Life Sciences and
Facility Management

Institute of
Computational Life Sciences

Feature engineering

Machine Learning

Norman Juchler

The data science workflow

- Problem definition
- Data acquisition
- Preprocessing
- Exploratory data analysis
- Feature engineering
- Modeling and training
- Model evaluation
- Model deployment
- Reporting

The data science workflow

- Problem definition
- Data acquisition
- Preprocessing
- Exploratory data analysis
- **Feature engineering**
- Modeling and training
- Model evaluation
- Model deployment
- Reporting

- Transform raw data into a format that is more suitable for modeling.
- Extract features that are sensitive to the target variable(s), increase the signal to noise ratio
- Feature engineering is important for model accuracy, reducing overfitting, and enhancing the generalization capability of models
- Represents one way of introducing inductive bias into the model, and to exploit prior knowledge.

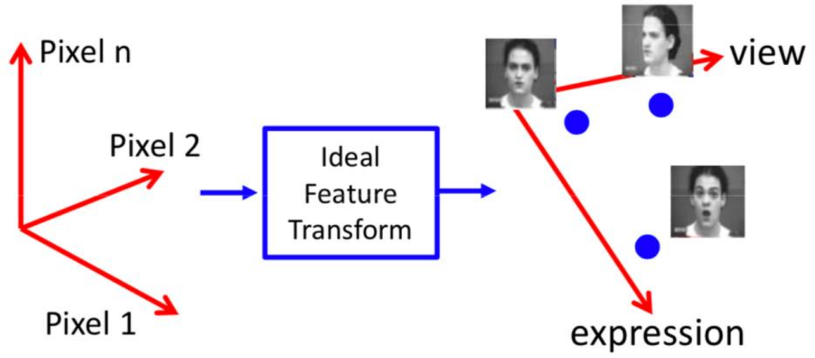
“Coming up with features is difficult, time-consuming, and requires expert knowledge. Applied machine learning is basically feature engineering.”

—Prof. Andrew Ng

Efficient representations

- Humans are often so good at extracting relevant features from the input that we are not even aware that we are seeing the world in a **latent space**.

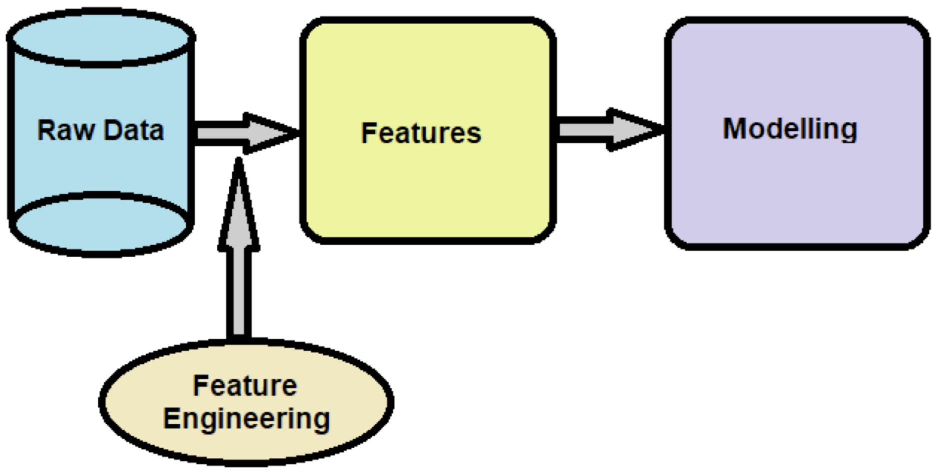
- Example: Human visual perception
  - Orientation of a human face, or facial expressions.
  - Brain extracts highly processed, derived features which might even incorporate information not contained in the input data, like common sense, cultural and intuitive psychological knowledge.



- **Latent space**: a compact representation of the data derived from the original data, that captures the most salient and abstract information about the data.

Feature engineering

- Feature engineering refers to the creation of derived features from the input features.



- **Objectives**:
  - Increase signal-to-noise ratio
  - Increase separation power (for classification tasks)
  - Minimize loss of information relevant for the ML task

- Feature engineering thus may incorporate commonsense and domain knowledge

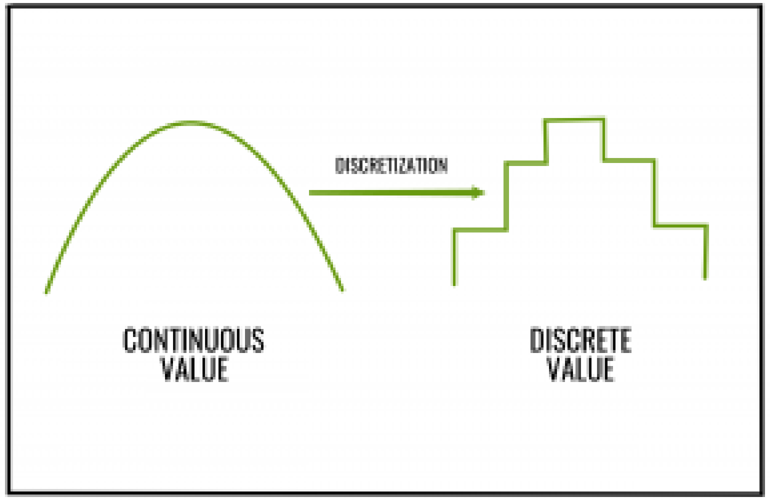
Feature engineering process

- Brainstorming or testing features
- Deciding what features to create
- Creating features
- Testing the impact of the identified features on the task
- Improving your features if needed
- Repeat

Discretization

- Discretization groups sets of values together into bins.

- Examples:
  - Dates → Days of the week
  - Days of the week → Weekday / Weekend
  - Binning of continuous variable into a small number of intervals
  - Use of a simple decision tree to sort data into subgroups



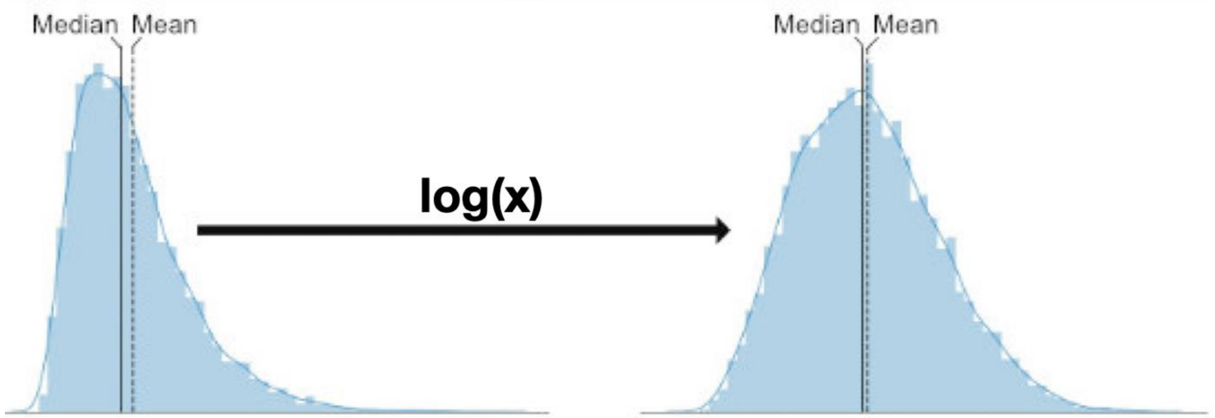
- Advantage: Increase in signal-to-noise
- Disadvantage: Loss of granularity

Simple transformations

- New features can be generated by applying simple mathematical formulas

- Examples:
  - Square of a single variable: Expand larger values
  - Logarithm of a single variable: Expand smaller and compress the larger values
  - Additive combinations: Sum of two or more variables
  - Multiplicative combinations: Product of two or more variables

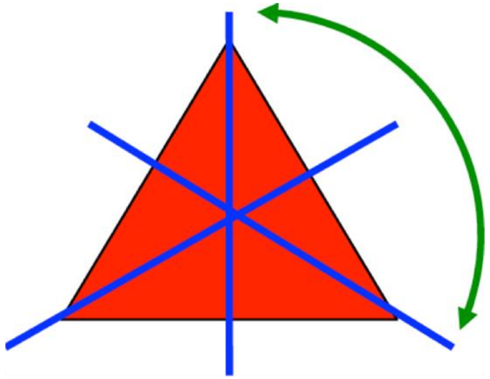
- Although directly derived from the input, simple transformations can have an important effect on model performance.



Symmetries in the input data

- A reliable way to increase the signal-to-noise ratio is to make use of symmetries that might exist in the underlying problem.

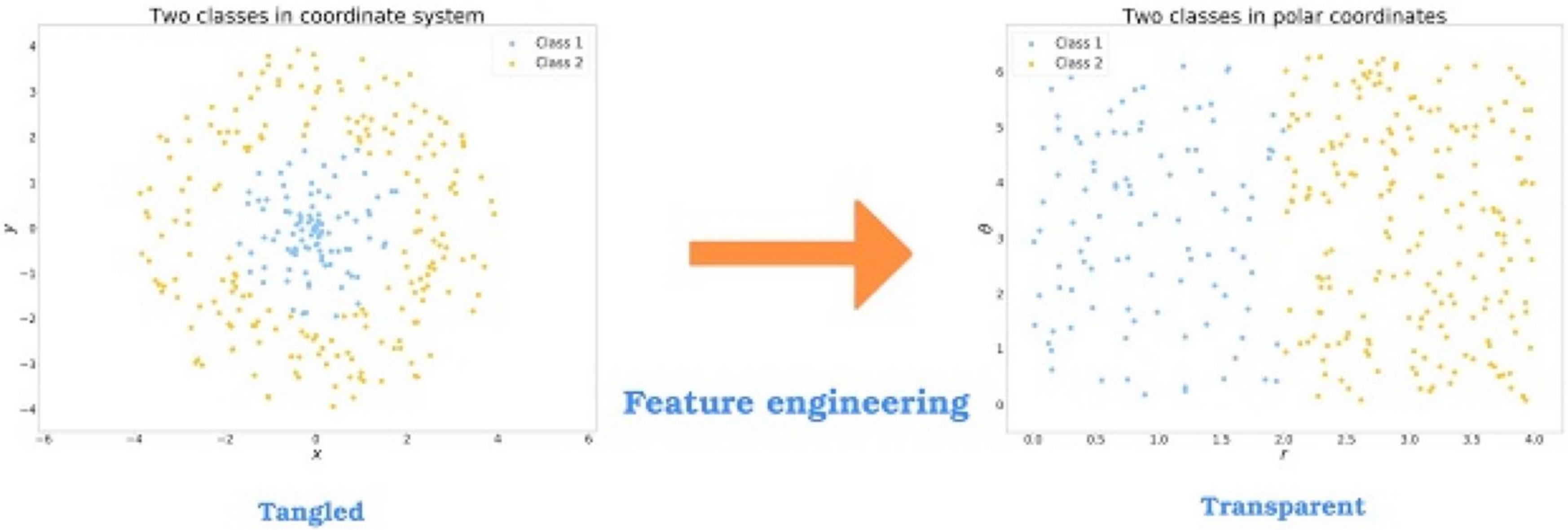
- Examples:
  - Weekly patterns
  - Symmetry of human faces
  - Left-right-symmetry of proteins



- Deviations in symmetrical features can indicate special information (e.g., presence of anomalies).

Coordinate transformations

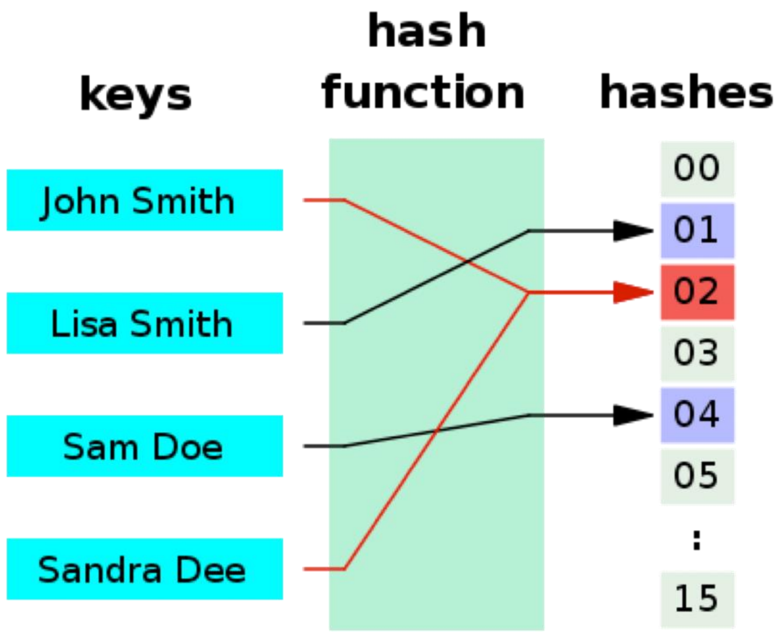
- Once we have found symmetries, we may exploit them to **disentangle** the data.
- Example: **radius** on x axis, **angle** on y axis



$$r = \sqrt{x^2 + y^2} \quad \theta = \arctan \frac{y}{x}$$

The hashing trick

- Hashing is a fast and space-efficient way of vectorizing features.
- Idea:
  - Turn arbitrary features into indices.
  - Use a hash function which maps features of arbitrary type and number to a fixed number of index values.



- Potential problem:
  - This may potentially result in collisions
  - However, their probability can be made very small by choosing the right hash function

Feature selection (Recap)

- Reasons why removing some features might be useful:
  - Some features may not contain any relevant information for the task at hand
  - Increased signal-to-noise ratio can improve model performance
  - Models of reduced complexity can be more robust and easier to understand
  - A smaller model is usually faster in training and prediction.

What do you think is the difference between feature selection and dimensionality reduction?

Historical note

- **Early days** (pre-deep learning era)
  - Handcrafted features. Before deep learning, feature engineering was largely manual, relying on domain expertise to create informative features tailored to specific problems.
  - Algorithm dependence. Features were sometimes designed to suit specific algorithms (e.g., kernel transformations and SVM, linear features for logistic regression)
  - High importance: Was often one of the most critical factors for ML success, often outweighing the choice of algorithm
- **Emergence of deep learning** (2010s)
  - Shift to automated feature learning: DL models can learn hierarchical representations of data directly from raw inputs, reducing the reliance on manual feature engineering.
  - End-to-end learning: Deep networks enabled end-to-end training pipelines where raw input data could directly map to predictions without handcrafted features.
- Still relevant today, where...
  - ...interpretability / explainability plays a role
  - ...one aims to incorporate domain / expert knowledge into ML models