(MSc) Data Mining

Topic 03 : Data Modelling - Introduction

Part 01 : Data Modelling - Introduction

Dr Bernard Butler and Dr Kieran Murphy

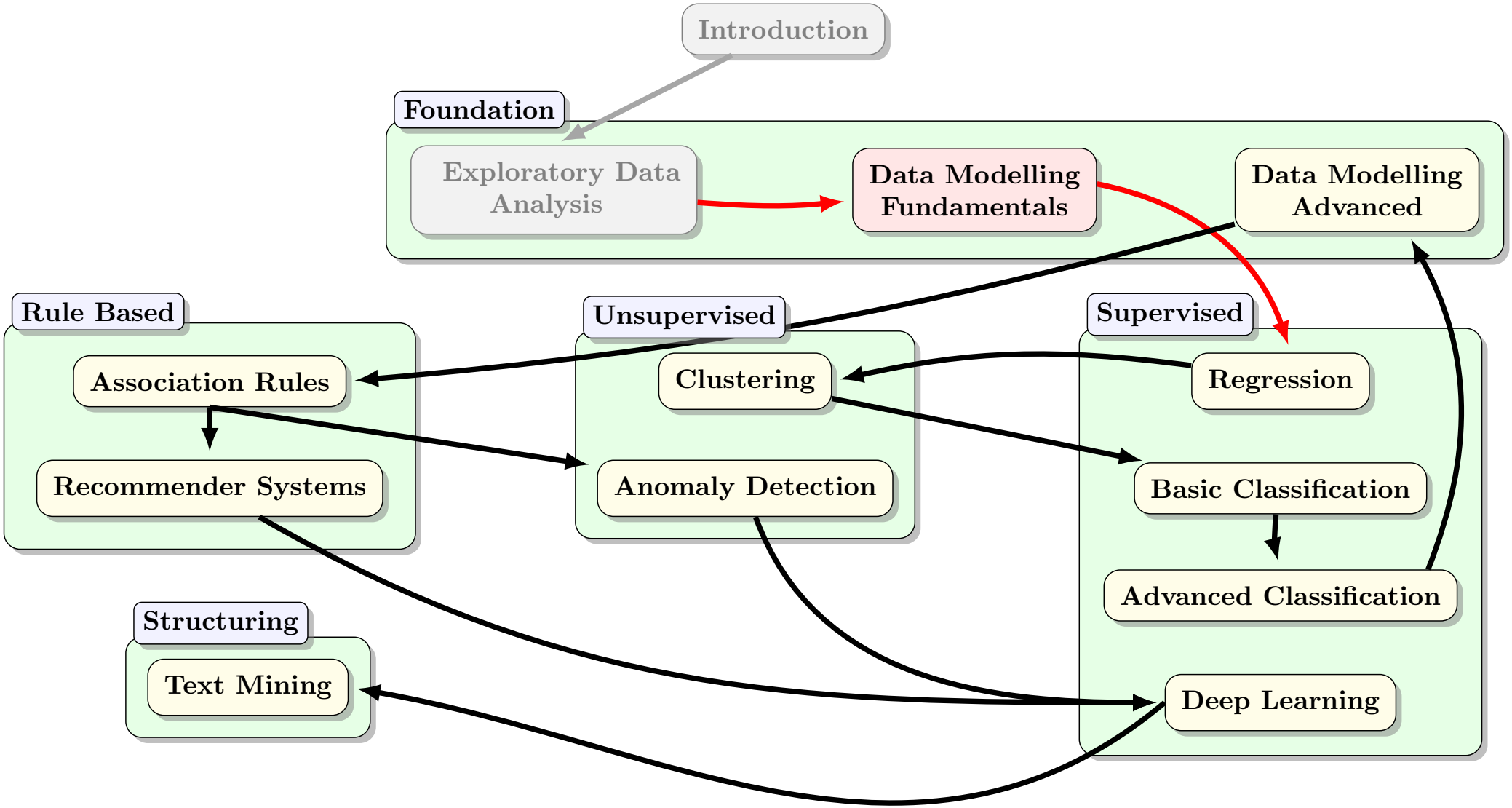
Department of Computing and Mathematics, SETU Waterford.
(bernard.butler@setu.ie; kmurphy@wit.ie)

Spring Semester, 2026

Outline

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

Data Mining (Week 3)



Three Components of a Machine Learning Problem

It is easy to get lost among the multitude of concepts to address and the 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.

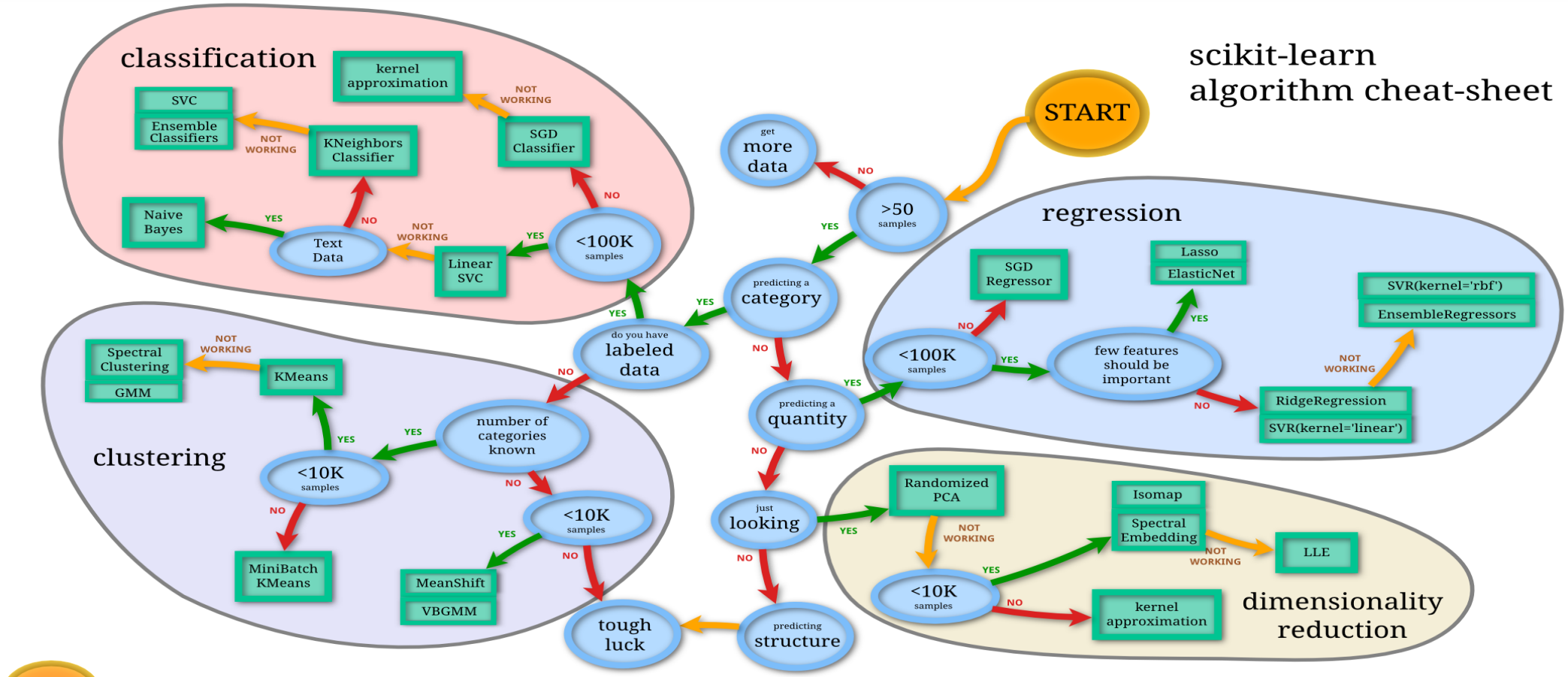
. . . by Algorithmic Properties

- 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

A neural network with more than one hidden layer is called a deep learner, all other learners are shallow learners.

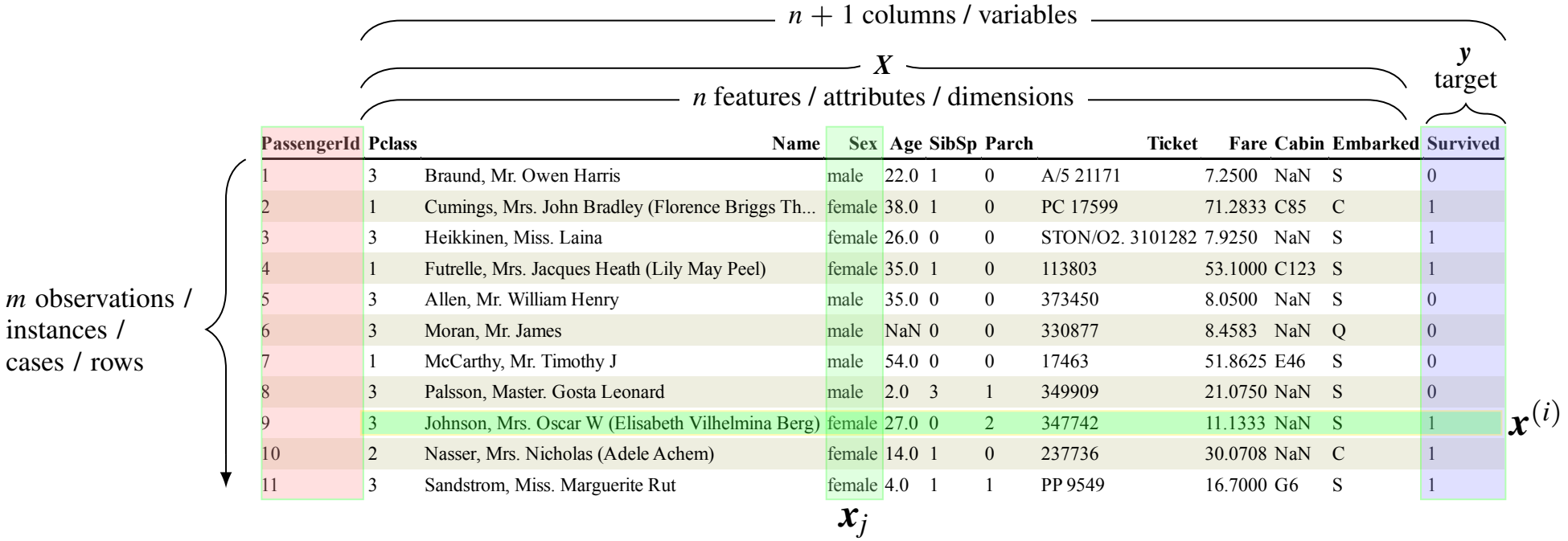
Statistical Models vs Machine Learning Models

	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 on 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	• Use empirical evaluation methods instead of theory—how well does it work on **unseen** data? • Don't estimate 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

Dataset Terminology / Notation



PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	Survived
1	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S	0
2	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C	1
3	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S	1
4	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S	1
5	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S	0
6	3	Moran, Mr. James	male	NaN	0	0	330877	8.4583	NaN	Q	0
7	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.8625	E46	S	0
8	3	Palsson, Master. Gosta Leonard	male	2.0	3	1	349909	21.0750	NaN	S	0
9	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.1333	NaN	S	1
10	2	Nasser, Mrs. Nicholas (Adele Achem)	female	14.0	1	0	237736	30.0708	NaN	C	1
11	3	Sandstrom, Miss. Marguerite Rut	female	4.0	1	1	PP 9549	16.7000	G6	S	1

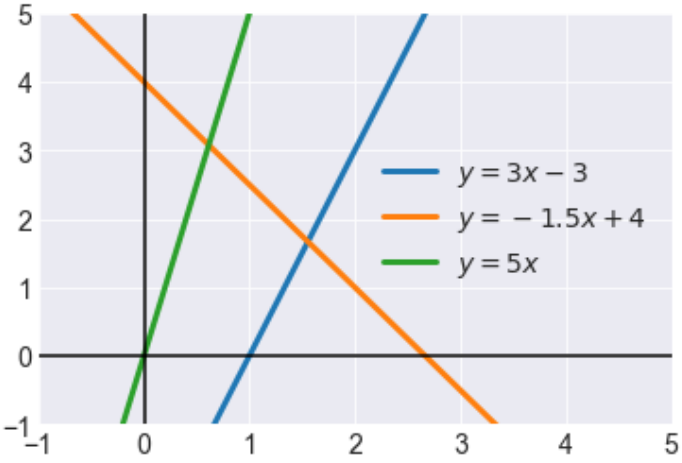
- A labeled dataset consists of m rows $\times$ $(n + 1)$ columns / variables.
- Use bold to represent vectors and matrices.
- Use subscripts to indicate particular feature / attribute / column ..... $\boldsymbol{x}_j$
- Use superscript in parenthesis to indicate particular observation / instance/ case / row ..... $\boldsymbol{x}^{(i)}$
- So $x_j^{(i)}$ (or $x_{i,j}$) is the i-th observation in the j-th feature ..... $x_j^{(i)}$

Aside: A reminder of functions from the past . . . I

Recall the formula for the line:

$$y = \underbrace{m}_{\text{slope}} x + \underbrace{c}_{\text{intercept}}$$

where given coefficients $m \neq 0$ and c.



Examples include

$$y = 3x - 3, \qquad y = -1.5x + 4, \qquad y = 5x,$$

Now consider the collection of all possible lines — this gives us the set/family of linear functions , i.e., all functions of the form

$$f(x; m, c) = f(x) = mx + c$$

name input parameters formula

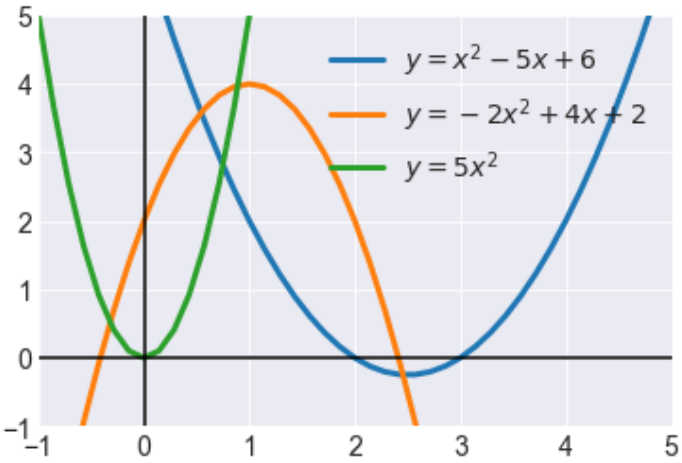
comma separator between inputs and between parameters + semicolon between input(s) and parameter(s)

Aside: A reminder of functions from the past … II

And the formula for a quadratic:

$$y = ax^2 + bx + c$$

with given coefficients $a \neq 0$, b and c.



Examples include

$$y = x^2 - 5x + 6, \qquad y = -2x^2 + 4x + 2, \qquad y = 5x^2$$

Now consider the collection of all possible quadratics — this gives us the family of quadratic functions, i.e., all functions of the form

$$f(x; a, b, c) = ax^2 + bx + c$$

name input parameters formula

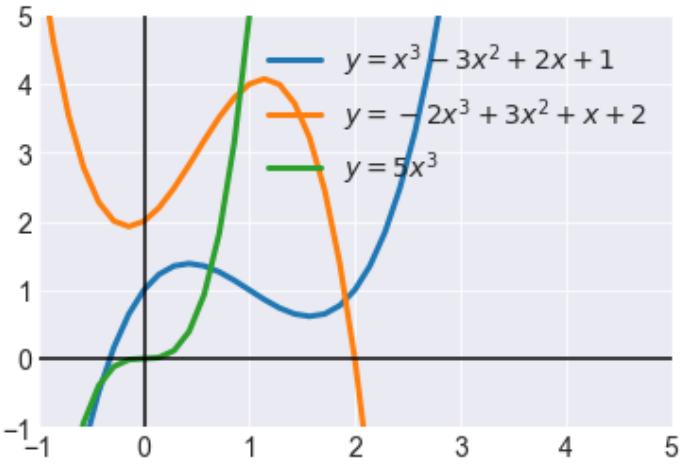
comma separator between inputs and between parameters + semicolon between input(s) and parameter(s)

Aside: A reminder of functions from the past … III

And the formula for a cubics:

$$y = a_3x^3 + a_2x^2 + a_1x + a_0$$

with given coefficients $a_3 \neq 0$, a_2, a_1 and a_0.



Examples include

$y = x^3 - 3x^2 - 5x + 2x + 1$, $y = -2x^3 + 3x^2 + x + 2$, $y = 5x^3$.

Now consider the collection of all possible cubics — this gives us the family of cubics functions, i.e., all functions of the form

$$f(x; a_3, a_2, a_1, a_0) = a_3x^3 + a_2x^2 + a_1x + a_0$$

We could continue this to quartics (power of 4), quintics (power of 5), sextics (power of 6), … or we could think about a bigger family — the family of polynomials.

Aside: A reminder of functions from the past . . . IV

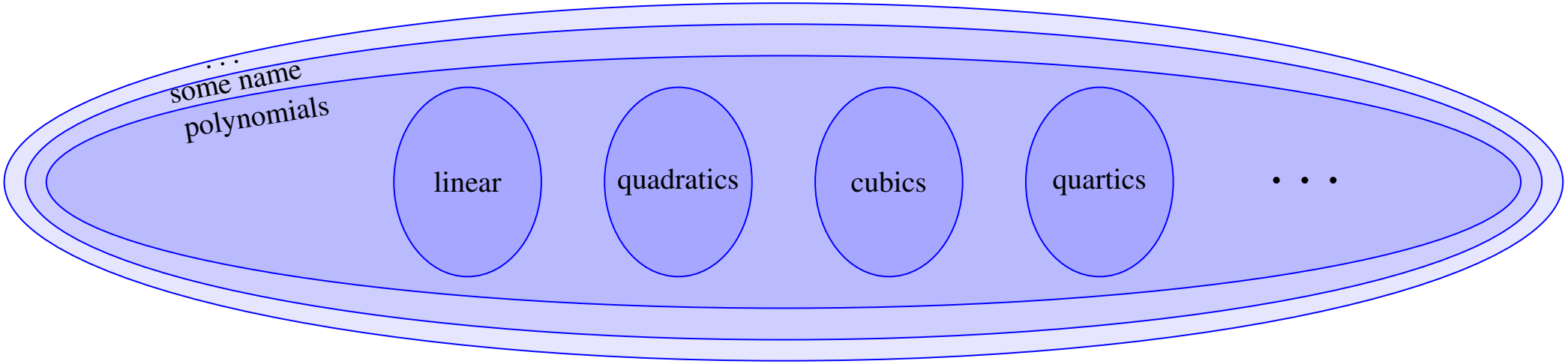
The family of polynomials is all functions of the form

$$f(x;\boldsymbol{\theta}) = a_d x^d + a_{d-1}x^{d-1} + \cdots + a_2 x^2 + a_1 x + a_0$$

with vector $\boldsymbol{\theta} = [a_d, a_{d-1}, \ldots, a_1, a_0]$.

remember we use bold font to indicate vectors.

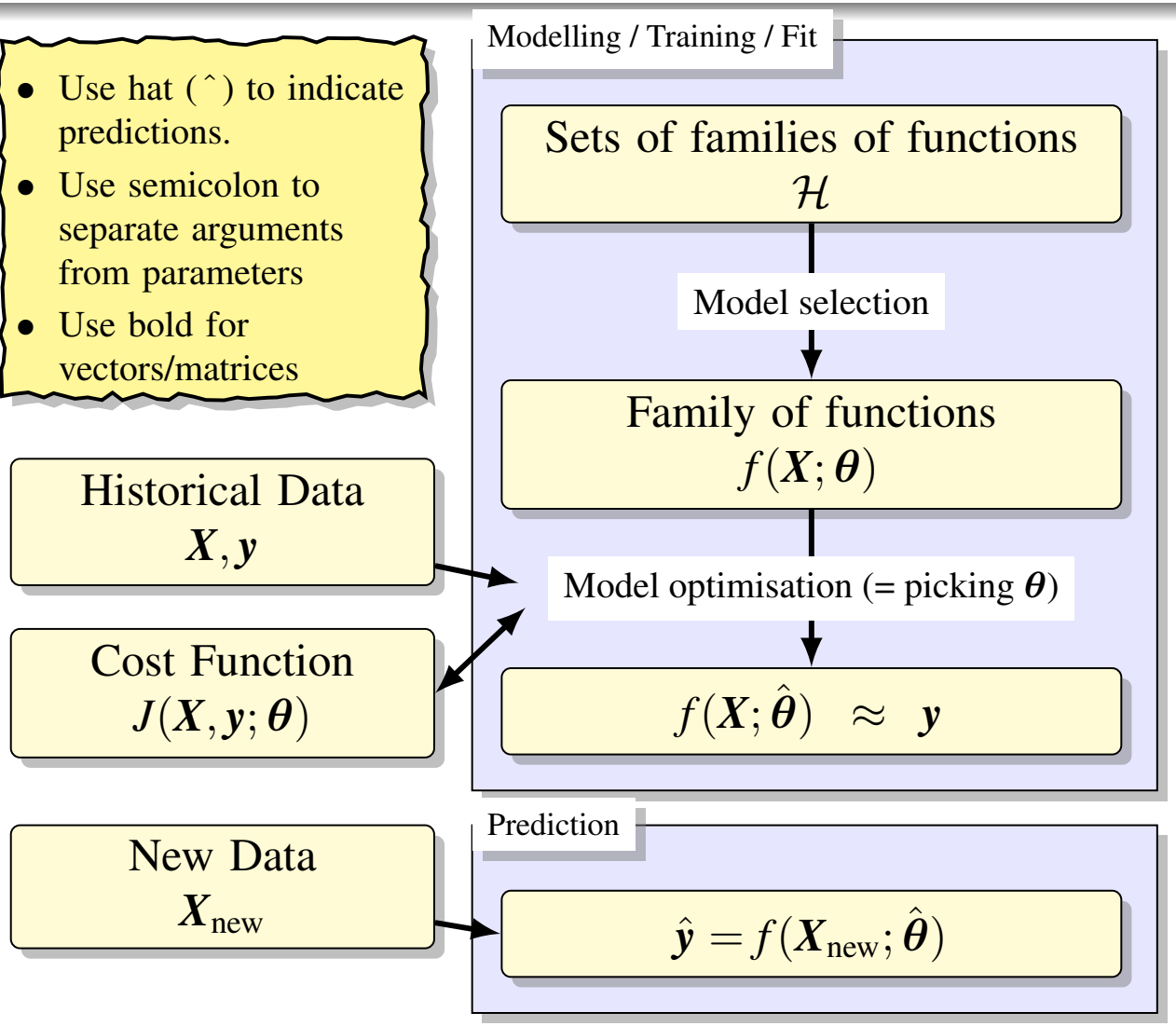
So we end up with this idea of (nested) families (or sets) of functions



Note: Instead of saying "sets of sets of functions", we often say "families of sets of functions" or "sets of families of functions".

The ML Problem

- Use hat ($\hat{}$) to indicate predictions.
- Use semicolon to separate arguments from parameters
- Use bold for vectors/matrices



Ingredents

- Historical, labeled data
- Unseen, unlabelled data
- Sets of families of functions

Recipe

- Select a family of functions from all possible families.
- Each member of this family is uniquely determined by its parameter values. Finding the best member (function) is equivalent to finding the corresponding parameter values.
- Using a cost function (which is a measure of the difference between generated labels and given labels) we determine our optimal parameters.
- Finally using best member (function) we can generate predictions for new, unlabelled data.

Cost Functions

- Cost is a weighted sum/average over (all) cases/rows/observations.
- In each case/row/observation, i, want to measure the difference between predicted label, $\hat{y}^{(i)}$, and given label, $y^{(i)}$.
  - Simplest options are based on the difference — so we have

$$\text{cost}^{(i)} = \left|\hat{y}^{(i)} - y^{(i)}\right| \qquad \text{or} \qquad \text{cost}^{(i)} = \left(\hat{y}^{(i)} - y^{(i)}\right)^2$$

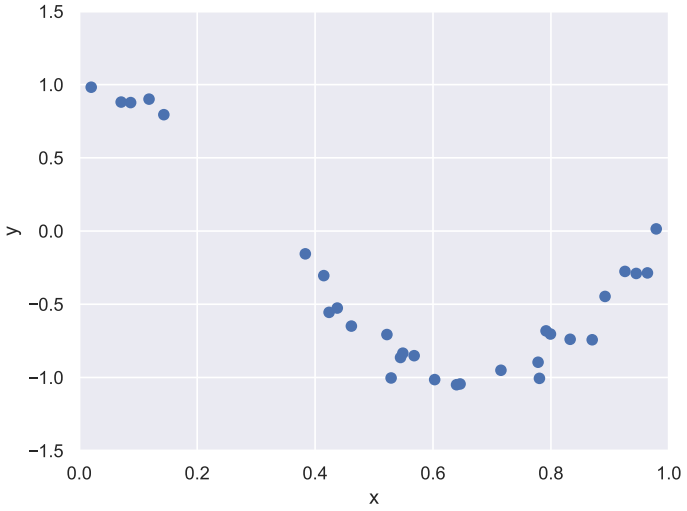
    Absolute value $||$ or squaring is used to make sure negative and positive costs don't cancel out.
- So a possible cost function, $J(\boldsymbol{X}, \boldsymbol{y}; \boldsymbol{\theta})$ is

$$J(\boldsymbol{X}, \boldsymbol{y}; \boldsymbol{\theta}) = \frac{1}{m} \sum_{i=1}^{m} \left(\hat{y}^{(i)} - y^{(i)}\right)^2 \qquad \text{(MSE)}$$

  where vector of predicted labels is $\hat{y} = f(\boldsymbol{X}; \boldsymbol{\theta})$.
- Many, many variants:
  - Only sum over a subset of cases — stochastic gradient methods — faster training.
  - Use log functions to magnify interval between 0.1, 0.01, 0.001, etc. so we punish a model more if it is wrong with a "probability/confidence/certainty" of, say 1/1,000,000, than when it is wrong with probability of 1/1,000 — cross entropy.

Example: Need for Unseen Data

Consider the problem of fitting a function (a curve) to the data in following figure ...



- We will simplify things by tightening the specification and work with polynomials only

$$f(x) = f(x;\boldsymbol{\theta}) = \theta_d x^d + \cdots + \theta_2 x^2 + \theta_1 x + \theta_0$$

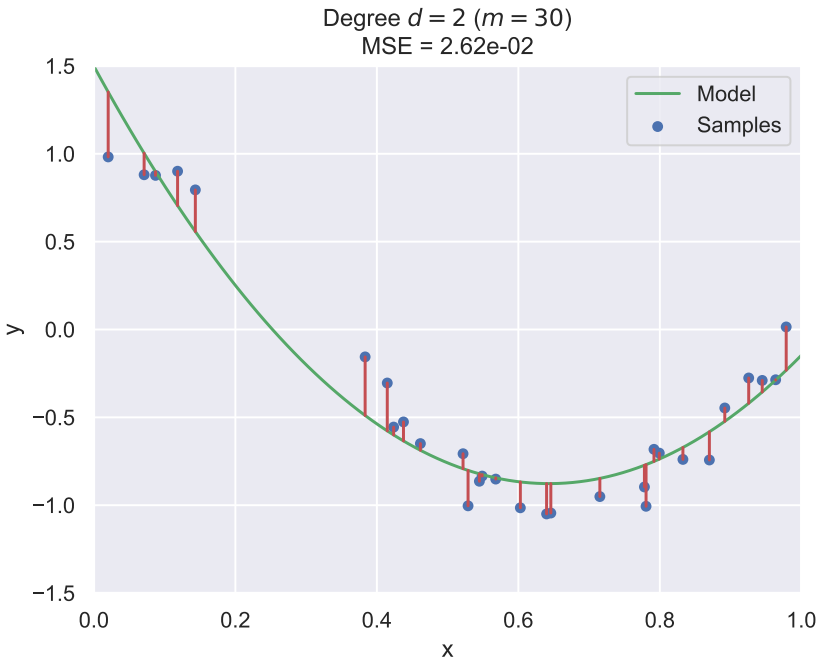
- Appear to have $d+2$ parameters: the order*, d, and the $d+1$ coefficients.
- Number of parameters is not fixed so have a non-parametric method.
- However the order parameter is fundamentally different from the coefficients parameters in the impact a change in its value has (consider switching from line ($d=1$) to quadratic ($d=2$) to cubic ($d=3$), etc.).
- General convention: We treat d as a meta-parameter†, and treat the coefficients as parameters‡, so we get a parametric method.

How do we pick d?
What if we get it wrong?
How will we know?

*In a polynomial the highest degree is called the order
†we (semi-manually) decide on appropriate value ourselves
‡use an algorithm to determine optimum values

Curve Fitting aka Regression/Classification/Clustering aka ALL ML Algorithms

Given data and a parameterised function, (model) curve fitting is the problem of determining function parameters (fit / train) so that the function output (predictions) matches desired output (true values / labels) as closely as possible. (error)



Input/Feature:

$$\boldsymbol{x} = [x^{(1)}, x^{(2)}, \ldots, x^{(m)}]$$

True value:

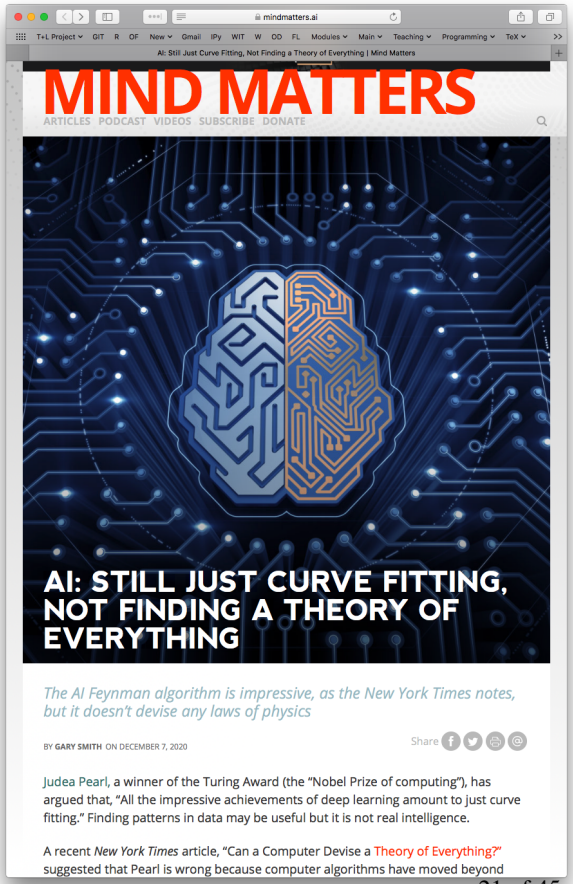
$$\boldsymbol{y} = [y^{(1)}, y^{(2)}, \ldots, y^{(m)}]$$

Predictions:

$$\hat{\boldsymbol{y}} = f(\boldsymbol{x})$$

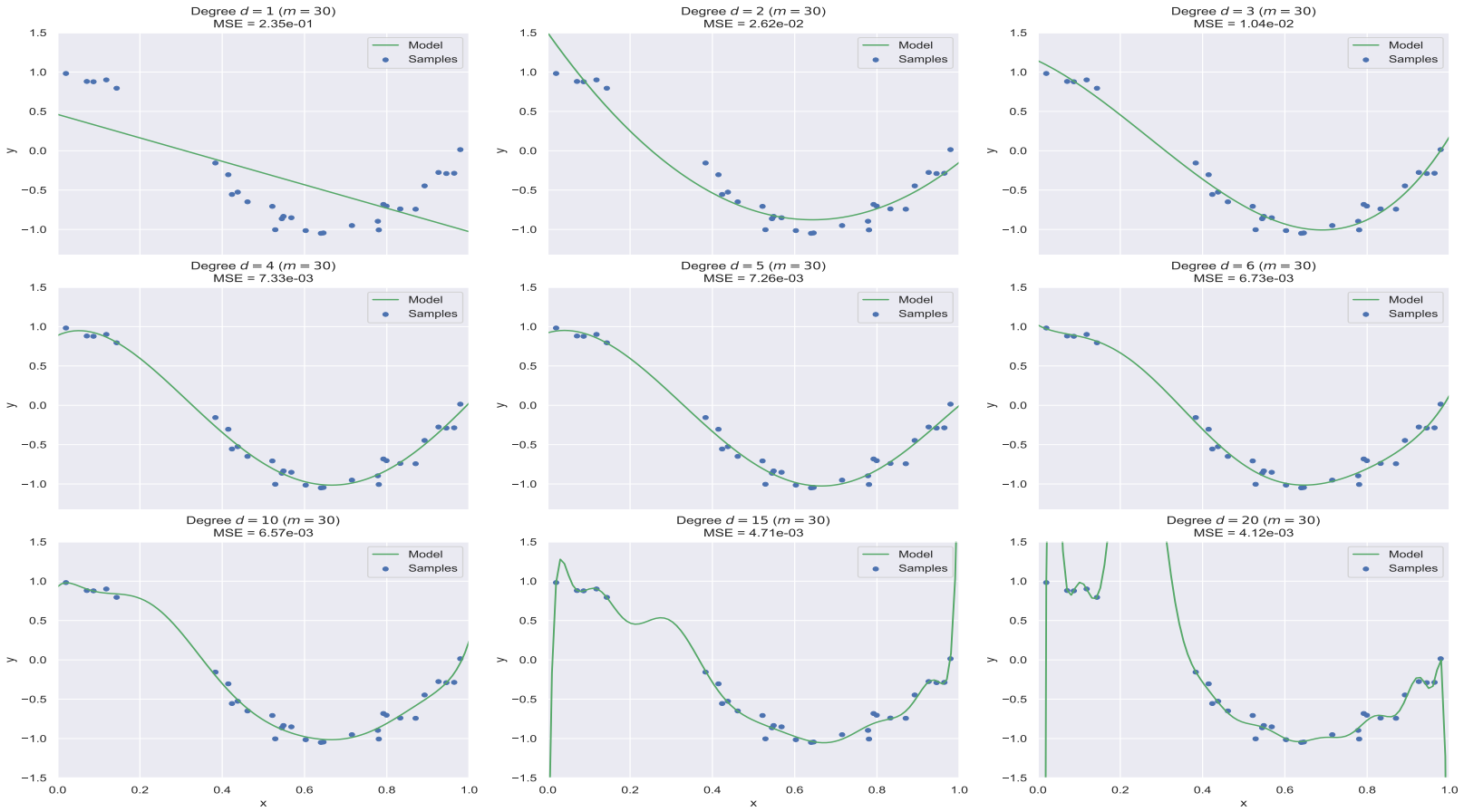
Mean Square Error:

$$MSE = \frac{1}{m} \sum (\hat{\boldsymbol{y}} - \boldsymbol{y})^2$$



Varying Meta-parameter, d

Question: What is the best fit for different values of $d = 1, 2, 3, \ldots$?

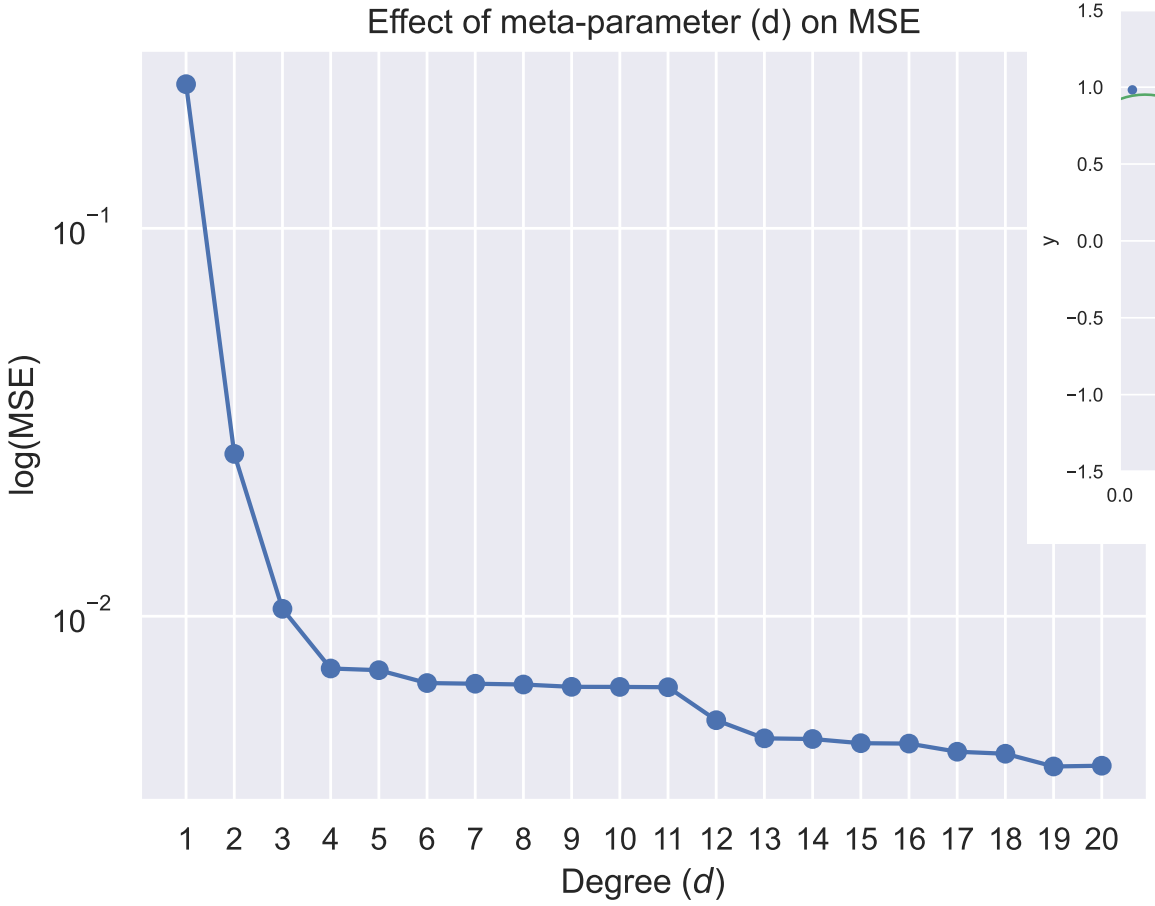


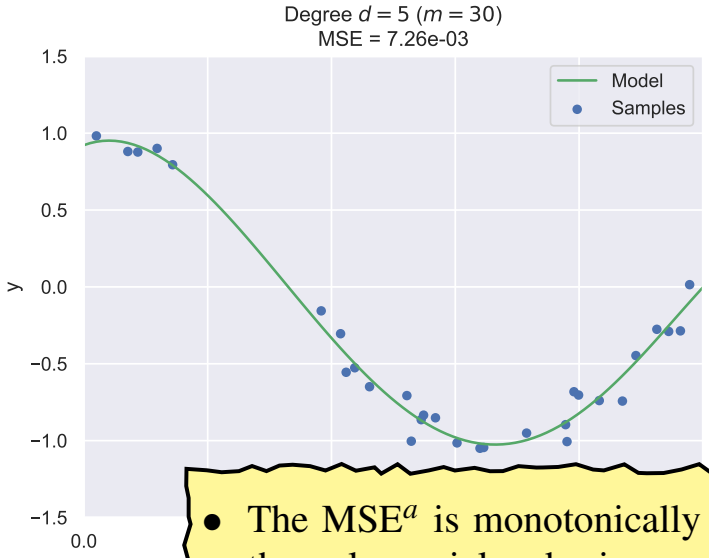
- Degree less than 3 are under-fitting, degree 10 and higher are over-fitting.
- What **objective** metric can we use to determine d? (**objective** $\approx$ **automatic**)
- Using the Mean Square Error (MSE) appears no good, it is decreasing as we increase the order.

$$MSE = \frac{1}{m} \sum (\hat{\boldsymbol{y}} - \boldsymbol{y})^2$$

Effect of Meta-parameter, d, on MSE

Using a for loop over $d = 1, 2, \ldots$ fit model and compute the error (MSE). Plot graph of error against degree, d:



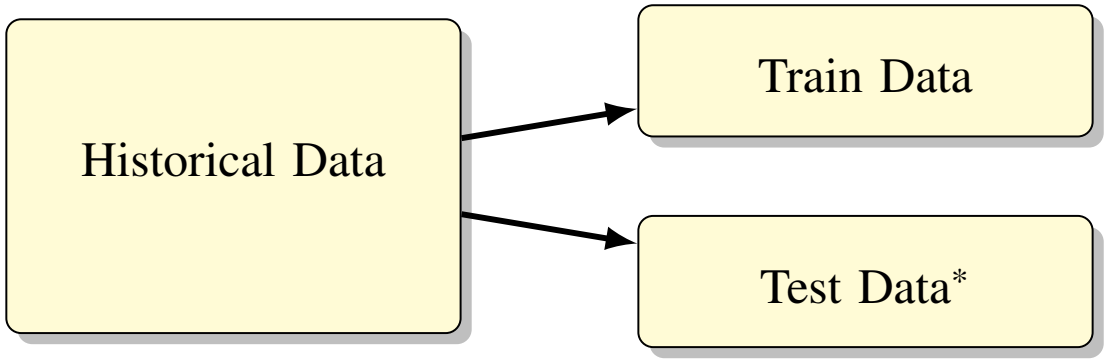


- The MSE[a] is monotonically decreasing as the polynomial order increases.
- Can only estimate the optimal value for d (about 5) by looking at plots of the fit — subjective, computationally prohibitive and not scalable.

[a] Note we use logarithmic scale when plotting values over relatively large scales

Splitting Historical Data into Train and Test Data

The issue is not with the MSE metric, but how we have used it.



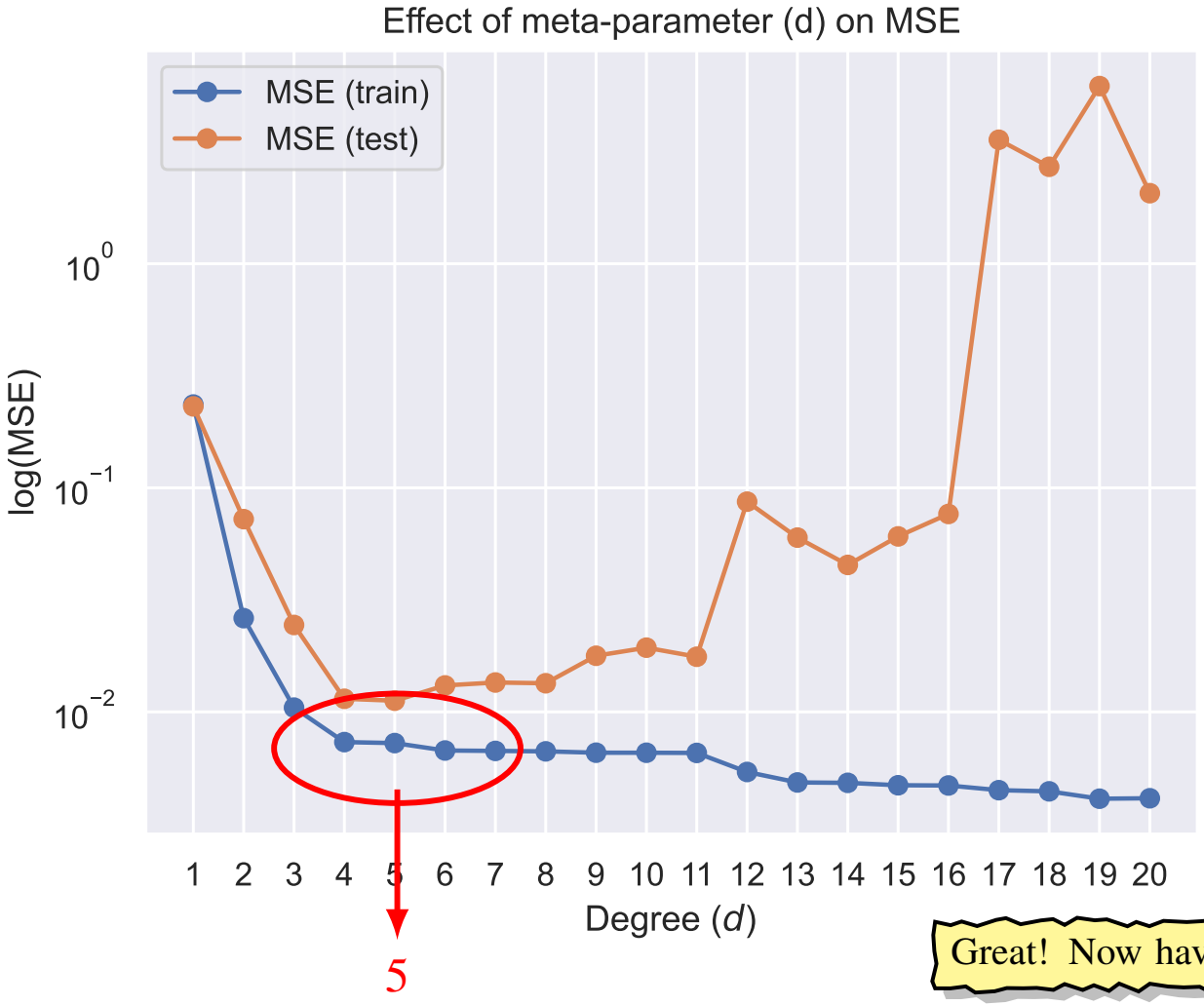
- Using the training data for both model fitting and evaluation purposes can lead to overly optimistic estimates about the performance of the model.
- This can result in choosing a suboptimal model that performs poorly when predicting on new data.
- Now my issues are
  - "*How do we split the historical data?*"
  - "*What happens if I don't have enough data?*"

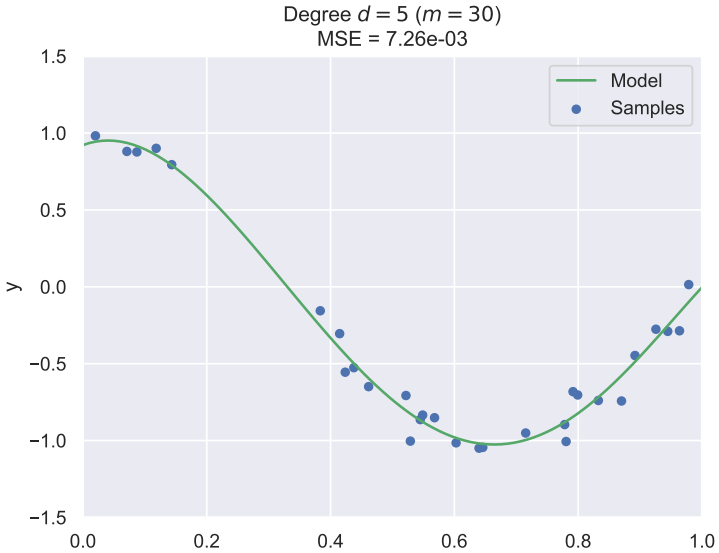
Default to train/test split of say, 60%/40%.

Use cross-validation (CV) methods.

*The label 'test' is problematic because this is often split again into 'test' and 'validation'. Some practitioners, eg Andrew Ng, use the term 'dev' set instead.

Effect of Meta-parameter on MSE (train and test data)





- Method: Fit the model to the train data and then evaluate it on the previously unseen test data.
- Select value of meta-parameter based on MSE (test).

Great! Now have a objective / automatic method

Bias and Variance

Under-fitting/over-fitting can be thought of in terms of high-bias/high-variance:

Bias

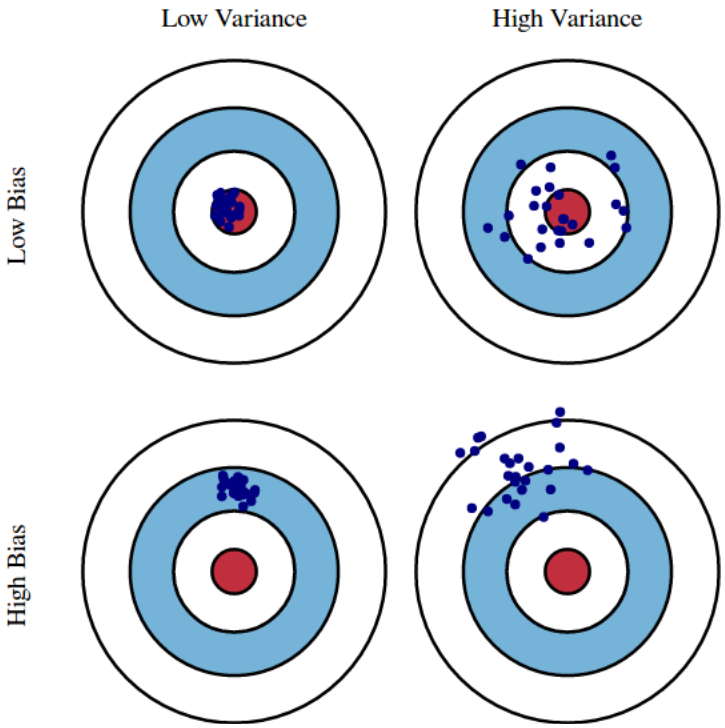
The error due to the difference between the expected (or average) prediction of our model and the correct value which we are trying to predict, over different realisations of the model.

Under-fitting results in high-bias

Variance

The error due to variance is taken as the variability of a model prediction for a given data point between different realisations of the model.

Over-fitting results in high-variance

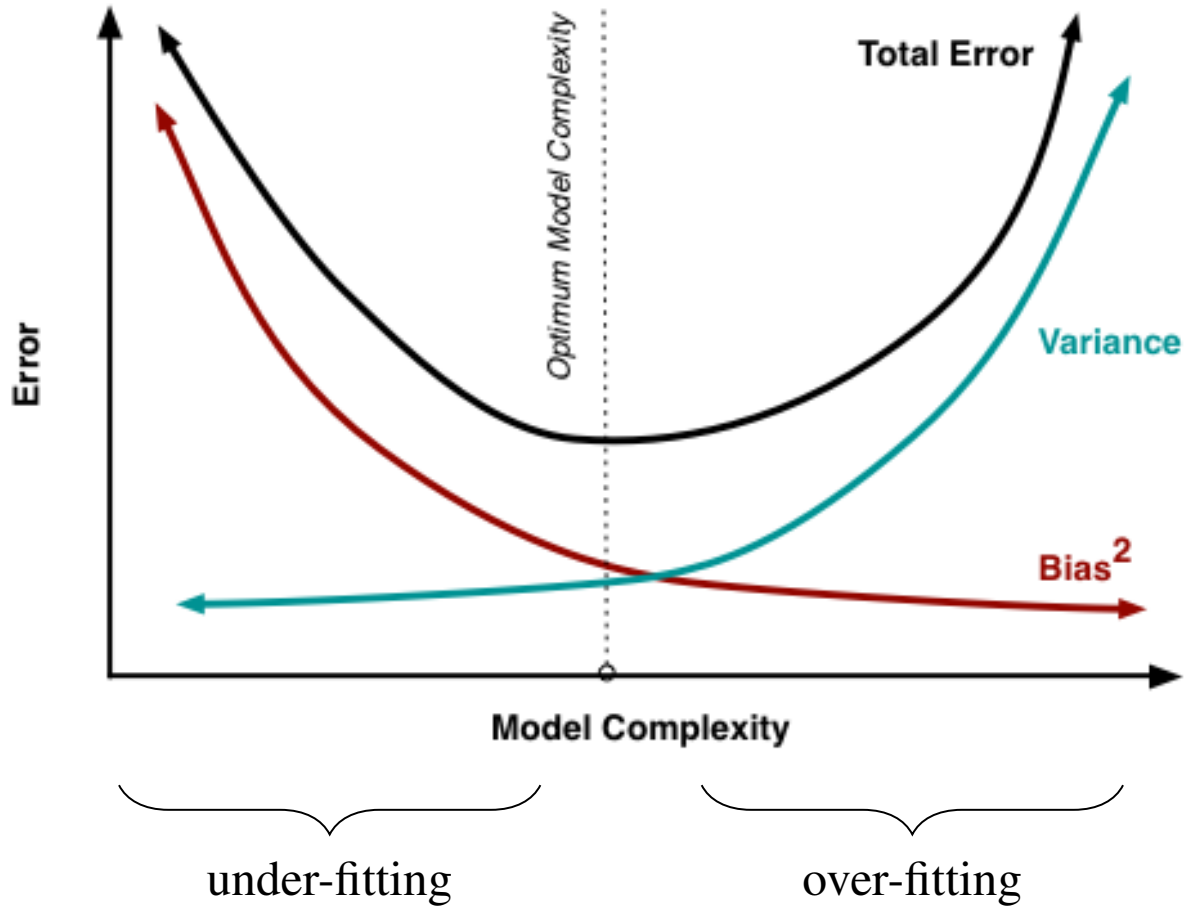


—-Scott Foremann-Roe, June 2012

The different realisations of the model can be computed by repeating the model building process over multiple datasets, or by sampling the given dataset.

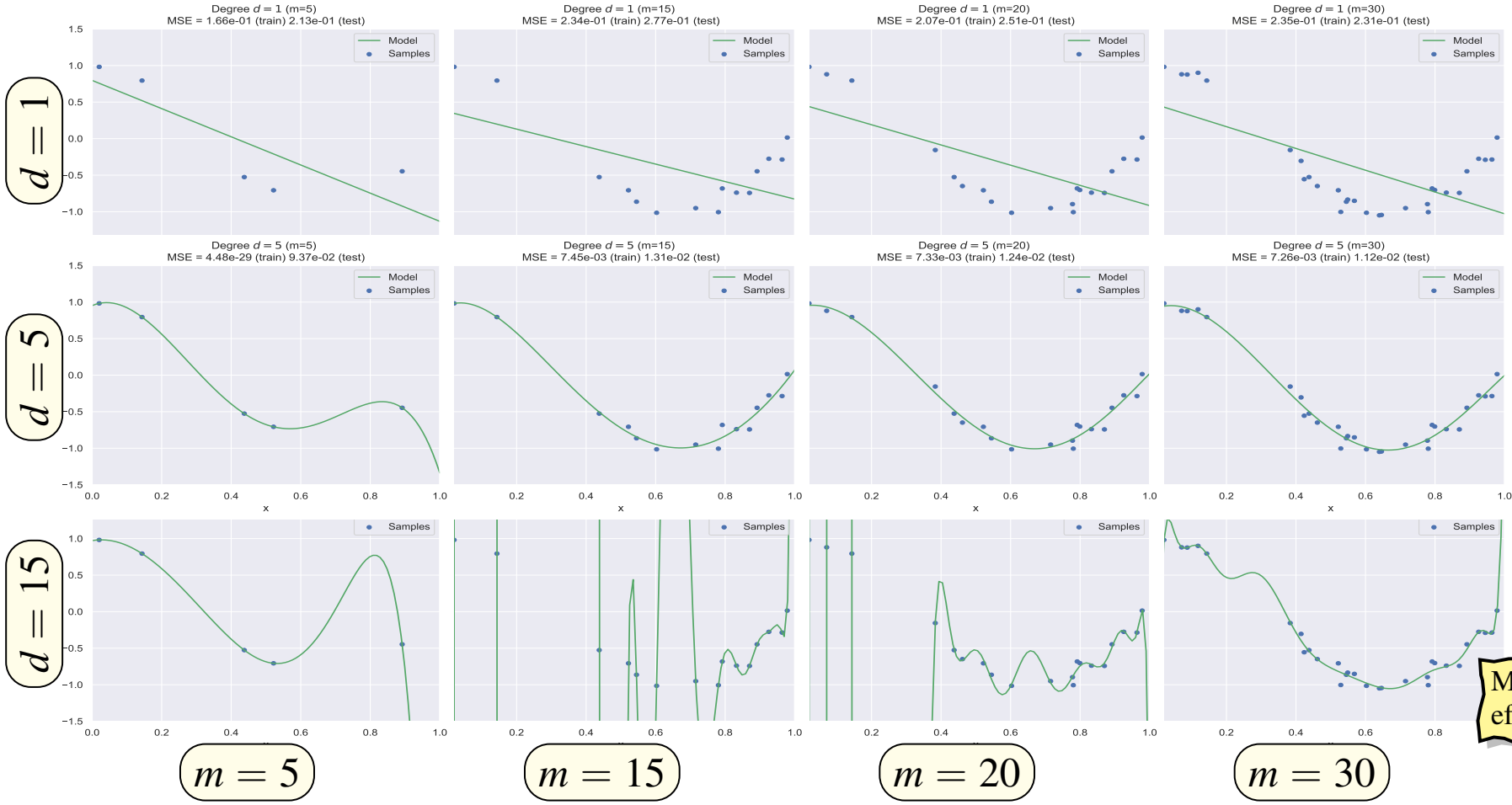
Bias–Variance Tradeoff

$$\text{Error} = \text{Bias}^2 + \text{Variance} + \text{Irreducible Error}$$



Effect of Number of Observations and Over/Under-fitting

Question: How does the best fit model change as, m, the number of observations change? . . .



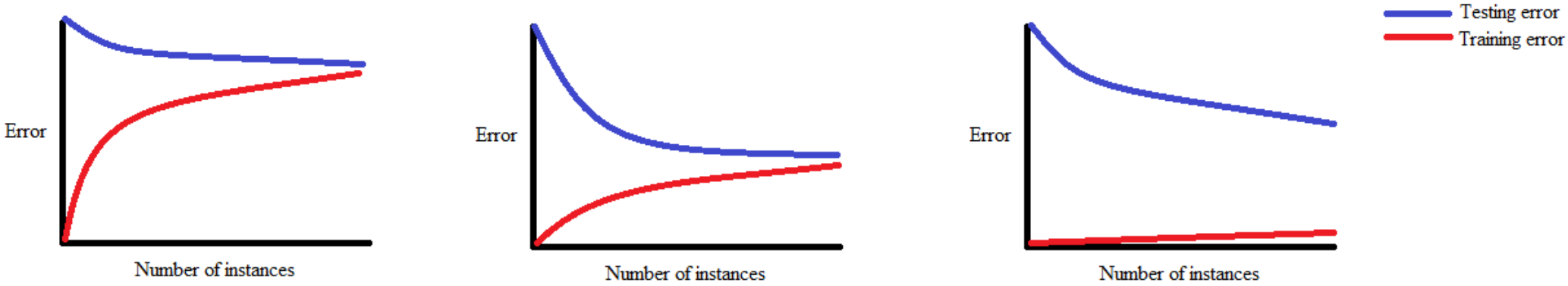
- When under-fitting (high-bias) the model is not sensitive (can't represent) to changes in the data.
- When over-fitting (high-variance) the model is too sensitive to changes in the data.
- More data does not affect an appropriate model much.

More data helps to reduce effects of overfitting.

Learning Curves

- The learning curve is a graph that compares the performance of a model on training and testing data over a varying number of training instances
- Should generally see performance improve as the number of training points increases.
- The comparison of performance on training and testing datasets is an indicator of how well the model can generalise to new data.
- Learning curve allows us to verify when a model has learning as much as it can about the data:
  - The performances on the training and testing sets reach a plateau.
  - There is a consistent gap between the two error rates.

Idealised Learning Curves

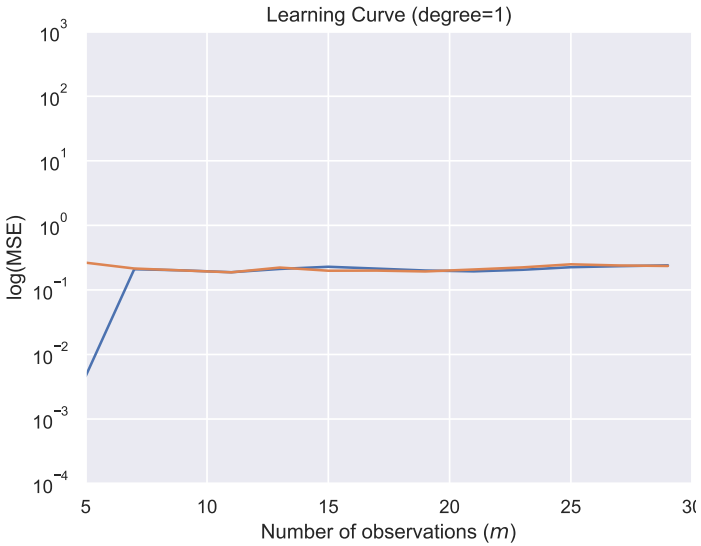


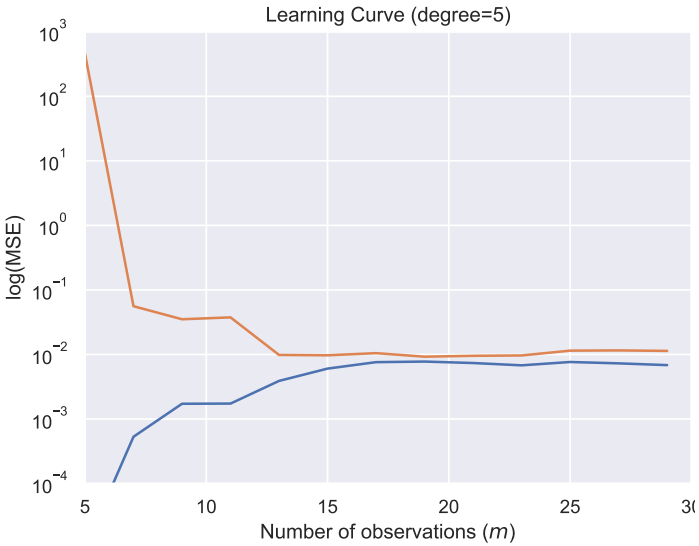
high bias

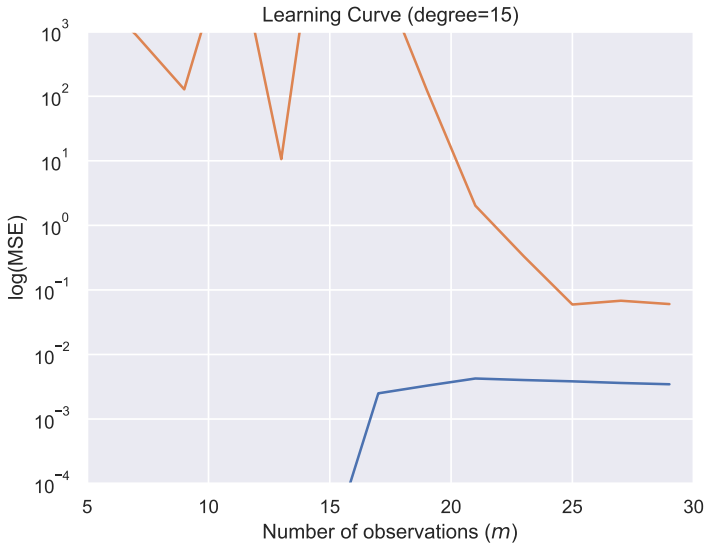
high variance

Practical Learning Curves

Real learning curves are more noisy, but the general structure is still visible — for our generated data set we have







high bias

high variance

Resolving High Bias/Variance

- High-Bias — Increase model complexity
- High-Variance — Decrease model complexity or/and increase number of observations.

Overview of Testing and Validating a Model

No single approach works, so need mixture of

- **theoretical** — various measures of statistical validity to determine whether there are problems in the data or in the model
  - Test for normality, uniform variance, ...
- **practical** — separate the data into train and test sets and evaluate on test (or validation set).
  - **accuracy** — how often/close does the model match the target?
    - Which metric should be used?
  - **reliability** — how robust is the model to different train sets?
    - How much data is needed for convergence (training)?
  - **utility** — can the model be used in practice? what is the marginal benefit of using the model?

    Metrics **lift** and **gain**: the improvement that you get from using a data mining model, when you compare it to random guessing (or other model).
- **reality check** – area experts to review model.

Cross Validation Methods

Splitting the dataset into train and test data results in another source of variation, especially for small datasets. Methods that address this:

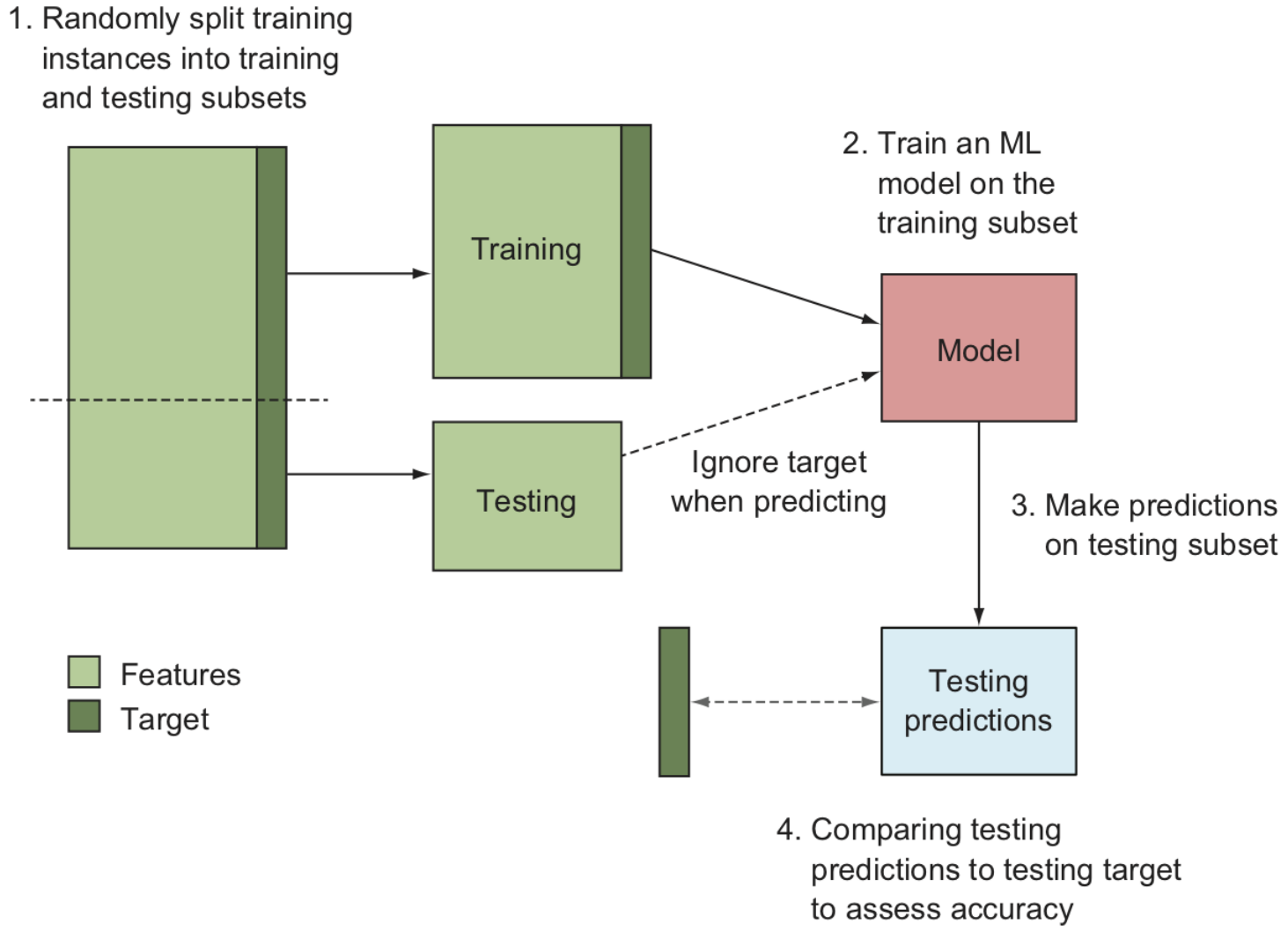
Holdout Method

- Simplest method — just select a train/test split ratio and randomly allocate observations to train and test datasets.
- The optimal parameters from training and the error estimates on the test dataset can be noisy.

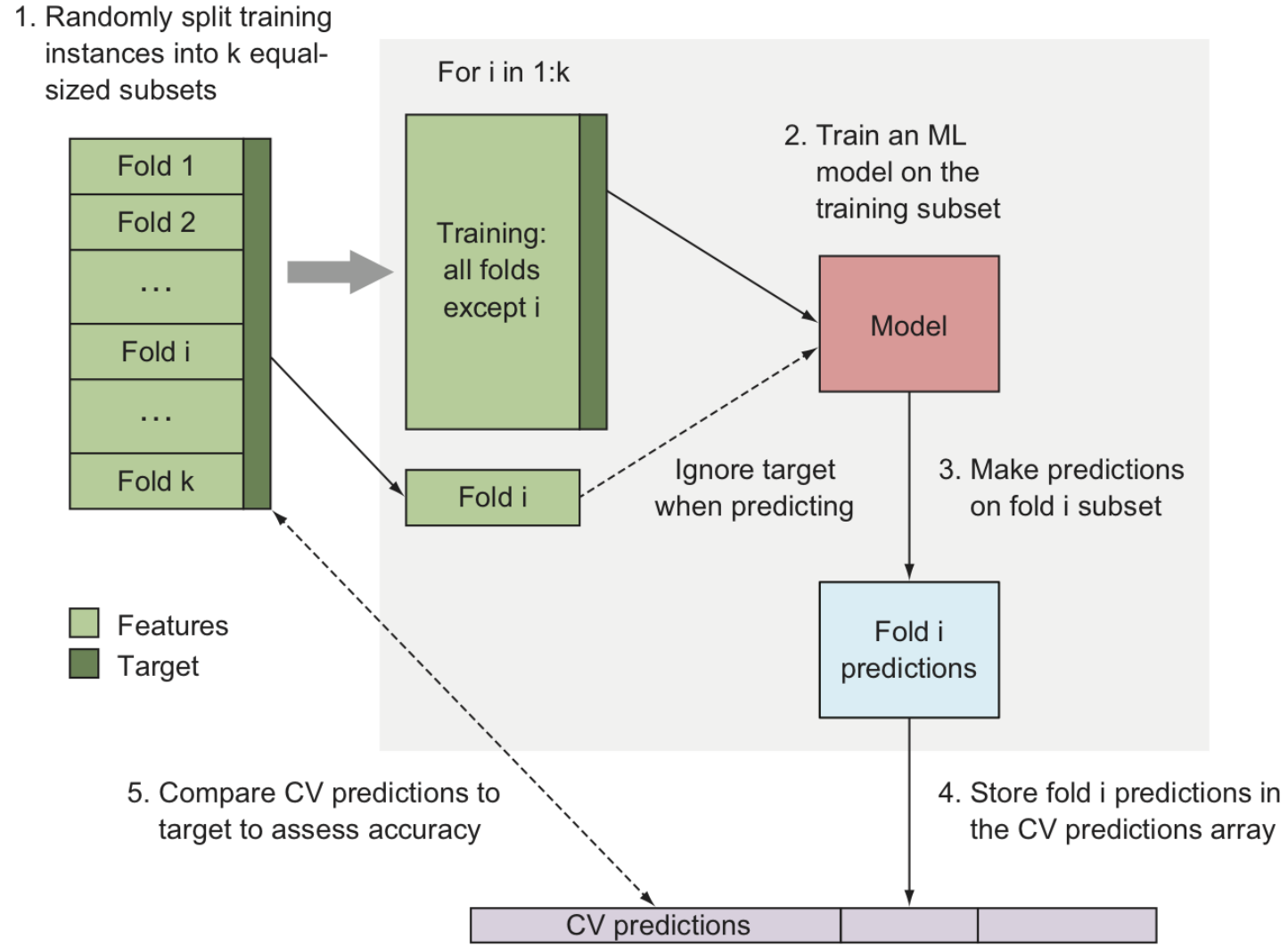
K-Fold Cross Validation

- Split the dataset into into k disjoint subsets, called folds (typical choices for k are 5, 10, or 20).
- For each fold, a model is trained on all the data except the data from that fold and is subsequently used to generate predictions for the data from that fold.
- After all k-folds are cycled through, the predictions for each fold are aggregated and compared to the true target variable to assess accuracy.
- This results in noise estimate for the parameters and the error but is more computationally intensive approach than the holdout method.

Holdout Method



K-Fold Cross Validation

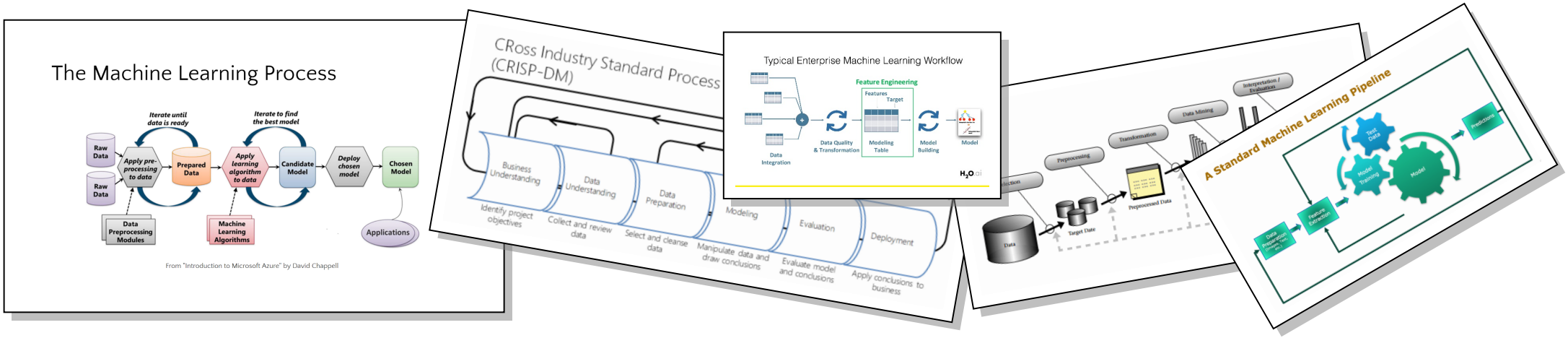


K-Fold Cross Validation

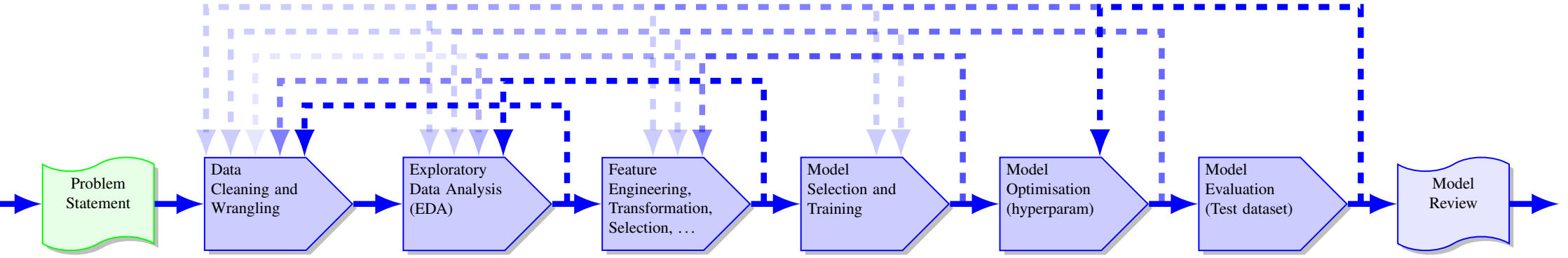
Typical process: Train model *k* times to generate *k* estimates of model performance and then average.

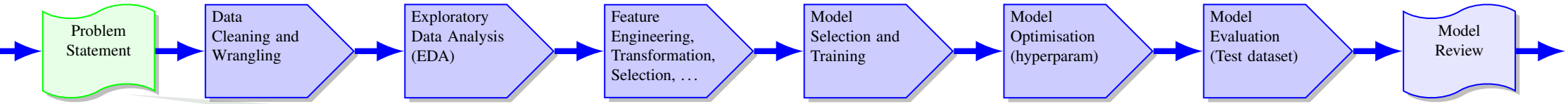
Data Mining Workflow

There are many, many . . .



So why not make YADMW (Yet Another Data Mining Workflow)?

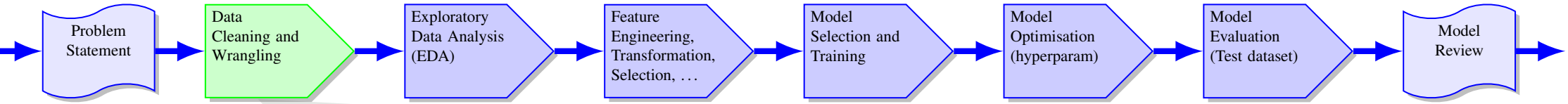




What type of problem is it?

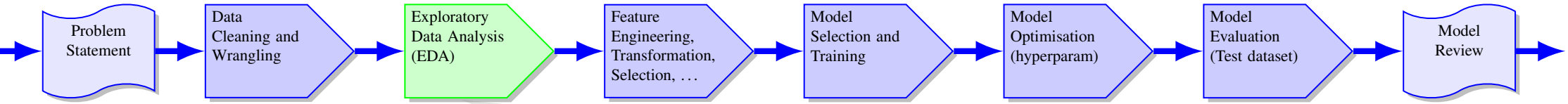
- Exploratory data analysis
  Do we just want to see what the data says?
- Association / Rule finding
  Are we searching for relations/patterns?
- Hypothesis testing (Statistical)
  Do we have a theory we wish to test?
- Model building
  Do we wish to build a representation of some pattern within the data?
  - Supervised ⇐ data spilt into input variables (features) and output variable(s) (target(s))
    - Classification (target is categorical) vs regression (target is continuous)
  - Unsupervised ⇐ no target
    - Clustering — grouping similar cases

Data mining / Machine Learning workflow



How to import and prepare data for subsequent analysis/processing?

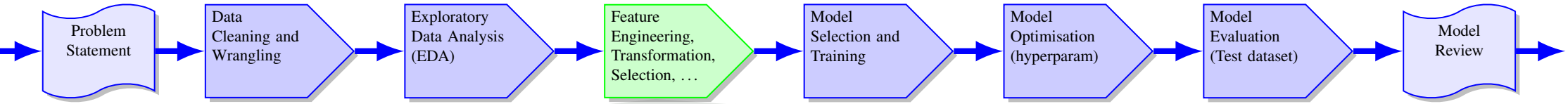
- Multiple file formats
  - Pandas supports a wide collection of file formats but default options often need to be changed to suit data.
  - Main file format (Comma Separated Values (csv)) does not support meta-data, is slow, and results in large files $\Longrightarrow$ use other formats (pickle, feather) to store datasets between steps in the workflow.
- Assumptions made by input parser can be important (i.e., bite you when you least expect)
  - Scientists rename human genes to stop Microsoft Excel from misreading them as dates
  - Pandas vs excel use different heuristics to decide on data type of each variable.
- Sub-tasks
  - Check dimension (number of rows/cases, number of columns/variables).
  - Check data types (categorical, ordinal, or numerical (discrete/continous)) of each variable.
  - Check for missing values, encoding errors, etc.
  - Merge tables, apply filters, and general data wrangling to generate (tabular) dataset suitable for EDA.



What is the data telling us?

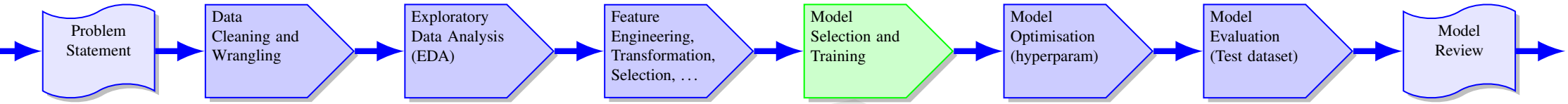
- Univariate descriptive statistics — examine each variable
  - What are typical values?
  - What is the variation / spread / range?
  - What does the data look like ...bell curve, bath tub curve, etc. ?
- Bi- / multi- variable descriptive statistics
  - Identifying relationships between variables.
- All descriptive statistics methods summarise data:

✔ A summary is good since it helps to focus on simpler and important aspects.
✘ A summary is bad if it focuses on irrelevant or the wrong aspects.
⇒ Need to use multiple methods, be aware of their strengths/deficiencies.



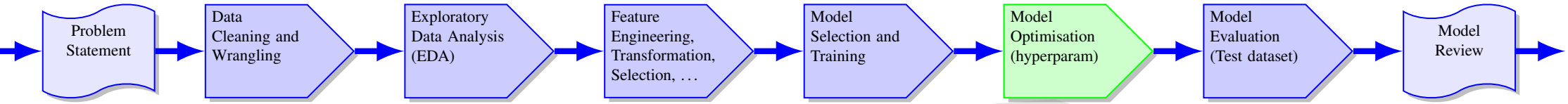
Can we transform, encode/bin, select, ..., the given features to improve model training?

- Better features can mean:
  - Better model performance and reduce training times.
  - Simpler models become applicable — think linear/logistic regression.
  - More explainable models — the future of machine learning (hopefully).
  - Cheaper and easier models to deploy.
- Feature selection reduces the number of features used in the model:
  - Drop features that have low variability.
  - Drop features that have no relation to target.
  - Drop features that are highly related to other features — multicollinearity.
  - Keep features whose addition to model have the largest improvement in model score.
- Feature extraction merges existing features to generate (hopefully) fewer features with essentially all the variation.



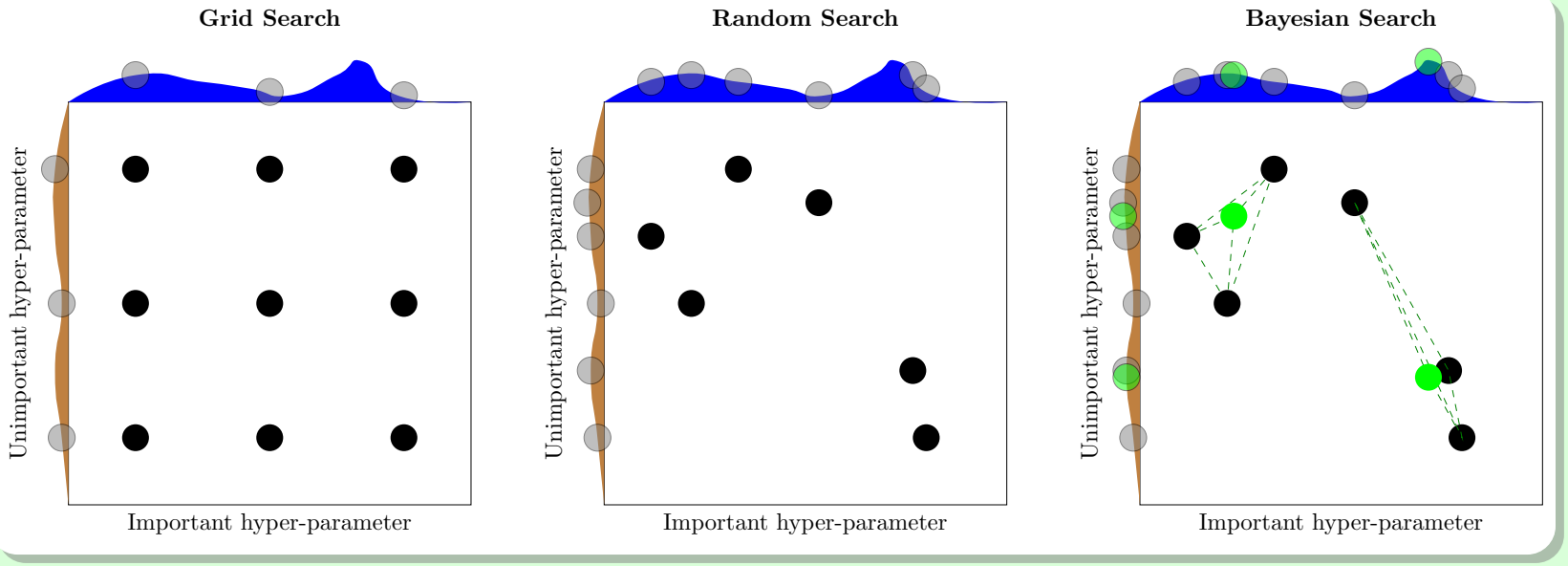
Which models are suitable?

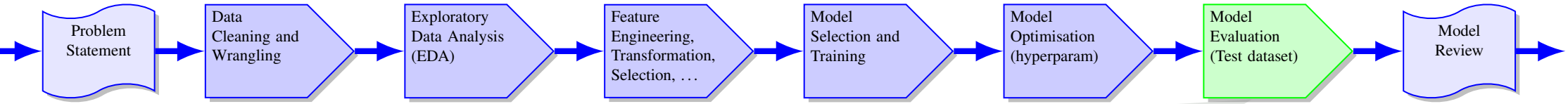
- Models vary greatly in terms of capabilities/deficiencies — usually aim to build a short list of candidate models, which are subsequently optimised in the next step.
- Select models based on different algorithms/approaches.
- Select (loss function and) evaluation metric.
  - Loss function is used to train model, evaluation metric is used to evaluate model (post training).
- Relative model performance can help identify issues with data.
  - Outliers can negatively affect linear regression but have smaller impact on decision tree based models.



How do we determine optimal values of the hyper-parameters?

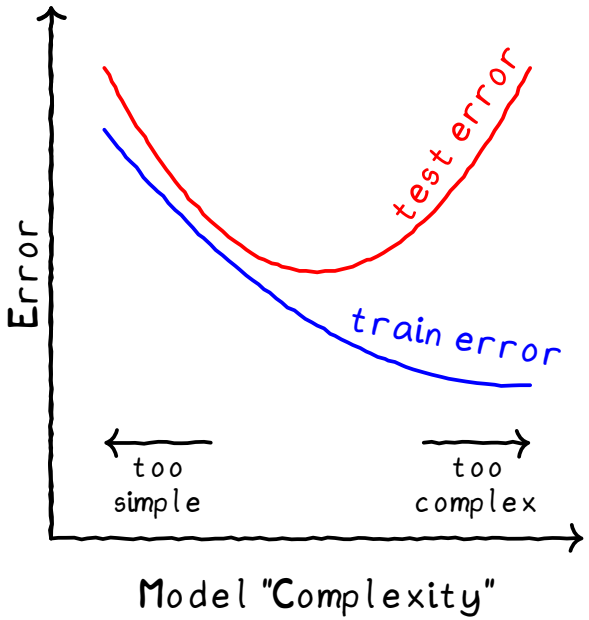
- Most models have options which control how a model "learns" from the training data.
- Three search stratagies: Grid search $<$ Random search $\ll$ Bayesian search

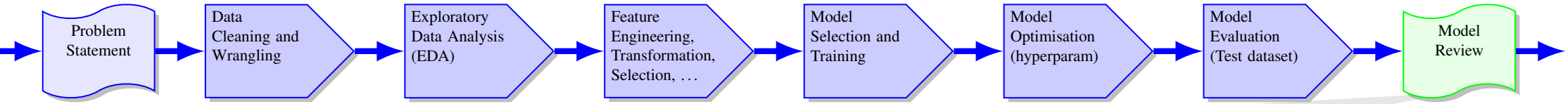




How well does the model generalise (to unseen data)?

- In the machine learning approach (vs statistical approach) we rely on model performance on **unseen data** to evaluate models.
  - Split data into train/test, only use train dataset for all modelling decisions.
  - Data leakage (MachineLearningMastery article), where information outside the train dataset is used in model building.
- Is there evidence for overfitting?
  - Does the model perform much better on training dataset than on the test dataset?
- Multiple techniques to address overfitting:
  - Regularisation (linear / logistic regression).
  - Trimming (decision trees).
  - Dropout (neural networks), Batch normalisation (CNN).





How well have we addressed the problem statement?

- A what level of accuracy (or other metrics) does a model become useful?
  - This is a business, medical, ... decision
  - The larger the relative payoff the weaker the model can be and still be useful.
- OK, finally ready to implement/deploy model ...
  - Separate skillset / concerns
  - MLOps = ML + DevOps
  - Monitoring of model drift needed.
- towards data science What is MLOps — Everything You Must Know to Get Started

Q: Why don't we automate all of this shtuff?
Tools are getting better and easier to use, but need intervention/direction (data can be weird in weird ways)

– xkcd.com/2054