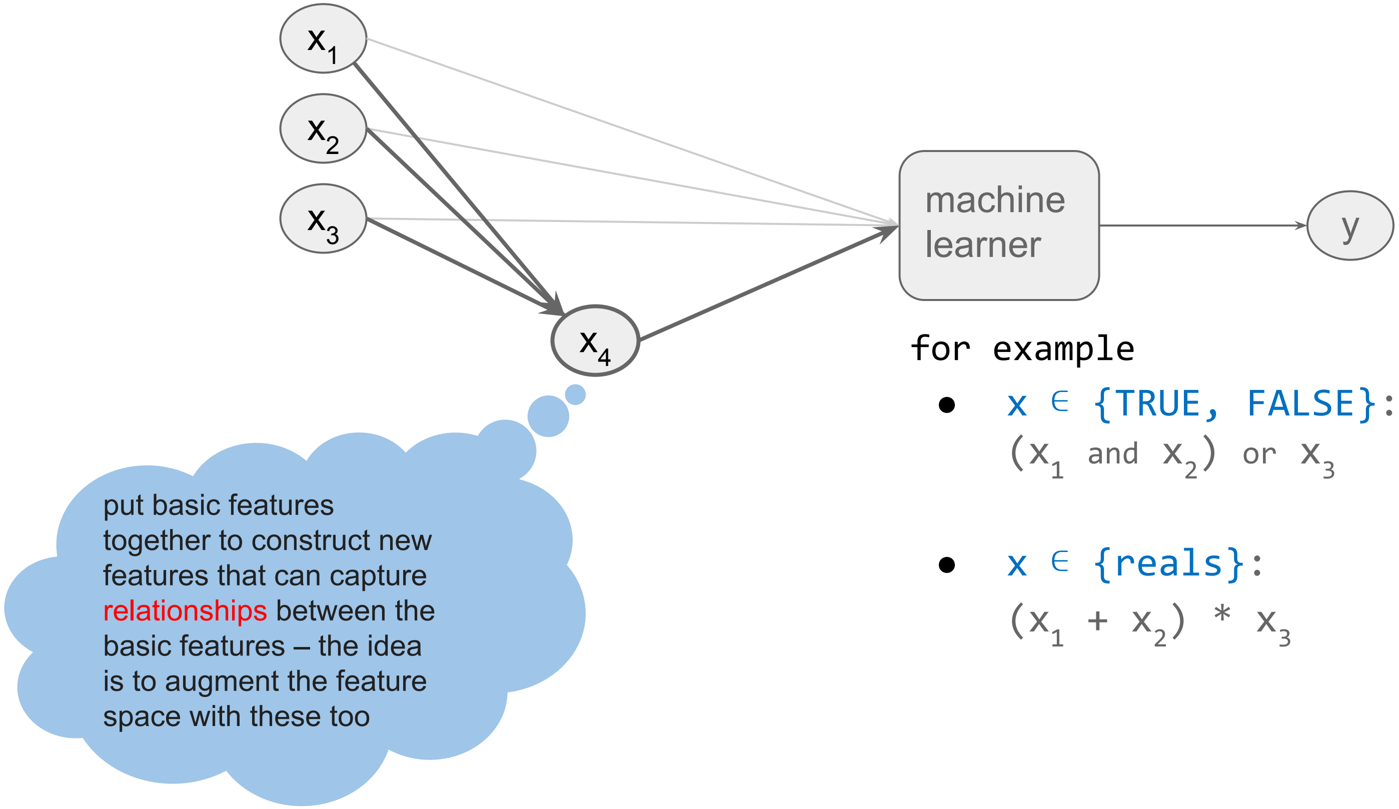
- `x ∈ {dog, cat, cow, eel}:`
  `cat  → (0,1,0,0)`

  “one-hot” encoding

- `x ∈ {reals}:`
  x → (x, x^2, x^3, x^4)

  polynomial expansion

many-to-one

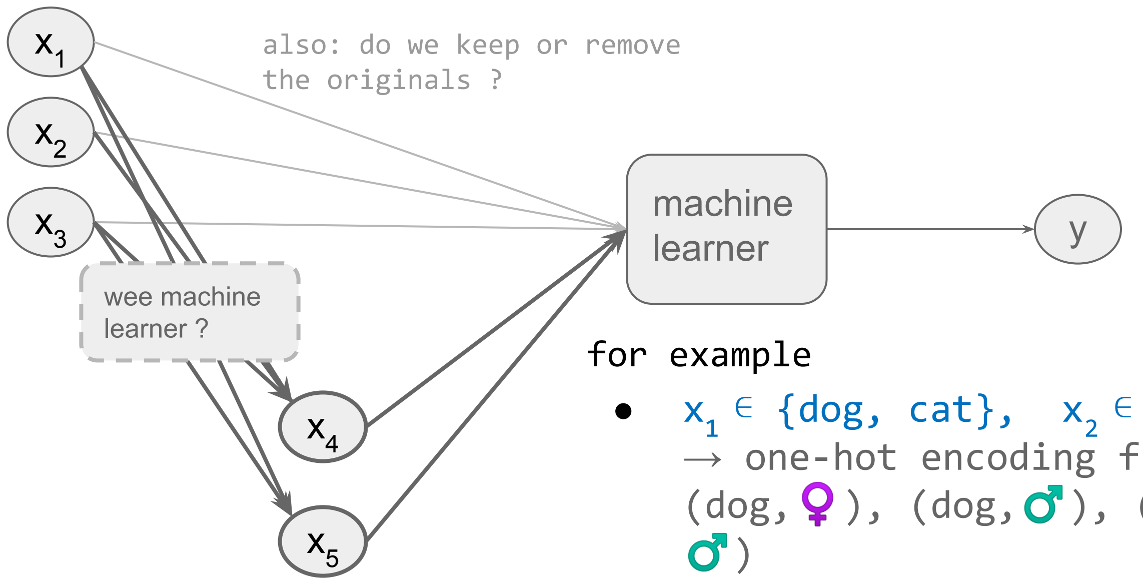


put basic features together to construct new features that can capture relationships between the basic features – the idea is to augment the feature space with these too

for example

- x ∈ {TRUE, FALSE}:
  (x_1 and x_2) or x_3
- x ∈ {reals}:
  $(x_1 + x_2) * x_3$

many-to-many



known as the cartesian product; allows learner to be "intersectional"

can consolidate redundant information across features

for example

- $x_1 \in$ {dog, cat}, $x_2 \in$ {♀,♂} :
  → one-hot encoding from (dog,♀), (dog,♂), (cat,♀), (cat, ♂)

- $x \in$ {reals}:
  → PCA? kernel PCA?
  → ICA?
  → a one-hot encoding from clustering?

many-to-one is a.k.a. “feature construction”

- we can also put basic features together to make constructed features that might capture relationships between the basic features – the idea is to augment the feature space with these too

E.g. Given base features [X_1, X_2, X_3]

linear constructions: $Xc = X_1+X_2$,

or $Xc = 4X_1+3X_2+6X_3$,

…

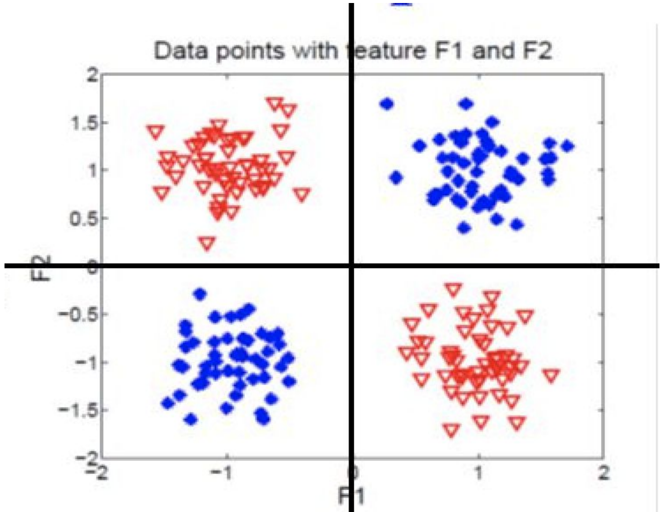
nonlinear constructions: $Xc = X_1*X_2$,

or $Xc = X_1*X_2^2$

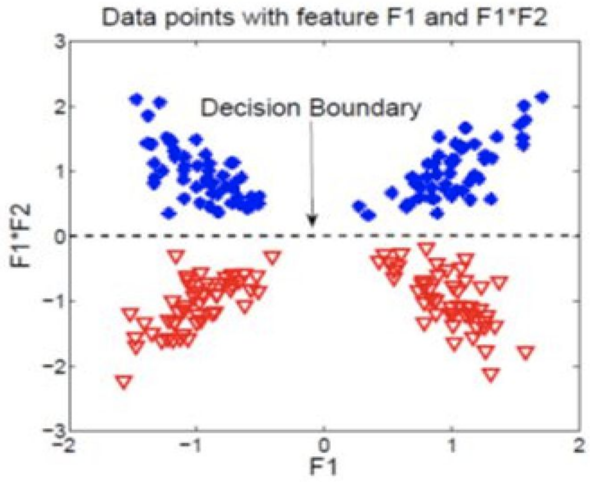
feature construction

the quality of such constructed features can drastically affect the learning performance

- -ve: Feature interactions introduce errors and add complexity
- +ve: more meaningful features that lead to more concise and accurate machine learning models



$F_3=F_1*F_2$



components of a feature construction system

Automatic feature construction systems need the following two components:

- a search strategy to search the space of possible functions (of original features)
- an evaluation mechanism to measure the goodness of a candidate function

operators for constructing new features

Boolean features:

- x1, x2 ∈ {TRUE, FALSE}
    - e.g. x3 ← not(x1)
    - e.g. x4 ← xor(x1,x2)

⇒ logical operators like negation, conjunctions, disjunctions,...

Nominal/Categorical features:

- x1 ∈ {dog, cat},x2 ∈ {rain, snow}...
    - e.g. x3 ∈ {(dog,rain), (dog,snow), (cat,rain), (cat,snow)}

⇒ cartesian product

Numerical features:

- x1 ∈ {ints}, x2 ∈ {reals},...
    - x4 ← min(x1, mean(x2,x3))

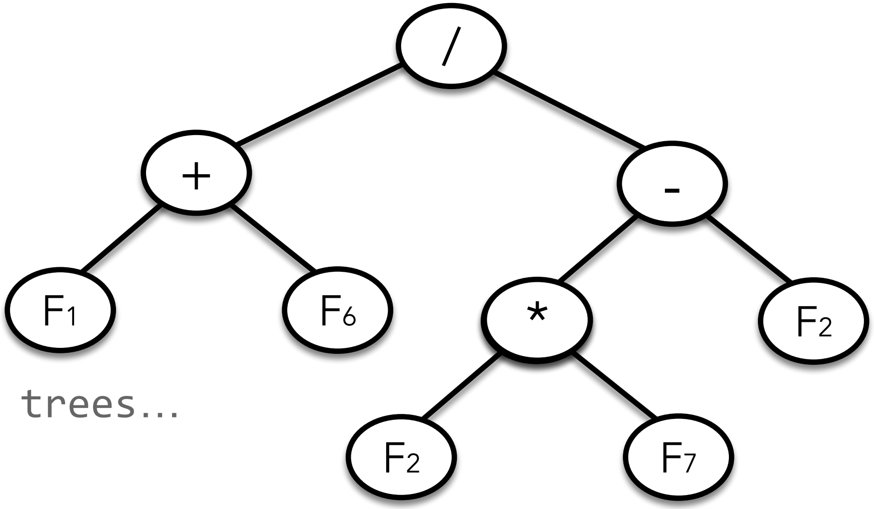
⇒ add, multiply, max, average, equals, ...

can we construct new features automatically with Genetic Programming?

Genetic Programming is a flexible way to make mathematical and logical functions

warning: there isn’t much structural (topological) information in the search space of possible functions

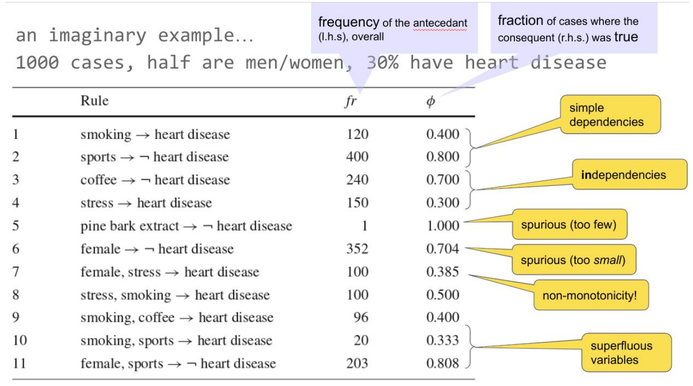
→ yes it’s possible,
but be prepared to spend a
lot of compute time trying out dud trees…



evaluating new features

- Not all constructed features are good!...

- Apply the usual feature selection techniques to remove redundant and irrelevant features
    - Q: what are the “usual” techniques?



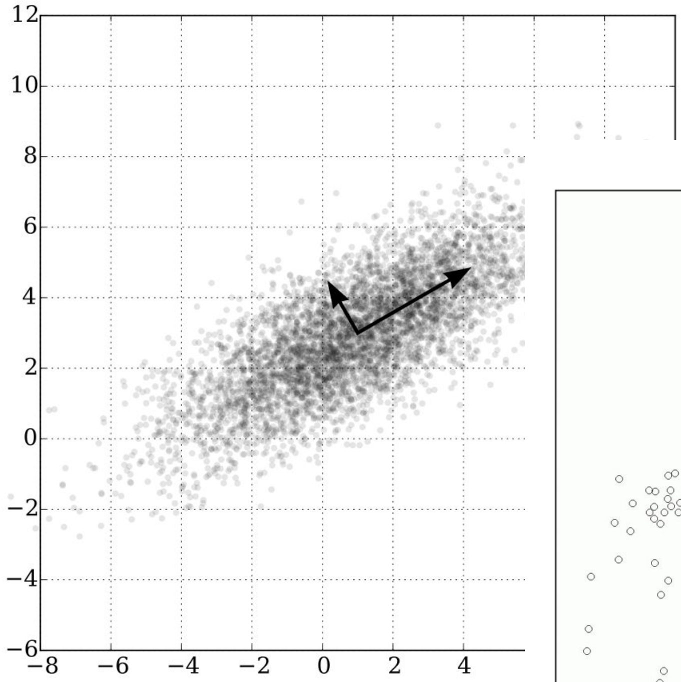
an imaginary example...
1000 cases, half are men/women, 30% have heart disease

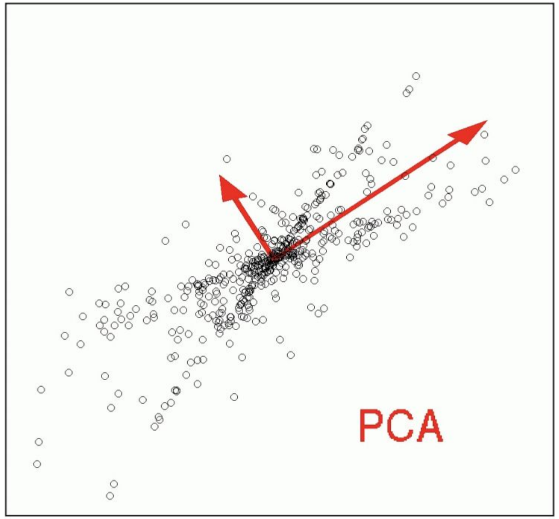
	Rule	*fr*	ϕ
1	smoking → heart disease	120	0.400
2	sports → ¬ heart disease	400	0.800
3	coffee → ¬ heart disease	240	0.700
4	stress → heart disease	150	0.300
5	pine bark extract → ¬ heart disease	1	1.000
6	female → ¬ heart disease	352	0.704
7	female, stress → heart disease	100	0.385
8	stress, smoking → heart disease	100	0.500
9	smoking, coffee → heart disease	96	0.400
10	smoking, sports → heart disease	20	0.333
11	female, sports → ¬ heart disease	203	0.808

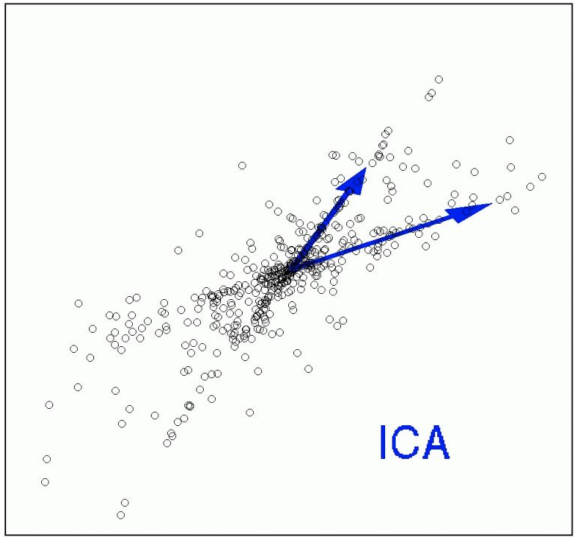
Some specific "many to many" feature methods:

- PCA
- ICA
- Kernel PCA

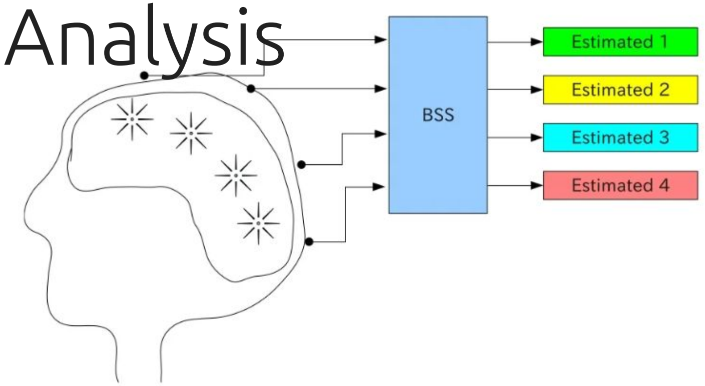
PCA (Principal Components Analysis)







ICA - Independent Components Analysis

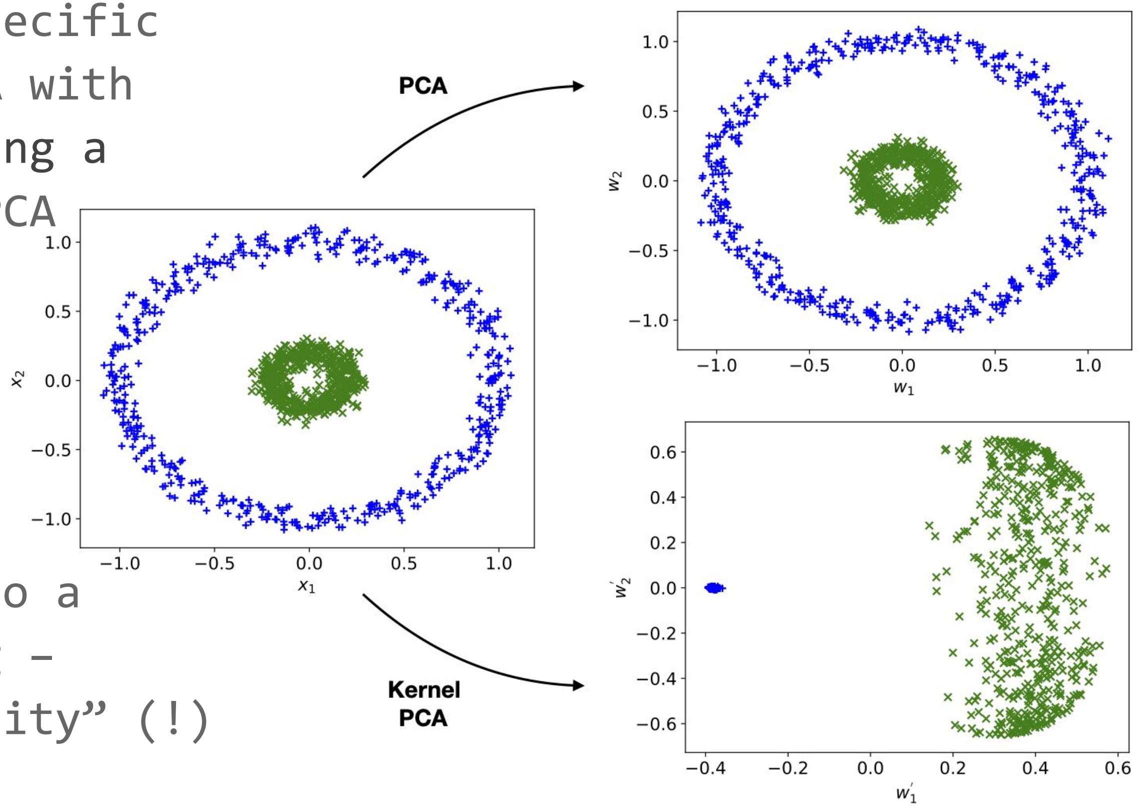


- PCA and ICA both create new components that are linear combinations of the originals
- ICA assumes the observed data arose as a linear combination of independent components (that are sadly unknown)
  - Goal of ICA: recover the components from X by finding a linear transformation
- it's like PCA but we're looking for components that are as statistically independent as possible
- PCA removes correlation, but doesn't usually find those independent components that might actually underly the data
- why might it be good to find truly independent components?

kernel PCA

- Kernel PCA combines a specific mathematical view of PCA with kernel functions, creating a nonlinear extension of PCA

- Perform PCA on data that has been transformed into a high-dimensional setting – “blessing of dimensionality” (!)



https://ml-lectures.org/docs/structuring_data/ml_without_neural_network-2.html