

SCC.460 Data Science Fundamentals

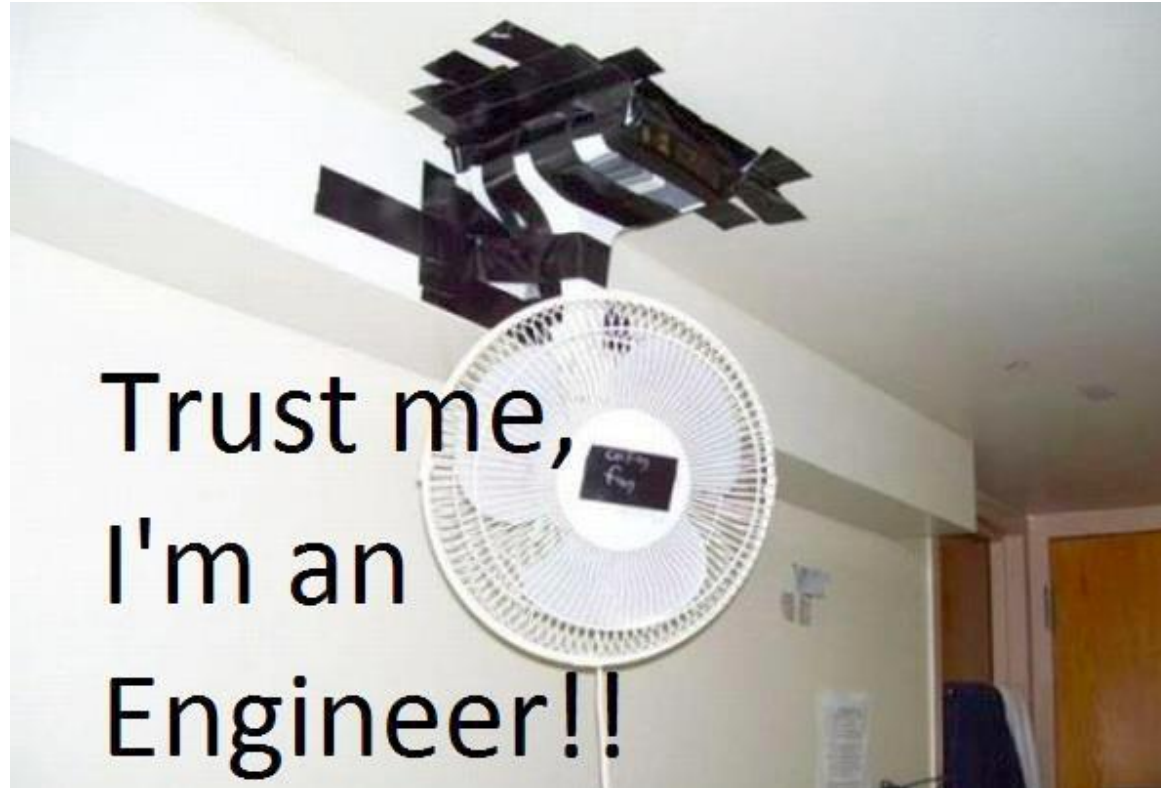
Modelling and Experimental Design

Slides originally developed by Dr Yehia El Khatib and edited by Dr Ioannis Chatzigeorgiou.
Lecture delivered by Dr Ignatius Ezeani.

Today's learning outcomes

- ❑ Feature engineering
 - What are features?
 - Different types
- ❑ Model fitting
 - Model quality
 - Over- and underfitting
- ❑ Experimental design
 - Different experimental setups
 - Splitting data

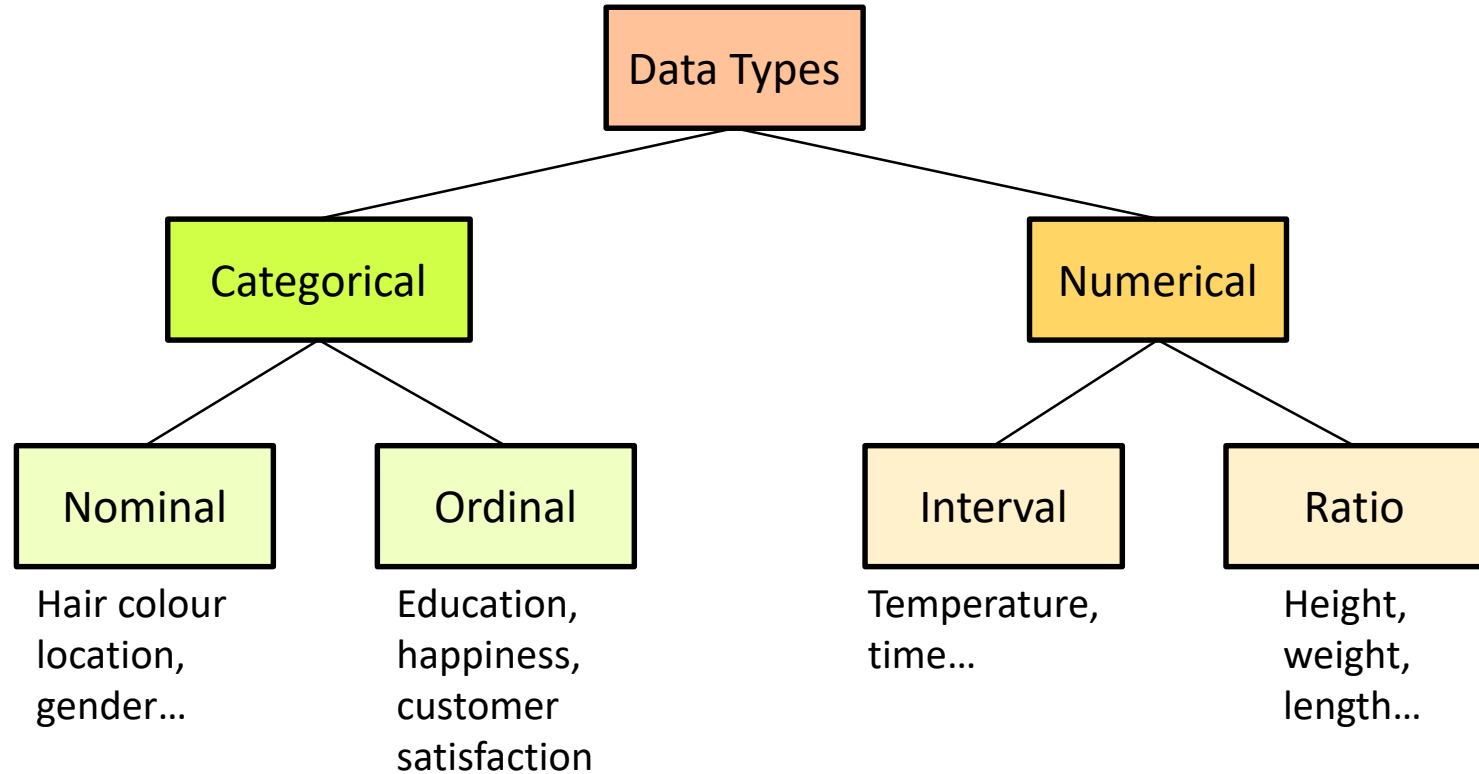
Feature engineering



What are features?

- A dataset consists of:
 - Features/attributes: qualities that we are measuring
 - Feature value: actual measurement of the feature
 - Instance: a record made up of one of more features (e.g. person, session)
 - Normally represented as a vector
- Features can be of different types:
 - **Categorical:** Characteristics that have no mathematical meaning
 - *Nominal*: qualitative values, cannot be ordered (special case: dichotomous)
 - *Ordinal*: qualitative values, can be ordered
 - **Numerical:** quality measured using a set of numbers (real, integer)
 - *Interval*: Ordered, same distance, no true zero, ratios have no meaning
 - *Ratio*: Ordered, same distance, true zero exists, ratios have meaning

Types of data



Feature engineering

- Machine learning and statistical models require numerical features.
 - i.e. for inducing a hyperplane through an n -dimensional space
- Hence, we need to convert nominal and ordinal features to numerical values.
- In many (**not all**) cases, it is possible to add qualitative information to models as quantitative scales or factors.
- Various coding schemes exist to achieve this (one hot, hash, count, target, polynomial ...).
 - The choice is domain-specific.
- Feature engineering could be one of the *most time consuming* parts of your job.

Feature engineering: nominal → 'numerical'

- Create a variable for each category (one hot encoding, here one of 2):

PersonID	Hair Colour
123	Blonde
456	Black



PersonID	BlondeHair	BlackHair
123	1	0
456	0	1

- This is how a binary 'bag of words' model works.
 - Equivalent to an order-less representation of a document (e.g. word clouds)

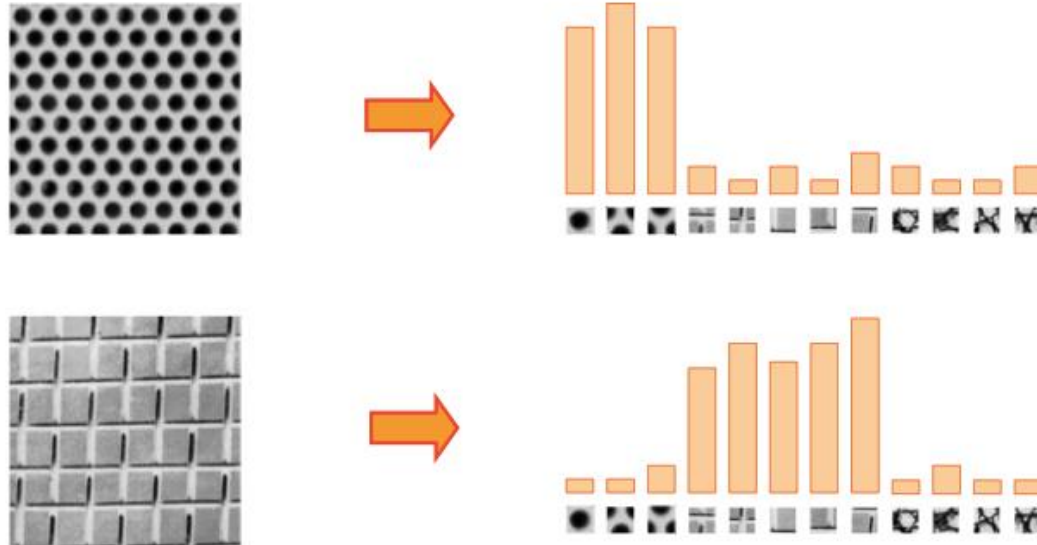
mi casa es su casa



mi	casa	es	su
1	2	1	1

Feature engineering: bag-of-features models

- A form of capturing feature identity and frequency, not interrelationships.
- Bag-of-words is a special case of the bag-of-features.
- Popular example: Texture recognition



Feature engineering: dichotomous → 'numerical'

- Similar to converting nominal to numeric: we simply change the feature's value range to be $\{0,1\}$

PersonID	Smokes	...
123	Yes	
456	No	



PersonID	Smokes	...
123	1	
456	0	

- The only difference here is that we need to decide on the coding scheme.
 - i.e. what constitutes a 1, and what is a 0

Feature engineering: ordinal → 'numerical'

- Choose a suitable coding scheme to change ordered discrete (categorical) values to numbers.

PersonID	Grade
123	B+
456	A

Ordinal	Numeric
A+	10
A	9.5
A-	9
B+	8.5



PersonID	GradeValue
123	8.5
456	9.5

- Important to choose a coding scheme that preserves the magnitude of differences (the ordering).

Model fitting



Model fitting


- Once the dataset is prepared with engineered features, we can fit a model to the data (hopefully).
 - e.g. *linear regression*, where we aim to find the best linear function $y=f(x)$ (i.e. line) to explain the data.
- Goals of applying a model are two-fold:
 1. To predict/forecast labels/outcomes for unseen data
 2. To analyse a model's diagnostics to understand the contribution of certain constructs
 - i.e. in terms of individual features' contributions and importance

General model types

1. Non-parametric Models

- Input: training dataset
- Output: model fitted to the dataset
 - i.e. induced parameter vector
- Examples: Naive Bayes, SVM, Perceptron, linear regression

2. Parametric Models

- Input: training dataset and hyperparameters (θ)
 - Output: model fitted to the dataset, θ -indexed
 - i.e. induced parameter vector specific to θ
 - Examples: regularised linear models, singular value decomposition
- 

Examples of linear regression

- Let $x_{i,1}, x_{i,2}, \dots, x_{i,p}$ be the explanatory / predictor / independent variables.
- Let $b_0, b_1, b_2, \dots, b_p$ be the model / regression coefficients.
- Let y_i and ε_i be the dependent variable and the error variable, respectively.

Linear regression (for $i=1, \dots, n$):

$$y_i = b_0 + b_1 x_{i,1} + b_2 x_{i,2} + \dots + b_p x_{i,p} + \varepsilon_i$$

$$\Leftrightarrow \varepsilon_i = y_i - b_0 - b_1 x_{i,1} - b_2 x_{i,2} - \dots - b_p x_{i,p}$$

Objective of linear regression
(non-parametric):

$$\min_{b_0, \dots, b_p} \sum_{i=1}^n \varepsilon_i^2$$

Objective of **regularised** linear regression
(parametric):

$$\min_{b_0, \dots, b_p} \left(\sum_{i=1}^n \varepsilon_i^2 + \theta \sum_{j=0}^p g(b_j) \right)$$

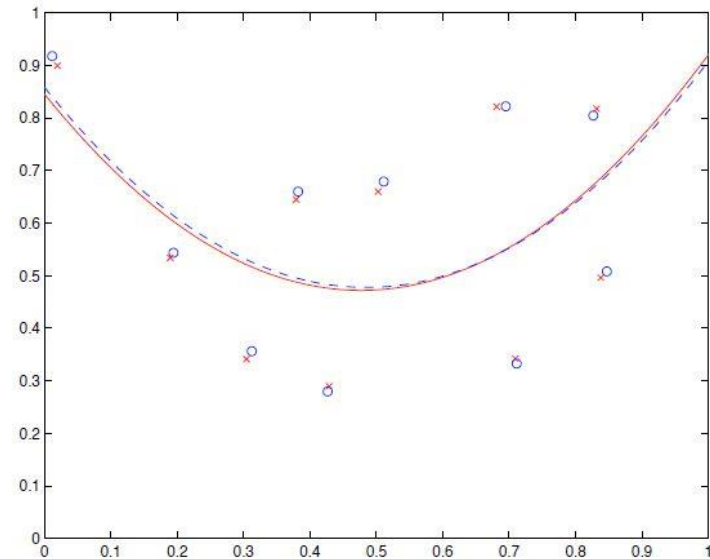
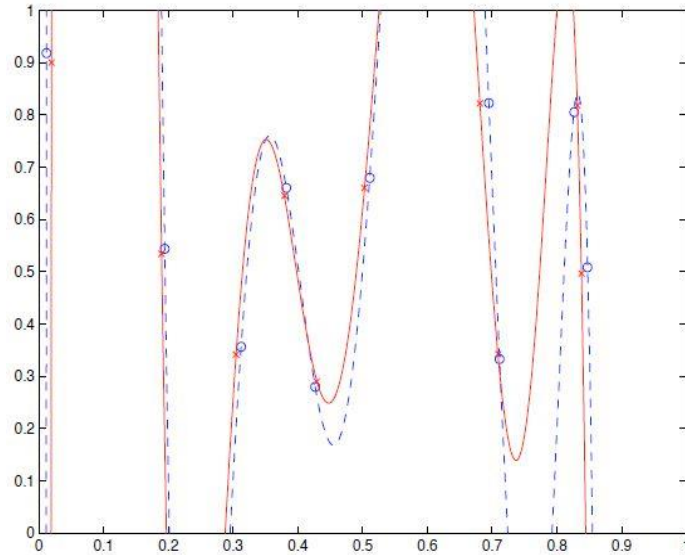
Model quality

- Almost every model optimizes some quality criterion:
 - Linear regression: Residual Sum-of-Squares
 - k-Means: Inertia = mean squared distance from each sample to cluster center
- The quality criterion is chosen to prove quality of fit (convexity) and provability (convergence).
- Other quality criteria:
 - Stability of the model (sensitivity to small changes)
 - Compactness (sparseness or many zero coefficients)
 - Silhouette score
 - Inter-cluster similarity
 - Intra-cluster entropy

} *regression models*

} *clustering models*

Example of stability



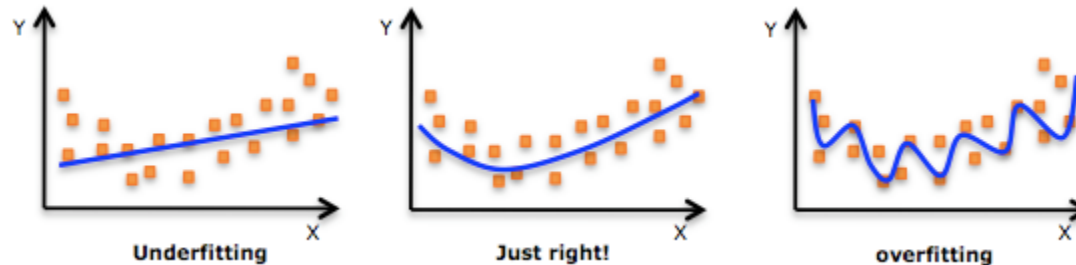
Overfitting may generate a smooth curve (red) but a small perturbation (change in the values of the points, from red to blue) will significantly affect the solution (blue). In the right-hand side figure, the solution varies only by a small amount.

Fitting a model

- Two main methods for fitting, depending (non-/parametric) model type.
 1. Maximising Model Fit to Data
 - Choose model parameters that maximise in-sample best fit criterion
 - i.e. *using just the training set*
 - Coefficient of determination (R^2), Akaike Information Criterion (AIC), Bayesian Information Criterion (BIC)
 2. Minimising Prediction Error
 - Learn from errors in prediction and update parameters accordingly
 - Use out-of-sample error calculation
 - i.e. learn over the training set, calibrate over the validation set
 - Derive the mean error and standard deviation to allow selection of best θ

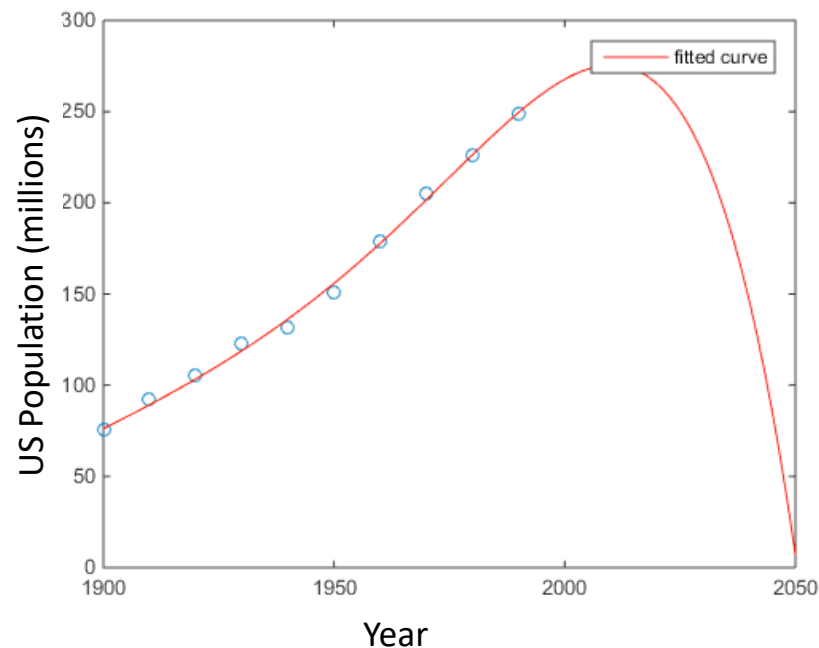
Overfitting

- Model fitting can be prone to overfitting, where our in-sample fit leads to poor generalisation.
- Ideally, the model would fit an infinite sample of features.
 - i.e. extend to represent out-of-sample predictions



- Higher order polynomial models will better fit the feature values.
- But it models the noise. Not good for inferring the trend (the signal).

Overfitting



Without looking out-of-sample you could not be sure of the model.

Controlling for overfitting

1. Regularisation

- Penalise models for excessive complexity using secondary criteria next to the main quality criterion being optimised.
 - L_1 regularization adds the sum of absolute value of model coefficients.
 - L_2 regularization adds the sum of squares of model coefficients.

2. Data splitting for extrapolation

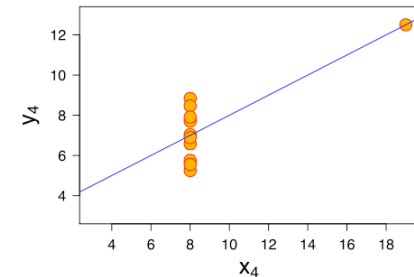
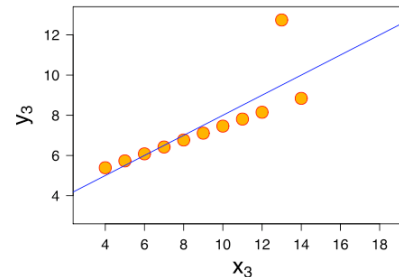
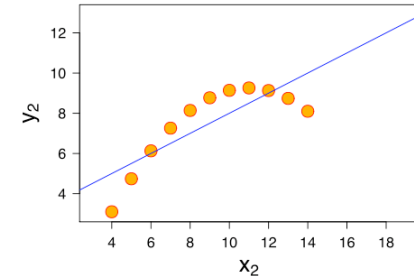
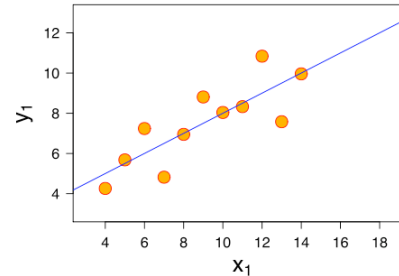
- Training: to estimate a model
- Validation: to choose and calibrate a model
- Testing: to evaluate the model

3. Fine tune through cross-validation

- Covered in extrapolation under Experimental Design (last part of the slides)

Underfitting

- The model will learn what you teach it.
 - e.g. you can *always* fit a linear model to a dataset, but do you actually know if there is a real linear relationship? (bias)
- Easily detected using standard performance tests.



Experimental design



Experimental studies

- Recall: an experimental study focuses on investigating cause-effect relationships.
 - Often done through a hypothesis.
 - Think of treatment experiments: alter treatment, examine impact.
- Experimental design ensures internal validity:
 - i.e. truth of cause-effect inference
 - e.g. teaching on Friday afternoon enables retention

Experimental design

- In its simplest form:
 1. Create two groups (e.g. of people) that are as similar as possible.
 2. Apply the treatment to one group, do not apply the treatment to another group (control).
 3. Observe the difference in the outcomes.
- Is this applicable in Data Science?

Experimental design

- Existing experiment design approaches assume ability to acquire observational data.
 - i.e. run an experiment where we apply a treatment, and observe the effect
- Often we don't have this:
 - Data is provided by other departments
 - External datasets are mined from the Web
 - Open data is used as a ground truth
- Often we are working with found data.

Experimental design

- A common approach to a data science task is:
 1. Engineer features from a dataset
 2. Induce a model from the data
 3. Apply it to new data to see how well it performs
 4. If it's a good model, inspect which features were important. If not, go to 2.
- Key components to this:
 1. We fit the model using data
 2. We apply the model to new data
- Experimental design plays a crucial role here...

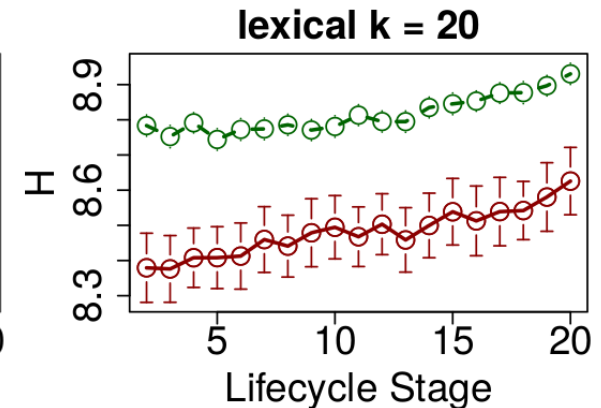
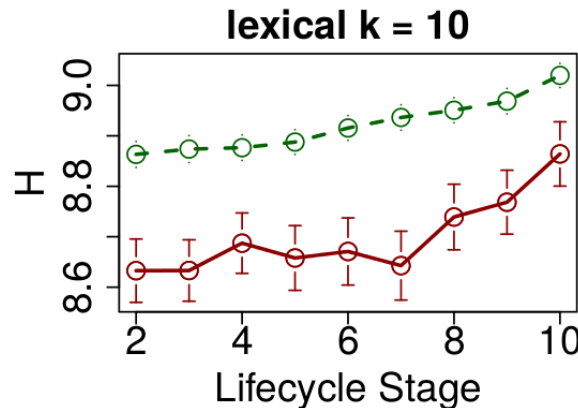
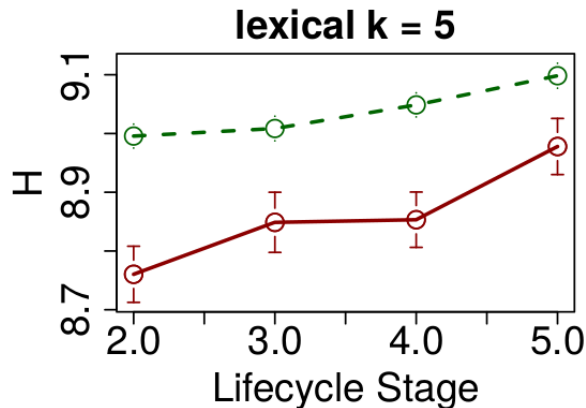
Experimental design: example*

- **Goal:** predict churners and non-churners
- **Motivation:** The identification of users who will drop out of a service is known as *churn detection*. It is important for telecoms, online games, social networks, etc. so that they propose countermeasures to reduce losses.
- **Data:** found online (four online community platforms)
- **Process:**
 1. Engineer features to capture user behaviour (social interactions – to other users and by other users; language – referred to as *lexical* information)
 2. Examine how churners and non-churners differ with varying lifecycle **fidelity**. The fidelity value determines the number of stages/periods that a given set of posts will be divided into (it is denoted by k in the following figures).

* Matthew Rowe, “Mining User Development Signals for Online Community Churner Detection”, *ACM Trans. Knowledge Discovery from Data*, 2015

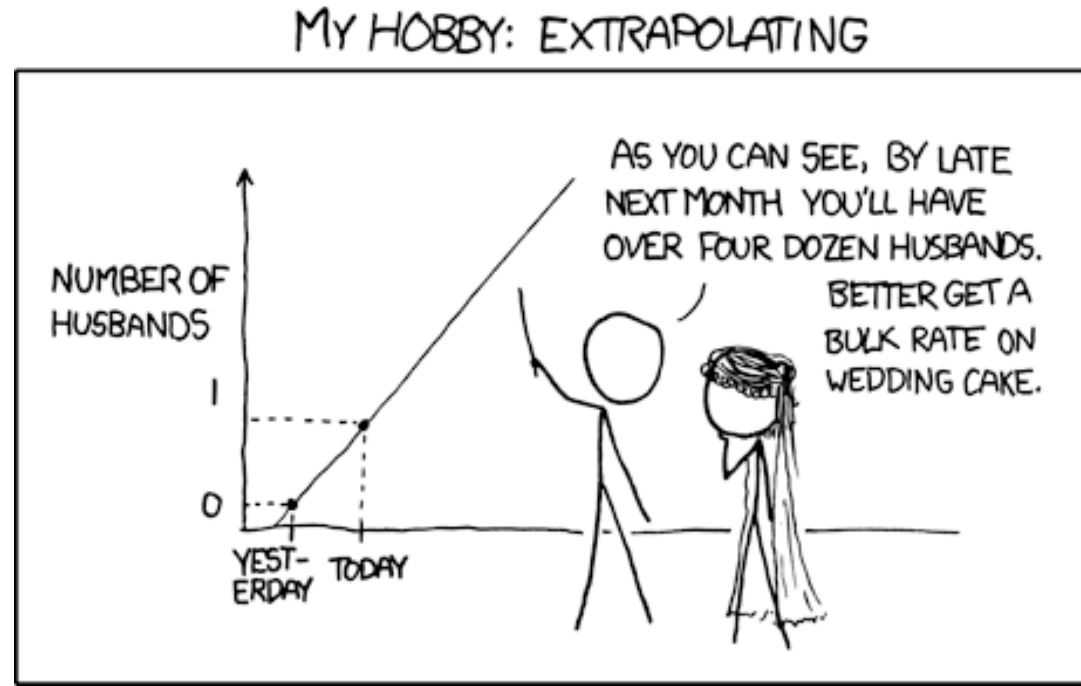
Experimental design: example (continued)

- **Lifecycle** is the period between a user's "birth" (earliest post within a set of posts) and "death" (final post within the same set of posts).
- The vertical axis in each graph represents the **entropy** (H), which describes the amount of variation within a random variable (i.e. it gauges how much a user is varying, including connections and terms used in posts).



Experimental design: extrapolation

- In order to test the 'generalisation' capability of the model, we need to extrapolate to new (**unseen**) data.
- Methods for **extrapolation**:
 1. Retrospective
 2. Forecasting

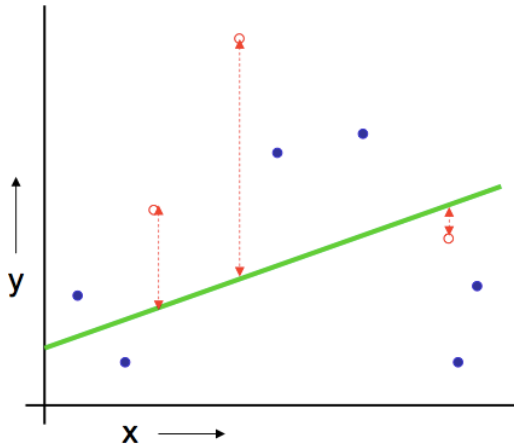


Experimental design: Retrospective extrapolation

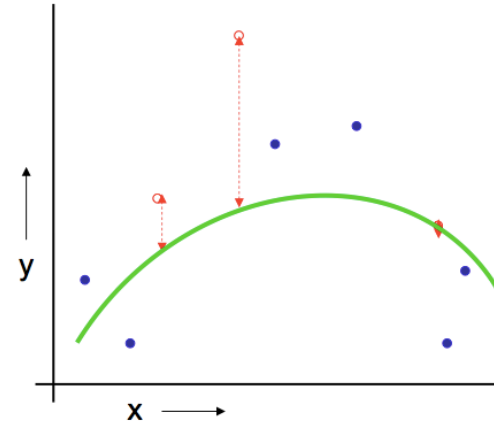
- Retrospective data analysis (hold out method)
 1. Gather a dataset.
 2. Use one portion to induce (train) the model
 - This is used for analysing features, tuning model parameters, etc.
 3. Hold out another portion for testing the model
 - Evaluate how well the induced model does on this data

Retrospective extrapolation: Regression example

- Perform the regression on the training set.
- Estimate future performance with test data.



Mean Squared Error = 2.4



Mean Squared Error = 0.9

Retrospective extrapolation: Splitting

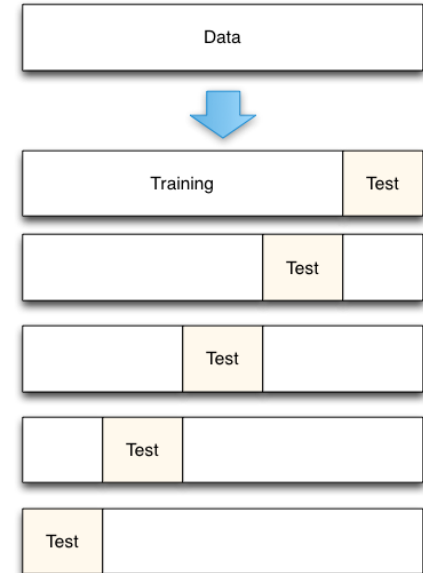
- Common splitting procedure for retrospective analysis:
 1. Segment dataset into a training and test sets (and validation for parametric models)
 - Typically based on 80%/20% or 70%/30% random split
 2. Induce the model on the training set
 3. Apply the model to the test set
- If each instance in the dataset is time-sensitive: then we use the first 80% as training, the rest as test.
- Be cautious not to introduce bias by the way you split the data.

Experimental design: Forecasting

- Forecasting:
 1. Gather a dataset up to the present
 2. Induce the model over this dataset
 3. Apply the model to data as it becomes available
 4. Continuously re-calibrate model based on its accuracy

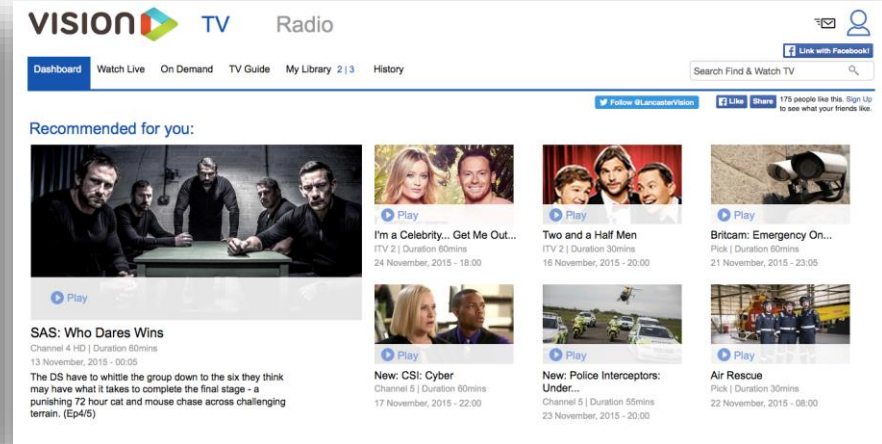
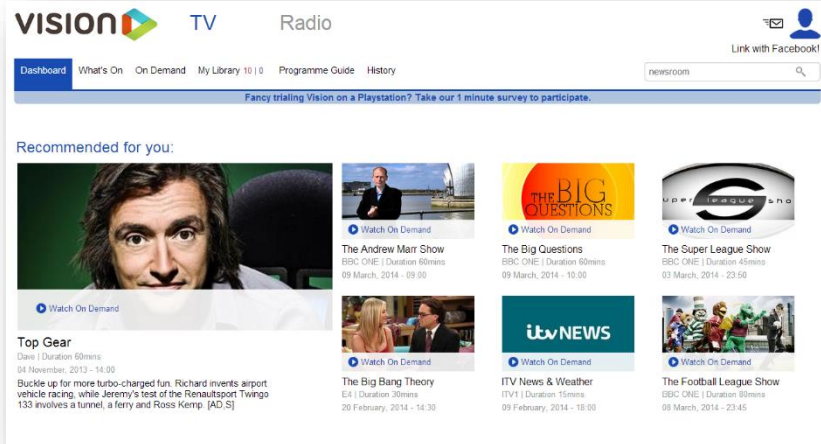
Experimental design: Repetition

- Repeating experiments allows us to see how much a model varies:
 - Large variance = poor generalisation capability
 - Small variance = Less deviation from future observations
- Repetition is 'hardcoded' into k-fold Cross-Validation:
 - For each i in $\{1, \dots, k\}$ do:
 - Train a model on all the other folds
 $A_1, \dots, A_{i-1}, A_{i+1}, \dots, A_k$
 - Test the model on A_i
 - Returns k performance measures
 - Measure the standard deviation in error
 - All data points are (eventually) used for both training and testing.



Experimental design: Randomised Controlled Trials

- Sometimes you would have an experimental setting, which allows you to examine the efficacy of an intervention.
- Randomised Controlled Trials (RCTs) enable this.
 - Often referred to as *A/B Testing* in Data (and Computer) Science



Experimental design: Randomised Controlled Trials

- **“Randomised”**: Avoids selection bias and maximizes statistical power (i.e., the power of a hypothesis).
- **“Controlled”**: Implies eligibility criteria, hypotheses, methods for enrollment and follow-up, rigorous monitoring, analysis plans and stopping rules.
- RCTs provide the “gold standard” for proof-of-concept and are widely used in medical trials.
- Key phases:
 - Eligibility and Enrollment
 - Randomisation and Blinding
 - Follow-up
 - Analysis, Monitoring and Stopping Rules

Today's learning outcomes

- ☑ Feature engineering
 - What are features?
 - Different types
- ☑ Model fitting
 - Model quality
 - Over- and underfitting
- ☑ Experimental design
 - Different experimental setups
 - Splitting data

Further reading

- N. Silver, “The Signal and the Noise: Why So Many Predictions Fail-but Some Don't”
- W.M.K. Trochim, “Experimental Design” in “The Research Methods Knowledge Base”.
- H.R. Varian, “Big Data: New Tricks for Econometrics”, *Journal of Economic Perspectives*, 28(2):3-28, 2014.
- S.L. Scott, “Multi-armed bandit experiments in the online service economy”.