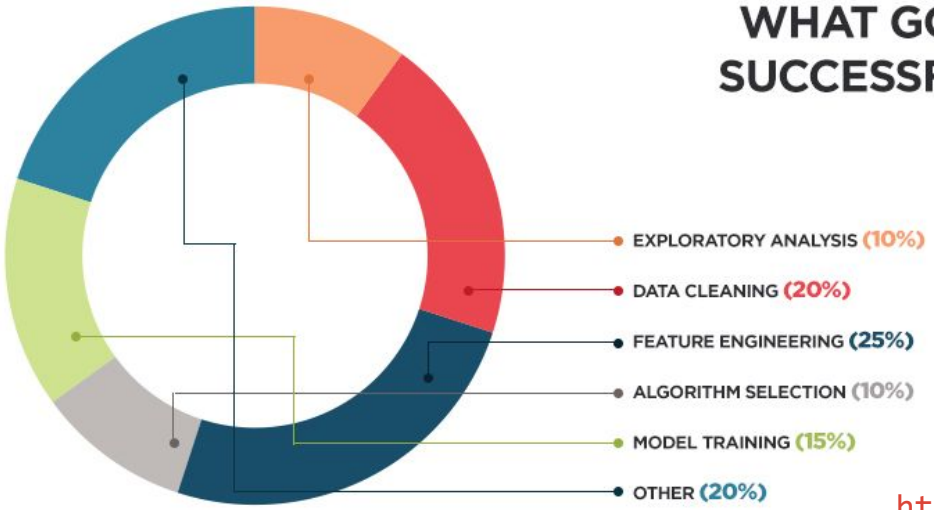
COMP309 in Week 07, 2024

feature engineering

Marcus Frean marcus@ecs.vuw.ac.nz

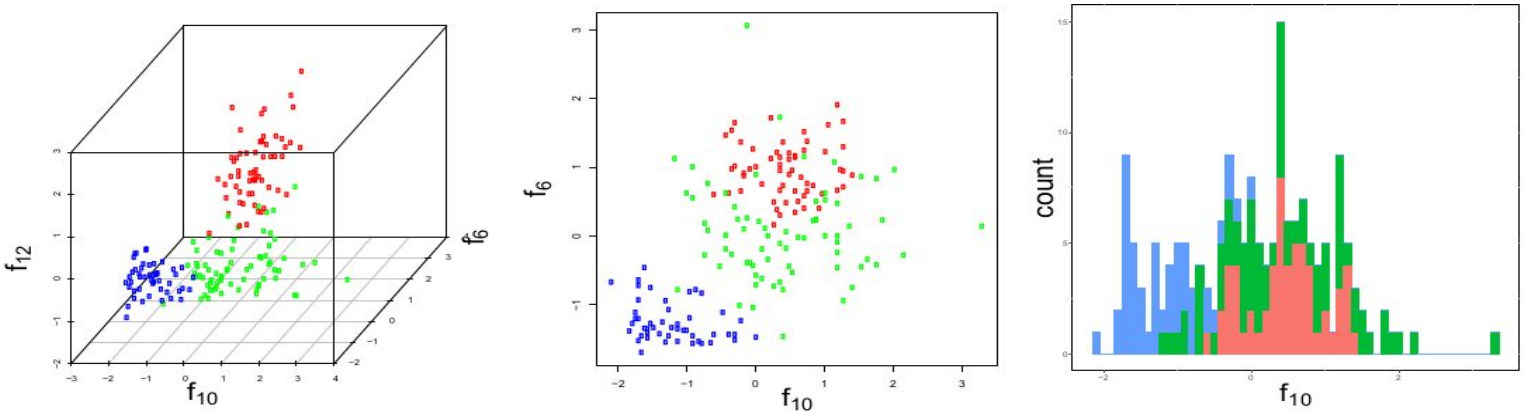
Q: how do you get the most out of your data, for predictive modelling?

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



https://elitedatascience.com/feature-engineering

More features = good?

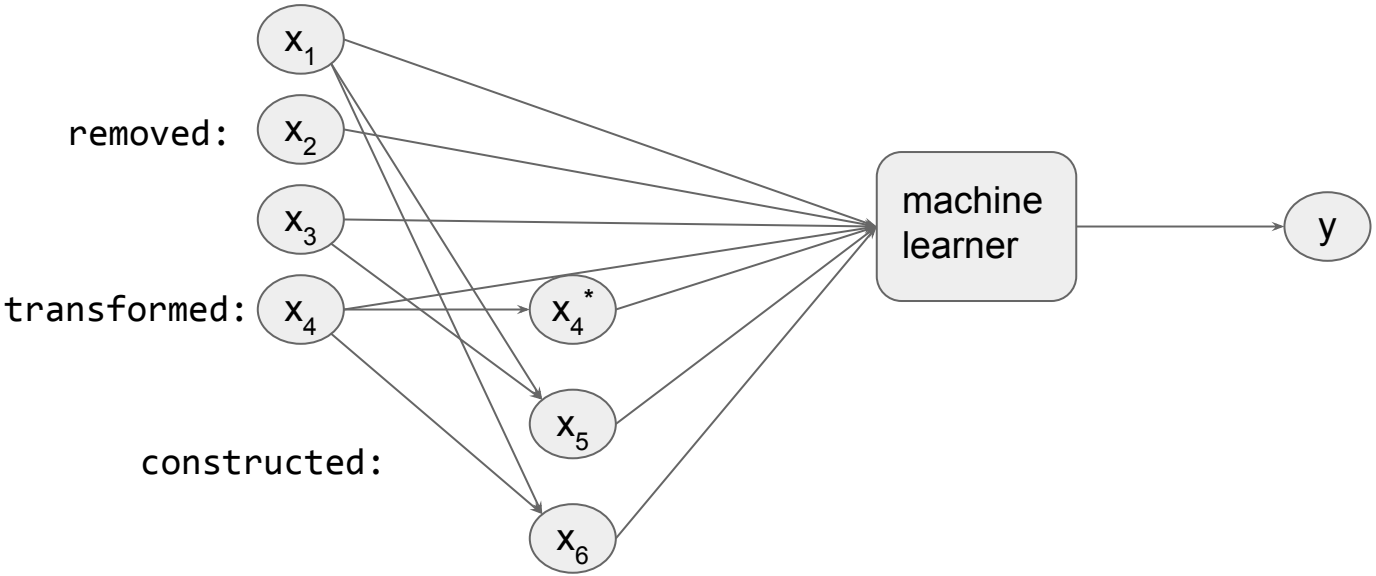


(a) 3 features: f_{10}, f_6, f_{12}. (b) 2 features: f_{10}, f_6. (c) 1 feature: f_{10}.

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

e.g. plot 3d demo

changes to features

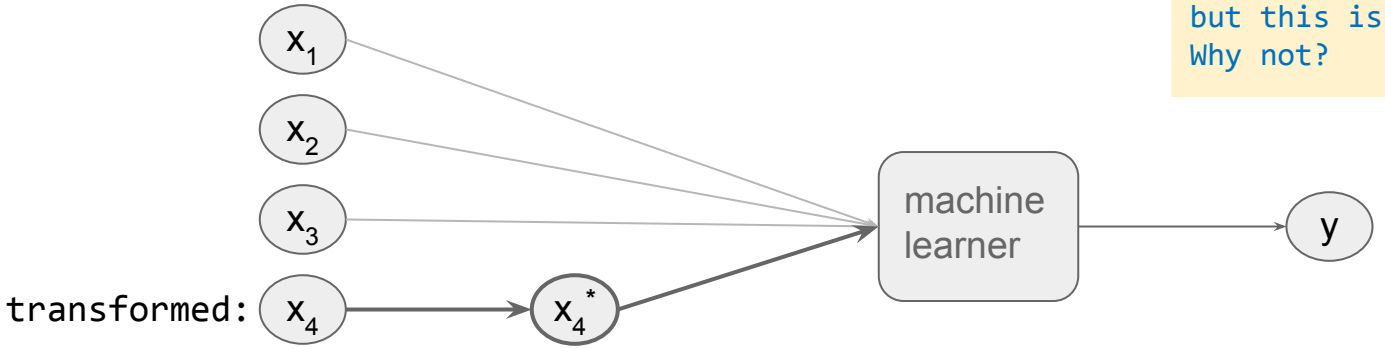


one-to-one

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

```
dog    → 1
cat    → 2
ferret → 3
```

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

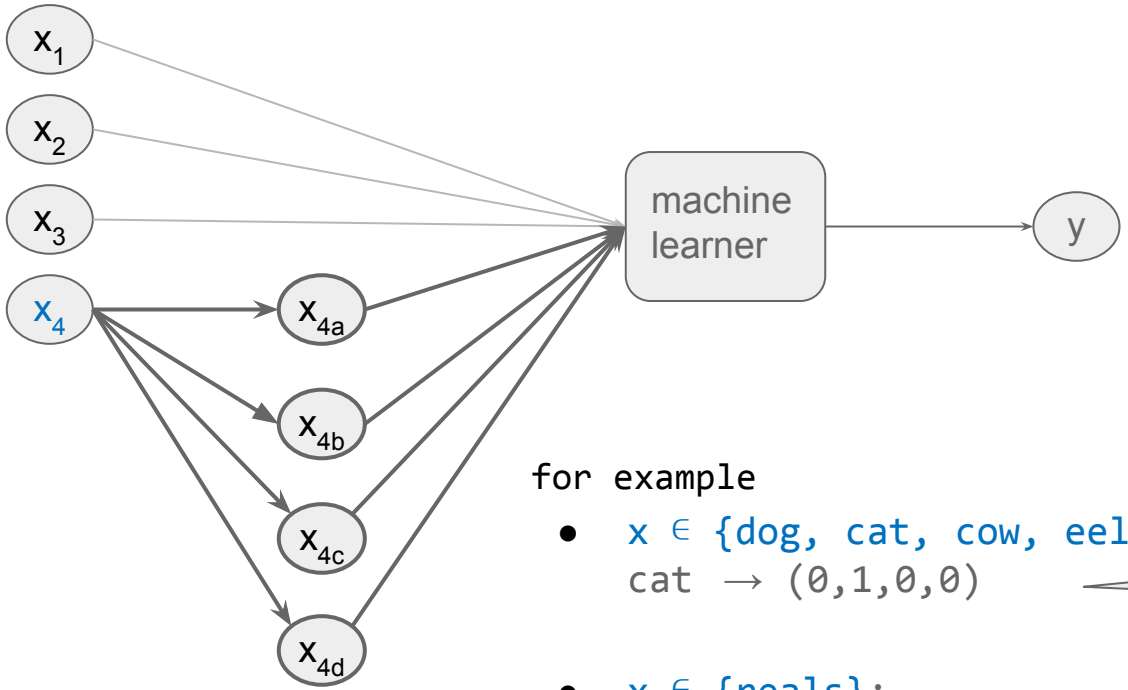


can get the original feature into a more useful range for the learner

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

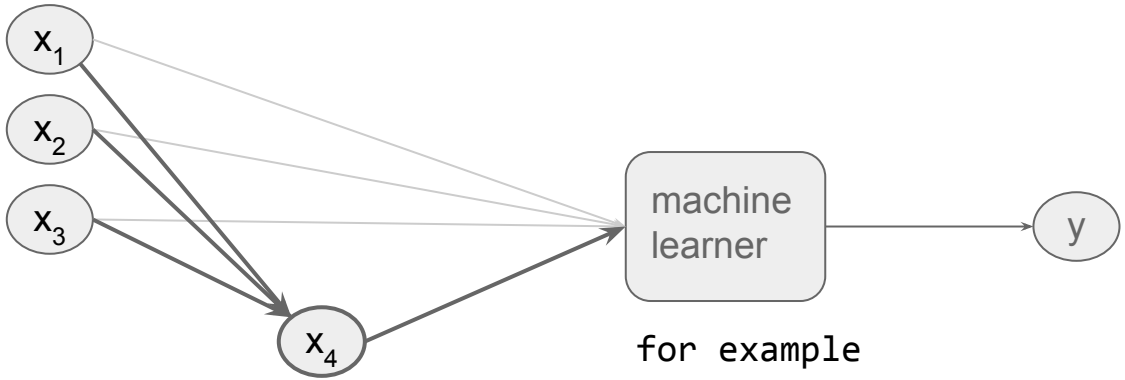
- $x \in \{dog, cat, cow, eel\}$:
  $cat \rightarrow (0,1,0,0)$

  "one-hot" encoding

- $x \in \{reals\}$:
  $x \rightarrow (x, x^2, x^3, x^4)$

  polynomial expansion

many-to-one

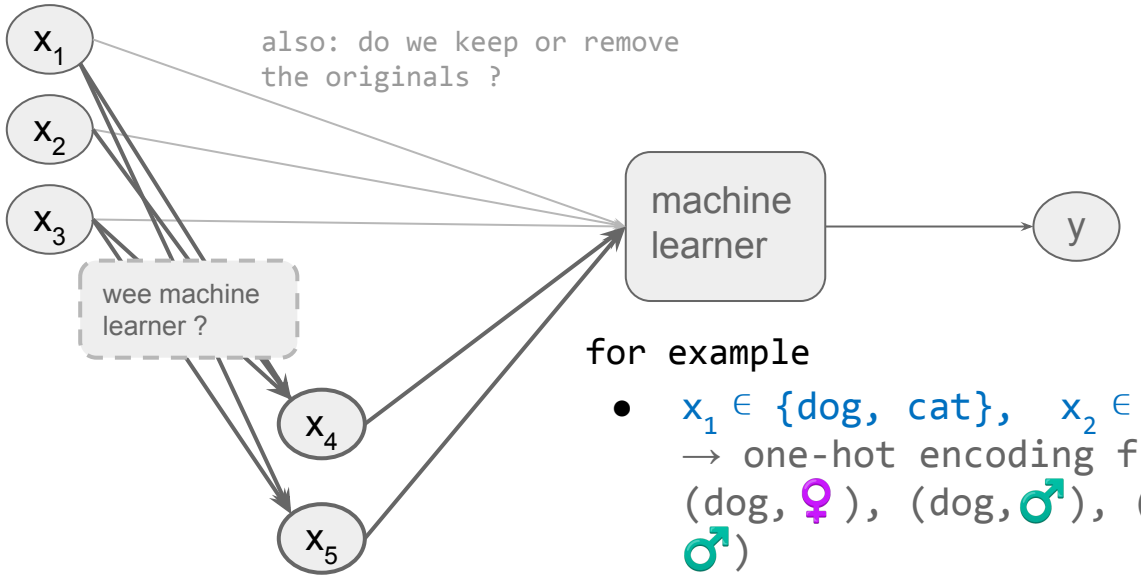


put basic features together to construct new features that can capture relationships between the basic features – the idea is to augment the feature space with these too

for example

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

many-to-many



known as the cartesian product; allows learner to be “intersectional”

for example

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

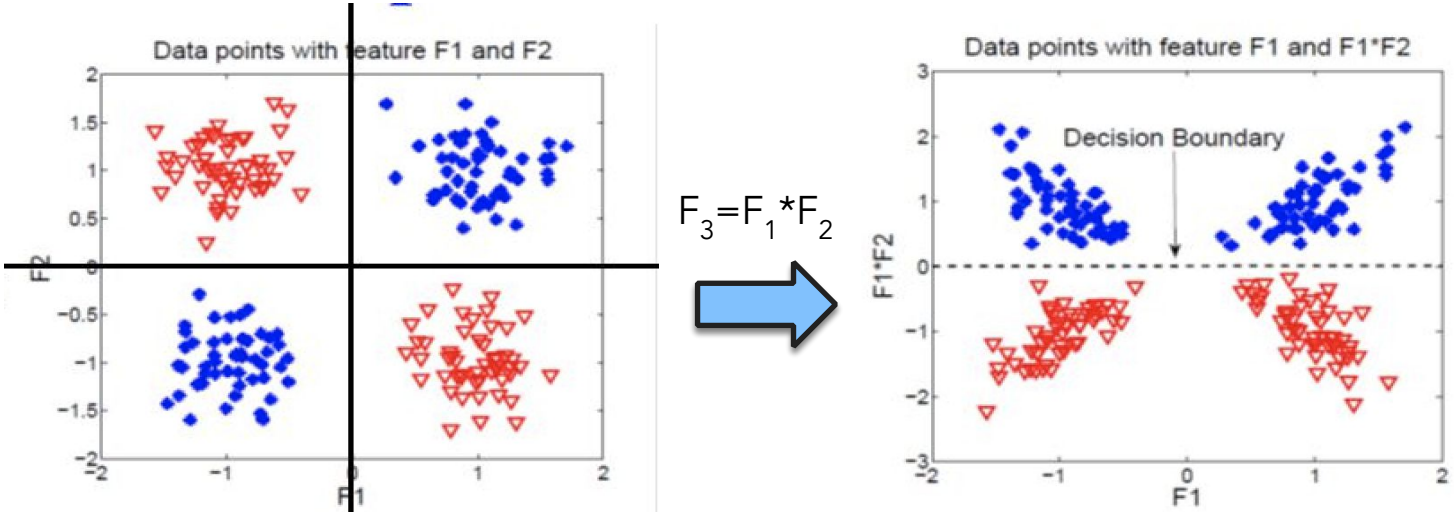
can consolidate redundant information across features

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

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



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:

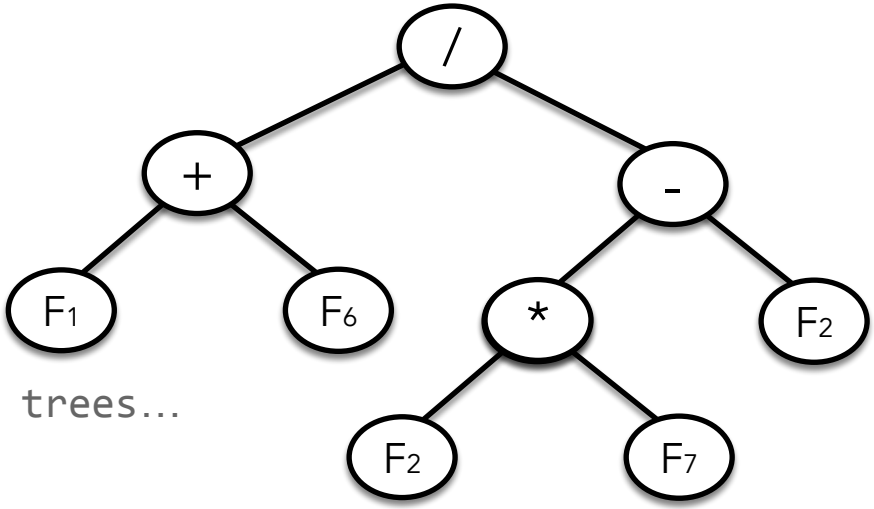
- x1 ∈ {ints}, x2 ∈ {reals},...
  - x4 ← min(x1, mean(x2,x3))

⇒ 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…

Some specific “many to many” feature methods:

- PCA
- ICA
- Kernel PCA

PCA (Principal Components Analysis)