dm22s1

Topic 06 : Data Modelling

Part 01 : Data Modelling - Introduction

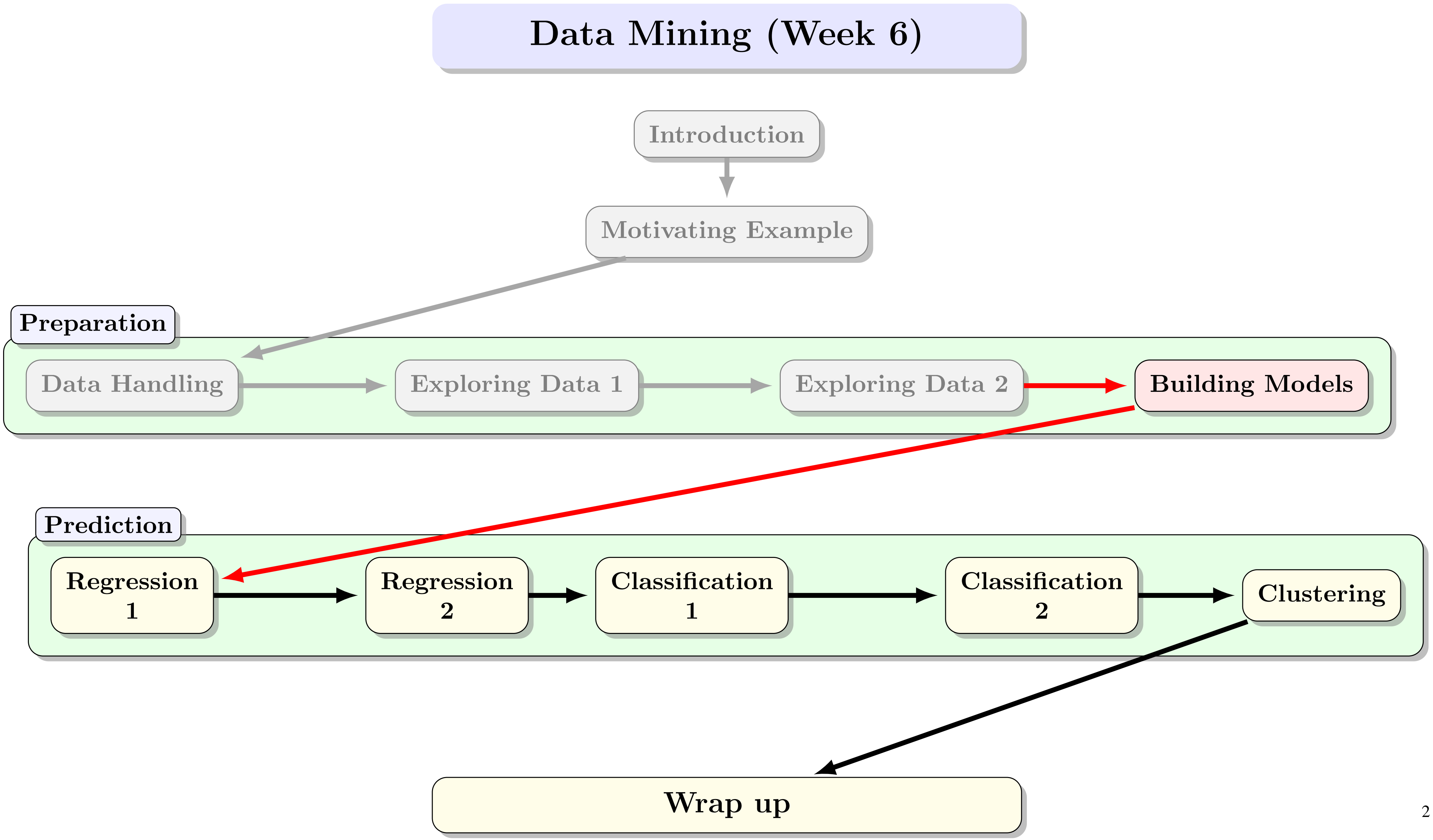
Dr Bernard Butler

Department of Computing and Mathematics, WIT.
(bernard.butler@setu.ie)

Autumn Semester, 2022

Outline

- Components of a machine learning problem
- Machine learning concepts and notation
- Bias vs variance
- Learning curves
- Regularisation


Data Mining (Week 6)
Introduction
Motivating Example
Preparation
Data Handling
Exploring Data 1
Exploring Data 2
Building Models
Prediction
Regression 1
Regression 2
Classification 1
Classification 2
Clustering
Wrap up

Three Components of a Machine Learning Problem

It is easy to get lost among the multitude of choices one needs to make when given data mining problem. A good decomposition is the following:

Representation	Evaluation	Optimization
Instances	Accuracy/Error rate	Combinatorial optimization
K-nearest neighbor	Precision and recall	Greedy search
Support vector machines	Squared error	Beam search
Hyperplanes	Likelihood	Branch-and-bound
Naive Bayes	Posterior probability	Continuous optimization
Logistic regression	Information gain	Unconstrained
Decision trees	K-L divergence	Gradient descent
Sets of rules	Cost/Utility	Conjugate gradient
Propositional rules	Margin	Quasi-Newton methods
Logic programs		Constrained
Neural networks		Linear programming
Graphical models		Quadratic programming
Bayesian networks		
Conditional random fields		

A Few Useful Things to Know about Machine Learning, Domingos, 2012.

Three Components of a ML Problem — Representation

Representation	Evaluation	Optimization
Instances	Accuracy/Error rate	Combinatorial optimization
K-nearest neighbor	Precision and recall	Greedy search
Support vector machines	Squared error	Beam search

Representation refers to formulating the problem as a machine learning problem — typically a classification problem, a regression problem or a clustering problem.

- How do we represent the input?
- What features to use?
- How do we learn additional features?
- With each type of problem, we have multiple subtypes:
  For example which classifier? a decision tree, a neural network, a support vector machine, etc.

Three Components of a ML Problem — Evaluation

Representation	Evaluation	Optimization
Instances	Accuracy/Error rate	Combinatorial optimization
K-nearest neighbor	Precision and recall	Greedy search
Support vector machines	Squared error	Beam search

Evaluation refers to an objective function or a scoring function, to distinguish a good model from a bad model.

- For a classification problem, we need this function to know if a given classifier is good or bad. A typical function can be based on the number of errors made by the classifier on a test set, using precision and recall.
- For a regression problem, it could be the squared error, or likelihood. Do we include regularisation? etc

Three Components of a ML Problem — Optimisation

Representation	Evaluation	Optimization
Instances	Accuracy/Error rate	Combinatorial optimization
K-nearest neighbor	Precision and recall	Greedy search
Support vector machines	Squared error	Beam search

Optimisation is concerned with searching among the models in the language for the highest scoring model.

- How do we search among all the alternatives?
- Can we use some greedy approaches, branch and bound approaches, gradient descent, linear programming or quadratic programming methods.

Taxonomy of Machine Learning Models . . .

. . . by Intuition/Motivation

- Geometric models use intuitions from geometry such as separating (hyper-)planes, linear transformations and distance metrics.
- Probabilistic models view learning as a process of reducing uncertainty, modelled by means of probability distributions.
- Logical models are defined in terms of easily interpretable logical expressions.

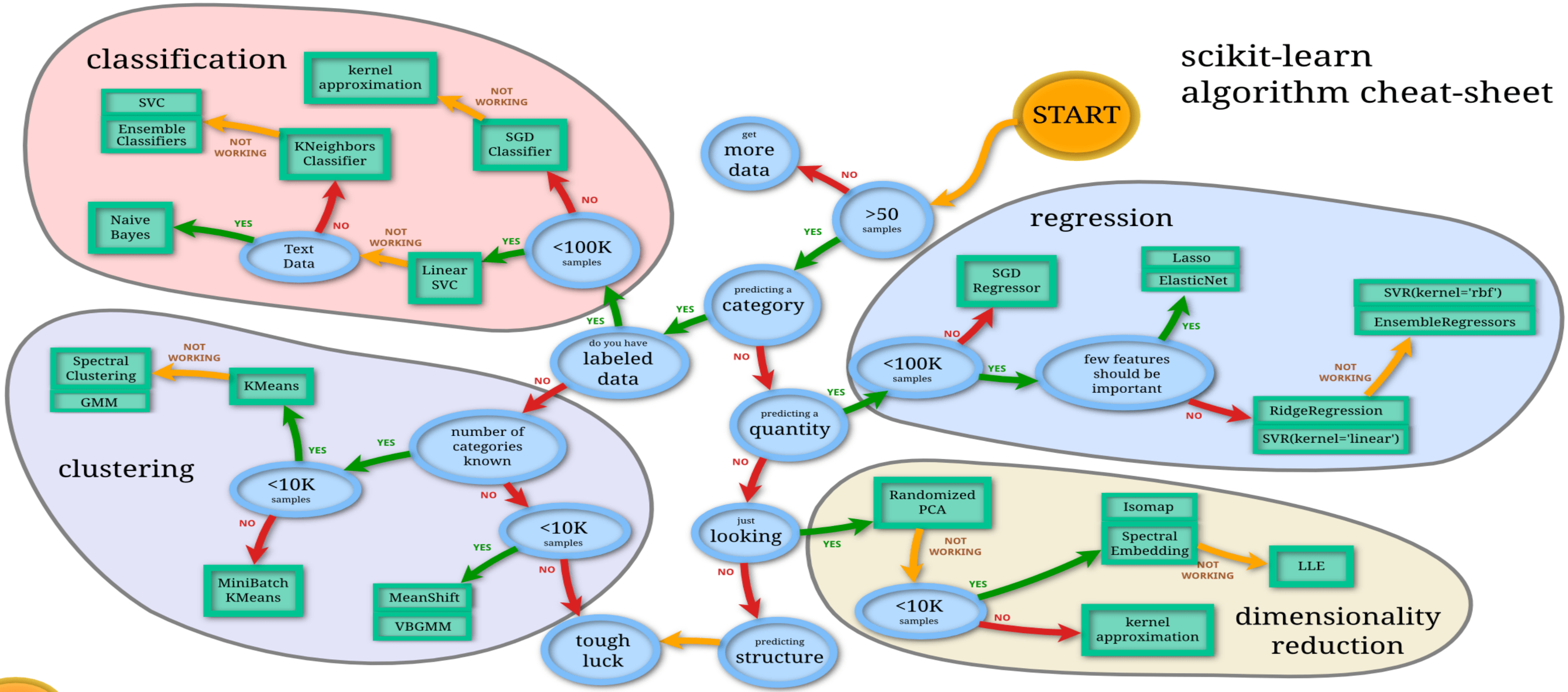
. . . by Algorithmic Properties

- Regression models predict a numeric output.
- Classification models predict a discrete class value.
- Neural networks learn based on a biological analogy
- Local models predict in the local region of a query instance.
- Tree-based models (recursively) partition the data to make predictions.
- Ensembles learn multiple models and combine their predictions.

. . . by Fixed/Variable Number of Parameters

- Parametric models have a fixed number of parameters.
- In non-parametric models the number of parameters grows with the amount of training data.

Aside: Scikit-learn Flowchart of Models (Shallow Learners)



Back

Statistical Models vs Machine Learning Models

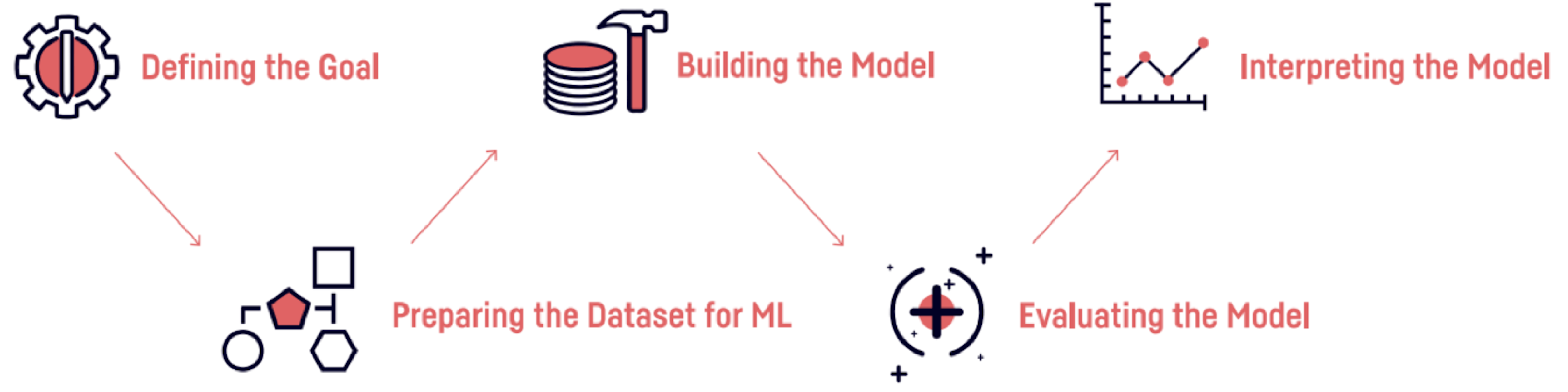
	Statistical Models	ML Models
Data	• Usually small ($<$ 1000 observations) • Low dimension ($<$ 10 variables) • Can have detailed understanding of data • Data is clean — human has looked at each data point	• Can be huge (million+ observations) • Large dimension (1000+, more for vision) • Too large for human to parse / understand • Data not clean — humans can't afford to understand/fix each point
Models	• Simple models — complexity limited by theory • Detailed/complex statistical assumptions re data • Model known, and data is carefully examined to verify assumptions.	• "No" upper limit on model complexity • Fewer statistical assumptions re data • Don't know right model? No problem! have multiple models and vote/weight results
Validation	• Evaluation based on theoretical estimates under stated statistical assumptions • Analysis of errors using theoretical distributions	• Empirical evaluation methods instead of theory — how well does it work on **unseen** data? • Don't calculate expected error, measure it from **unseen** data.

Statistics would be very different if it had been born after the computer instead of 100 years before

Splitting data into train+test(+validation) is vital

The Pipeline Metaphor

Model Building Pipeline



Source: Dataiku

Comments

- We saw the first two stages in previous weeks
- This week we look at the remaining stages
- Of course this pipeline is a simplification. In reality it is iterative.

What does a (supervised learning) model look like?

Definition 1 (Linear Model)

General form of linear model used in this module looks like

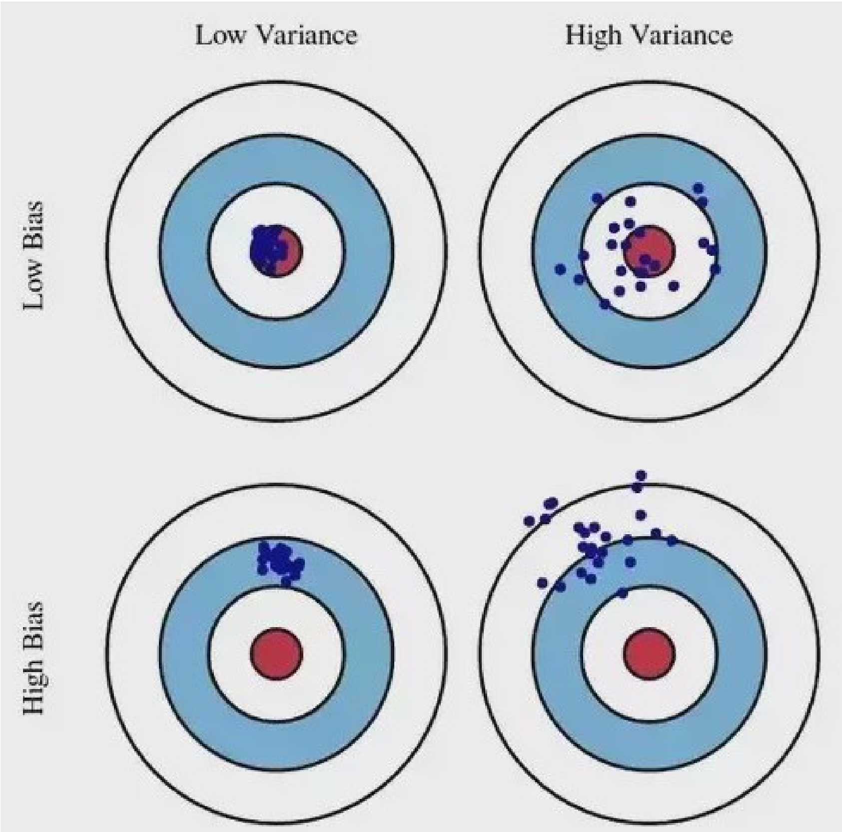
$$y_i \sim f_i^{(1)} + f_i^{(2)} + \ldots + f_i^{(n)}$$

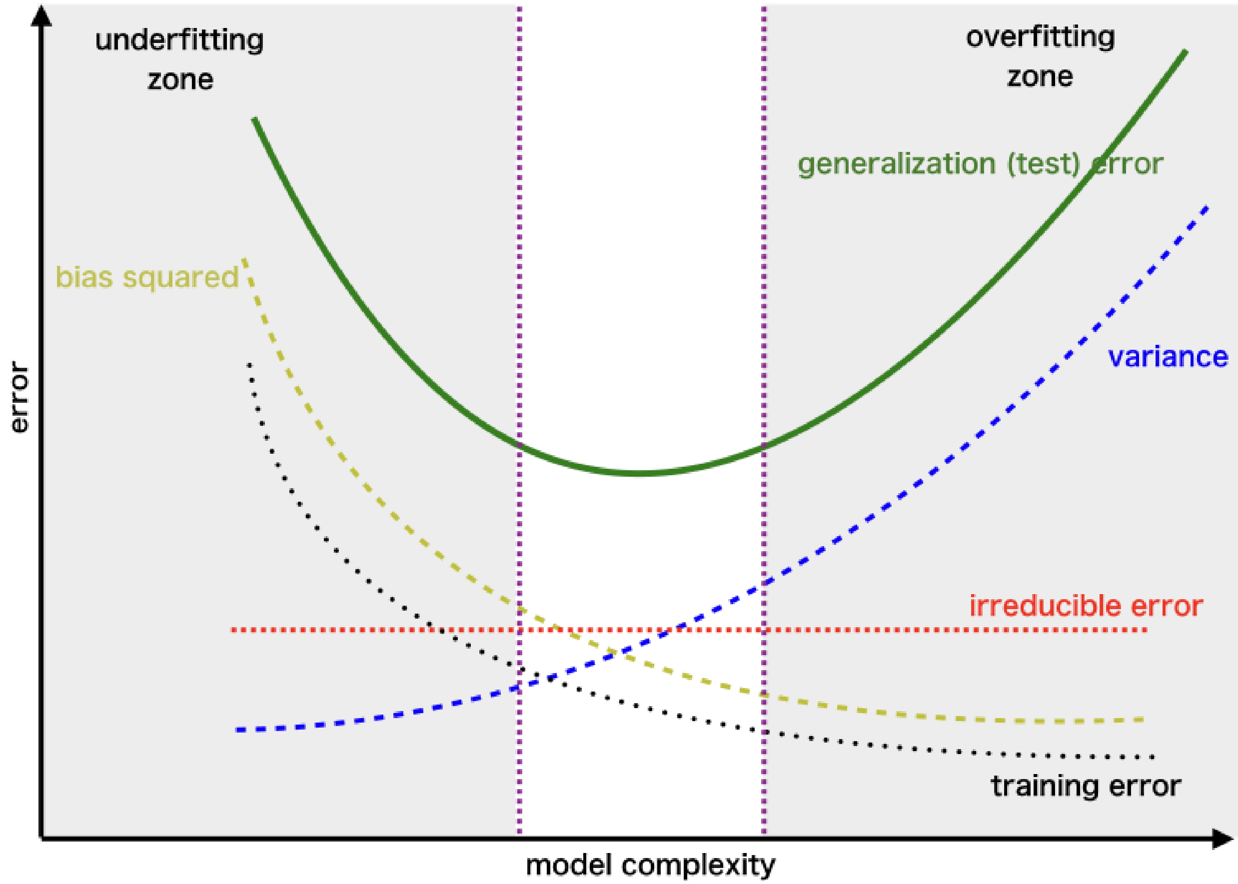
where y_i is the value of the response variable for observation i, and $f_i^{(j)}; j = 1, \ldots, n$ is the value of the j^{th} feature for that observation.
The model is linear in the sense that it can be turned into the following linear equation:

$$y_i = a_0 + a_1 f_i^{(1)} + a_2 f_i^{(2)} + \ldots + a_n f_i^{(n)} + \varepsilon_i$$

- Note that the features f can be nonlinear but the model parameters a must appear linearly.
- The goal of modelling is to find a so that the *prediction error* is a minimum.

Bias-Variance and Total Error





Look for *a* that minimise the generalization error (estimated using the test set)

Example Model Types

Model	Applications	Concerns
Logistic Regression	X-ray classification	Regression with transformed variable
Fully connected networks	Classification	Classical ANN: choose encoding and size
Convolutional Neural Networks	Image processing	deep learning - choose segmentation
Recurrent Neural Networks	Voice recognition	ANN with feedback - how much?
Random Forest	Fraud Detection	Ensemble method - how many?
Reinforcement Learning	Learning by trial and error	Choose goal and penalties
Generative Models	Image creation	Choose parameters
K-means	Segmentation	Choose distance function and k
k-Nearest Neighbors	Recommendation systems	Choose distance function and k
Bayesian Classifiers	Spam and noise filtering	Deal with imbalances

Before you start...

Does a *pre-trained* model exist?

Transfer Learning

- Building a model from scratch is resource-intensive
- Open source data and model exist, particularly for deep learning (not in this nmodule)
- Most frameworks provide example models that can be used as a template
    - Select a similar model
    - Prune it (remove unnecessary terms)
    - Train using the pruned model as a starting point

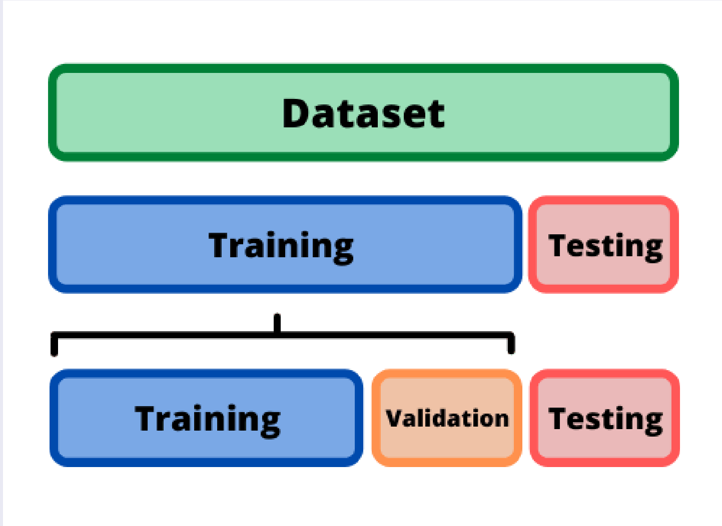
Training, test and valuation subsets: 3-way Holdout

Why Split?

Hold back some data to check how the model is doing.

- Training data is sample used to fit the model parameters.
- Test data is sample used to test the final model fitted to the training data.
- Validation data is sample used to test each interim model while tuning it.

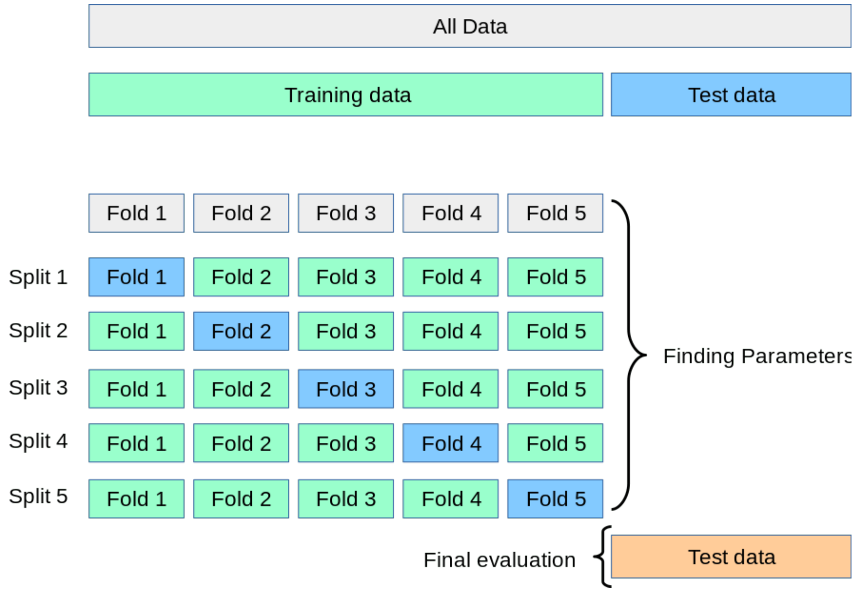
Typical Splits



sklearn example

```
from sklearn.model_selection import train_test_split
trainVal, test = train_test_split(df, test_size=0.2)
train, validation = train_test_split(trainVal, test_size=0.1)
```

K-fold cross validation



Source: `https://scikit-learn.org/stable/modules/cross_validation.html`

sklearn example

```
from sklearn.model_selection import cross_val_score
# clf is some classifier, X and y are the features and labels
scores = cross_val_score(clf, X, y, cv=5)
```

`scores` is a $k = 5$ element array, can be used to estimate the prediction error (or other score) while building a model

Featuring engineering 1: Scaling of numerical variables

Scaling - what it does

- If numeric features have different scales, e.g. [-0.005, -0.003] and [10000, 10001] some terms dominate, others are "lost"
- Better: transfer the scaling from the feature to the model parameter
- A min-max scaling is often a good choice:

$$\tilde{X} = \frac{X - X_{\text{min}}}{X_{\text{max}} - X_{\text{min}}}$$

- Note that X is in the range $[X_{\text{min}}, X_{\text{max}}]$ but $\tilde{X}$ is in the range $[0, 1]$.
- Other options include StandardScaler (subtract mean and divide by standard deviation) and a max-abs scaler (scales to [-1,1])

sklearn example

```python
from sklearn.preprocessing import MinMaxScaler
# df is a dataframe with numeric features
scaler = MinMaxScaler()
dfScaled = scaler.fit(df))
```

`dfScaled` can be used instead of `df` with the advantage that the fitted parameters are more accurate.

Feature Engineering 2: Choice of Features

- How many to include? Use metrics to decide. Will see some when considering regression and classification.
- How do we handle different feature types? Need to encode categorical variables.
- Can we derive new numeric features? Yes, $f' = \log(f)$ etc. is possible

Summary

- We have reviewed different types of models and considered their general form
- We looked at the goals of modelling: minimise predictive error
- We considered how feature engineering can help.
- In subsequent weeks we will put this theory into practice.

Resources

- **A Summary of the Basic Machine Learning Models**
  towardsdatascience.com/a-summary-of-the-basic-machine-learning-models-e0a65627ecbe

- **Train-Test Split for Evaluating Machine Learning Algorithms**
  https://machinelearningmastery.com/train-test-split-for-evaluating-machine-learning-algorithms
  This week I have focused on the theory rather than its (python) implementation. This is a nice article that covers the implementation side of things.

- **Cross-Validation: Estimator Evaluator**
  medium.com/swlh/cross-validation-estimator-evaluator-897d28afb4ff
  Nice article that covers cross-validation in a lot more detail — we will be using many of these variants in later weeks, especially k-fold stratified.