Week 1 - Introduction to Applied ML

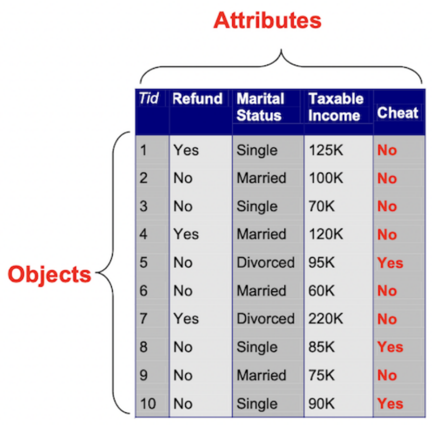
Data

What is data?

Most data comes in a **data frame** like the one on the right.

- Columns are per-instance attributes:
  - e.g. age, height, eye color, column ID, ...
  - Also called: feature, variable, field, ...
- Rows are instances
  - Also called: object, data point, sample, ...

Several steps before $\mathbf{X} \in \mathbb{R}^{N \times P}$...!


Attributes
Objects

Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Data types

- Numerical (real-valued)
- Categorical
- Ordinal
- Dates and times
- Coordinates (geospatial...)
- Text
- Unique identifiers
- Images, video, audio, ...

Two types of data sources

Cheap (infinite) data

Predict observable events

- Ad clicks
- Product recommendations
- Stock market
- House numbers
- Transcription autocorrect
- ...

Expensive data

Automate complex processes

- Medical diagnosis
- Drug trial
- Microchip design
- ...

Big data headaches

More data is not always better

- Is dealing with the large size worth the hassle? Data scientist / analyst time is expensive!
  - (can you get away with just using a subset that will fit in RAM?)
- Where did the data come from? Is it representative of future data?
- Would a smaller dataset of higher-quality data be more useful, or less useful?
  - (this can be situation dependent!)

Feature Pre-processing

Numerical feature pre-processing

Pre-processing

- Scaling and centering (whitening)
- Rank tranformation
- Clipping
- Non-linear transformations (e.g. $\log$ or $\sqrt{\ldots}$)

Scaling and centering are not desirable for **sparse data** because these transforamtions **remove the sparsity** of the data.

Categorical feature pre-processing

- One-hot encoding
- Label encoding
- Frequency encoding
- Target encoding
- Learn embeddings

	boro	salary	vegan
0	Manhattan	103	No
1	Queens	89	No
2	Manhattan	142	No
3	Brooklyn	54	Yes
4	Brooklyn	63	Yes
5	Bronx	219	No

Categorical feature pre-processing

Other ideas:

- **mean** encodings replace the categorical variable with some other statistic, e.g. the mean of the target variable
  - ▶ . . . many potential pitfalls
- **Frequency** (or count) encodings record how often a particular value occurs
  - ▶ . . . e.g. “Bag-of-words” representations for text
  - ▶ We’ll come back to this later in the module
- Learned encodings / **embedding layers** in neural networks map each discrete value onto a vector in $\mathbb{R}^D$
  - ▶ . . . a good option if trained end-to-end with the rest of the model
  - ▶ common in NLP (e.g. word or character embeddings)

Generalization

Central goal of machine learning

In machine learning, we care about the error on test data!

This is different than the error on our **training** data.
Analogy to sitting an exam:

- Training error is the score on a practice exam
- Test error is the score on the real exam
- We want to do well on the **real** exam, not the practice one.

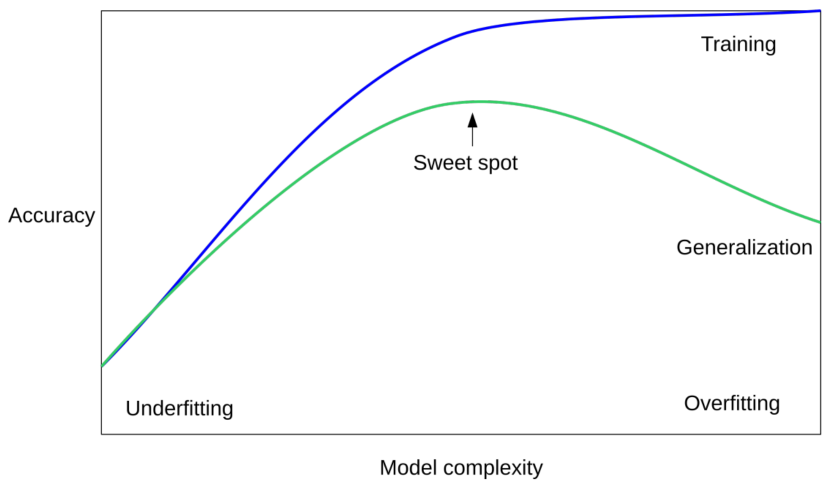
Memorizing the practice exam will produce a low training error.
Learning is necessary to also have a low test error!

Why does machine learning work?

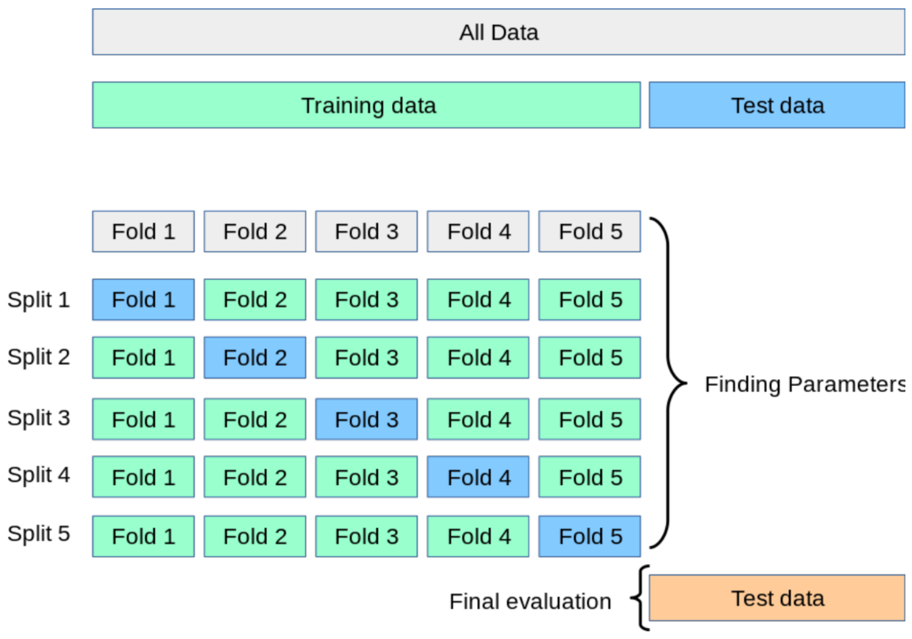
Fundamental assumption: **Test data is similar to training data**

- If the training data and test data are unrelated, then we can’t learn anything
- Generalization requires assumptions!

Training error and generalization



Cross-validation

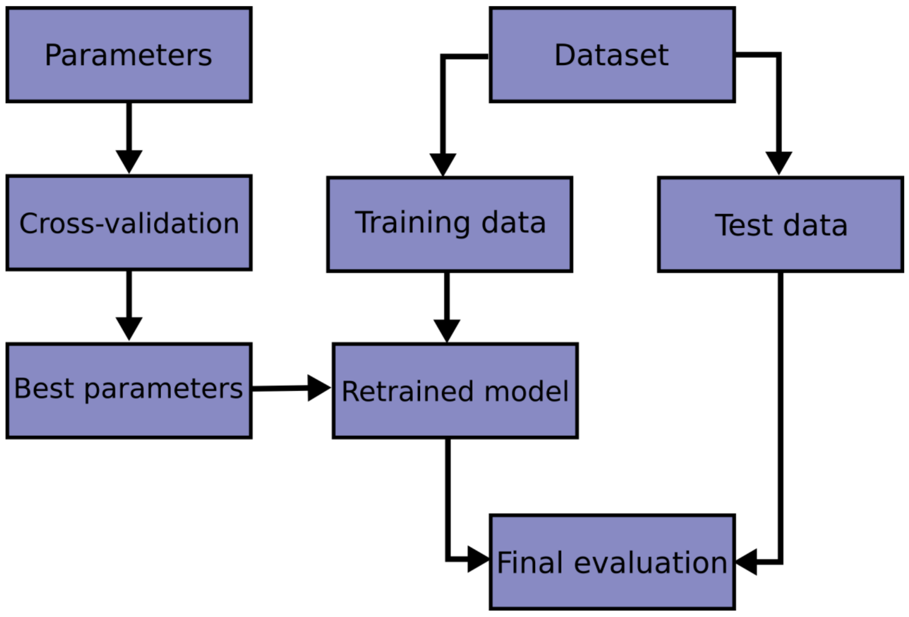


Validation sets for non-iid data

Data is often not independent and identically distributed (*iid*)! A **random split** might not be appropriate.

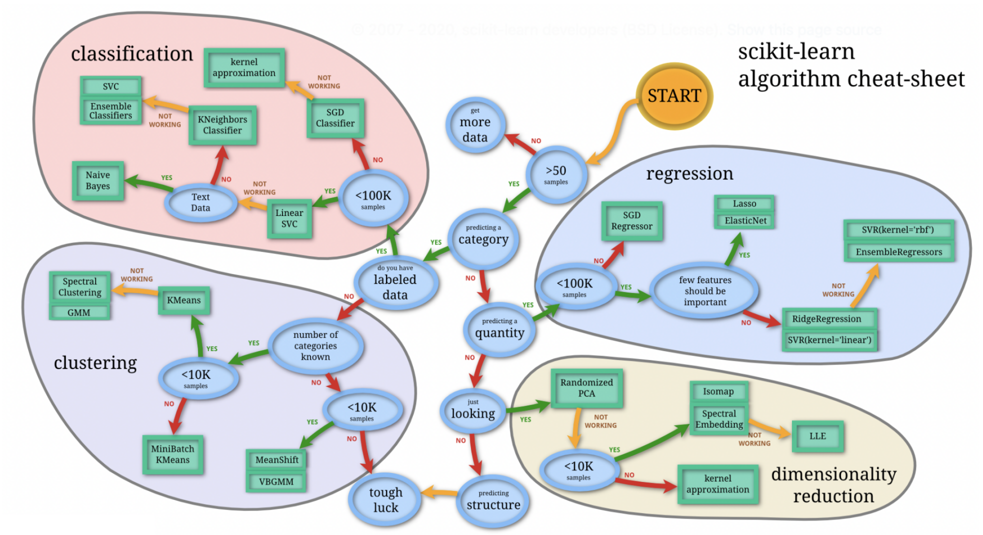
- **Grouped data**: e.g. data collected from different cities, classrooms, etc, where we expect to see higher correlation within-group than between groups
  - **Stratified sampling** of cross-validation sets
- **Time-series data**: in a random split, you are effectively testing interpolation between points, instead of extrapolation into the future
  - Temporal splits ensure validation / testing only on held-out future data
  - **Careful:** more data than you think is actually time series data! Data is typically collected sequentially over a long period of time (even if you don't know it)

Cross-validation



Model Selection

Machine learning as a bag of tricks



Evaluation Metric

Accuracy: what fraction of examples are classified correctly?

$$\frac{1}{N}\sum_{i=1}^{N}\mathbb{I}[f(x_i)=y_i]$$

- Hard predictions! Does not take into account uncertainty
- Can't be optimized with gradient-based methods
- Misleading for imbalanced classes

Two classes: positive and negative.

- **Precision**: "How many of those predicted positive are actually positive?"

$$\frac{\text{true positives}}{\text{true positives} + \text{false positives}}$$

- **Recall** or **sensitivity**: "How many of those which are actually positive are correctly predicted as positive?"

$$\frac{\text{true positives}}{\text{true positives} + \text{false negatives}}$$

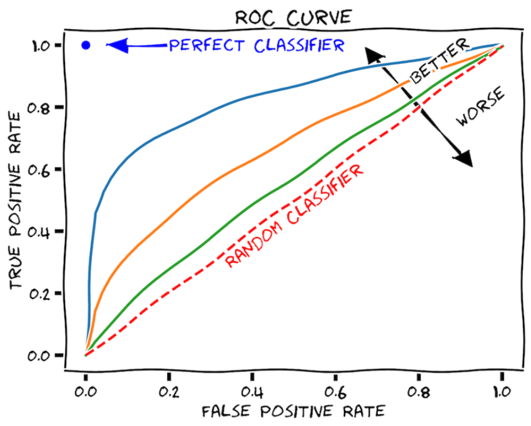
- **Specificity**: "How many of those which are actually negative are correctly predicted as negative?"

$$\frac{\text{true negatives}}{\text{true negatives} + \text{false positives}}$$

Solution: look at the area under the curve (**AUC**)

- True positive vs false positive
- Precision vs recall
- ...

Exercise: think about how this is affected by class imbalance



The **ROC curve** is more stable when applied to datasets with **different label distributions.** On the otherhand, the **precision-recall curve** changes rapidly with the distribution of labels.

Therefore, use the P-R curve when you want to **visualize the variation** in model performance on different datasets.

Moreover, this means that the ROC curve is not a good evaluation metric for **imbalanced dataset**, because the **false positive rate** does not drop drastically when the total negative is huge. In other words, when positive is minority, the same false positive rate suggested by the ROC curve actually means **a lot more false positive cases** (i.e. the model is bad).

Instead, **precision** is highly sensitive to flase positive while not being affected by a large total negative denominator. Therefore, the P-R curve is a better fit when the dataset is imbalanced.

Classifier performance: log-loss

Maximum likelihood estimation:

- Define $f_\theta(x_i)$ which returns a value in the range $[0, 1]$
- Interpret this output as a probability $p(y_i = 1|x_i; \theta)$
- Optimize the parameters θ by maximizing the (log-)probability of the labels given the data:

$$\begin{aligned}\hat{\theta} &= \arg\max_\theta \sum_{i=1}^{N} \log p(y_i = 1|x_i; \theta) \\ &= \arg\max_\theta \sum_{i=1}^{N} y_i \log f_\theta(x_i) + (1 - y_i)\log(1 - f_\theta(x_i))\end{aligned}$$

This (with flipped sign) is also called **binary cross-entropy** loss.

Most of the above evaluation metrics are **non-convex**, making them not available to be used for model training with **gradient descent**. Instead, the **binary cross-entropy**, or **log-loss**, is a common used loss function based on the **maximum likelihood estimation.**

Mean squared error (MSE):

$$\frac{1}{N}\sum_{i=1}^{N}(f(x_i) - y_i)^2$$

- Easy to optimize directly! (gradients, second-order methods)
- RMSE $= \sqrt{\text{MSE}}$
- Analogous to assuming normal-distributed errors
- Exercise: what is the best constant predictor?

Mean absolute error (MAE):

$$\frac{1}{N}\sum_{i=1}^{N}|f(x_i) - y_i|$$

- Also easy to optimize directly (though not by second-order methods)
- Less sensitive to outliers
- Analogous to assuming Laplace-distributed errors
- Exercise: what is the best constant predictor?