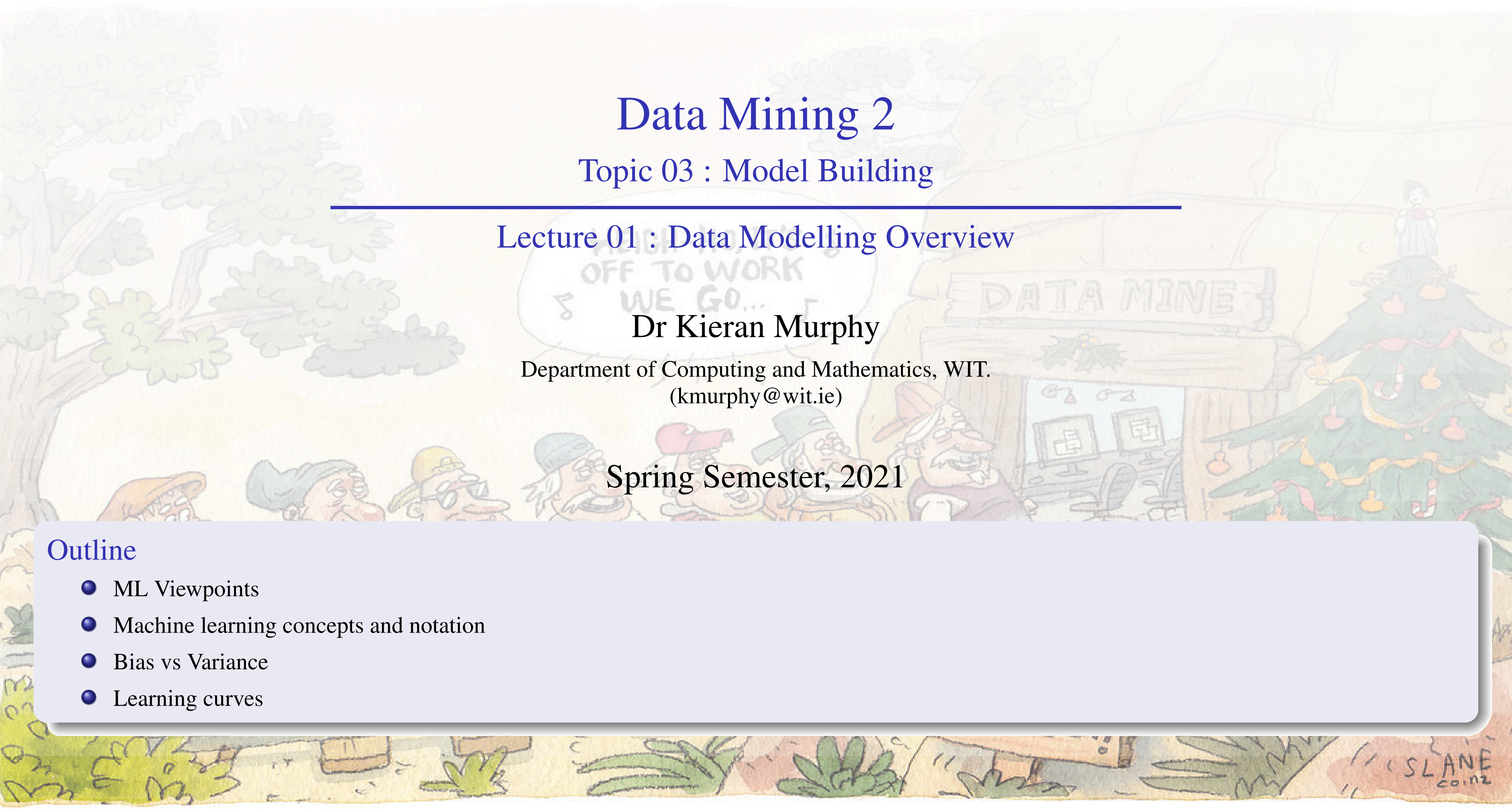


Data Mining 2

Topic 03 : Model Building

Lecture 01 : Data Modelling Overview

Dr Kieran Murphy

Department of Computing and Mathematics, WIT.
(kmurphy@wit.ie)

Spring Semester, 2021

Outline

- ML Viewpoints
- Machine learning concepts and notation
- Bias vs Variance
- Learning curves

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.

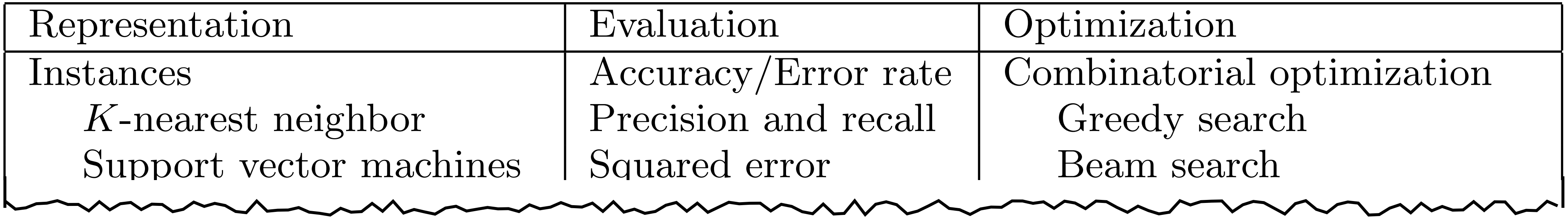
Three Components of a ML Problem — Representation

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

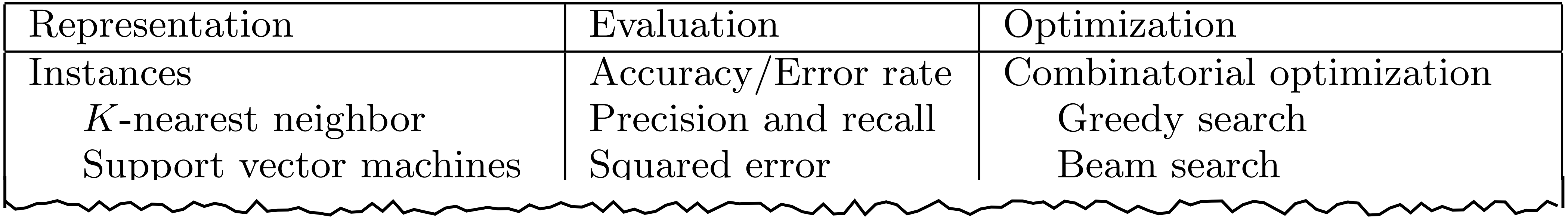


Representation	Evaluation	Optimization
Instances	Accuracy/Error rate	Combinatorial optimization
K-nearest neighbor	Precision and recall	Greedy search
Support vector machines	Squared error	Beam search

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.

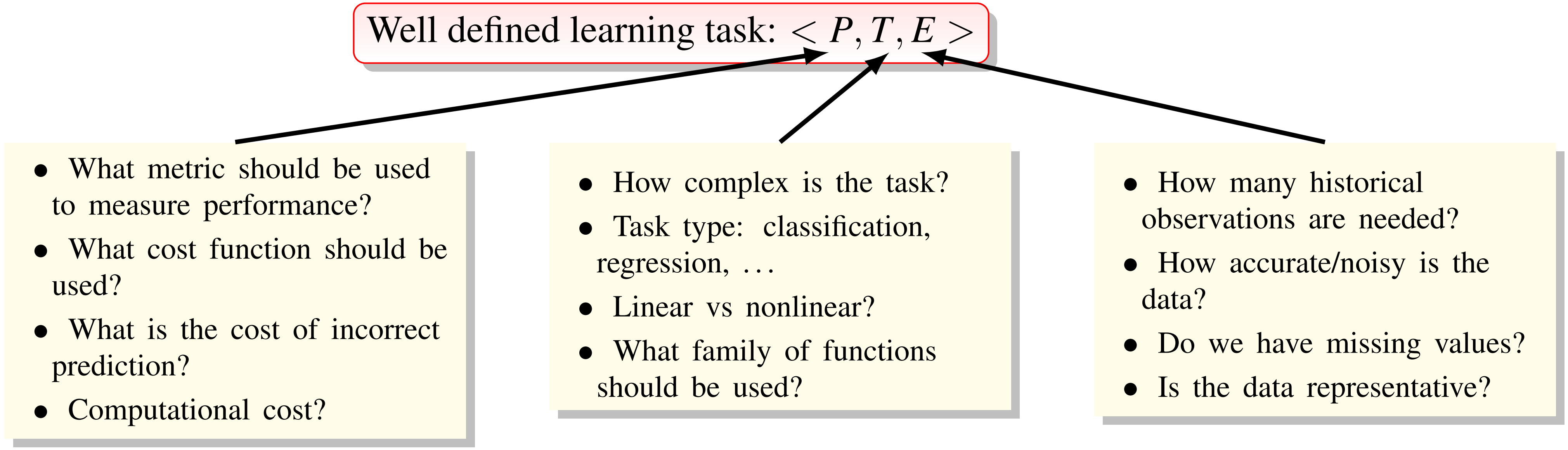
- How do we search among all the alternatives?
- Can we use some greedy approaches, branch and bound approaches, gradient descent, linear programming or quadratic programming methods.

Data Modelling (aka Machine Learning)

As alternative to the three component (Representation / Evaluation / Optimisation) viewpoint we can think of a machine learning problem as

Definition 1 (Machine Learning)

Study of algorithms that improve their performance P at some task T with experience E.



Well defined learning task: $< P, T, E >$

- What metric should be used to measure performance?
- What cost function should be used?
- What is the cost of incorrect prediction?
- Computational cost?

- How complex is the task?
- Task type: classification, regression, …
- Linear vs nonlinear?
- What family of functions should be used?

- How many historical observations are needed?
- How accurate/noisy is the data?
- Do we have missing values?
- Is the data representative?

Taxonomy of Machine Learning Models . . .

. . . by Intuition/Modivation

- 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.

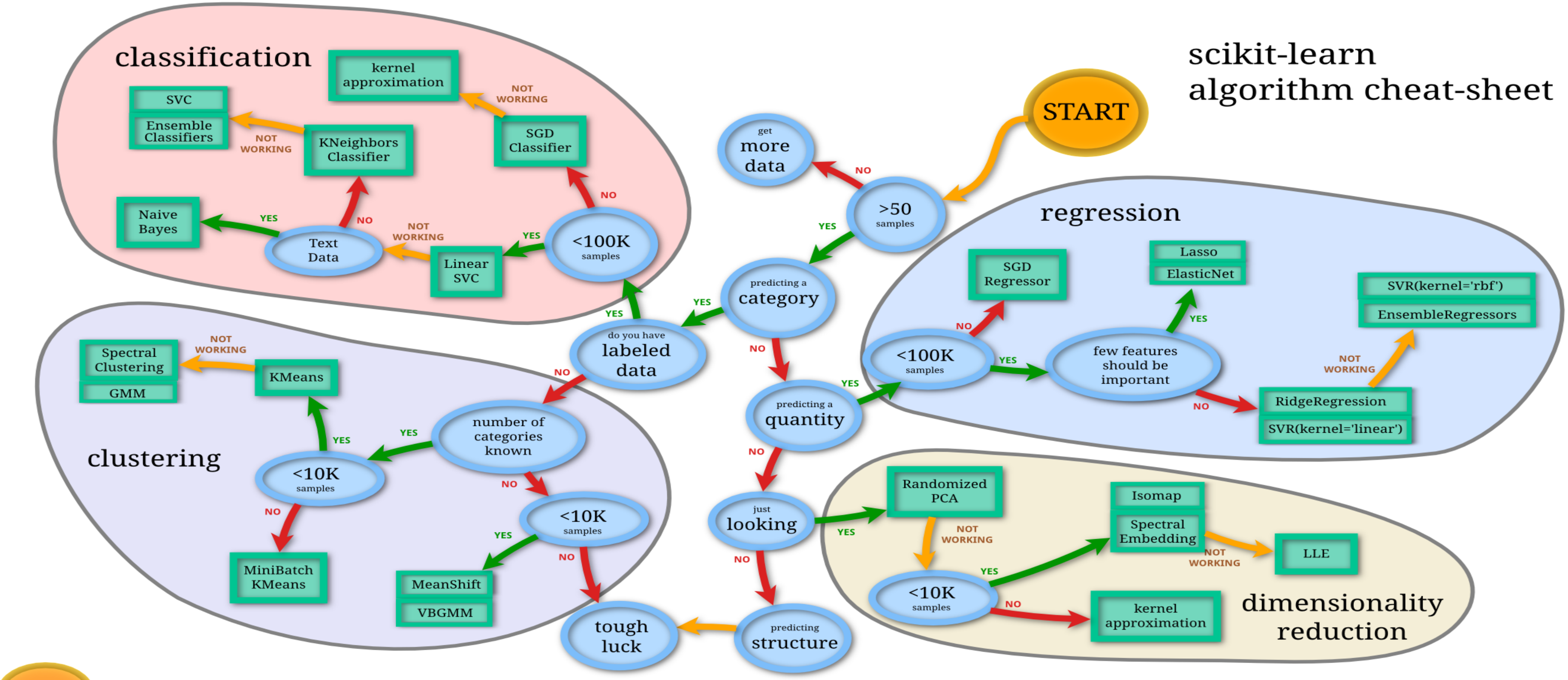
. . . by Algorithmic Properties

- 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.

Scikit-learn Flowchart of Models (Shallow Learners)



Back

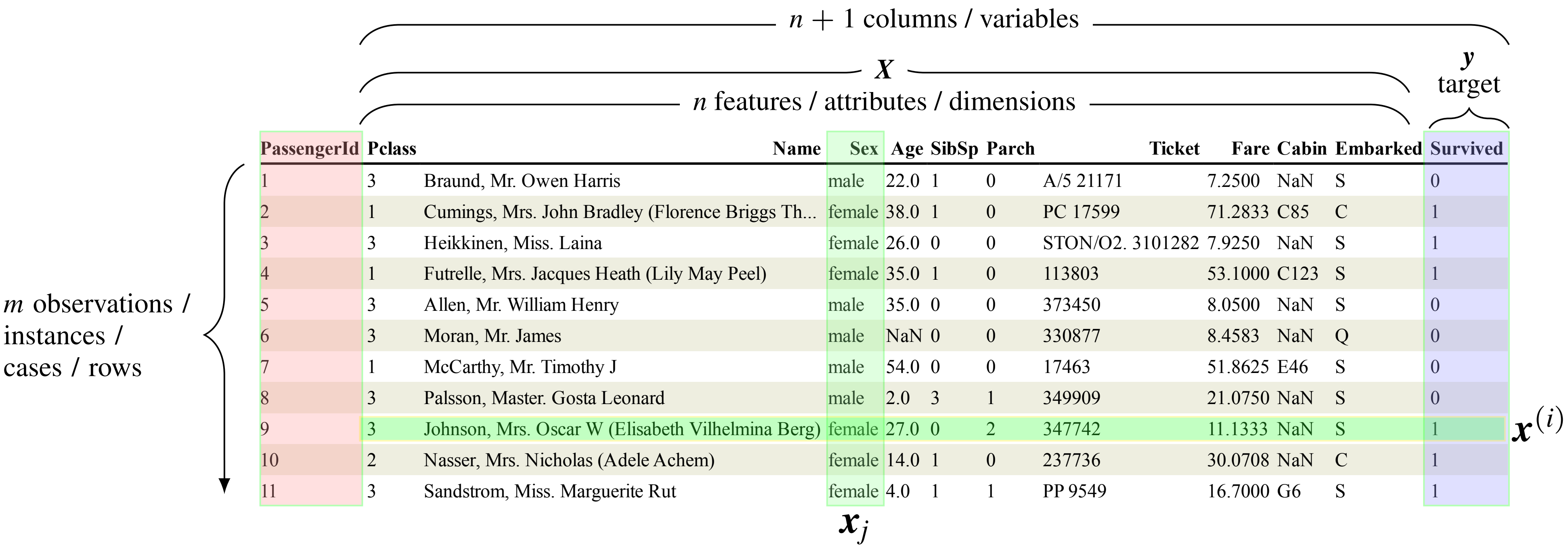
Statistical Models vs Machine Learning Models

	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

Dataset Terminology / Notation



$n + 1$ columns / variables

$\boldsymbol{X}$

n features / attributes / dimensions

$\boldsymbol{y}$ target

m observations / instances / cases / rows

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

$\boldsymbol{x}^{(i)}$

$\boldsymbol{x}_j$

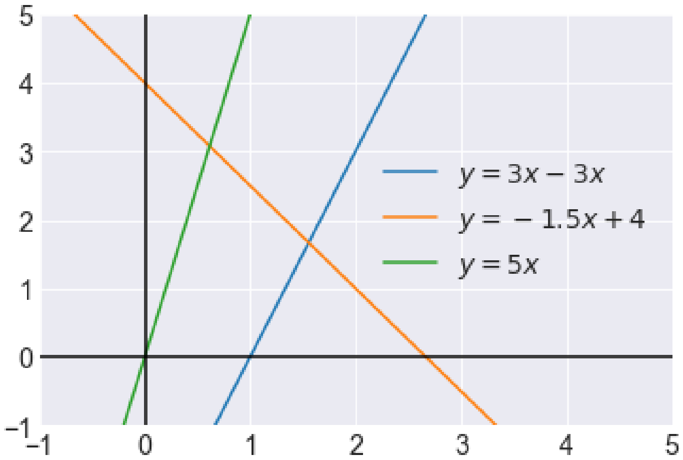
- A labeled dataset consists of m rows $\times$ $(n+1)$ columns / variables.
- Use bold to represent vectors and matrices.
- Use subscripts to indicate particular feature / attribute / column . . . . $\boldsymbol{x}_j$
- Use superscript in parenthesis to indicate particular observation / instance/ case / 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:

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

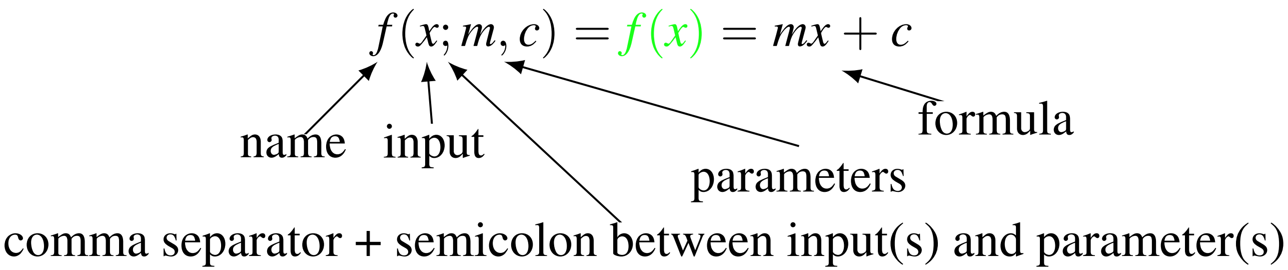
where given coefficients $m \neq 0$ and c.



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



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

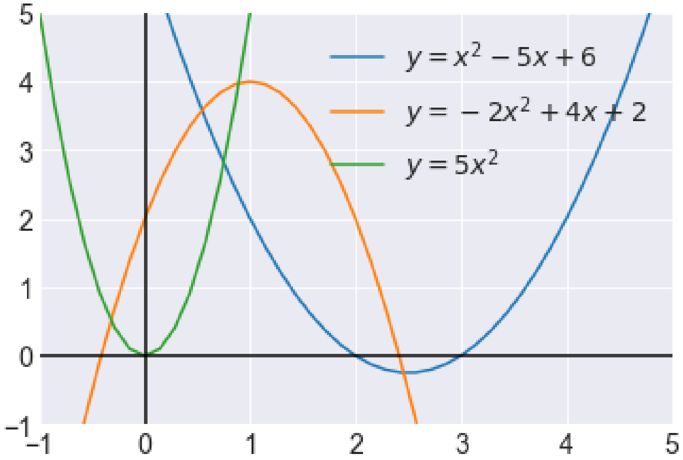
name
input
formula
parameters
comma separator + semicolon between input(s) and parameter(s)

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

And the formula for a quadratic:

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

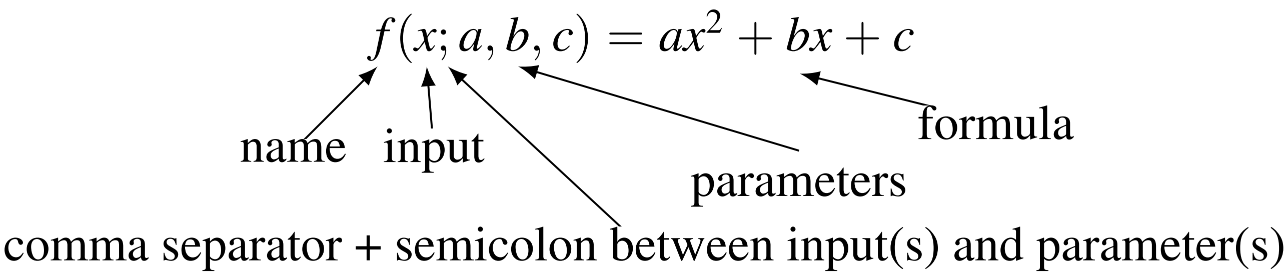
with given coefficients $a \neq 0$, b and c.



Examples include

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

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 + semicolon between input(s) and parameter(s)

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

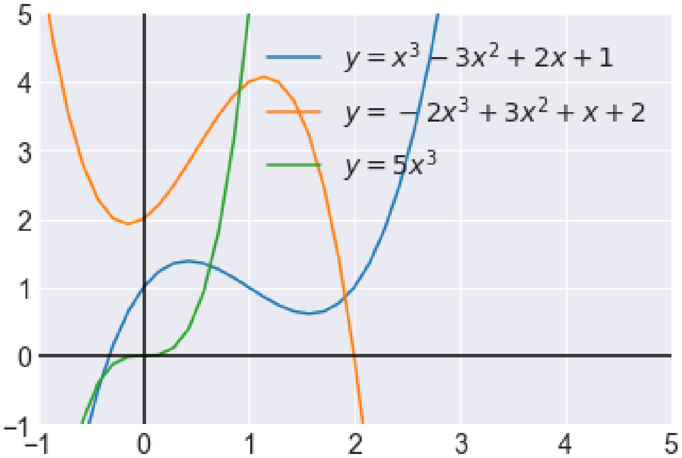
And the formula for a cubics:

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

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

Examples include

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



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

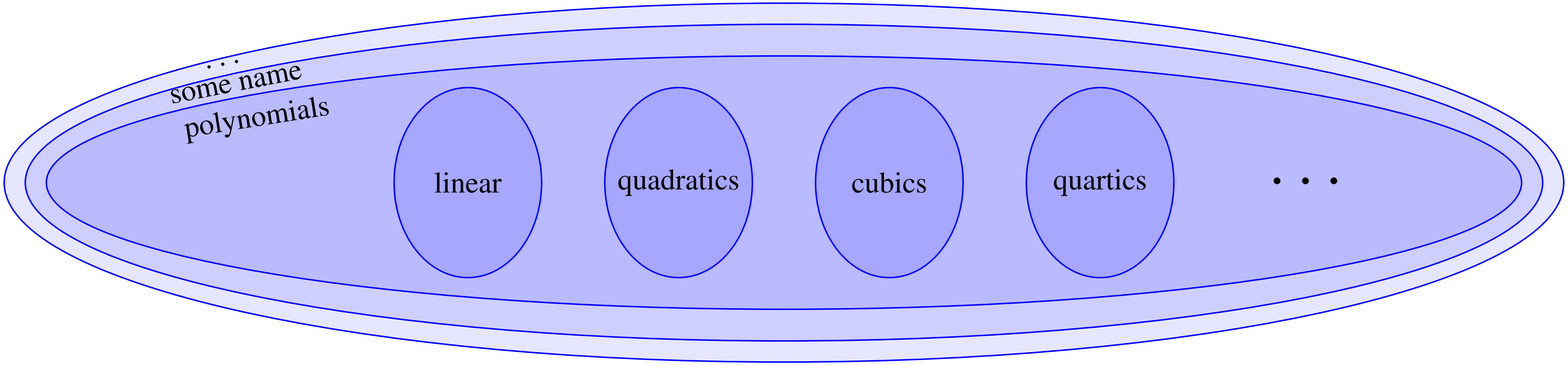
The family of polynomials is all functions of the form

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

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

remember we use bold font to indicate vectors.

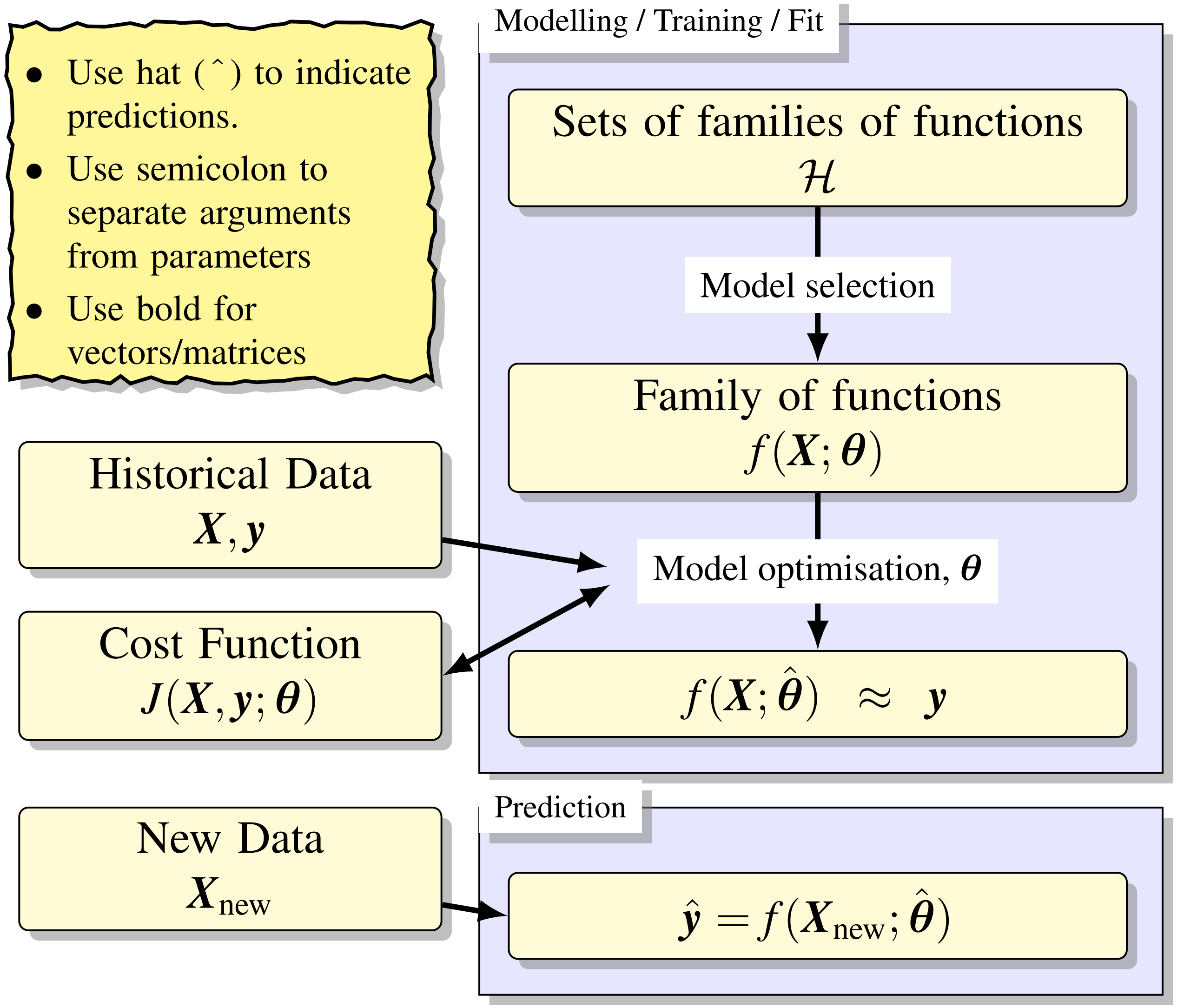
So we end up with this idea of (nested) families (or sets) of functions



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

ML Problem Terminology

- Use hat (ˆ) to indicate predictions.
- Use semicolon to separate arguments from parameters
- Use bold for vectors/matrices



Modelling / Training / Fit

Sets of families of functions $\mathcal{H}$

Model selection

Family of functions $f(\boldsymbol{X};\boldsymbol{\theta})$

Historical Data $\boldsymbol{X},\boldsymbol{y}$

Model optimisation, $\boldsymbol{\theta}$

Cost Function $J(\boldsymbol{X},\boldsymbol{y};\boldsymbol{\theta})$

$f(\boldsymbol{X};\hat{\boldsymbol{\theta}}) \approx \boldsymbol{y}$

Prediction

New Data $\boldsymbol{X}_{\text{new}}$

$\hat{\boldsymbol{y}} = f(\boldsymbol{X}_{\text{new}};\hat{\boldsymbol{\theta}})$

Ingredients

- 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

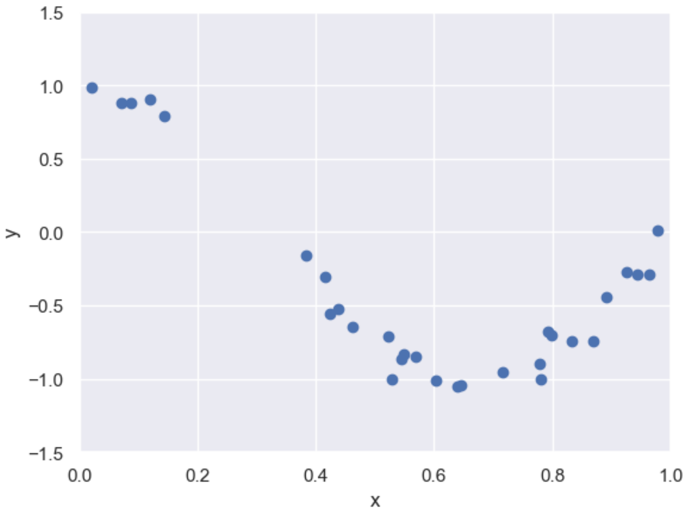
$$J(\boldsymbol{X}, \boldsymbol{y}; \boldsymbol{\theta}) = \frac{1}{m} \sum_{i=1}^{m} \left|\hat{y}^{(i)} - y^{(i)}\right|$$

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.
  - 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 probability, say 1/1,000,000 than when it is wrong with probability 1/1,000 — cross entropy.

A Motivating Problem

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



- We will tighten the specification and work with polynomials only

$$f(x) = \theta_k x^k + \cdots + \theta_2 x^2 + \theta_1 x + \theta_0$$

- Appear to have $k + 2$ parameters: the order*, k, and the $k + 1$ coefficients.
- Number of parameters is not fixed so have 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 ($k = 1$) to quadratic ($k = 2$) to cubic ($k = 3$), etc.).
- General convention: We treat k as a meta-parameter†, and treat the coefficients as parameters‡, so we get a parametric method.

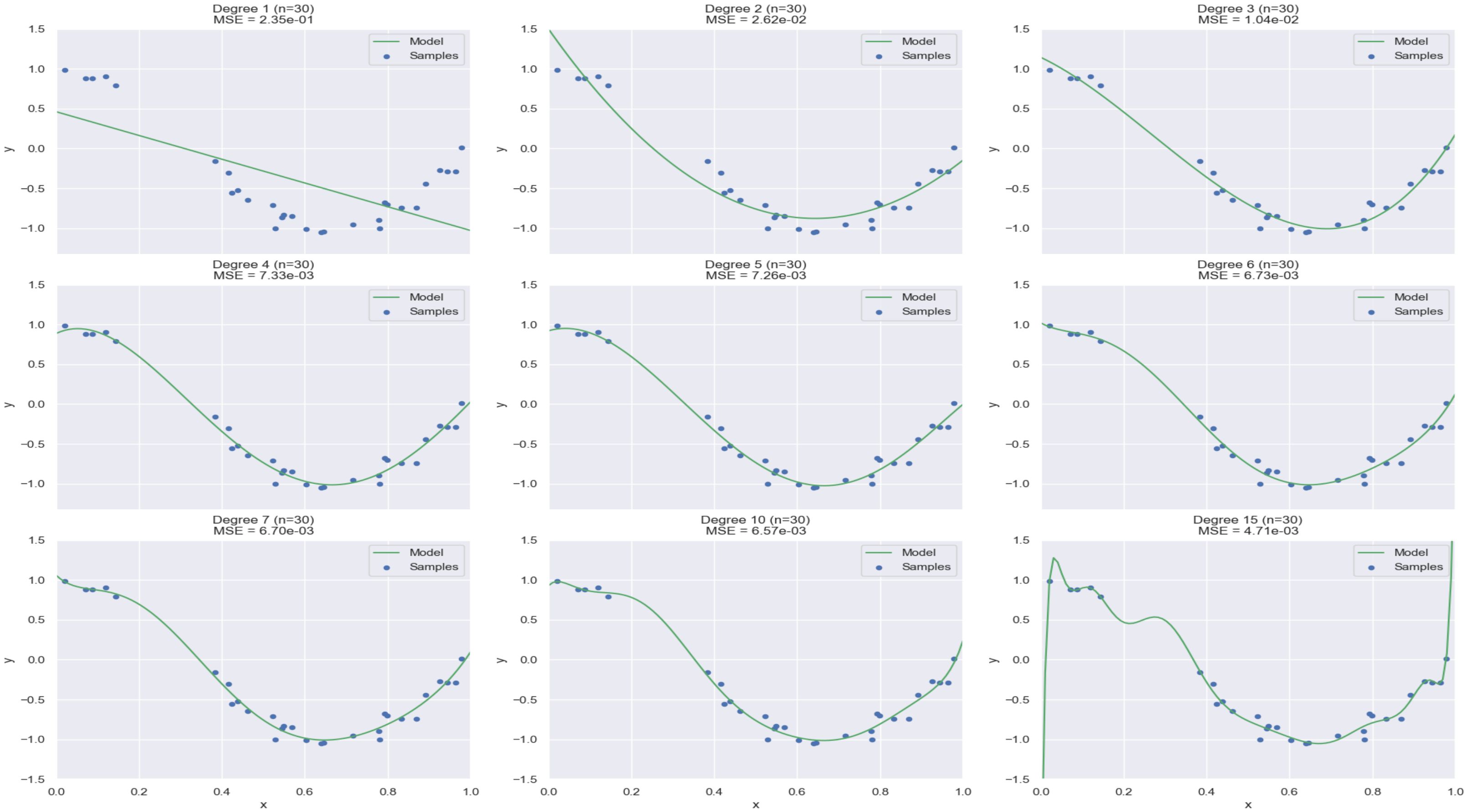
*In a polynomial the highest degree is called the order

†we decide on appropriate value ourselves

‡get a algorithm to determine optimum values

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

Varying Meta-parameter, *k*


Degree 1 (n=30) MSE = 2.35e-01

Degree 2 (n=30) MSE = 2.62e-02

Degree 3 (n=30) MSE = 1.04e-02

Degree 4 (n=30) MSE = 7.33e-03

Degree 5 (n=30) MSE = 7.26e-03

Degree 6 (n=30) MSE = 6.73e-03

Degree 7 (n=30) MSE = 6.70e-03

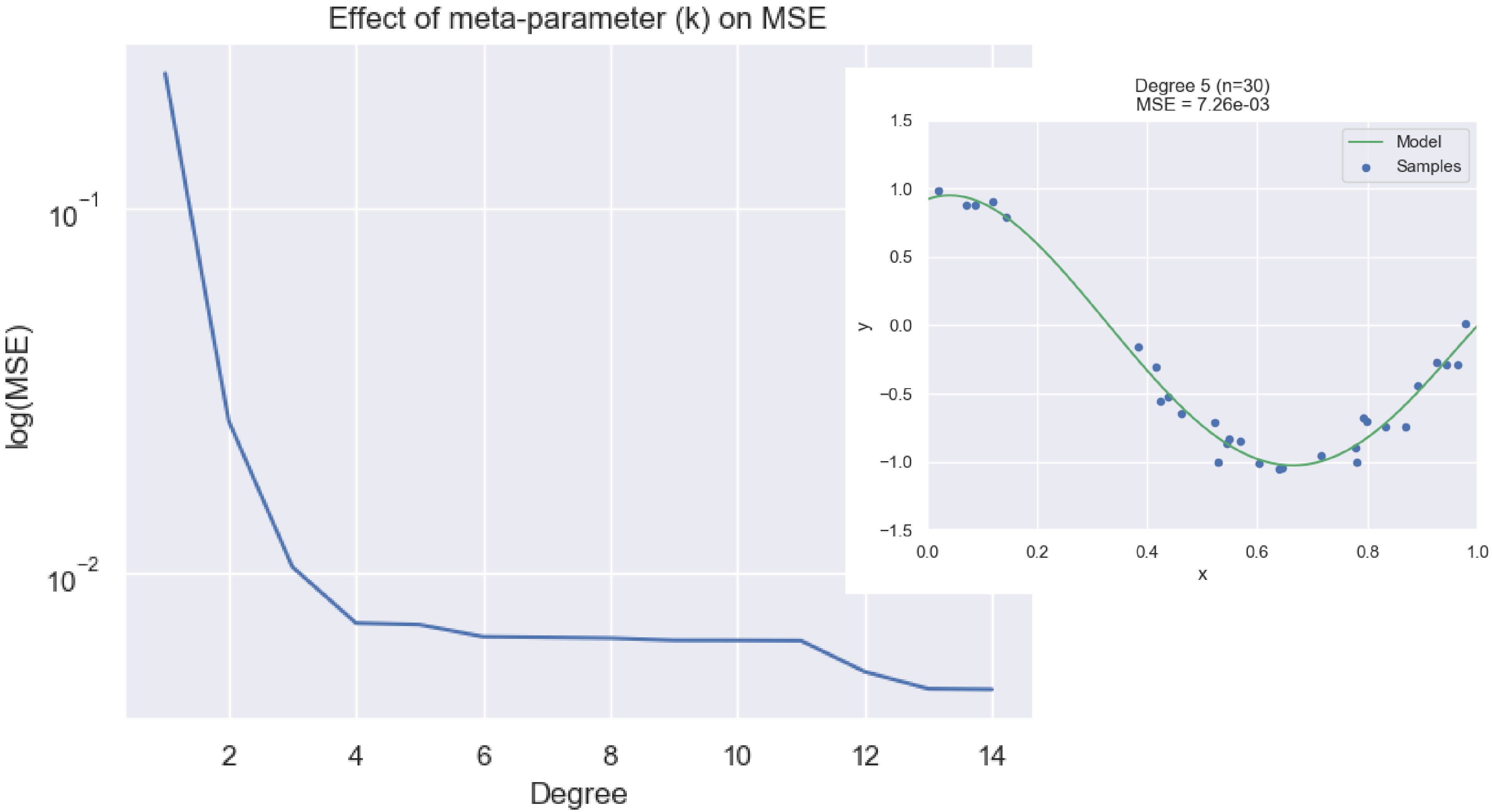
Degree 10 (n=30) MSE = 6.57e-03

Degree 15 (n=30) MSE = 4.71e-03

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

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

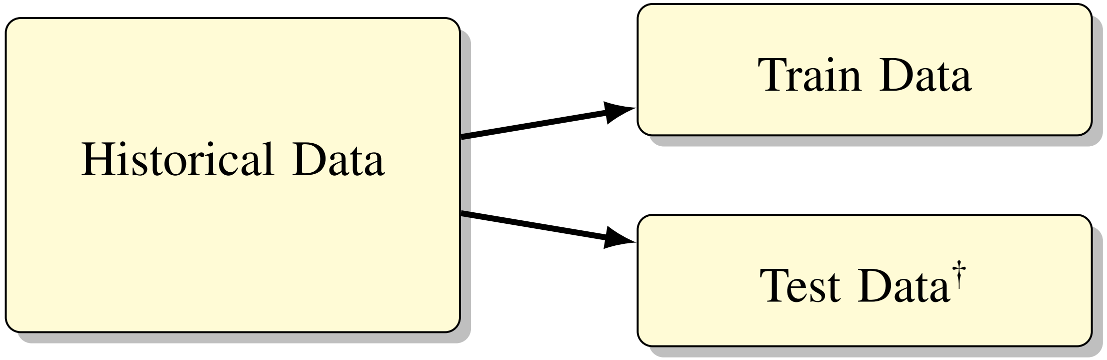
Effect of Meta-parameter on MSE



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

Splitting Historical Data in Train and Test Data

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



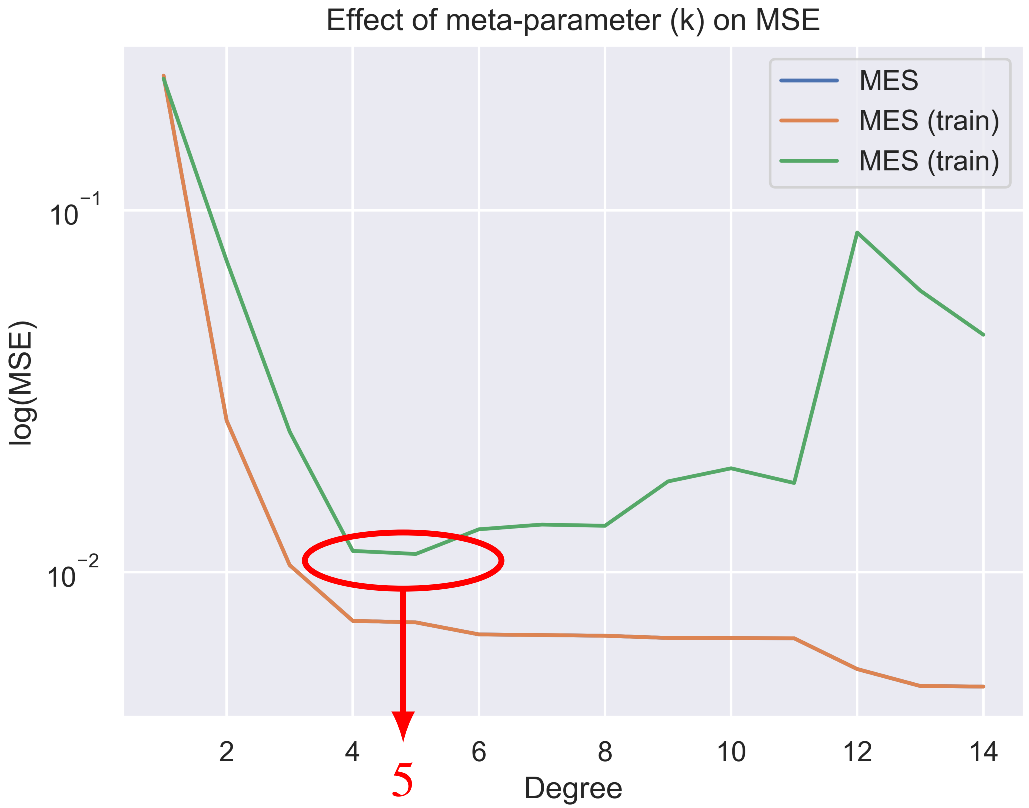
Used to train model

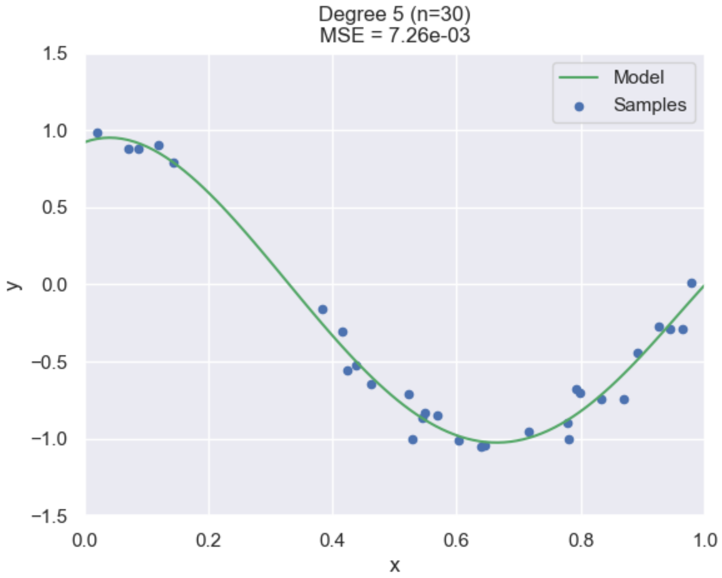
Used to evaluate model

- 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?*” Default to train/test split of 80%/20%.
  - “*What happens if I don’t have enough data?*” Use cross-validation 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(train).

Danger of Overfitting Example: The Fukushima Power Plant Disaster I

The problem

Want to design a power plant to withstand earthquakes. Strength of structure is measured by expected number of years between earthquakes of magnitude up to the expected failure magnitude.

Implementation

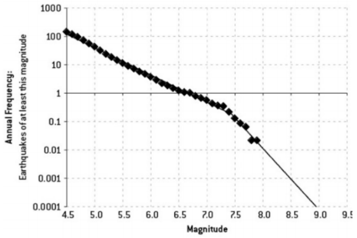
- Engineers used the Gutenberg-Richter Law[‡], which allows the predicting the probability of a very strong earthquake from the frequency that very weak earthquakes occur.
- This is useful because weak earthquakes[§] happen almost all the time and are recorded by geologists, so the engineers had quite a large dataset to work with.

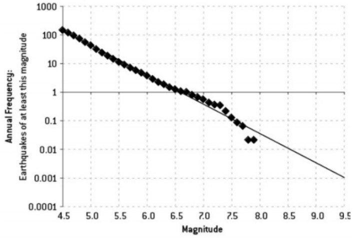
[‡] Law states that the relationship between the magnitude of an earthquake and the logarithm of the probability that it happens is linear.

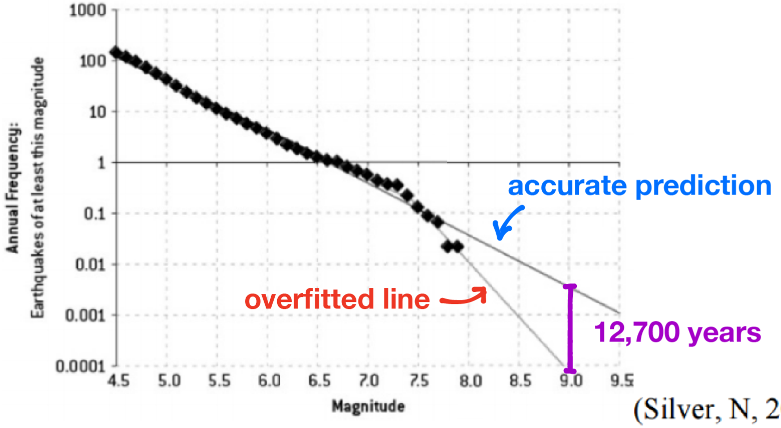
[§] ones that are so weak that you can't even feel them

[¶] Source: Machine Learning Crash Course: Part 4 - The Bias-Variance Dilemma

Danger of Overfitting Example: The Fukushima Power Plant Disaster II







accurate prediction

overfitted line

12,700 years

(Silver, N, 2012)

- The diamonds represent actual data while the thin line shows the model (regression). Model hugs the data points very closely. In fact, their model makes a kink at around a magnitude of 7.3.
- A linear model does not fit the data as closely (but in this case is theoretically more justified).
- When extrapolating† errors have an ever increasing effect.
- The 2011 earthquake that devastated the plant was of magnitude 9.

†Predicting outside initial data range is extrapolating otherwise it is interpolating.

Bias and Variance

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

Bias

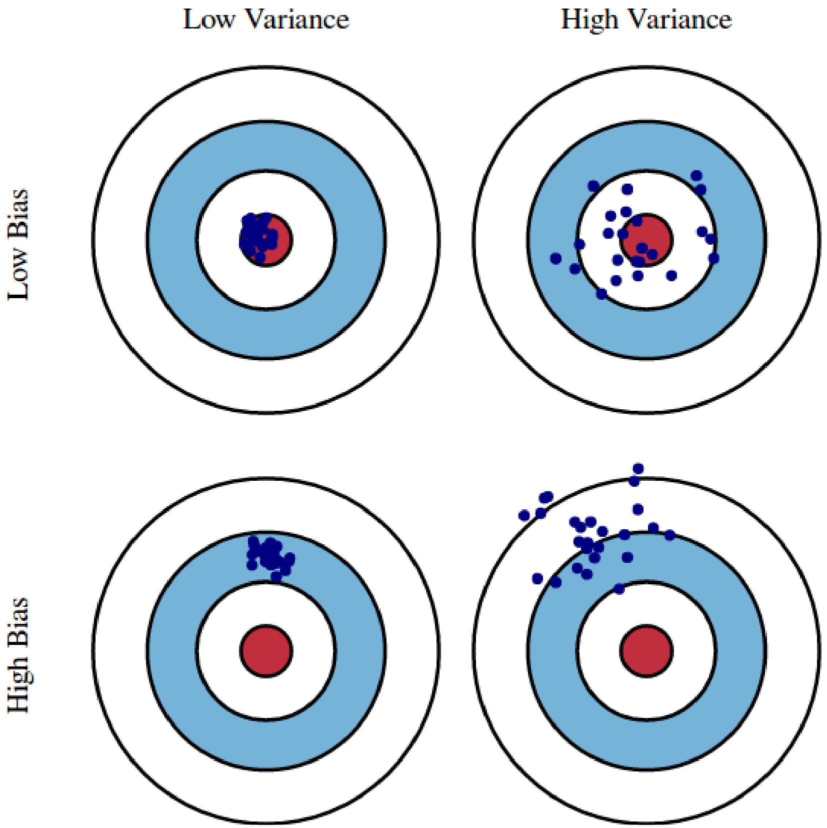
The error due to the difference between the expected (or average) prediction of our model and the correct value which we are trying to predict, over different realisations of the model.

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

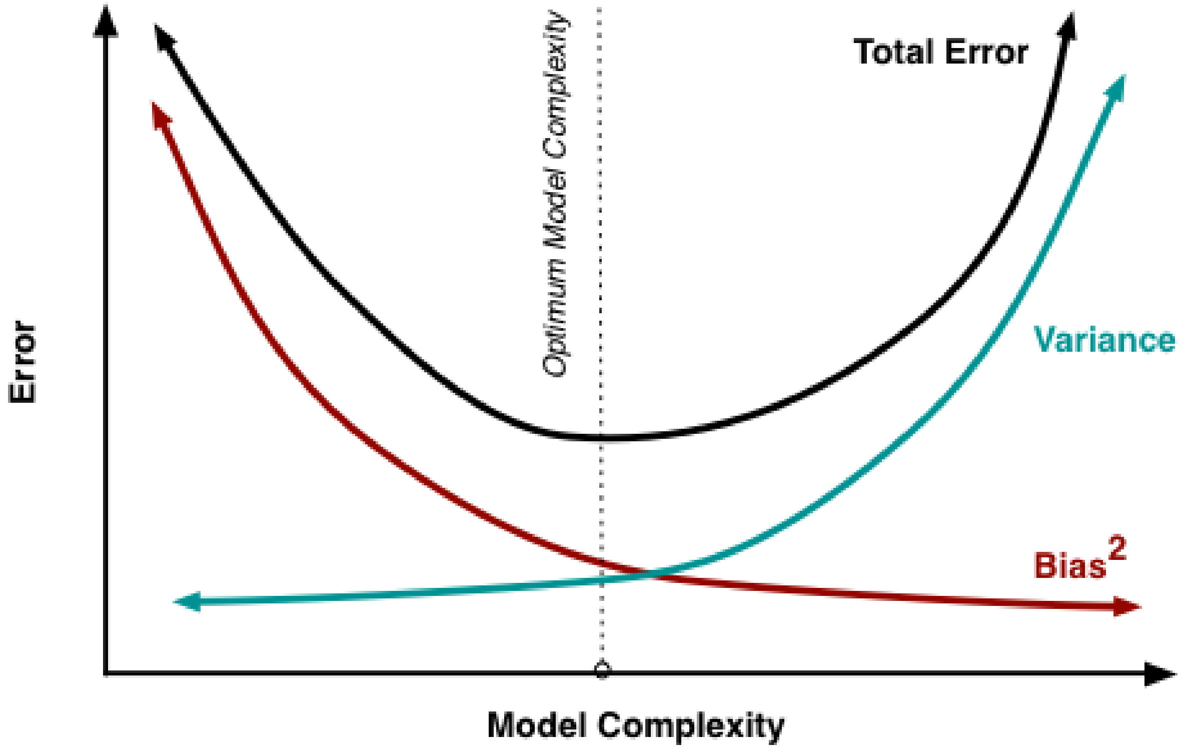


—-Scott Foremann-Roe, June 2012

- 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

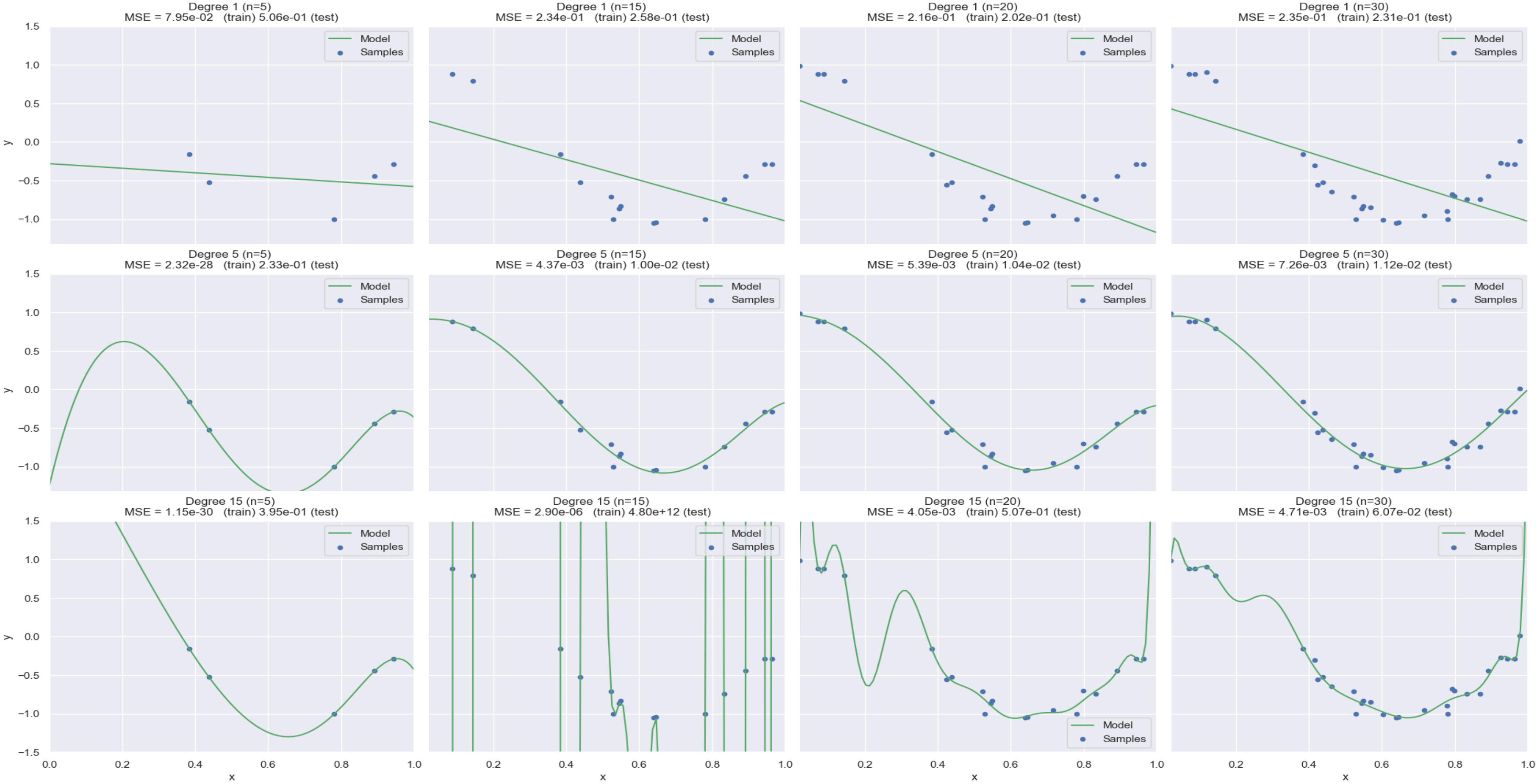
$$\boxed{\text{Error} = \text{Bias}^2 + \text{Variance} + \text{Irreducible Error}}$$



$$\underbrace{\qquad\qquad\qquad\qquad}_{\text{under-fitting}} \qquad \underbrace{\qquad\qquad\qquad\qquad}_{\text{over-fitting}}$$

Effect of Number of Observations and Over/Under-fitting

Fitting model to $n = 5$, $n = 15$, $n = 20$ and $n = 30$ observations . . .

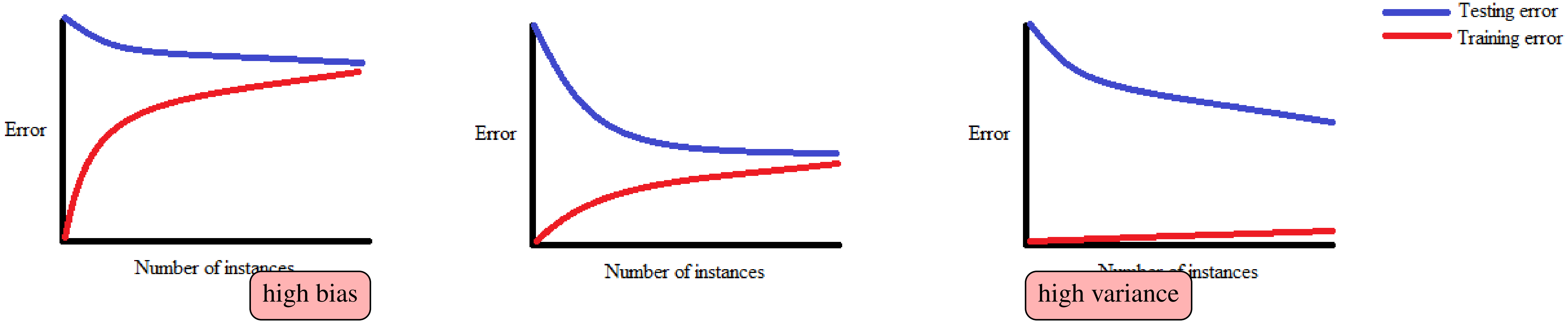


- When under-fitting (high-bias) the model is not sensitive (can’t represent) to changes in the data.
- When over-fitting (high-variance) the model is too sensitive to changes in the data.

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

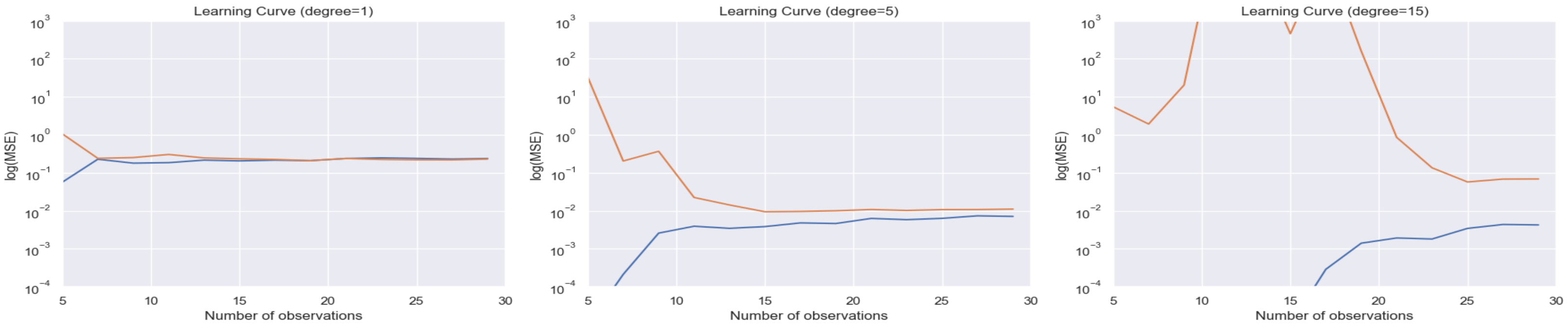


high bias

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 obsdrvations.

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 (see Data Mining 1).
  - Test for normality, uniform variance, . . .
- **practical** — separate the data into train and test sets and evaluate on test (or validation set). (this module).
  - **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 previous 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:

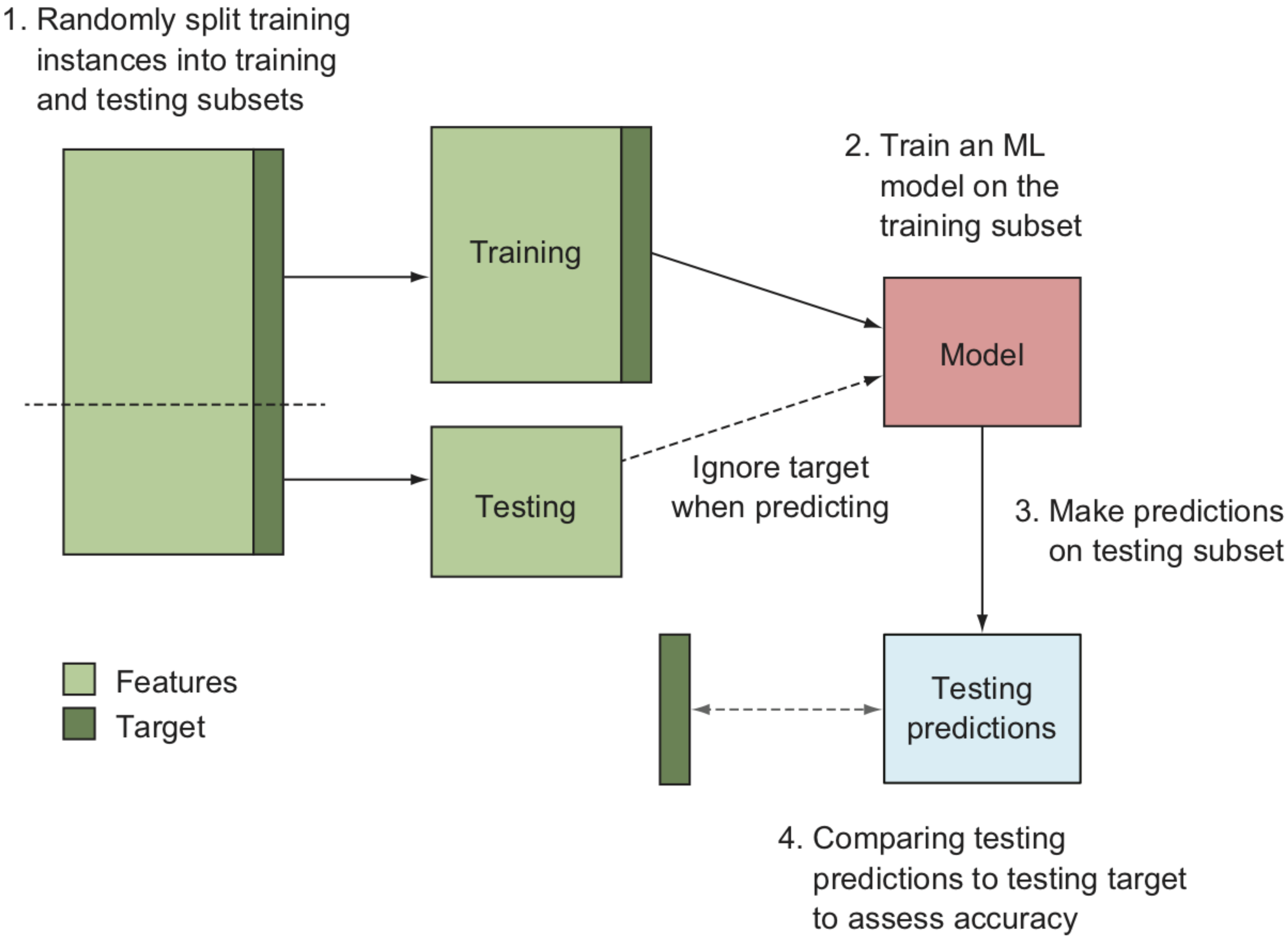
Holdout Method

- 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.

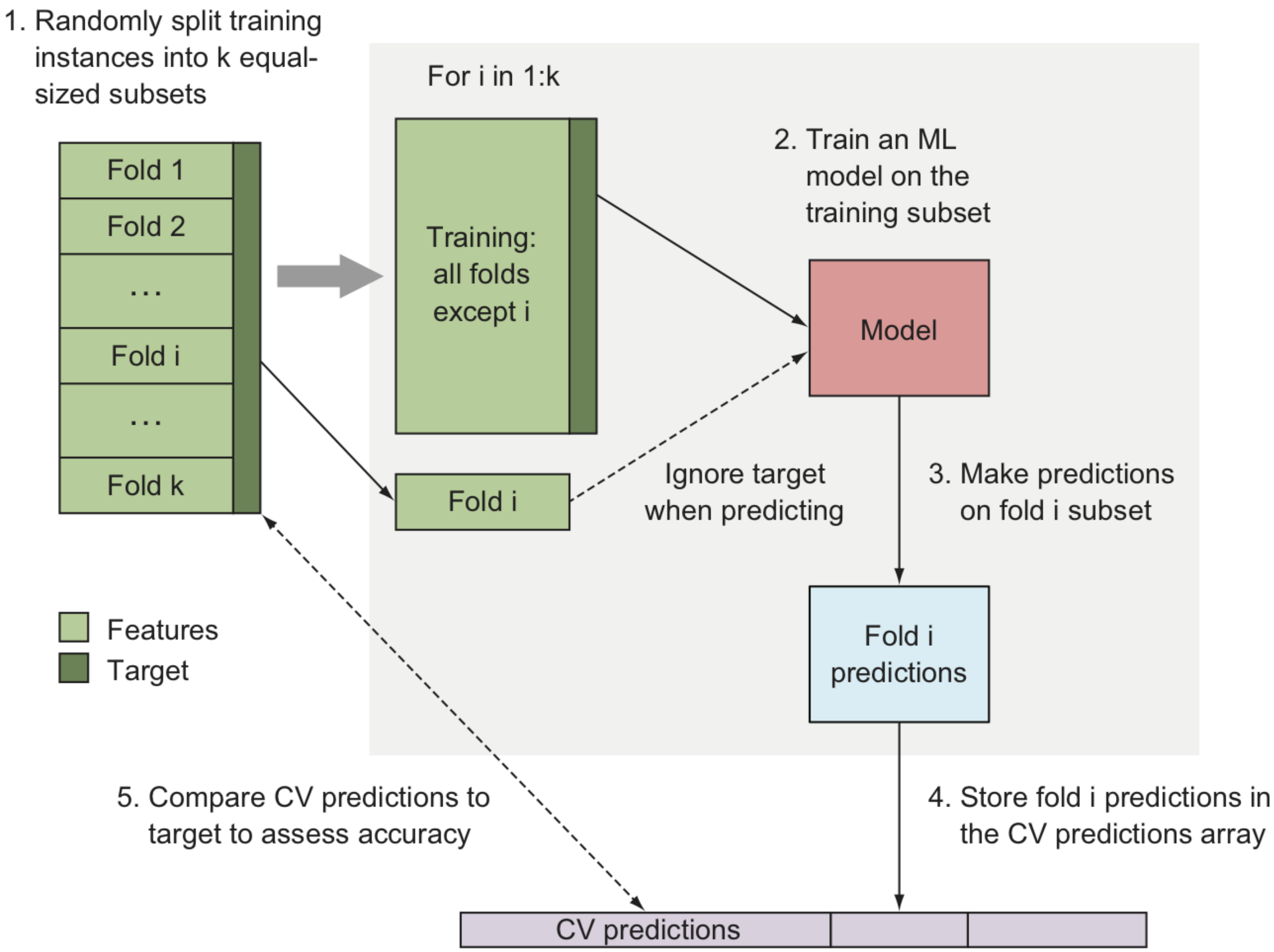
K-Fold Cross Validation

- 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


1. Randomly split training instances into training and testing subsets
Training
Testing
2. Train an ML model on the training subset
Model
Ignore target when predicting
3. Make predictions on testing subset
Testing predictions
Features
Target
4. Comparing testing predictions to testing target to assess accuracy

K-Fold Cross Validation


1. Randomly split training instances into k equal-sized subsets
For i in 1:k
Fold 1
Fold 2
. . .
Fold i
. . .
Fold k
Training: all folds except i
2. Train an ML model on the training subset
Model
Fold i
Ignore target when predicting
3. Make predictions on fold i subset
Fold i predictions
Features
Target
5. Compare CV predictions to target to assess accuracy
4. Store fold i predictions in the CV predictions array
CV predictions

Regularisation

Regularisation refers to a broad range of techniques for artificially forcing models to be simpler.

- The method will depend on the type of model (learner). For example, you could prune a decision tree, use dropout on a neural network, or add a penalty parameter to the cost function in regression.
- Often times, the regularisation method is a hyper-parameter as well, which means it can be tuned through cross-validation.

Returning to our polynomial fitting problem, rather than having a cost function only based on the error (MSE) we could have

$$J = MSE + \lambda \left[(1-\alpha) \sum_a |\theta_a|^2 + \alpha \sum_a |\theta_a| \right]$$

where

- The meta-parameter $0 < \lambda$ determines to relative importance of the regularisation term.
- The meta-parameter $0 \leq \alpha \leq 1$ determines the weighted average between the $L_2 = \sum |\theta_a|^2$ and the $L_1 = \sum |\theta_a|$ regularisation terms.