Empirical models - understanding and validation

Author **Jan-Willem Bikker, Bert Schriever, Matthijs Tijink**

Date **2023, December 7**

Reference **version 0.9**

The ASIMOV project

This report is written as part of the ASIMOV project (https://www.asimov-project.eu/). The ASIMOV project (AI training using Simulated Instruments for Machine Optimization and Verification) is a collaborative research project running under the ITEA4 programme within the Eureka framework. It was proposed under the Eureka AI call 2020.

This report focuses on empirical models: models that are fit using data obtained in “the real world”, as in supervised machine learning and statistics. Such models are examples of *digital models,* which is part of the sequence digital models, digital shadows and digital twins, where distinctions are made in the relation between digital and physical twin. The concept of digital twins is a wider topic than these empirical models, see e.g. [1]. Sometimes in building a digital model, a digital model or submodel is fit from example data and can be called an empirical model. Such models are often encountered in CQM’s projects.

The first topic addressed in this report is validation of the quality of the model’s predictions. A common method is to split the data at hand into a training set and a test set. However, often data is scarce, and other approaches such as k-fold cross validation are taken. This report investigates methods to obtain confidence intervals for the prediction quality obtained in this way: instead of reporting a single number such as “97% accuracy”, the confidence interval gives lower and upper bounds for that 97%.

The second topic is about understanding - “quantifying effects” of predictor x on outcome y in some model $y = f(x)$. This is commonplace in models used in statistics, such as linear regression, where the effects coincide with the regression coefficients. In projects, cleaning the input data for a statistical model can be a lot of work. Prediction models from machine learning can be more forgiving on how clean the input is. The report describes the investigation into obtaining confidence intervals for the effects. Confidence intervals for effects are not commonly reported in machine learning models.

Content

1. Introduction

1.1 Background

For more than 40 years, CQM has been performing projects that have quantitative elements. In those projects, the CQM consultants collaborate with the customer to solve some problem or achieve an improvement. Modelling and optimization often play a role, and CQM's projects can take on the form of early exploration with workshops, coaching and training, to developing models for improvement and problem solving.

CQM's consultants have quantitative backgrounds from various disciplines: statistics, operations research, business economics, econometrics, social sciences, and computer science. This means that different viewpoints on modelling co-exist within CQM. This variety of modelling also follows from a wide range of focus area; supply chain analytics, warehousing, research and development, data science, production planning.

The recent article [2], titled "Analytical problem solving based on Causal, Correlational and Deductive models", gives an insightful taxonomy that applies to much of CQM's modelling. The article states that many problems can be framed using some model $Y = f(X)$, from forecasting, decision making, to optimizing in planning and industry. It considers the construction of the models and the steps afterwards in using them to address the problem at hand. The **deductive models** $Y = f(X)$ are based primarily on first principles and there are many examples in logistics and planning. For instance, in scheduling tasks or resources, the structure of a model is shaped by the modeller in terms of costs, durations, capacities, constraints on those, etc. Some projects involve *simulation* of a (logistic) process; here the model takes on a stochastic nature (random demand and supply) where the randomness is governed by the parameters of probability distributions. At CQM, the term "deductive models" is not used in practice; instead terms like simulation, digital twins, planning, scheduling etc are used; these projects are strongly linked to the scientific discipline of Operations Research. One other class of deductive models are physics-based models sometimes used in industry.

For the deductive models, the numerical values for model parameters in the above example typically have a concrete interpretations and are determined in a separate activity, sometimes from domain knowledge, or derived from historical data. However, this does not directly affect the *structure* of the model. The structure of the model can be quite complex involving many variables (e.g. each edge of a large logistic network) and is usually written down as a "paper model".

Data-driven models are the causal and correlational models mentioned above and commonplace in statistics and machine learning ([2]). In contrast to the deductive models, the relationship $Y = f(X)$ is largely determined by the available data. A typical example is linear regression. The terms in the model, e.g. whether the x^2 term included or not in $y = c_0 + c_1 x + c_2 x^2$ can be informed from domain knowledge. The example data y, x are used to estimate model parameters c_0, c_1, c_2. They are also typically estimated *jointly* – there is often not a direct route from data to a single parameter. In practice, the model structure is altered in an iterative way after considering the results of estimating the model from data. This way of developing the model and thinking about it in data-driven model is different from that of deductive models.

The focus of this report is on data-driven models.

1.2 Examples of data-driven models

- Prediction and forecasting; forecast demand for energy or goods based recent events, weather forcasts; predicting product quality from inputs that are available earlier. The article [2] in table 3 mentions for *correlational models* problems as types of problems:
    - decision optimization (e.g. decide when to perform maintenance to a system, based on performance forecast)
    - Predictive control (reduce variation in y by an intervention on some controllable variables x, based on forecasts of y from controllable and uncontrollable x)
- Understanding and quantifying effects; here domain knowledge should have a strong voice in the mathematical formulation of the model. The example data may be observational or the result from a targeted experiment. The business questions are typically related to the effects of predictors (inputs) $x_1, \dots, x_p$ on the response (output) on y. With limited amounts of data, statistics give assess the uncertainty of those effects using concepts such as statistical significance, p-values, confidence intervals. Examples are finding the most important factors among the x_i in a manufacturing process, comparing prototype vs reference, or indeed clinical trials where the business question is about stating the amount of evidence for an effect. The article [2] in table 3 mentions for *correlational models* problems as types of problems:
    - Response-surface optimization (find good setting of x to maximize/minimize y)
    - Robust design (find x so that future variation in x causes as little variation in y)
    - Tolerance design (impact of future variation in x on variation in y; possibly reduce variation in x)
    - Decision optimization (as in the prediction and forecasting above)
    - Predictive control (as in the prediction and forecasting above)
- Surrogate models / Compact models, on top of a detailed detailed (simulation) model of a physical system is available but which is slow to evaluate. Using a careful scheme of runs ("Design of Computer Experiments"), the slow model is run in batch mode to generate data. Then models are created based on those as a fast surrogate model or compact model. Note that the slow model may itself be considered a deductive model, a prime example is if such a model is based on physical laws and this model can be considered a causal model.
- Complex inputs /deep learning, where the inputs are large and complex such as images. The output may be classification of objects or defect detection, along with indicating possible locations. Techniques from the world of AI such as deep convolutional neural networks, and GAN (generative adversarial networks) are employed.
- Finding structure: suppose many variables are given such as concentrations of substances, or many questions on a questionnaire; then all variables can be considered on equal footing (as opposed to roles of input and output). Questions could be to find clusters of data or describe typical relations between the variables. This is called unsupervised learning.

This report focusses the first two points mentioned above (predictions, understanding).

1.3 Model – detailed setting

We consider data-driven models, where output is related to input for prediction. Using the fundamental setup from statistics, the dataset at hand is a *sample* from an imaginary *population*; e.g. in a clinical trial the population can be all patients with certain properties, when studying a factory production line it can be all produced items so far or in the future. Our dataset is seen as a *sample* from the *population* and used for analysis and model building, but the ultimate interest lies in properties of the population.

For instance, a medical treatment is intended not for the small group in our study but for all such patients. Beware; in machine learning lingo, “sample” often refers to a single observation.

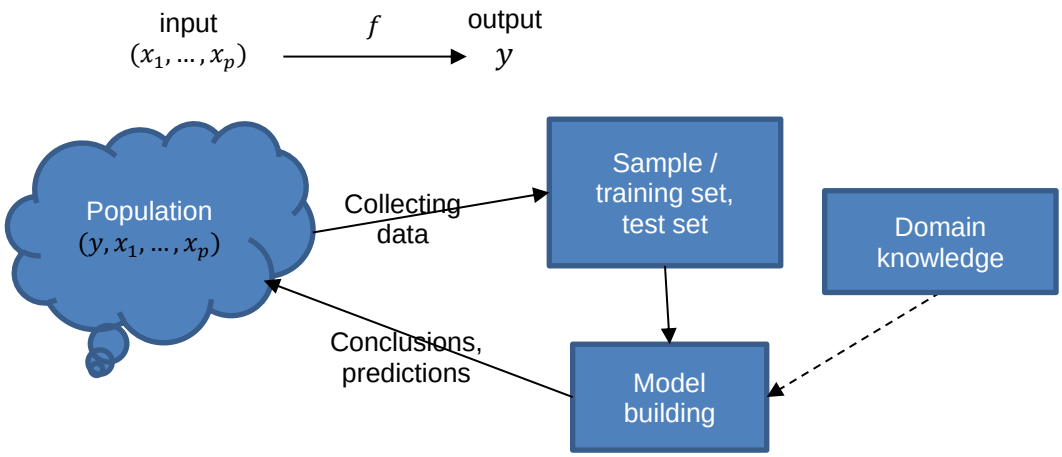


Figure 1: population and sample

1.3.1 Some notation

Further setup,

- we have example / training data $(x_1, .., x_p, y)$, and the inputs / predictors have been pre-processed as nicely structured vectors of length p.
- We have the statistical setup of a *population* of cases, and consider the training set or data as a *sample* from the population; the goal is to have a model f from which we will make statements about the population. In machine learning this distinction between population and trainingset is not always explicit.[1]
- The training data may be from observational data or targeted experiments, e.g. a grid of values of $(x_1, \dots, x_p)$.
- The predictors x_i may be continuous, categorical, or some mix.
- The y variable may be a continuous number, or a binary outcome, or categorical.
    - Much of this report also applies to other structures of y, e.g. a vector, or count outcomes. Some models naturally handle such a vector; otherwise each component of a vector y may be modeled separately.
- There is some inherent stochastic error, i.e., it is inherently impossible to predict perfectly $y = f(x_1, \dots, x_p) + \epsilon$ for continuous y.
    - This sets the situation apart from when the available data comes from a detailed physical model for which we make a surrogate model: there, the errors between surrogate model f and a detailed simulation model value y are structural and deterministic in nature.
    - Section 2.1.2 gives an alternative notation for binary y.

[1] A “sample” in machine learning can mean a single “observation” in statistics; the “sample” in statistics usually refers to the entire “training set”.

- Possibly, the data comes in groups, e.g., multiple measurement within the same machine, or fthe group of measurements after tedious calibration. From social sciences, if data is collected on school children, there can be an elaborate hierarchical grouping: province – city – school – class – student – repeated values over time. In the semiconductor industry this could be machine – major part – lot – wafer. See section 2.1.5.
- A related phenomenon is dependency over time, e.g. for a model of demand over time, if demand is unusually high (given x) on a certain day, it may be unusually high for a range of days, i.e. the errors are positively dependent.
- In real life, there may be x-variables that are important for y but are not in the dataset. This can lead to bias and confounding when interpreting the model. In [2], the first steps of problem solving is finding the relevant x and y variables. In this report, we take the set of x-variables as a given.

1.3.2 Goal of prediction

A major goal of a model may be *prediction*, where the model output should be as close as possible to future outcomes. Prediction is the focus in supervised machine learning.

- **Prediction error / generalization error** is of interest: a metric how well predictions match observations. Note that *generalization* error refers to the error on all new observations from the population, and that prediction errors on the training set tend to be smaller.
- Example metrics for prediction performance for continuous y are RMSE (root mean squared error), MAE (mean absolute error), or R2 (% of variation in y explained by the model). For binary y, examples are accuracy, sensitivity/recall, specificity, AUC (area under curve).
- Possibly, there are explicitly known costs involved of a wrong prediction of Y is problem/no problem. If also the rate of problems in the application is assumed, this leads to a metric that measures the quality in euros. This may be captured in the phrase “prediction performance” as opposed to the narrower “error”.

A topic of this report (chapter 2) is to assess the statistical uncertainty on such metrics, e.g. using a 95% confidence interval. This goal is very natural in statistics, but not always used in the fast-evolving field of machine learning.

1.3.3 Goal of understanding / quantifying effects

Once a model $y = f(x)$ is estimated, this empirical relation can help understand the engineer/developer/scientist the relation between y and the x_i. In Research and Development, engineering this can guide decisions on design of some product or process. Other domains are clinical trials (treatment vs control), or A/B testing in website design. In these last examples, the predictor would simply be binary, indicating the group, and typically, there has been some randomized experiment where group membership was based on random choice. In other settings, data may be observations without explicit experimentation. Other classical setting is George Box’ RSM, response surface methodology.

Some specific questions in this setting can be

- What are the names of the most important predictors?
- What are the top 3 predictors out of 10?
- What are the signs of the predictor effects?
- Is there any statistical evidence of an effect of a predictor (“statistical significance”); this includes treatment effect and clinical evidence.
- For highest Y what direction should the x_i’s move in?
- What is the optimal setting (for highest or lowest Y)?
- If in a future application some x_i will show variation, how much variation in Y will that cause?

Many disciplines teach basic linear regression, $Y = c_0 + c_1 x_1 + \cdots + c_p x_p + \epsilon$. Typically, the regression coefficients are then interpreted "ceteris paribus", i.e., what happens if only one variable changes, and all others are held constant. (In contrast, "mutatis mutandis" is about changing all variables simultaneously to reflect the relations between x, as in correlational models). Such a model implies that a planned change to a predictor x_i to $x_i + d$ will mean an increase in the mean of Y of $c_i d$. In a cost-benefit assessment, the exact value of c_i is of immediate interest. For this reason, the *standard error* or *confidence interval* of c_i is of interest: the error of the estimated vs true (unknown) value of c_i; see Figure 1: population and sample. For instance, if $c_i = 4.5$ with a standard error of 1.5, a 95% confidence interval is approximately $4.5 \pm 2 \cdot 1.5 = [1.5, 7{,}5]$. This means that the increase in Y is expected to be within $1.5 \cdot d$ and $7.5 \cdot d$. If there is a considerable cost in increasing x is involved (e.g., an expensive ingredient or expensive raw material), this wide range in benefit may warrant further investigation before making a decision.

Besides linear regression, there can be many other models in this setting; some names are ANOVA, RSM (response surface models), regression using transformed predictors and/or response, GLM (generalized linear model), all of those combined with random effects, SEM (structural equation models). The statistical tool Stata has a large suite of regression type models that have such regression coefficients-with-standard-error. It can also determine the confidence intervals of general (non-linear) functions of those coefficients which can be very powerful in problems for robust design or tolerancing.

1.3.4 The two modelling cultures

The field or discipline of statistics and that of machine learning/AI/data science both have approaches for this prediction setting. These fields have different roots, customs and culture. Very roughly, statistics has evolved from the early 20th century onward, and data science has roots in business in big tech and in computer science from the early 2000s. See e.g. a famous article by Breiman [3], [4] or [5] that give perspective on these differences, as does the earlier mentioned analytical problem solving article [2]. In the last 5 to 10 years or so, more and more work is done on combining both worlds. In this report, we combine those worlds as well.

The goal of specifying statistical uncertainty on any number supporting a conclusion, such as a regression coefficient, is very much part of the statistical culture. This contrasts with the custom in some ML activities where the predictive performance is described using a single number, evaluated on the test set. Here, we aim to give a confidene interval on this number.

1.4 The questions and overview

Chapter 2 focuses on procedures for assessing prediction performance metrics and how to quantify the uncertainty on it. The procedure of splitting data into training set and test set is compared to procedures of k-fold cross validation. Constructing confidence intervals can be computationally intensive and results may be biased; some recent results are described.

Chapter 3 assumes familiarity about quantifying effects in case of a linear regression model. CQM investigated in Felix Kapulla's internship a method where this is done using a Random Forest, which is a black-box model from machine learning, in an attempt to combine the best of both worlds (the convenient model building from machine learning and rigorous questions and uncertainty quantification from statistics).

2. Prediction

2.1 Introduction to prediction

2.1.1 Uncertainty on prediction errors: big data can easily be small data

There can be good reason to consider the uncertainty on a metric of prediction error. Suppose we work on a model that predicts whether a system in a given situation is going to lead a particular problem. The predictors are measurements on the system describing the state of the system. We have followed the system in actual use and collected a dataset on 1000 situations. In this data, the problem occurs 10% of the time; $y = 1$ in case of a problem, $y = 0$ for no problem. We use a machine learning to make a prediction model from the vector of predictor variables $(x_1, \dots, x_p)$, and do this by splitting the data into a trainingset of n=700, and evaluate on the test set of the remaining 300. We made the split randomly but in such a way that we have 10% of 300 =30 problem cases in the test set. One common measure of performance is **sensitivity**, a.k.a. **recall**: the fraction of problem cases that are correctly predicted as such. Suppose 27 of 30 cases are correctly predicted, resulting in sensitivity = 27/30=90%. However, we are interested in the prediction performance on all future cases or an infinitely large test set. A 95% confidence interval[2] for sensitivity is 73% to 98%, meaning that the actual sensitivity might as low as 73%. In a project situation where this prediction model is developed, decisions need to be made on the model, where choices may be:

- Model is good enough for daily use in practice,
- Collect more data,
- Extend the set of predictors to include the more difficult and expensive measurements.

Such decisions should be based on a confidence interval[3] and the lower confidence value of 73% should be taken into consideration of how poorly it may perform. In this example, we predict a binary response variable (problem occurring or not). The same issue arises for continuous responses (e.g. some quality measure) in terms of e.g. RMSE (root mean squared error). The wide confidence interval for prediction performance in the example above can be made narrower (see section 2.4) by using K-fold cross validation as alternative to the train-test-splitting, but there are some subtle issues involved. Note the contrast to the setting of having a single number on a test set, e.g. in machine learning competitions where different prediction algorithms are compared. The question is how do the small differences between the competitors compare to the uncertainty? A blog [6] by Tijink does consider this question in its appendix.

2.1.2 Setup

We use the setup of 1.3.1. We have an observational dataset consisting of observations $(x_1, \dots, x_p, y)$. For a continuous response Y we write $Y = f(x_1, \dots, x_p) + \epsilon$; here we write Y with a captial to stress it is a random variable as is the error ϵ. An alternative notation is $Y \sim N(\mu, \sigma^2)$ with $E(Y) = \mu = f(x_1, \dots, x_p)$. This notation is easily extended to the class of generalized linear models, of which logistic regression for a binary Y is an example. If Y is coded as 0 and 1, logistic regression is $\text{logit}(E(Y)) = f(x_1, \dots, x_p)$ with $\text{logit}(p) = \log\left(\frac{1}{1-p}\right)$; in this model there is no error term. The notation for categorical Y is slightly different and extends this notation. A note on jargon: in statistics, these models are considered as variants of the same theme with regression in the name (e.g. logistic regression for binary Y,

[2] There are different ways of calculating a confidence interval; here we used an exact Pearson-Clopper interval.

[3] There are also other types of intervals expressing uncertainty in statistics, e.g. a Bayesian credible interval.

multinomial logistic regression for categorical Y). In machine learning, *regression* usually refers specifically to continuous Y and *classification* refers to the case of binary or categorical Y.

2.1.3 Aspects of building prediction models

Many important steps occur in practice before arriving at the idealized situation of having data $(x_1, \dots, x_p, y)$. In CQM's experience, at least the following points touch on the technicalities of this document.

- First steps are the identifiation of the relevant variables $x_1, \dots, x_p$ and y from the problem at hand.
- Most of the hard work lies in arriving at actual real world data to processed, cleaned, versions of $(y, x_1, \dots, x_p)$.
- Domain knowledge should play a role as large as possible. In an extreme case, the model can be written down without any data (e.g. in simple physics). Another example is the cooling down of a hot cup of coffee to the ambient (room) temperature, $T(t) = A + (S - A)\exp(-c \cdot t) + \epsilon$ with A the ambient temperature, S the start temperature, c determines how fast it cools down, and an error term.
- Simplicity and explainability. Black box models are rarely used for any important predictions that affect operational decisions. A simple regression model that predicts only somewhat worse than say a random forest is usually preferred. For instance, CQM has seen situations where a model for a binary response of the form "$\hat{y} = 1 \text{ if } x_1 > 3;\ \hat{y} = 0 \text{ otherwise}$" performs as well as a random forest on $(x_1, \dots, x_7)$.

Although black box models have limitations for final application, they can be useful in model development as a benchmark for how good prediction performance may get. They also tend to indicate the most important inputs.

2.1.4 Some example prediction methods

In this subsection and the remainder of the text we often refer to the book Elements of Statistical Learning, second edition [7] which is sometimes abbreviated as ESLII.
Examples of models that are easy to explain are linear regression and CART (classification and regression trees). Classical linear regression can be extended in many ways with transformed predictors such as $x_1 \cdot x_2$ and x_3^2 and $\log(x_4)$. These linear models are convenient and often easy to interpret and connect to domain knowledge. These are close to RSM (response surface models); when used in combinations with DoE (design of experiments) it is called the RSM methodology.

The observed values of the predictors in the training data can pose challenges.

- "Small n big p", many predictors p compared to the number of observations n; e.g. $p < 10n$ is nice for classical methods
- Strong correlations, relations, dependencies between the predictors.

There can be many more issues, however we focus on these two for now. Some methods that are sort of in the middle of machine learning and statistics can be helpful: Random Forests, Lasso, Ridge. For instance ESLII [7] describes these methods from mathematical and statistical perspectives. There are many more types of models such as gradient boosting, support vector machines or support vector regression that may be suitable as well. Briefly, random forests takes the average predicted value over many trees (CART, classification and regression trees) and the large number of CARTs makes direct interpretation difficult. Lasso and ridge regression are linear models where the coefficients are pushed towards zero to improve prediction performance. Lasso has the property that many coefficients are set to zero, so that it can be viewed as a selection method of predictors, where the predictors are selected that get a non-zero coefficient. Ridge regression does not select and may take more of weighted averages of sets of similar predictors. These models (Random Forst, Ridge, Lasso) have the following desirable properties,

- They can handle response variables Y that are continuous, binary, or categorical.

- Only a little tuning is needed. Tuning is about carefully choosing numerical algorithm settings, called hyper parameters, to obtain good results.
- They give a rough prediction performance estimate for future test sets “for free” as part of the model estimation, at least for continuous Y. This facilitates model building. The book ESLII [7] section 15.3.1 compares it to the LOOCV (leave one out cross validation) method.

Random forests is a method that will capture strong non-linear relationships to some degree. A downside is that the fitted relation looks like a staircase: it is a piecewise constant function with many small jumps so that using them in optimization can be more difficult: the first derivatives are mostly zero (except at the places where there is a jump, there the derivative is undefined). In [8] it became apparent that random forests introduce quite some bias in the sense that effects are pushed towards zero: the effect plot is much flatter.

2.1.5 Groups and dependencies in the data

In practical settings, often the observations cannot be regarded as independent events. Here, it should be understood that the dependence of the resulting Y values is conditional on known predictor values $(x_1, \dots, x_p)$. For instance, suppose we regard the weather forecast and study the temperatures hour by hour. If the observed temperature is much lower at 7:00 than forecast, this is probably also the case at 8:00: the deviations are dependent because they are close in time.
In machine learning and statistics, the term “leakage” refers to information leaking from the training set to the test set, see https://en.wikipedia.org/wiki/Leakage_(machine_learning)

Examples

- Starting with a fictional and frivolous example. A company is interested in the weight of employees where the comparison is made between employees choosing healthy or unhealthy food in the canteen. They decide to collect 100 measurements for both types of food by investigating people who just bought their lunch. Because asking people to participate is a lot of work, 20 instead of 100 people per food type are asked to participate, but each participant steps on the weighing scale 5 times, so that 100+100 data points are collected after all. Here, each participants forms a group of observations. A proper statistical analysis takes these groups into account which basically means that conclusions are based on 20+20 “relevant” observations rather than 100+100 so that confidence intervals of the difference get the correct width.
- In the semiconductor industry, a *lot* consists of 25 *wafers.* Suppose individual wafers each give an observation which is the vector $(x_1, \dots, x_p, y)$, then the lots form the groups.
- 10 machines are tested in 20 configurations where configurations represent different contexts, inputs, circumstances and each give a single observation. The 10*20=200 observations are grouped by machine.
- To study a prediction model in farming, data from 30 farms is collected with one observation per week for 20 weeks. The farms constitute groups of observations.
- In situations with time, errors that are close in time are often positively correlated like in the weather example forecast. Here, as an approximation, groups can be chosen e.g. all measurements within a 24 hour period, and errors from different groups can be considered independent.
- The daily value of a stock market index is predicted using public economic indicators. If the prediction error is high on Monday, likely it is high on Tuesday as well as the economy does not change much and the model treats the short term dynamics only as random: the errors are likely to be auto-correlated.

In statistics, taking such groupings into account for models is standard practice; for instance, linear mixed models are an extension of linear regression. Here, the model equation for linear regression is

considered fixed, and random terms are added with randomness occurring on the group level, e.g. $y = f(x) + U_g + \epsilon$ where $U_g \sim N(0, \sigma_g^2)$ are random intercept contributions for each group g and $f(x)$ is a linear expression in the predictors. This makes generalization to new, unseen groups explicit albeit random. Such models also have counterparts for y binary, ordinal or count data (categorical with >2 predictors is more complex). There have been attempts to generalize this setup to the situation where $f(x)$ is a more general model; e.g. (GAM with random intercepts in [9]) or a random forest with random intercepts in a CQM internship [10]; the latter showed difficulties estimating the size of random intercepts).

In prediction problems, assessing the uncertainty should obey the group structure of the data generating process by e.g. assigning the observations of a group entirely to either the training or test set. This takes some care in coding.

It is perhaps suprising that many standard implementations of machine learning methods ignore the groups in the data when tuning the model complexity (e.g. a penalty parameter). Ignoring the groups can lead to optimistic performance estimates, and hence the chosen model complexity tends to be too great. Lasso and ridge from glmnet [11] have an option in cv.glmnet to manually supply the foldid, so that one can keep groups together in choosing the folds used in k-fold CV. In Stata's Lasso, the same can be enforced by a combination of options. In random forests, possibly the bootstrap could be replaced by a "clustered bootstrap" that respects the groups, but this is not available in the widely known R packages randomForest and ranger.

2.1.6 Expressing prediction performance in the binary case

In machine learning, often theory uses the concept of a loss function: some function that expresses the loss caused by the difference between a single observation and prediction; for instance, squared difference in regression. The total loss is then a sum over all observations in trainingset or setset.

For binary responses, there is the question what is predicted. In introductory machine learning texts often the main form of predictions is a 0 or 1. In statistics, the focus is on $\Pr[Y = 1]$. Of course, on this predicted probability a threshold can be applied to arrive at a binary prediction. The threshold can be chosen depending on the application and costs involved with the two types of errors. The ROC (receiver operator characteristic) and AUC (area under curve) is a way of evaluating a model for $\Pr[Y = 1]$ without having to decide on a threshold first. (https://en.wikipedia.org/wiki/Receiver_operating_characteristic)

If a threshold is chosen, sensivity and specificity are straightforward metrics that are evaluated on the testset. It is based on the 2x2 table of true values of Y and predicted values of Y.

- Sensitivity = proportion of correct predictions for Y=1, where Y=1 usually refers to a problem or issue or disease that should be detected. How sensitive is the method to detect the problem?
- Specificity = proportion of correct predictions for Y=0. Is the detection specific to the problem at hand?

2.1.7 Beyond loss functions and metrics not being a sum over observations

In the binary case example (previous section), the AUC (area under curve) is calculated on a larger set of observations and cannot be expressed as a sum of losses. Methods for cross validation ideally are suited for situations beyond a loss function. Many texts (e.g. chapter 7 in [7]) use sums over individual observations as the basis of an overall prediction metric, but in practice, different forms do occur such as AUC. To our knowledge, AUC cannot be expressed as a sum over observations of some loss. Possibly, sensitivity and specificity can, where the loss of sensitivity is 1 if ground truth=1 and prediction =0, and 0 in other cases; that for specificity defined analogously. However, this loss was not encountered in literature whereas misclassification loss is common. The AUC is attractive for classification as it allows postponement of choosing the threshold.

CQM has not yet found more literature on the distinction between metric-per-fold or metric-per-observation. There are some expressions in appendix I in [12] and in [13].

2.1.8 Prediction performance is not just error and may be in euros

In practice, it may be possible to make the costs of two types of errors in the binary case explicit in euros: costs of failing to detect a problem (e.g. costs involved in solving the problem later), and costs of a wrong detection (e.g. scrap a good product). Here it is possible to consider a range of thresholds. If a further assumption is made on the rate of failures to expect, this direcly leads to expected costs as function of the threshold. The expected costs may then be seen as a measure of prediction performance when comparing models.

Much of literature uses terminology involving the word “error”, and “generalization error” but many methods applying to error metrics may also be applied to these settings.

2.1.9 Comparing methods or just the prediction quality

In the subsequent sections, we describe methods to assess how well some model predicts. There are two common reasons to do this (see e.g. [13])

- Comparison of candidate models
- Assessing prediction quality / generalization error, in view of applying the model in operation.

The sections ahead focus on the latter: assessing in absolute sense some given model. Of course, two models can be compared on these metrics. However, for the purpose of comparison, it appears that it may be better to go beyond a staightforward comparison, and consider on e.g. observation level the losses of both models (section 4 in [14]). This would be similar to having a test set for assessment, consider squared errors for each observation, and consider a paired t-test or other paired test to compare errors.

2.2 Train-test set and K-fold cross validation

In building a model for prediction purposes, often the data is split at random into a training set and a test set. The test set can be quite small in view of a confidence interval for a prediction performance metric. A well-known variant to the training-test split is K-fold CV. Here, the data is partitioned in k subsets, called *folds*, that do not overlap and together form the entire set. In turn, each fold acts as test set where the k-1 remaining folds act as trainingset. The model is built on a fraction $\frac{k-1}{k}$ of the data and assessed on the test fold and the prediction performance is calculated. The k resulting values are averaged and this is given as the best guess of prediction performance for the final model, which is trained on the entire dataset. Figure 2 shows an example.

fold1	fold2	fold3	fold4	fold5	
test	train	train	train	train	MSE1
train	test	train	train	train	MSE2
train	train	test	train	train	MSE3
train	train	train	test	train	MSE4
train	train	train	train	test	MSE5
train	train	train	train	train	avg

Figure 2: K-fold cross validation using MSE (Mean Squared Error) as example for prediction performance. The final model is based on the entire dataset.

This well-known setup can be applied to many situations and metrics of prediction performance. A non-standard example occurs for a binary response e.g. of a problem occurring, where benefits or costs in euros are given for false positives and false negatives. Given a percentage of problems in future cases, the performance metric is measured in euros.
Note that the K-fold output depends on the random choice of the folds: a different randomization results in a different response value. A variant of this scheme is *repeated K-fold cross validation* where the folds are chosen repeatedly and the final answer is the average over the repeated evaluations. In case of groups, the partition into folds needs to be respect the groupings.

2.2.1 Doing K-fold CV well is more difficult than train-test split

A danger using the K-fold CV is that not all steps from model building are included when rebuilding the model on the k-1 folds subsets of the data, e.g. in choosing which predictors to include or how to pre-process them. Say that one predictor is a time series that can be smoothed in different ways, and then different ways of expressing the middle and spread of the data are considered. This can easily result in potentially 1000s of predictors of which perhaps two summaries are chosen (middle and spread of the data). It could be that the choice of predictor originally was made somewhat subjectively, e.g. by studying plots of the raw, trial-and-error, etc. In principle, this selection should be part of the model building and therefore different predictor sets could result in each fold's corresponding training set.

Another example is when different high-level models are considered, say support vector machines, neural nets, k-nearest neighbors, lasso, or random forests. This choice should be part of the model selection as well.

In practical situations, this requires more careful coding and it is tempting to skip the informal steps of model building, and cross-validate only subsequent steps. See 7.10.2 in [7] for a more detailed description and examples that this a practical danger.

2.2.2 Which estimate of generalization error

The book ESLII [1] makes a point on the different thing that is estimated by K-fold CV compared to train-test set. Ideally, one builds the model using all available data, and then magically evaluates prediction performance on an infinitely big test set to have the future prediction performance.

We compare this idealized setup with the two practical methods.

Infinite test set (not possible)	Training-test	K-fold CV
100% of data for training	e.g. 70% of data for training	100% of data for training
Independent large test set	30% of data for test	~100% of data for test
Exact value for prediction performance of future data	Estimation of prediction performance of the trained model: the expected value matches, but with large uncertainty due to possibly small test set (30%)	Estimation of expected prediction performance over many new hypothetical datasets + trained models from the same population
(Unbiased)	Unbiased (averaged over many repeated newly obtained test sets, it is on average correct)	Bias (complicated)

The bias in the K-fold CV case comes in part from the fact that the errors correspond to models based on a fraction $\frac{k-1}{k}$ of the data. Generally, more data improves the model so there is a bias involved towards worse prediction performance. However, the word "bias" also implies what you compare the estimate to, and different versions of this comparison exist in literature. The article [12] is quite detailed on this in its analyses (see just below).

It is telling that [7] concludes section 7.12 with "We conclude that estimation of test error for a particular training set is not easy in general, given just the data from that same training set. Instead, cross-validation and related methods may provide reasonable estimates of the expected error Err." There is also a footnote stating that the subtle distinction between types of errors was added only to the second edition, illustrating that focus on these issues is relatively recent. The authors Hastie and Tibshirani

together with main author Bates wrote an excellend article in much the same style with many more details and insights, [12].

A more recent article [15] constructs confidence intervals for R-squared, which is effectively a re-scaled MSE version: $R^2 = 1 - MSE/var(Y)$ where MSE is the mean squared error and var(Y) is the MSE for the simplest possible model using a constant only. It is consistent with the views in [7] but opts for the equivalent of Err rather than the generalization error based on the current dataset $\mathrm{Err}_{\mathrm{T}}$.

It is surprisingly tricky to reason about the errors in practice. Setting up code for a simulation is done quickly enough, but they can easily use a reference error that is not relevant: the error of the true, unknown model. However, we are primarily interested in the generalization error of the model at hand, as evaluated on a big testset. In case of little data, we may use K-fold CV and settle for the expected generalization error (expected over many similar trainingsets from the population). On the website Stackexchange, there are a few conversations mostly about LOOCV versus K-fold CV. For instance, a first highly rated answer (on Stackexchange) to such a question is given [16]. It is a thoughtful piece with references to papers and includes a simulation. From the description and the code, it appears that over many new trainingsets, the resulting MSE estimates are considered: as a distribution, and the bias is taken in reference to the the true model. The same is true for the simulation in another thoughtful question on the same topic on Stackexchange, [17]. In contrast, Elements of Statistical Learning [7], compare the estimated error to the generalization error $\mathrm{Err}_{\mathrm{T}}$ in the simulation of figure 7.7, where a large test set is used. Also, [12] is very explicit in these notations, an improvement to the texts on Stackexchange. This corresponds to our interest as well.

In code a simulation at CQM on different occasions, it became apparent how easy it is to make a thinking error in the many steps of aggregating over folds, repetitions, taking averages, standard deviations, divide by sqrt(K), etc.

2.2.3 Repeated K-fold CV: remove the randomness from choosing folds

In the K-fold CV, the data is split in a random way. This random choice ultimately makes the estimate of the generalization error random. To reduce this random influence, one could repeat the procedure for many random splits and average the result.

2.2.4 Nested cross validation schemes: hyper parameter tuning

As explained in section 2.1.9, one goal of evaluating predictions is to choose between candidate models, e.g. choosing a penalty parameter to balance model complexity and overfit. Often, K-fold CV is used to pick the best penalty parameter in terms of a performance metric, e.g. glmnet [11]. This best value can be taken as evaluation for the entire model, but this is optimistic in nature, as the test set equals the trainingset, at least for training the penalty parameter, which is part of model fitting. To remove that optimism without needing a separate test set, a basic scheme is

- **Outer loop:** For folds k=1,..K
    - For penalty parameter values c from a set
        - For training data = all minus fold k:
            - **Inner loop:** For folds m=1..M (newly chosen folds)
                - Construct model given penalty parameter c
                - Save prediction metric for fold m
        - Overall prediction metric from the folds
    - Choose best c for the model
    - Save prediction metric for fold k
- Overall prediction metric from the folds

This scheme results simply from regarding the hyper parameter tuning as part of model building, and construct an independent outer K-fold CV loop around it.

There are of course variants possible, e.g. [12] uses the existing K-1 outer training folds for the inner loop for a K-1 fold CV (where a model is trained on K-2 folds).

2.2.5 The dataset: random sample or structured

Our main setup is about a population with a random sample, and generalization being about the population ("a very big testset"). In practice, the situation can deviate from this setup. One example is a DoE, where predictors take on specific values on a grid. Say there are 10 predictors each taking 2 values, and what is known as a "2^{n-k} design" is used, to get say a dataset of $2^6 = 64$ points ($n = 10, k = 4$), where the dataset has particular properties (e.g. all four combinations of (x_i, x_j), occur with frequency 64/4, for all $i \neq j$). A K-fold CV scheme would create trainingsets that destroy such properties.

Another example of structure is a categorical response y where the dataset is constructed to contain each value equally often:

> Both stackexchange pages [16] and [17] contain references to a 1995 paper of Kohavi, [18], which focuses on the K-fold CV vs LOOCV case when the response is "labeled data", i.e., y is categorical. This paper is written from the machine learning / algorithmic perspective. On one hand, the importance of a confidence interval for accuracy is acknowledged as in statistics. On the other hand, the population – sample foundation is not made explicit. The remark on training and testset being dependent in the famous Iris dataset rests on each label of y occurring exactly 1/3 of the time in the dataset, i.e. the statement is conditional on this particularly nicely balanced dataset. In this example, this leads to severe errors in prediction accuracy estimation. However, if the data had been sampled from a larger population, this phenomenon would not occur.

ESLII [7] is explicit in training and testset being random in the notation of chapter 7.

In some settings with severe imbalance in frequencies of y-labels, the training data is sampled with different frequencies. E.g., if some issue y=1 occurs only once every 1000, and non-issues y=0 in the remaining cases, a large fraction of the y=0 cases may be discarded to have less imbalance (e.g. 1:10). As long as selection happens randomly, the schemes should still not introduce or increase bias for specificity, sensitivity, AUC; a case-control study in biostatistics uses the same principle.

2.2.6 Handling groups and dependencies in the data

In case of groups in the data, all splitting / K-fold / bootstrap procedures can be adapted by keeping groups together in splitting. For bootstrap this is usually called clustered bootstrap, where the random selection in bootstrap samples is taken per-cluster. See Stata's bootstrap command and cluster option.

For dependencies-over-time in data, as an approximation one can make "rational subgroups" of the data.

- Divide up the time axis in say 10 about equally sized intervals
- Consider training sets of groups 1-5, then 1-6, …, 1-9.
- The corresponding test set is the single interval immediately following the training set.
- A variant is to take the entire part following training interval as test sets.

This is called "evaluation on a rolling forecasting origin" (https://otexts.com/fpp3/tscv.html)

A variant of such a scheme is to no longer respect time order and do a leave-one-interval-out cross validation: in turn each interval is testset, and all other intervals are trainingset. A conceptual disadvantage is that the past is predicted party using the future.

2.2.7 LOOCV (Leave One Out CV)

Leave one out cross validation is in fact K-fold cross validation with K=n= size of the dataset. In some models this is computarionally cheap: OLS (ordinary least squares) where it can be done using a trick

related to studentized residuals. Random Forests give an estimate of generalization error for free based on the out-of-bag samples. Section 15.3.1 of ESLII [7] states that it is nearly identical to LOOCV. The CQM tool Compact [19] for Kriging models uses LOOCV where the slow fitting of "hyper parameter" is sped up by having the main model's values as good starting values for each of the models.

Considerations for choosing K-fold CV (with K<<n) over LOOCV are

- Computational (typically far fewer models to fit)
- LOOCV has larger variance than K-fold CV according to e.g. section 7.10.1 of [7]; here variance is w.r.t. hypothetical new datasets. LOOCV will have less bias from the "(k-1)/k" data size difference for the final model and the k models used for evaluation.
- With very small sample sizes the concern, for K-fold CV may be the randomness from the choice of split. Or for practical reasons, the fact that given the dataset the resulting model depends on a random seed may be seen as a disadvantage. This is absent in LOOCV.
- With not too many groups (section 2.1.5), LOOCV on the group level, i.e. leave-one-group-out CV, may be the natural choice.

2.3 Confidence intervals for Testset – trainingset

In this section we consider the situation where data is split into training and test set, and the model performance is determined on the test set. Possibly, group structure is considered (section 2.1.5).

2.3.1 Bootstrap

From the statistical viewpoint, we regard the prediction performance calculated from the testset as an estimate of prediction performance on the entire population. Note that in this setting this is a case of population → random sample → [some number] meaning that bootstrapping applies, which will give a standard error of that estimate. Here, bootstrapping is applied to the testset:

- Loop 50 – 1000 times
    - Take a bootstrap sample of the testset (i.e. construct a set of new observations where each observation is randomly taken from the testset, with replacement; some original observations will not occur in the bootstrap, some multiple times).
    - Repeat the calculation: evaluate prediction performance on the bootstrap sample. Collect these numbers.
- The standard deviation of the collected numbers is taken as standard error of the estimate.
- Work with the original point estimate and standard error, e.g. a 95% C.I. is estimate +- 2*standard error.
- It is tempting to consider low and high percentiles as well. These may give biased intervals though, and more bootstrap samples are needed. There are bias correction methods available, see the manual on Stata's bootstrap command or [20].

2.3.2 Statistics on simple metrics

Section 2.1.6 briefly introduces specificity and sensitivity in case for prediction of two classes. Both are proportions of a slice of the testset (Y=1 and Y=0). Therefore, an exact Pearson-Clopper interval can be calculated, even in case of 100% correct in the data.

An overview of more metrics on this 2x2 table is given in https://en.wikipedia.org/wiki/Confusion_matrix; some of these are proportions to which this simple method applies.

In OLS (ordinary least squares, i.e. classical regression), the size of errors is summarized in R-squared and RMSE (root mean squared error). The distribution and confidence intervals for RMSE are known; see https://en.wikipedia.org/wiki/Standard_deviation#Confidence_interval_of_a_sampled_standard_deviation where the

degrees of freedom are according to https://en.wikipedia.org/wiki/Mean_squared_error#In_regression. An underlying assumption that is used is that the errors themselves are on average 0, which is true in OLS for errrors from the trainingset. In prediction models in general, such as Lasso or ridge, MSE (mean of squared errors) is built up from both bias (i.e. the unsquared errors have a non-zero average) and variance.

2.4 Confidence intervals for K-fold cross validation

2.4.1 Bootstrapping as outer loop

The bootstrap method applies to a population-sample situation, where a sample leads to any number estimated from it (e.g. mean, standard deviation, a percentile): it is a form of population → random sample → [some number]. We observe that an estimate for generalization error by K-fold CV is also a function from the data. As such, it can be bootstrapped. The scheme then is as follows, for any metric expressing generalization error.

For K-fold CV

- K-fold CV: for each fold k=1..K,
  - build a model on the folds excluding k
  - evaluate using fold k as testset
- generalization error estimate is average over the k folds

To construct a confidence interval

- For 50-1000 bootstrap replicates
  - make a bootstrap sample
  - On the bootstrap sample, perform K-fold CV to get an estimate of the error metric
- Take the SD of the error metric as standard error SE of the metric
- Perform K-fold CV on the original dataset to get the point estimate E
- An approximate 95% confidence interval is E +- 2*SE.
- There is a conservative bias from the fact that the underlying models use $(k-1)/k$ part of the data (typically, more regularization is applied in case of smaller trainingsets).

Note that for e.g. K=5 and 50 bootstrap samples, this involves 50*(5+1)=300 models to be built.

In statistics, the bootstrap method can beconsidered as Plan B if Plan A in estimation fails. Here, plan A is to have the sampling distribution properties of an estimator by mathematical derivation which works for model parameters after maximum likelihood, or OLS.

It is hard to find literature on the internet for this method This scheme is found in a 2012 MSC thesis from South Africa, [21]. A similar scheme, BCV, “Bootstrap Cross-Validation” is found in [22]. This paper mentions [23] which focusses on statistical tests on performance metrics for comparing candidate models, and which suggest slight adaptations in the scheme.

The scheme BCV is also described in [24] (as referenced by [22]). The motivation is to have a simple procedure, BCV, especially for small samples, but the focus is on getting an accurate error estimation. This is achieved by the bootstrap *average* of the bootstrap estimates. Note that in statistics, usually only the *standard deviation* of bootstrap estimates is taken as standard error, where the point estimate is based on the original (un-bootstrapped) data. The point here is that this seems to work better on small samples, as small as n=16 is stated, and that for bigger samples you would not use this method. The article also features standard errors, although this is not the main focus.

It also discusses the concern the whole complicated scheme: in bootstrap samples, an original observation can occur multiple times and one copy can end up in a training fold and another copy in

the test fold. The article argues why the entire scheme will still work. Another argument why it works is from the mathematical way of thinking: K-fold CV gives an estimate for generalization error which usually is a “nice” engouh function from the sample; and nice functions from a sample can be treated as black-box and therefore be bootstrapped. To clarify further, this is analogous to a joke about mathematicians. If a mathematician knows how to make tea from cold water, he/she can then also make it from warm water: just let the water cool down first, then apply the previous knowledge. The additional step of cooling from warm to cold is the bootstrap, the core activity of preparing tea from cold water is the K-fold CV.

2.4.2 The “standard error method”

We repeat the figure of K-fold CV.

fold1	fold2	fold3	fold4	fold5	
test	train	train	train	train	MSE1
train	test	train	train	train	MSE2
train	train	test	train	train	MSE3
train	train	train	test	train	MSE4
train	train	train	train	test	MSE5
train	train	train	train	train	avg

Figure 3: K-fold cross validation using MSE (Mean Squared Error) as example for prediction performance. The final model is based on the entire dataset.

The estimate of generalization error is the average of numbers per fold. A common consideration is to take the standard error of the mean of these k numbers in K-fold CV, e.g. figure 7.9 in [7]. However, it is also common in literature to formulate this scheme in terms of squared errors (or, more generally, other loss metrics) on the *observation* level. Consider the difference in case of squared error loss. If the k folds each have exactly the same size, first taking averages within the folds, and then average those averages will result in the same overall average. If the folds have slightly different sizes, some observations are weighted slightly heavier in case fold averages are averaged. Section 2.1.7 gives examples where a per-fold metric can be very handy in practice.

An important example of this type standard error is in in model building of Lasso and Ridge in the well-known R-package glmnet [11] plots the generalization error +- standard error vs a model complexity parameter. Then there are two typical choices of penalty parameters: the one minimizing estimated generalization error, and one according to the “1 SE rule”, where the penalty is chosen for simpler models such that the thus selected penalty has error within 1 se of the minimum.

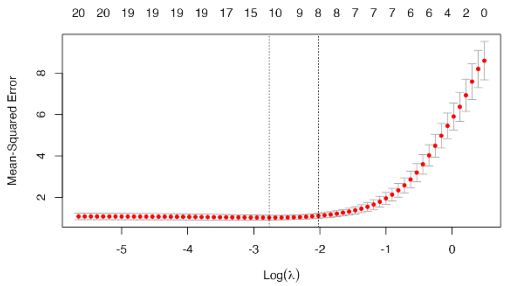


Figure 4: Figure with standard errors in glmnet taken from [11]

The statistical tool Stata uses the same options [25]. This illustrates a common use of this standard error, allbeit for a slightly different purpose.

Is taking the standard error over the folds' results a "good" method? The article [12] has a detailed study of a method that is strongly related and which is called the "naïve CV"; this method uses the standard error of the invidual squared errors (the article's appendix I makes the link). The mathematical and simulation analysis warn that the resulting confidence intervals are too narrow. It also introduces an alternative, called nested CV (see the next section, 2.4.3). In the article's discussion section:

> "A fundamental open question is to understand under what conditions the standard CV intervals will be badly behaved, making the nested CV computations necessary. Roughly speaking, we expect the standard CV intervals to perform better when n=p is larger and when more regularization is used. In our experiments, we saw that even in the mundane linear model with n=p = 10, the miscoverage rate of standard CV was about 50% larger than the nominal rate. As n increases, however, the violation decreases."

In CQM's straightforward simulation for just estimating a mean, the method's standard error matched the simulated distribution of MSE estimates quite well.

2.4.3 Further details on the standard error method

The book ESLII [7] introduces the standard error method but does not really explain the "correctness" of the standard error. In principle, the k numbers being averaged have a dependency. Here, we consider the entire dataset as a set of random vectors $(x_1, \ldots, x_p, y)$ from the population. The models are fit using this data, and in that sense random as well.

Consider fold k acting as testset. Here, $y_k = f_{-k}(x) + e_k$ where the errors come from fold k, and the model f_{-k} is based on the other folds. For say folds 1 and 2, the models f_{-1} and f_{-2} use training data from folds 3, 4, 5 which means that the quantities $f_{-1}(x)$ and $f_{-2}(x)$ are dependent random variables (for any given x). This means that if we take one observation from fold 1, and one observation from fold 2, that their errors $e_k = y_k - f_{-k}(x)$ (for $k = 1, 2$ and the prediction vector x from the respective observations) are dependent as well via the indirect route of folds 3, 4, 5. However, one might expect that in "typical" cases the variance of e_k is much larger than the variance of $f_{-k}(x)$; after all, the last term is based on many observations and e_k is associated with only a single observation. Writing the true, unknown relation as $y = F(x) + \epsilon$ we would have $f_{-k}(x) \approx F(x)$ (where $F(x)$ is non-random) and $e_k \approx \epsilon_k$ for the true random and independent contributions ϵ. This argument of near-independent results over the folds is used in [18] in the proof of a proposition (Figure 5).

Proposition 1 (Variance in k-fold CV)
Given a dataset and an inducer. If the inducer is stable under the perturbations caused by deleting the instances for the folds in k-fold cross-validation, the cross-validation estimate will be unbiased and the variance of the estimated accuracy will be approximately $\mathtt{acc}_{CV} \cdot (1 - \mathtt{acc}_{CV})/n$, *where* n *is the number of instances in the dataset.*

Proof: If we assume that the k classifiers produced make the same predictions, then the estimated accuracy has a binomial distribution with n trials and probability of success equal to the accuracy of the classifier. ∎

Figure 5: Proof of a proposition in [18] using the argument that the models ("classifiers") from each of the folds are approximately equal. The proposition is specific to the studied case of the response being categorical and using the accuracy metric "acc".

A next formal step is to rewrite $y_k = f_{-k}(x) + \left[\epsilon_k + \left(F(x) - f_{-k}(x)\right)\right]$. The error term within the brackets is partitioned into the ϵ components that is by construction independent between observations, and

comparatively "small" deviations of estimated functional form and true form, $F(x) - f_{-k}(x)$. If the "small" property holds, we have approximate independence between the folds k of the estimates of MSE from all observations i of fold k, $MSE_k = \sum_i e_{ki}^2$. These arguments are heuristic; section 4.1 of [12] gives a more precise description that does consider dependencies between the errors.

Note that this argument is easily extended to the case of AUC, which is not a sum over observations of some term.

If repeated K-fold CV is employed, each repeated CV results in a new estimate and standard error value. The estimates are averaged over the repeats to a new, less noisy estimate (noise from the random split is reduced by averaging). The standard error values squared are also averaged (as in the pooled standard deviation) and taken as standard error for the estimate. Note that there is a conservative step here, as the noise reduction is not taken into account here.

In Wikipedia's phrasing: "the *standard error* (SE) of a statistic (usually an estimate of a parameter) is the standard deviation of its sampling distribution or an estimate of that standard deviation." (https://en.wikipedia.org/wiki/Standard_error). In this line, we performed a simulation to compare the SE from this standard error method to the actual spread of the estimator of many simulated datasets for a simple problem of estimating a mean, where the (repeated) K-fold is employed. There is a near-perfect match for the standard error and the standard deviation over simulated datasets. Moreover, in the simple setup the bias is very small, both to the generalization error and average generalization error.

The simulation from 2.4.2 is described in further detail in an CQM internal report[4]. The setting used is: n=40 observations, k=5 folds, 10 repetitions, 1000 new datasets. As true model, we use the simplest possible example of $Y \sim N(\mu, 1)$ for unbiased model estimates. As the whole procedure might be different for biased models, we use a form of pulling the estimate to some other value. We put a prior on $\mu \sim N(0, \sigma_p = 0.5)$, and we have $\sigma = 1$ so that the posterior mean of μ is $w \cdot 0 + (1 - w) \cdot \bar{Y}$ with $w = \frac{1}{\sigma_p^2} \cdot \left[n * \frac{k-1}{k} + \frac{1}{\sigma_p^2} \right]^{-1}$ (https://www.statlect.com/fundamentals-of-statistics/normal-distribution-Bayesian-estimation).
We call this Bayesian estimate "regularized", as a stand-in for e.g. Lasso and ridge, which give biased estimates.

2.4.4 Nested cross validation

The Bates paper [12] describes an alternative scheme with better coverage property of the confidence interval of the error estimates. There is good news on the *genereralization error*, i.e. on the prediction performance of the model based on the given dataset if applied to future large test sets, which is the error we are interested in as practioner.

- The error estimate is closer to the average generalization error Err (i.e. averaged over many hypothetical new datasets) than to the Err_T (the generalization error of the current model based on dataset T).
- Still, the confidence intervals from the paper are for Err_T

A good coverage property of a 90% confidence means that in reality (or in a simulation), the true value lies in the confidence interval 90% of the cases; i.e., the interval is neither too wide, nor too narrow. The direct standard error method from section 2.4.2 has poor coverage, the article's nested CV better coverage.

The scheme is somewhat similar to repeated K-fold CV:

- For e.g. 50 repetitions
    - Create K folds randomly

[4] For CQM internal reference: CQM technical document rep kfold with se.docx

    - Do an inner (K-1) fold crossvalidation where one of the folds is held apart
    - Collect and calculate specific intermediate results
- Return $\widehat{Err}, SE$

There is an R package available on Github, https://github.com/stephenbates19/nestedcv . The experiments from the article using this are given in https://github.com/stephenbates19/nestedcv_experiments.

2.4.5 BBC-CV (Bootstrap Bias Corrected CV): A method doing it all

The 2018 article titles "Bootstrapping the out-of-sample predictions for efficient and accurate cross-validation" by Tsamardinos, Greasidou, Borboudakis [26] combines many of the ideas and requirements mentioned into a promising method. It is not references by Bates, Hastie, and Tibshirani in [12]. The article [26] is clearly written and consider the issues discussed so far,

- Beyond loss functions on individual observations, such as AUC (2.1.7)
- Nested CV for removing optimism due to hyper parameter fitting (2.2.4)
- Bootstrap for errors (2.3.1, 2.4.1)
- Goals: model selection, evaluation, hyper parameter tuning (2.1.9)
- Need for confidence intervals (2.1.1)

These issues are all addressed by this method, as well as very effecient computation by requiring not so many models to be fit.

The basis is "CVT", cross validation with tuning. Here, tuning refers to hyperparameters, and any other choices (perhaps pre-processing choices), called *configurations*.
At first, the data is partionioned into K folds. For each configuration, K-fold CV is performed. The out-of-sample predictions from each test folds are collected into a big matrix Π for many configurations (the columns); and the rows are individual observations grouped in folds.

Fold	Conf 1	Conf 2	Conf 3	Conf 4
1				
2				
3				
4				
5				

Note that the number of model fits equals [nr configurations]*K.
Next, a scheme is used that deals with the bias resulting from cherry-picking the best configuration, by bootstrapping the *already calculated predictions*. Here, the choice of best configurations is considered.

- For bootstrap samples b=1...100
    - Take a bootstrap sample of observations (rows in the matrix)
    - Pick the configuration with the best prediction metric value: i.e. choose a column
    - Find the observations (i.e. rows) that are not in the bootstrap sample.
    - Evaluate the prediction metric on those and store the result.
- From the bootstrap sample, take the mean and SD and percentiles for a point estimate and CI.

The article proceeds with simulations and an example from real data and finds that bias is small, and good coverage rates of the CI (i.e. fraction of time that the CI contains the true value). Note that they can investigate real datasets by using large hold-out sets that give a "ground truth".

The article mentions further extensions:

- BBC-CV with repeats: as in repeated CV (2.2.3); the matrix gets a third index and the bootstrap samples should be clustered over the repeats.
- BBCD-CV: BBC with Dropping. This adaptation is to speed things up. It fits the model fold by fold in the matrix (so within a chosen test fold, go over the configurations). However, if after only a few test folds certain configurations really seem to be inferior to the best, these inferior configurations are removed from the candidate lists so that the remaining part of K-fold CV does not need to compute models for them.

The authors provide Matlab code on Github, https://github.com/mensxmachina/BBC-CV .

2.5 Conclusions: which method for assessing generalization error?

The above sections considered a number of methods and a number of considerations. There does not seem to be a clear general recommendation from literature. The advice below may be seen as a rough guideline.

The setting we are interested in has a dataset, on which prediction models are built and assessed. For model comparison, one could simply use comparable metrics directly; or consider some scheme in which pairs of prediction errors are considered (each observation has an error from model 1 and model 2).

For the assessment of a single model, the simplest and clearest setup is to split the data into training and test set ("hold-out set"). Considerations why this may be unsatisfactory are as follows:

- Is the test set "large enough"? Say at least 30 observations?
    - For a binary response variable Y, are there at least say 30 cases of Y=1 and at least 30 cases of Y=0?
- Is the resuling confidence interval (section 2.3) of the generalization error narrow enough for useful interpretation?
- Explainability to stakeholders: below we consider more complex schemes but these are much more difficult to convey than trainingset-testset.
- Trust in the procedure. If before the model building starts, the testset is taken apart and "locked up" without being looked at, it can be considered as an independent testset and presented as such. Did you do that to preserve this interpretation advantage?
    - You should use the testset only once (and not retrain the model if it gives new insights). As subsection 2.2.1 explains, this is much more difficult in K-fold CV as really all design and model choices should be made formal in scripts. How much confidence in the procedure should you convey to the stakeholder at the end?
- Computational effort, if model building takes long computational time. Is there any room left for more elaborate schemes?

If in the above considerations for train-test split a disadvantage becomes apparent, a k-fold CV may be usedful. Considerations for k-fold CV are as follows.

- Consider if there are groups in the data. If there are not too many, consider LOOCV where one group is left out at a time.
- K=5, K=10 are mentioned as common choices. LOOCV is simple, but computationally expensive, may have large variance, and is less robust.
- Handle the model building / tuning / pre-processing choices from the evaluation by putting it into the model building part (a form of nested cross validation, subsection 2.2.4)
- How desirable is robustness: repeated K-fold CV with say 5 repeats results in less random noise in point estimate and standard error, but takes 5 times as long.

- Is it ok to report a rough confidence interval ok with approximate coverage, with the remark that it may be optimistic? If so, use the straightforward standard error method
- If better confidence intervals are desired, also re-consider whether all analysis steps are included in the model building function. Skipping these steps likely has more impact than all the variant schemes for calculating an interval.
- For better CI coverage property and less bias, we found the following alternatives in literature.
    - For small sample sizes (say <20, <30) consider bootstrap CV (subsection 2.4.1) as they report good performance for this specific situation.
    - **BBC-CV** is for considering different model "configurations" explicitly and is as expensive as K-fold CV for each configuration. Reasonable guarantees for bias and CI coverage. Matlab code is available.
    - Other alternatives require say 50 times as much computation time. Two options,
        - Option 1, Use the nested CV as by [12] and the R-package. The confidence intervals have reasonable guarantees from the article and the confidence interval deals with the bias. It needs many repeats, the default is 50. The result is "stable" as in repeated K-fold CV (no influence of how folds are chosen)
        - Option 2, Use an outer bootstrap loop as in 2.4.1. Some more care than usual may be needed in considering artefacts as double points, empty folds, etc. Perhaps 50 bootstrap samples are enough.
    - A reason for choosing option 2 is the article [24] which emphasizes good properties for small samples; also, the procedure is simpler to explain: a straightforward combination of two common techniques (bootstrap and k-fold CV). Option 1 has well studied properties of the correctess of the CI.

For model comparison, one could simply use comparable metrics directly; or consider some scheme in which pairs of prediction errors are considered (each observation has an error from model 1 and model 2). The BBC-CV method includes such model comparison in the method. We finish with a summary in Table 1.

Method	**Nr of model fits**	**SE and CI**	**remark**
Train-testset split	1	Direct, or bootstrap	
K-fold CV	K+1	"straightforward standard error method"	
Repeated K-fold CV with R=5 repeats	R*K+1	Same, pool the R estimates	Randomness of choice of folds is averaged out.
LOOCV	n	SE of error	(for some methods it is quick)
Nested CV from [12] with R=50 repeates	R*K+1	R-package; CI also capture bias	Relatively good guarantees for CI
Bootstrap CV with R=50 bootstrap samples	R*K+1	Direct from bootstrap	BCA; mean of bootstrap samples for small samples
BBC-CV	K+1	Direct from bootstrapped OOS predictions	.. times the nr of "configurations", but that also applies to all others methods
BBC-CV with repeats	R*K+1		

Table 1: Overview of methods for generalization error and a CI on it

3. Understanding and quantifying effects

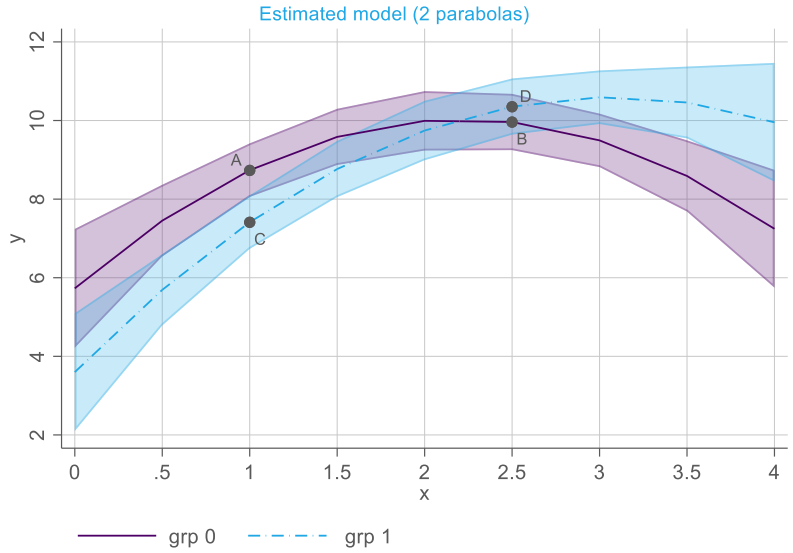
3.1 Introduction

See section 1.1 for the background of many of CQM's projects. We focus on data-driven models for $y = f(x)$. The recent article [2], titled "Analytical problem solving based on Causal, Correlational and Deductive models" gives a relevant taxonomy where *data-driven* models can be *causal* or *correlational*; the problem at hand determines what type of model is needed. Section 1.2 starts with two important categories of problems: Prediction or "Understanding and quantifying effects". Prediction calls for correlational models and is considered in chapter 2.

For the goal "Understanding and quantifying effects", causal models are needed. Here, the relationship between y and x itself is the focus. The fact that the dataset is not infinitely large is handled by statistical techniques to put numbers on the resulting uncertainty of the effects; section 2.1.1 motivates that this is relevant. Typical examples of understanding effect problems are finding the most important factors among the x_i, comparing prototype vs reference, and clinical trials where the question is about stating the amount of evidence for an effect. An example of the latter is a clinical trial comparing two groups of patients, where a single variable x is binary and indicates group membership; the effect of x is then the difference between mean outcomes in y.

3.2 Effects and statistical uncertainty

The statistics discipline has developed many models for many situations, often with a strong mathematical foundation. In those models, there is a precise mathematical formulation of probability distribution of y and structure of x, e.g. as in regression. An example: observations of Y are random and independent variables $Y = c_0 + c_1 x + c_2 x^2 + \epsilon$ with ϵ normally distributed with mean 0 and standard deviation σ. Given the data, the model is fitted, i.e. values for model parameters c_0, c_1, c_2, σ are determined. The fit is described by R-squared (percentage variation explained) and the σ describes how uncertain individual predictions are. In machine learning, it usually ends here, but statistics additionally assesses uncertainty of the model parameters: confidence intervals for c_i, hypothesis tests for whether c_2 can be 0, or functions of the c_i. We extend the example to have two groups "grp". We simulated data according to $y = 10 - (x - 2)^2 + \epsilon$ for grp=0 and $y = 10 + (x - 2.75)^2 + \epsilon$ for grp=1, with $\epsilon \sim N(0, \sigma = 4)$ for 250 observations in either group. We fit a regression model $y = c_{0i} + c_{1i} x + c_{2i} x^2$ for group $i = 0{,}1$ and get the following fits with confidence intervals for predicted values.



i

The regression output in this case is technical and we skip reporting on the coefficients directly. Instead, we use software to plot the above graph corresponding to the model. As an illustration, we investigate $x = 1, 2.5$ and the predictions marked with A, B (group 0) and C, D (group 1). The model predicts the expected value at these points, e.g. $y_A = E(Y|x = 1, grp = 0)$. The table below gives these values under y (se), where the *standard error* is given in brackets. The standard error indicates the uncertainty of the estimate, resulting from the size of the dataset and the noise. A 95% confidence interval is given roughly by estimate +- 2*standard error.

	grp	x	y (se)
A	0	1	8.73 (0.35)
B	0	2.5	9.96 (0.36)
C	1	1	7.41 (0.35)
D	1	2.5	10.35 (0.36)

effect within group		interaction	
	delta (se)		
B-A	1.23 (0.37)		
		(B-A)-(D-C)	1.71 (0.52)
D-C	2.94 (0.37)		

The right hand side of the table proceeds with effect sizes: the amount by which y increases as x increases from 1 to 2.5; B-A and D-C. The difference between these effects is the *interaction* effect (of group and x on y). As 1.71 is more than two times its standard error (2*0.52), the interaction effect is statistically significantly different from zero: the two groups give different curves, and the 1.71 (with 95% confidence interval 1.71 +-1.04) gives a measure of how different the curves are.

This toy problem actually has large underlying spread of the data which is not displayed (the standard deviation of 4 is large in the vertical scale). This toy problem uses knowledge that a parabola / polynomial of degree 2 is approximately appropriate for the relation, and not e.g. $a \cdot \exp(-b \cdot x) + c$. Under that assumption, these effects and their uncertainties are properly assessed.

In the example, we make a small set of points at which to evaluate the function. At these points, we make a vector of model predictions $y_{pred} = (y_A, y_B, y_C, y_D)$ that are functions of the model parameters c_{ji}. This 4-vector has a 4 by 4 variance-covariance matrix associated with it that gives the uncertainty; at cell i, j we have $Cov(y_i, y_j)$. On the diagonal, this is $Cov(y_i, y_i) = var(y_i) = se(y_i)^2$, the square of the standard error of the prediction. This matrix is used for construction of the standard error of linear functions of the predictions. For instance, the interaction is $a \cdot y_{pred}$ with $a = (-1,1,1,-1)$ and working

out the math gives that the standard error squared of the interaction is $0.52^2 = Var(a \cdot y_{pred}) = a \cdot Cov(y_{pred}) \cdot a^T$ (row vector times square convariance matrix times column vector).

In summary, for a small set on a grid we make predictions, we obtain the variance-covariance matrix of it, and then can construct main effects, deltas, differences from those to go back to the problem.

3.3 Procedures for effect calculations

A realistic part of the example from the previous section is the mix of a categorical predictor (group) and continuous predictor (x) with more-complex-than-linear relationship. In statistics, models are often formulated and discussed in terms of main effects, two-way interactions and higher-order interactions; indeed the classical ANOVA table is centered around these. However, note that the reasoning above works for any shape of model $y = f(x, grp) + \epsilon$. The formulation of effect plots is somewhat different than most statistical texts it can be easily extended to many types of models and many types of effects.

A sketch of the procedure for investigating effects from a model is:

- Estimate a model $y = f(x_1, \dots, x_p) + \epsilon$ where the predictors may be continuous or categorical.
- Study the problem at hand by formulating a number of effects. These can be thought of as predictions that depend on one or a few predictors.
    - The simplest plots have one predictor at a time. The example above has two. Panels, colored lines, different symbols may help visualize 3 in one plot, more becomes tedeous.
    - what to do with the other predictors, see below
- Plot the predictions using curves / lines / effect plots as above; similar to the effect plots that go alongside ANOVA.
- Combine the different effect plots into one overall plots with shared vertical axis.
- For a single effect plot, tabulate the points, and relevant differences between them. This uses the standard errors and variance-covariance matrix of the vector of predicted values.
- Statistical significance tests for whether the effects could be zero and corresponding confidence intervals help in interpretation.

In a situation with several predictors in a model and an effect plot, some predictors appear as elements in the plot elements and other predictors do not. The simplest method of dealing with those in generating predictions is to set them at an middle or average or reference value. Another strategy is to "average over them" in some sense. This is closely related to causality as explained in a paper by Zhao and Hastie [27], where such plots are called *partial dependence plots*. Suppose a plot is constructed for x_1, x_2 (e.g. $x_1 = x, x_2 = grp$ above). Loop over all observations in the trainingset. Loop over all points in the grid for the plot (e.g. one such gridpoint is $x = 2.5, grp = 1$). Substitute for each observation just the x_1, x_2 values of the grid point but leave $x_3, x_4, \dots$ as they were. Make a model prediction for each observation. Average these predictions over all observations; this becomes the value for y for the current (x_1, x_2) gridpoint. This way of working is Friedman's *partial dependence plot.* The paper [27] makes a connection to Pearl's causality and causal thinking [28] that is useful in case the data is observational.

The partial dependence plot (PDP) is also the default method of making plots and tables in Stata after model estimation, see the margins and marginsplot commands. Stata does not mention the term PDP, and we do not know other references that confirm that Stata's powerful margin commands are in fact PDPs: a notable example separation between the statistical and machine learning world. Stata has an exceptionally well thought-through syntax for a large suite of models, from survival analysis to variants of logistic regression, possibly combined with random effects. The quantity for the vertical axis in effect plots (called "marginal effects" in Stata) may be transformed. For instance, for logistic regression, both

the linear scale (log-odds) and the predicted probability are useful; for survival plots the probability of survival beyond a given time point, for regression the $\mu + 2\sigma$ point could be plotted, etc. Such flexibility has proven useful in numerous CQM projects.

For more common situations, there are R packages and point-and-click tools such as Minitab and Jmp. For random forests, the PDP take on a somewhat different character, see section 3.6.

3.4 Bootstrap as a general method for standard errors

The bootstrap method is a general method applicable to situations where a random sample is taken from a population, and some value is calculated (Figure 1 from 1.2). An example when the bootstrap is not applicable is an experiment with designed "settings" such as in design of experiments in statistics, or when an observation is collected in equal time intervals and time plays a role as a predictor.

The bootstrap method is a "plan B" for obtaining standard errors or confidence intervals in case "plan A" of dedicated statistical models as in 3.2 fails. This can be the case if for instance some non-standard pre-processing is part of the model building such as imputation of missing values, calculation of a node in a piecewise linear function etc. Section 2.3.1 briefly explains the bootstrap. It involves a loop of 50 to 1000 times of taking a bootstrap sample from the data, do the calculations such as estimating the model or calculating a summary value, evaluate the quantity at hand, and construct standard errors or even the variance-covariance matrix of a vector of predictions.

It is assumed that the quantity under study depends in some "continuous" way on the original data.

There is statistical theory from Efron, e.g. [20] to back up that this calculation method indeed gives a reasonable standard error of the estimated quantity, i.e. an estimate of standard deviation by how much the estimation of our quantity varies in imaginary, repeated datasets – the bootstrap samples are random samples with replacement and approximations of those imaginary repeated datasets.

3.5 From machine learning: feature importance, explainable AI

Some methods from machine learning $y = f(x)$ can give a measure of the importance of the predictors (called *features* in machine learning). One example is Lasso/Ridge regression. Here, the predictors are typically normalized so that they have mean 0 and standard deviation 1. The regression coefficients for the normalized predictors can be compared in size and are a measure for feature importance in $f(x)$. Note that a causal interpretation is difficult for several reasons. Typically, in observational datasets, the predictors are related; e.g. in a dataset with people's body measurements, length of left leg x_L will be strongly correlated to length of right leg x_R, assuming no irregularities. Ridge regression will typically result in about equal coefficients for x_L and x_R, but arguably, the thought experiment of holding x_L constant while varying x_R to assess the impact of x_R is difficult to imagine. Lasso would typically pick one of both predictors setting the other coefficient at zero; predictors with zero coefficients in Lasso could both be unimportant and important. Although there are difficulties with causal interpretation, it still holds that large standardized coefficients point to important predictors.

Random forest is another technique (see next section) which is the average of many trees t_k, $y = f(x) = \sum_k t_k(x)$ so that the every day use of the word "forest" is correct in that sense. From the way the random forest is constructed, a table with *relative importance* measure for each predictor can be constructed. Here, the value expresses the overall importance of the predictor including interactions with other predictors. For instance, in the thesis [29] written at CQM, in section 4.3 the importance metric of a large problem is aggregated over groups of predictors, so that ultimately insight is gained in which parts are most relevant in a larger context of reinforcement learning.

Explainable AI (XAI, eXplainable AI) is a theme and research area that is encountered in machine learning more and more. The book [30] gives a nice overview. There are global and local explanations where the aspects above are examples of global explanations that are true for the overall model and data. Local explanations focus on a single prediction, e.g. a model assessing a loan request by a bank's customer. If an AI model approves or turns down the request, a local explanation seeks to answer the question of why. Possible techniques are Shapley values, given from game theory, and considering what-if scenarios, e.g. how much more income is needed for the decision to go the other way, or at what different age.

Both concepts (global, local) are still somewhat different from the effect described in section 3.2 that are convenient in many problems. These effects are more common in social sciences and econometrics, where the predictors may be properties of people and indicators of government policies, where different policies are considered. In industry, we seek similar causal insights.

3.6 Random forest with standard errors

In 2022 and 2023, Felix Kapulla wrote his thesis at CQM ([8]) in which he researched and developed methods for estimating effect sizes in the style of section 3.2 using random forests, and at the same time give a standard error, which is unusual in that part of the literature. Figure 6 summarizes the setting.

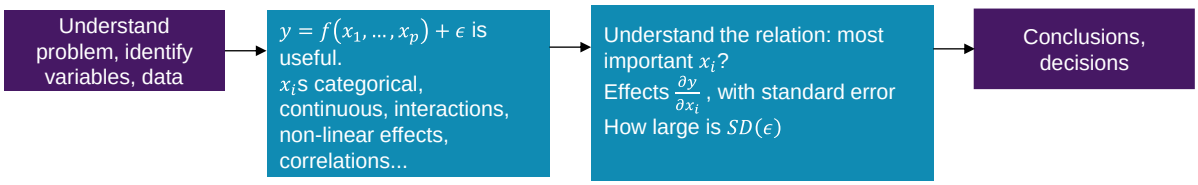


Figure 6: generic problem setting for effect sizes

The hope of using a machine learning algorithm such as random forest instead of a statistical regression-type model is that it is much faster in a project: for regression, the data quality needs to be better (outliers removed, severe colinearity studied and predictors replaced by summary values, model assumptions checked). Of course, these extra steps also result in much better understanding of the problem. The role of a more automatic black box algorithm such as a random forest is that an approximate answer is obtained more quickly; if in a project an exploration can be reduced from 3 days to 3 hours, this is a very useful option to have.

In principle, there are many machine learning methods besides random forests that could take the role of $f(x)$, e.g. support vector regression (SVR), gradient boosting, and neural networks. Advantages of random forests are

- Ability to handle mix of continous and categorical predictors
- Little tuning needed (mainly on making sure there are enough trees)
- A built-in reporting of generalization error for new datasets
- Fairly well researched with a mathematical and statistical angle, with extensions for survival times, quantile regression, e.g. [31]

A disadvantage is that the resulting surface is a non-continuous function so that it does not lend itself for optimization.

A starting point of the investigation in [8] was a paper by Wager [13] that gives a method for a standard error for an individual predictions such as y_A and y_B from section 3.2. By studying the mathematics closely, we could extend the method to calculate standard errors for $y_B - y_A$ effects, and also the general effects from section 3.2. The thesis [8] gives an introduction to random forests, and

summarizes the workflow. An application of the workflow to a real dataset describes the use of the PDP, a type of effect plot that can take on a non-linear shape: the random forest can give very detailed shapes compared to polynomial models. In Figure 7 below, age ("leeftijd") has a large value only for a narrow low range. In a problem with several predictors with possible correlations, PDPs can reveal such patterns that direct visualizations of the data might miss.

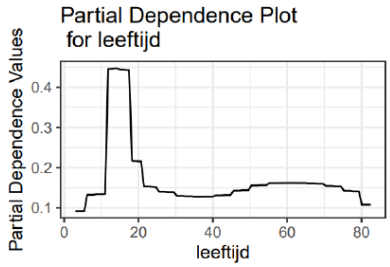


Figure 7: example Partial Dependence Plot for a random forest

The thesis also identifies the complication that estimates of the effects are *shrunk* towards zero. From the conclusions in 6.1 in [8]:

> As a first step, local main and low-order interaction effects have been defined as discrete derivatives. Local means that we fix some point in the predictor space and see how a fitted machine learning model responds to changes for a variable of interest. Therefore, by fixing such point in the middle of the predictor space, statements about effect sizes for an average observation can be made. However, by means of a simulation study it was shown that such effects are consistently shrunk towards zero for the random forest algorithm. The magnitude of shrinkage depends on several factors such as the sample size, the dimensionality of the predictor space, the correlation between predictors and certain hyperparameter settings. Therefore, such effects do not necessarily reflect the effect situation in the real world and interpretation must be done with care. Nonetheless, standard errors of such effects are estimated by applying a bias-corrected version of the jackknife estimation method to the inherent bootstrap samples of the random forest. With this approach the variability of the estimated effect is quantified as if infinite bootstrap samples were used to eliminate Monte Carlo noise down to the level of the inherent sampling noise. With the simulation study it was shown that on average this estimation method reflects the true variability of shrunk effects accurately for a sufficiently large sample size. However, stable estimates are only obtained if a sufficiently large number of base learner is used with regard to the sample size.

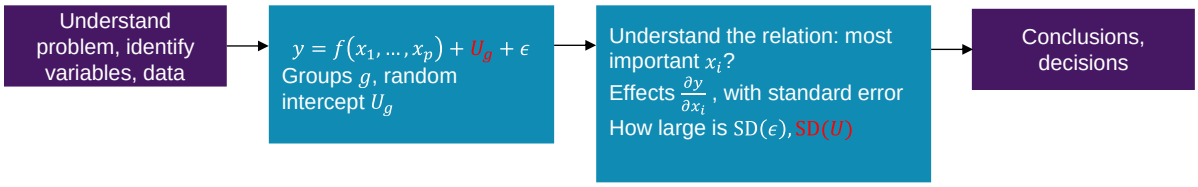
The shrinking of effects towards zero is well-known in machine learning, related to regularization. It is explicit in Lasso and ridge regression. It is a price to pay to handle observational data with correlations and many predictors. A specific difference between Lasso/ridge and random forest is the ability of the latter to use detailed non-linear effects of a single predictor as in Figure 7. The assumption in random forest of how smooth that relation may be, is implicit.

For CQM some important points are

- Effect estimates from random forests can be biased, in the most severe cases in the simulation study (more predictors, more highly correlated, smaller trainingset) by 50%, i.e. the average estimate is only 50% of the true unknown value.
- The standard error estimates require more trees than just for prediction. In the examples studied, 2000 trees were enough. There are practical plots to study to make this judgment, similar to the OOB performance metric vs tree number.
- The standard error of an effect is useful for interpreting the effect size.
- There are extensions for "custom" effects in the style of section 3.2.
- In the R eco system, the R-package for ranger is a good option.

3.7 Random forests handling groups

Another direction in which a regression-type analysis can be extended using machine learning methods to speed things up is the is including random effects to account for groups.



The effects of x on y can be more subtle in the presence of groups as Figure 8 shows: per group the effect of x on y is negative, overall it is slightly positive.

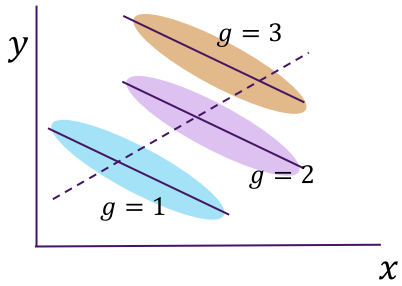


Figure 8: Simson's paradox

Such a study is well understood using linear mixed models / regression. As in 3.6, the thesis by Thomas Rutten [10] written at CQM investigates the variant of a random forest with a random intercept, called MERF (mixed effects random forest). An important starting point was the article [32] which claims to have such a model. The article does feature a simulation study with some fixed effect structure and a random intercept on some group level as in section 2.1.5. However, the article mainly focuses on prediction and does not mention any investigation into the estimate of the standard deviation of the random intercept. The thesis [10] does its own simulation and evaluates this standard deviation, which is a useful part of an analysis on its own. In certain cases, the estimate of this standard deviation has a bias of a factor 3, which also casts doubt on whether one would ever want to use this implementation of the MERF. The thesis [10] gives more detail on the situations where problems are more severe.

3.8 Outlook

This chapter does not start from an overview of possible techniques. Over the last several years, CQM gradually gained insights into how machine learning techniques may be used, as predictive method and as a method for understanding effects. We mention a few avenues.

- GAM, generalized additive models [33] and gamlss (GAM for location, shape, and scale) can be used with not-too-many-predictors and not-too-high-interactions, using a large library of basis functions that allow e.g. seasonal effects, complex domains such as points on the earth's surface, and give some statistical results for non-spline terms.
- Kriging / Gaussian process, a flexible interpolation method, especially useful for surrogate modelling. Usually, the points in predictor space should be far apart and not repeated.
- Gradient boosting, xgboost, makes use of a weighted average of trees as random forests. Some implementations allows missing values in the predictors, effectively assuming that the domain is $\mathbb{R} \cup \{\infty\}$. Although there will be strong assumptions behind the reason of

missingness, it can be very handy to have a first quick model without the need to sort out missing values.

- We look out for an implementation of random forest
    - that takes in missing values as xgboost as it should be possible
    - that uses clustered bootstrap instead of bootstrap in building the trees as it at least in part would handle groups in terms of more regularization (section 2.1.5)
    - has the ability to include random intercepts and structures
    - has an implementation in both the R and Python ecosystems that give the same results.

Note that for standard errors, “plan B” of bootstrapping effect sizes (section 3.4) would work with any model, although the model should be fit about 50 times more.

3.9 Conclusion

Understanding and quantifying effects has played a significant role in CQM’s projects. In an observational dataset, questions are often translated into concrete variables y and x where the relation between them is of interest to understand the problem, support decisions, etc.

The methods from statistics such as regression and its variants often can be applied, but much work may be needed to understand the predictors x, deal with structures, outliers etc. Sometimes in a project it can be handy to have a quick, approximate answer by employing a prediction model from machine learning.

In problem solving, the main questions for a model $y = f(x)$ are investigating the effects of the predictors x_i on y. This reports the emphasis of the statistical way of formulating the questions: keeping all but one predictor constant, and investigate what happens with the expected value of y if you vary one predictor; in addition, put a number of that uncertainty. This is different from most approaches taken in eXplainable AI to explain a model.

There are many models from machine learning that could be used for $y = f(x)$, e.g. [7] explains several of them. Random forests have some practical advantages in that they require little tuning and are flexible in terms of capturing non-linear effects and interactions. Some investigations in the last years at CQM have taken this path. In a recent investigation recorded in Kapulla’s thesis [8], we learned that bias in the effect estimates can be severe (i.e. expected value of estimated effect size may be only 50% of true effect size). Simulations give an indication in what situations bias may be small. Given the bias, the statistical uncertainty of the effect is of interest just as in classical statistics. The method could be extended to give standard errors to the effect estimates.

Although this chapter reports mostly on random forests, in principle there are many techniques from machine learning. Random forests are special in that they allow computationally efficient calculations of standard errors.

4. Bibliography

[1] Á. Bárkányi, T. Chován, S. Németh, and J. Abonyi, 'Modelling for Digital Twins—Potential Role of Surrogate Models', *Processes*, vol. 9, no. 3, Art. no. 3, Mar. 2021, doi: 10.3390/pr9030476.

[2] J. de Mast, S. H. Steiner, W. P. M. Nuijten, and D. Kapitan, 'Analytical Problem Solving Based on Causal, Correlational and Deductive Models', *The American Statistician*, vol. 77, no. 1, pp. 51–61, Jan. 2023, doi: 10.1080/00031305.2021.2023633.

[3] L. Breiman, 'Statistical Modeling: The Two Cultures (with comments and a rejoinder by the author)', *Statistical Science*, vol. 16, no. 3, pp. 199–231, Aug. 2001, doi: 10.1214/ss/1009213726.

[4] D. Donoho, '50 Years of Data Science', *Journal of Computational and Graphical Statistics*, vol. 26, no. 4, pp. 745–766, Oct. 2017, doi: 10.1080/10618600.2017.1384734.

[5] B. Efron, 'Prediction, Estimation, and Attribution', *Journal of the American Statistical Association*, vol. 115, no. 530, pp. 636–655, Apr. 2020, doi: 10.1080/01621459.2020.1762613.

[6] Tijink, Matthijs, 'How to win silver in the CodeCup'. [Online]. Available: https://tij.ink/entropy/

[7] T. Hastie, R. Tibshirani, and J. Friedman, *Elements of Statistical Learning: data mining, inference, and prediction. 2nd Edition.* Accessed: Jul. 03, 2023. [Online]. Available: https://hastie.su.domains/ElemStatLearn/

[8] F. Kapulla, 'Building an automatic workflow for an explorative Random Forest Model in Industry', Universiteit Leiden, Leiden.

[9] G. Simpson, 'Using random effects in GAMs with mgcv'. [Online]. Available: https://fromthebottomoftheheap.net/2021/02/02/random-effects-in-gams/

[10] T. A. S. Rutten, 'Mixed-effects random forest model for quantifying relations in clustered data', Eindhoven university of technology, 2021feb22. Accessed: May 22, 2023. [Online]. Available: https://research.tue.nl/en/studentTheses/mixed-effects-random-forest-model-for-quantifying-relations-in-cl

[11] 'An Introduction to `glmnet`'. Accessed: Jul. 03, 2023. [Online]. Available: https://glmnet.stanford.edu/articles/glmnet.html

[12] S. Bates, T. Hastie, and R. Tibshirani, 'Cross-validation: what does it estimate and how well does it do it?', arXiv.org. Accessed: Jul. 05, 2023. [Online]. Available: https://arxiv.org/abs/2104.00673v4

[13] S. Wager, 'Cross-Validation, Risk Estimation, and Model Selection', arXiv.org. Accessed: Jul. 05, 2023. [Online]. Available: https://arxiv.org/abs/1909.11696v1

[14] P. Bayle, A. Bayle, L. Janson, and L. Mackey, 'Cross-validation Confidence Intervals for Test Error', arXiv.org. Accessed: Jul. 05, 2023. [Online]. Available: https://arxiv.org/abs/2007.12671v2

[15] S. Hawinkel, W. Waegeman, and S. Maere, 'The out-of-sample R^2: estimation and inference', *The American Statistician*, pp. 1–11, Jun. 2023, doi: 10.1080/00031305.2023.2216252.

[16] X. Bourret Sicotte, '[TL:DR] A summary of recent posts and debates (July 2018)', Cross Validated. Accessed: Jul. 03, 2023. [Online]. Available: https://stats.stackexchange.com/questions/61783/bias-and-variance-in-leave-one-out-vs-k-fold-cross-validation/357749#357749

[17] J. Westfall, 'Variance of K-fold cross-validation estimates as $f(K)$: what is the role of "stability"?', Cross Validated. Accessed: Jul. 03, 2023. [Online]. Available: https://stats.stackexchange.com/q/280665

[18] R. Kohavi, 'A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection'. Accessed: Jul. 03, 2023. [Online]. Available: https://www.semanticscholar.org/paper/A-Study-of-Cross-Validation-and-Bootstrap-for-and-Kohavi/8c70a0a39a686bf80b76cb1b77f9eef156f6432d

[19] E. Stinstra, 'COMPACT', CQM. Accessed: Jul. 11, 2023. [Online]. Available: https://cqm.nl/nl/compact

[20] B. Efron and T. Hastie, *Computer Age Statistical Inference*. Cambridge University Press, 2016.

[21] M. Muna Balla Elshareef, H. G. Mwambi, and L. E. Dodd, 'Evaluation of Strategies to Combine Multiple Biomarkers in Diagnostic Testing', School of Mathematics, Statistics and Computer Sciences, University of KwaZulu-Natal, Pietermaritzburg, South Africa, 2012.

[22] B. B, M. O, T. H, and W. C, ‘Resampling methods for meta-model validation with recommendations for evolutionary computation’, *Evolutionary computation*, vol. 20, no. 2, Summer 2012, doi: 10.1162/EVCO_a_00069.
[23] T. Hothorn, F. Leisch, A. Zeileis, and K. Hornik, ‘The Design and Analysis of Benchmark Experiments’, *Journal of Computational and Graphical Statistics*, Jan. 2012, doi: 10.1198/106186005X59630.
[24] W. J. Fu, R. J. Carroll, and S. Wang, ‘Estimating misclassification error with small samples via bootstrap cross-validation’, *Bioinformatics*, vol. 21, no. 9, pp. 1979–1986, May 2005, doi: 10.1093/bioinformatics/bti294.
[25] ‘Stata Bookstore | Lasso Reference Manual, Release 18’. Accessed: Jul. 03, 2023. [Online]. Available: https://www.stata.com/bookstore/lasso-reference-manual/
[26] T. I, G. E, and B. G, ‘Bootstrapping the out-of-sample predictions for efficient and accurate cross-validation’, *Machine learning*, vol. 107, no. 12, 2018, doi: 10.1007/s10994-018-5714-4.
[27] Q. Zhao and T. Hastie, ‘Causal Interpretations of Black-Box Models’, *Journal of Business & Economic Statistics*, Jul. 2019, Accessed: Jul. 12, 2023. [Online]. Available: https://www.tandfonline.com/doi/abs/10.1080/07350015.2019.1624293
[28] J. Pearl and D. Mackenzie, *The Book of Why: The New Science of Cause and Effect*. Basic Books, New York, 2018.
[29] F. De Ruiter, ‘Practical reinforcement learning with large action space’, CQM. [Online]. Available: www.cqm.nl/asimov/Practical_reinforcement_learning_with_large_action_space
[30] C. Molnar, *Interpretable Machine Learning: A Guide For Making Black Box Models Explainable*. Munich, Germany: Independently published, 2022. [Online]. Available: https://christophm.github.io/interpretable-ml-book/
[31] S. Athey, J. Tibshirani, and S. Wager, ‘Generalized random forests’, *The Annals of Statistics*, vol. 47, no. 2, pp. 1148–1178, Apr. 2019, doi: 10.1214/18-AOS1709.
[32] A. Hajjem, F. Bellavance, and D. Larocque, ‘Mixed-effects random forest for clustered data’, *Journal of Statistical Computation and Simulation*, vol. 84, no. 6, pp. 1313–1328, Jun. 2014, doi: 10.1080/00949655.2012.741599.
[33] ‘gamlss: Generalized Additive Models for Location Scale and Shape’. [Online]. Available: https://www.gamlss.com/

CONSULTANTS IN QUANTITATIVE METHODS

Countless variables, innumerable x's. Are they coincidental or structural? How do they interrelate and what effect do they have? What is really important and what not? We help organizations make complex processes transparent. Using quantitative models, we create a framework to analyze processes and make decisions based on the facts, enabling you to optimize your planning and logistics, and improve your product and process innovation. Intelligence, that takes your organization to a lasting, higher level. We analyze and clarify, with a genuine understanding of the issues you face. That is the way we do things.

From x to u

CQM B.V.

T +31 40 750 23 23
F +31 40 750 16 99
E info@cqm.nl
I www.cqm.nl

Vonderweg 16
5616 RM Eindhoven
P.O. Box 414
5600 AK Eindhoven
The Netherlands

Trade register 17076484
IBAN NL61RABO0359340598
VAT NL801228505B01