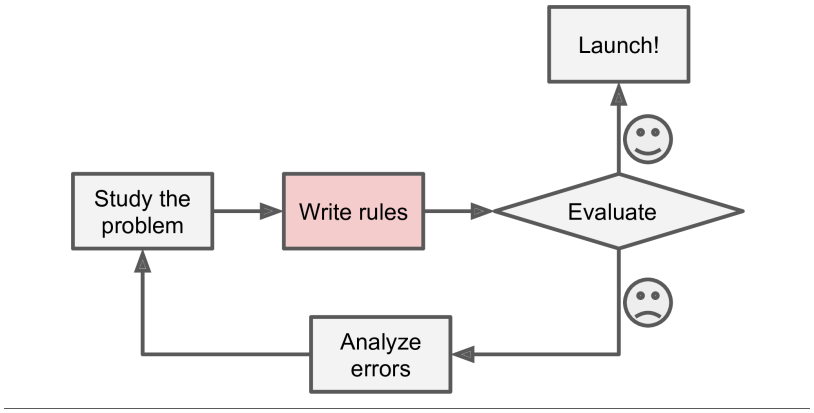
Machine Learning Workflow

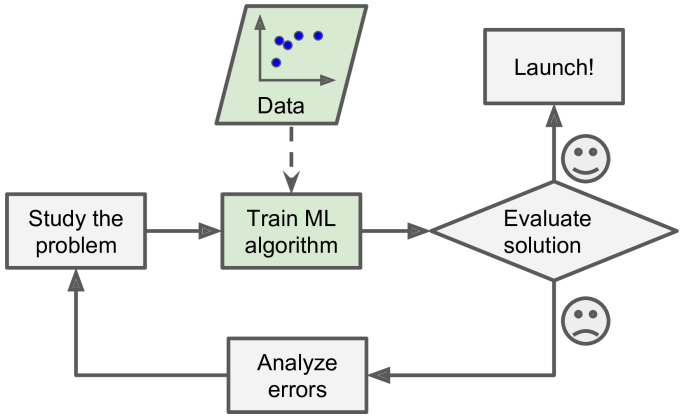
Using Machine Learning Tools

Géron, Chapter 1

What is machine learning?

- Imperative programming: develop ***algorithm*** to ***solve*** a problem and then write this as an explicit program for the computer
- Machine learning: program a computer to learn from the data ***how*** to solve the problem

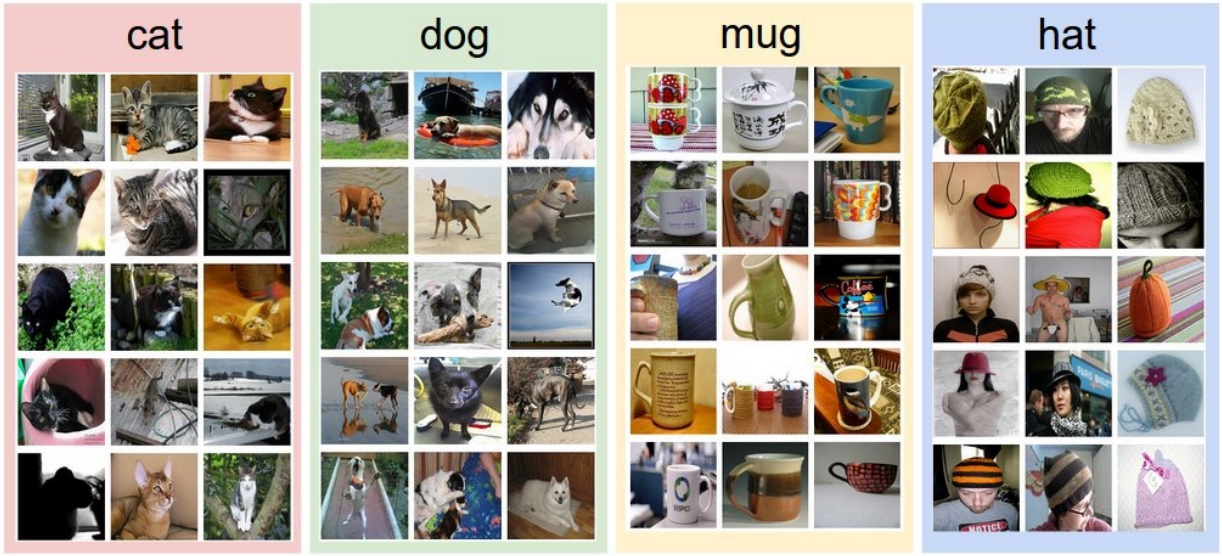




From Géron, Hands-On Machine Learning

Machine Learning Examples

- Junk email detection
  - avoids hand tuning of parameters
  - adapt to new environments and settings
- Visual recognition
  - complex problem with no known solution
  - analysis and mining of large data sets

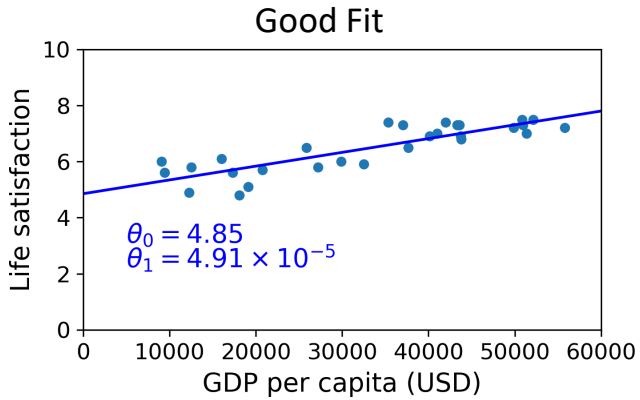


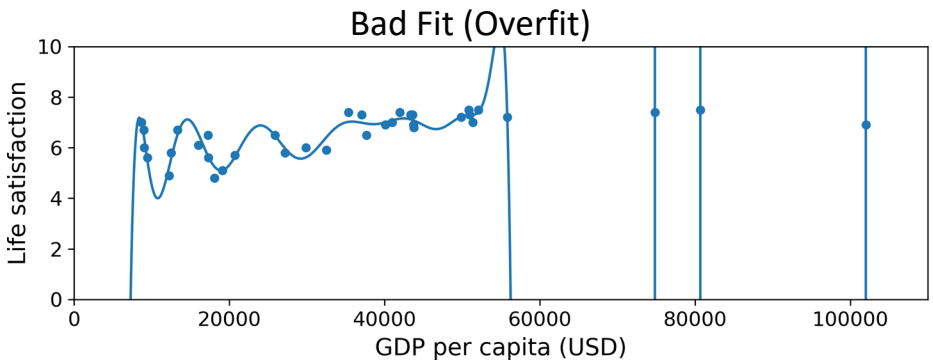
Types of ML: Supervised Learning

- **Supervised** learning means your training data includes **labels** or **values**
  - i.e. the desired result
- One type is **classification**
  - Uses discrete labels, for example:
    - Predict if a new email is junk, based on previous emails that have been marked as junk or not
    - Predict the type of animal in the image, based on large dataset of labelled animal images
    - Predict the topic of a paragraph, based on previous examples (e.g. news articles, adverts)
    - Predict whether a person has a disease or not, based on many examples of medical images (healthy & with disease)

Types of ML: Supervised Learning

- Another type is **regression**:
  - Uses real values to predict a real number, based on training data that contains values and possibly other features



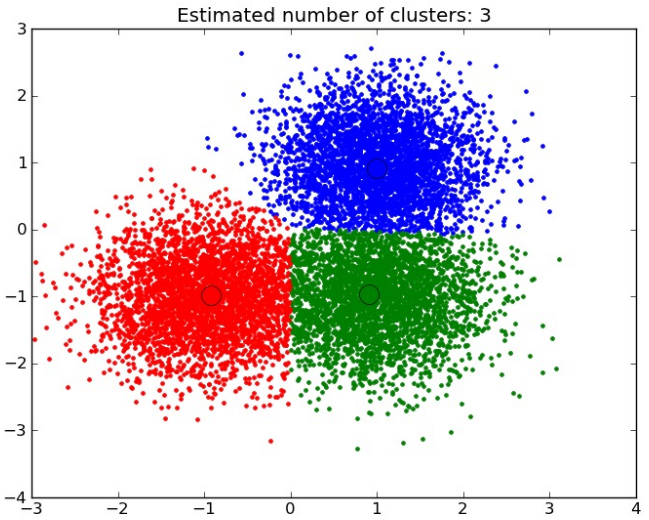


From Géron, Hands-on Machine Learning

  - For example:
    - Predict life expectancy, based on historical/medical records
    - Predict next month's sales, based on records

Types of ML: Unsupervised Learning

- Unsupervised learning means your training data does ***not*** include labels or values to predict
- For example:
  - Clustering data
    - Market research
    - Image segmentation
  - Dimension reduction
    - Visualization
    - For preprocessing
  - Anomaly detection
    - Fraud detection
    - Security
    - Pathologies in medical data

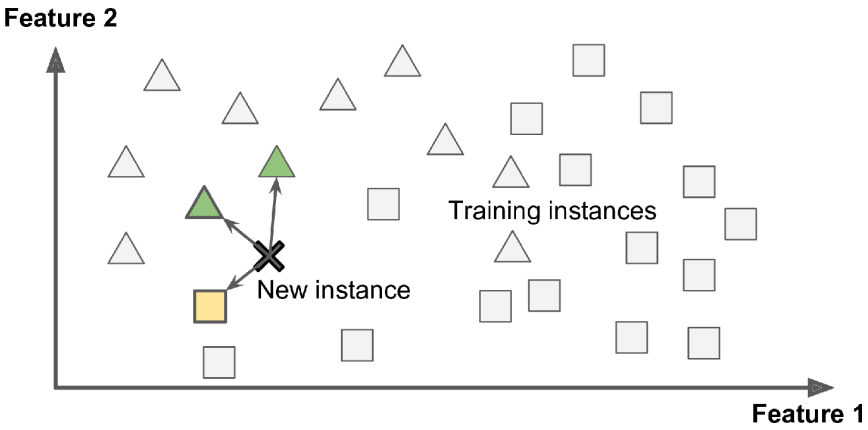


Types of ML: Reinforcement Learning

- Algorithm actively collects data, by ***interacting with the environment***
  - environment provides reward signals
  - learns a **policy** to maximise reward over time
- For example:
  - playing games

Approaches to ML

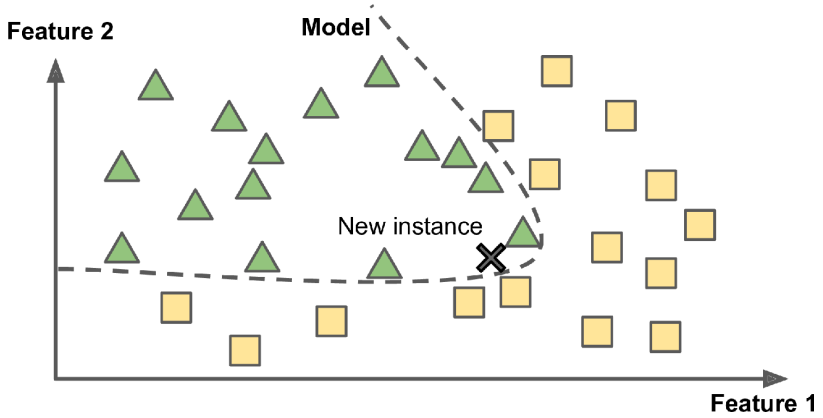
- **Example** based approach:
  - Find similar examples in existing data
  - e.g. nearest neighbours: follow the example of nearby training data points
    - K-nearest-neighbours (KNN) – majority or average of k closest points



From Géron, Hands-On Machine Learning

Approaches to ML

- **Model** based approach:
  - Fit a model to existing data
    - e.g. linear model, support vector machine, deep neural network, …
    - "Fitting" = training = optimizing model parameters
  - Apply the fitted/trained model to new data



From Géron, Hands-On Machine Learning

Approaches to ML

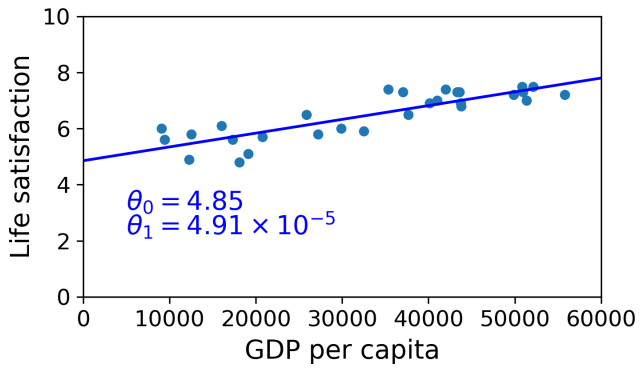
- ML includes many types of task and possible algorithms, but all rely on a key assumption:

- Existing data will apply (**generalise**) to new data
- In other words, we can **predict (infer)** outcomes for new data, based on existing data
  - Goal is to measure **generalisation error** – how well does this assumption hold?

- What are some situations or tasks that are a challenge for generalisation?

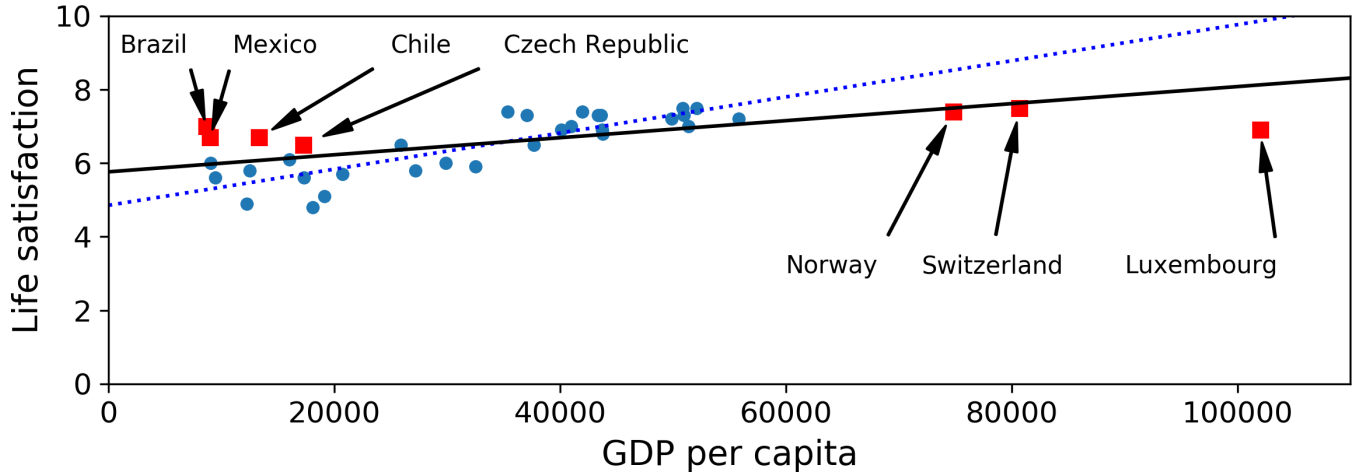
Problems with data

- **Quantity** of data
  - How much is enough?
  - Depends on the complexity of the problem and the dimension/richness of the data
    - e.g. 2D line fitting vs visual recognition



Problems with data

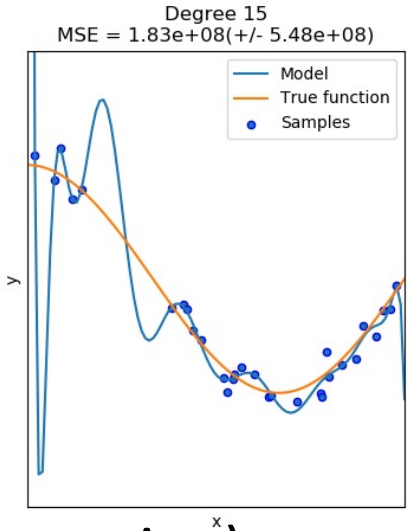
- **Range** or domain of data
  - does the available data represent (or cover) future cases that you want to predict?



Problems with data

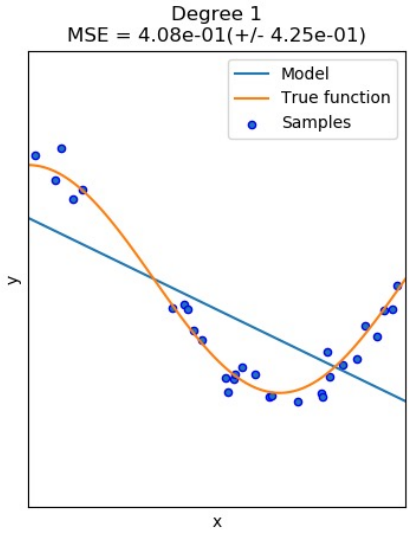
- Real data contains **noise, errors and outliers**
  - How to deal with this?
    - Detect and discard?
    - Attempt to fix or fill *(impute)*?
- Some features are less useful than others
  - Less related to the target you want to predict
  - We can select or combine features using *dimensionality reduction* methods
    - e.g. measure correlation with the target value

Problems with models



- **Overfitting**
  - *model is too complex for the data*
  - essentially fitting to noise in the data
- Could treat as a data problem:
  - collect more data, or reduce noise (e.g. averaging)
- Or as a problem with your algorithm/model:
  - choose a less complex model (fewer parameters)
  - **regularisation**
    - "soft" reduction in parameters
    - penalise parameter values that are far from zero

Problems with models



- **Underfitting**
  - *model is too simple for the data*
- Again, could treat as a data problem
  - feature reduction, pre-processing
- Or modify the model
  - choose a more powerful model (more parameters to fit)
  - reduce regularisation penalty

Selecting a model

- We want a model that **generalises** from existing data to new, unseen cases
- In most cases, impossible to know in advance what model to choose
  - so we pick a few likely candidates!
  - and select the best model by comparing them
  - this requires us to measure **generalisation error**
    - expected error, when apply model to new cases
  - How?
    - by definition, we don't have access to "new" cases during development

Training and test sets

- Assuming the data we have is representative of future data, split it into:
  - **training set** - used to fit model parameters
  - **test set** - used to measure prediction accuracy after fitting *(generalisation error)*
- Train model on training set, measure prediction error on both training and test sets, but separately
  - high training set error – **underfitting**
  - low training error, high test set error - **overfitting**

Model validation

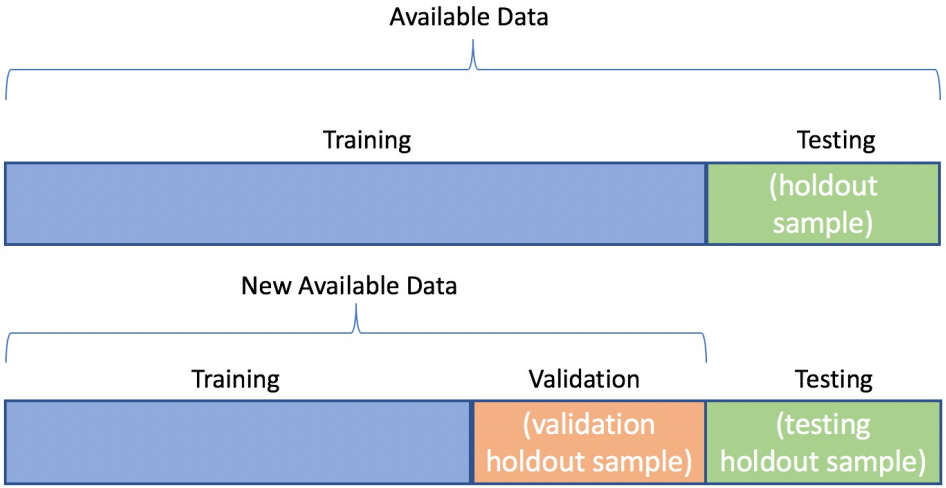
- Typically, want to select a model from many alternatives
  - several model types, possibly 100s or 1000s of different settings for each type of model
  - model comparison is for different types or different settings

- By training and testing on the same train/test split, we select for best performance **on this split**
  - single error value per model
  - no measure of error variance or sensitivity to this split

- The test set should only be used to estimate generalisation error **after** a model is selected!
  - Otherwise it is ***cheating*** and the results will be biased and performance in practice will be worse than this estimate
  - Comparing models or optimizing (hyper-)parameters is just the same as fitting the model parameters, so test set must not be used

Validation sets

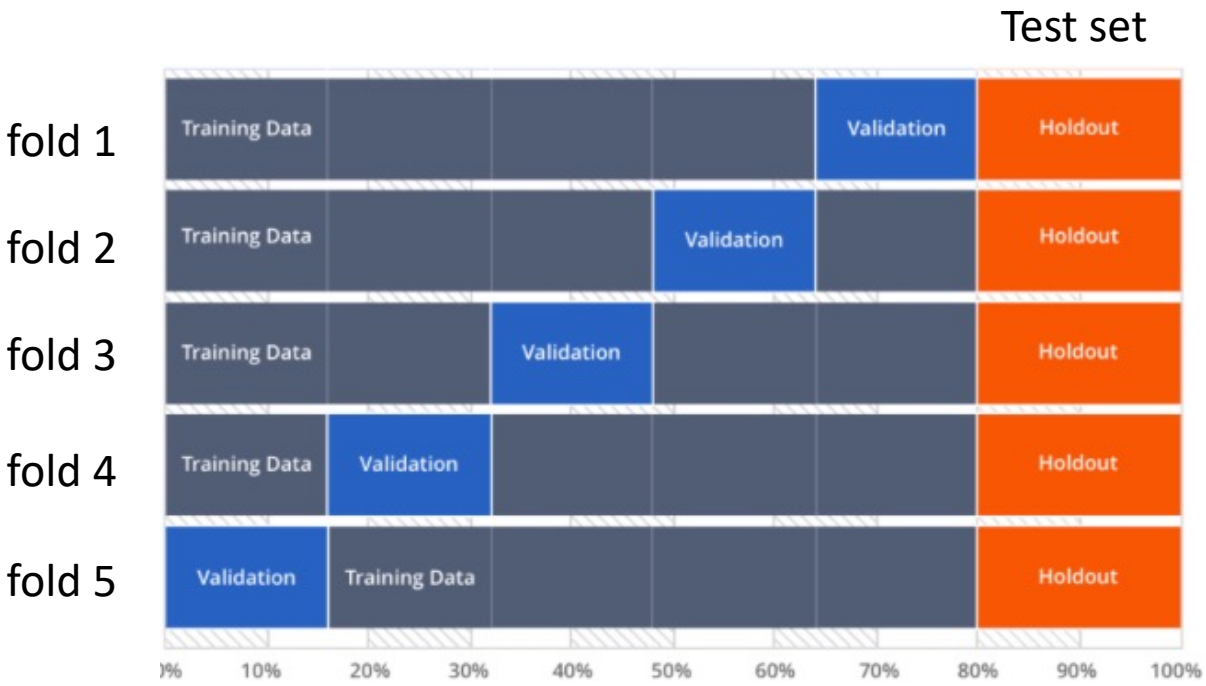
- Further split training set to get a third set (validation)
- Simplest option is a holdout (fixed) version:
  - hold out a validation set from the training set
  - train candidate models on reduced training set, measure error on validation set
  - choose model/settings with lowest validation error
  - estimate generalisation error using test set



Cross validation

- Select multiple validation sets
- Same as before, but now we train multiple times, using different training/validation splits
  - e.g. 5-fold cross validation (shown below), “leave one out” (LOO) cross validation (validation = one sample)

Perform training K times for K-fold CV



Cross validation

- Cross validation:
  - Gives you multiple error measures per model/setting
  - Compute mean and variance of errors
  - Better for smaller training sets, not dependent on a single (small) split
  - But more time consuming, as you must train the models many times

- Finally, re-train chosen model (best settings) on the full training set (merging training and validation)
  - Then apply that model to the test set to estimate generalisation error (only way to avoid bias/cheating)

Summary: typical ML workflow

1. **Look at your data** for common problems
   - missing, invalid, outliers, noise, scale etc. (remember GIGO)
2. Choose some candidate models based on data and task
   - use your knowledge/experience & that of others (*e.g.* docs, papers, blogs, …)
3. Split data into **training** and **test** sets
   - need to do this carefully (more on this next week)
4. Split training data into (reduced) training and **validation** sets
   - single holdout or cross validation
5. Train candidate models on training sets
6. Select best model type/settings based on validation set errors
7. Retrain model on the full training set (merging train/validation)
8. Apply best model to test data
   - this gives your estimate of the **generalisation error**
   - your best ***estimate*** of the error when applying the system to new data

Summary: key concepts

- Generalisation error: the goal of supervised learning
- Data can't always be trusted – so look at it!
- **Overfitting** and **underfitting**
- **Training** and **test** sets
- **Validation** and **cross-validation**
  - separate the task of selecting your model type/settings, from the task of estimating generalisation error
  - ***must*** use separate data sets for each!