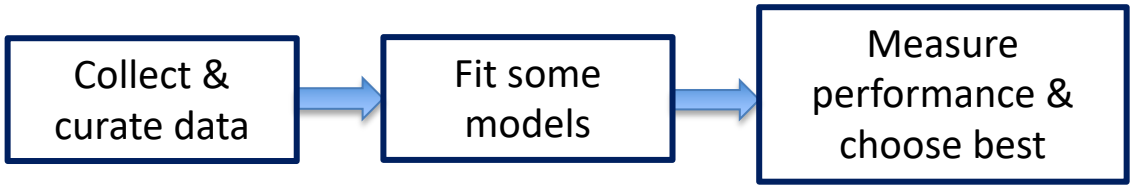
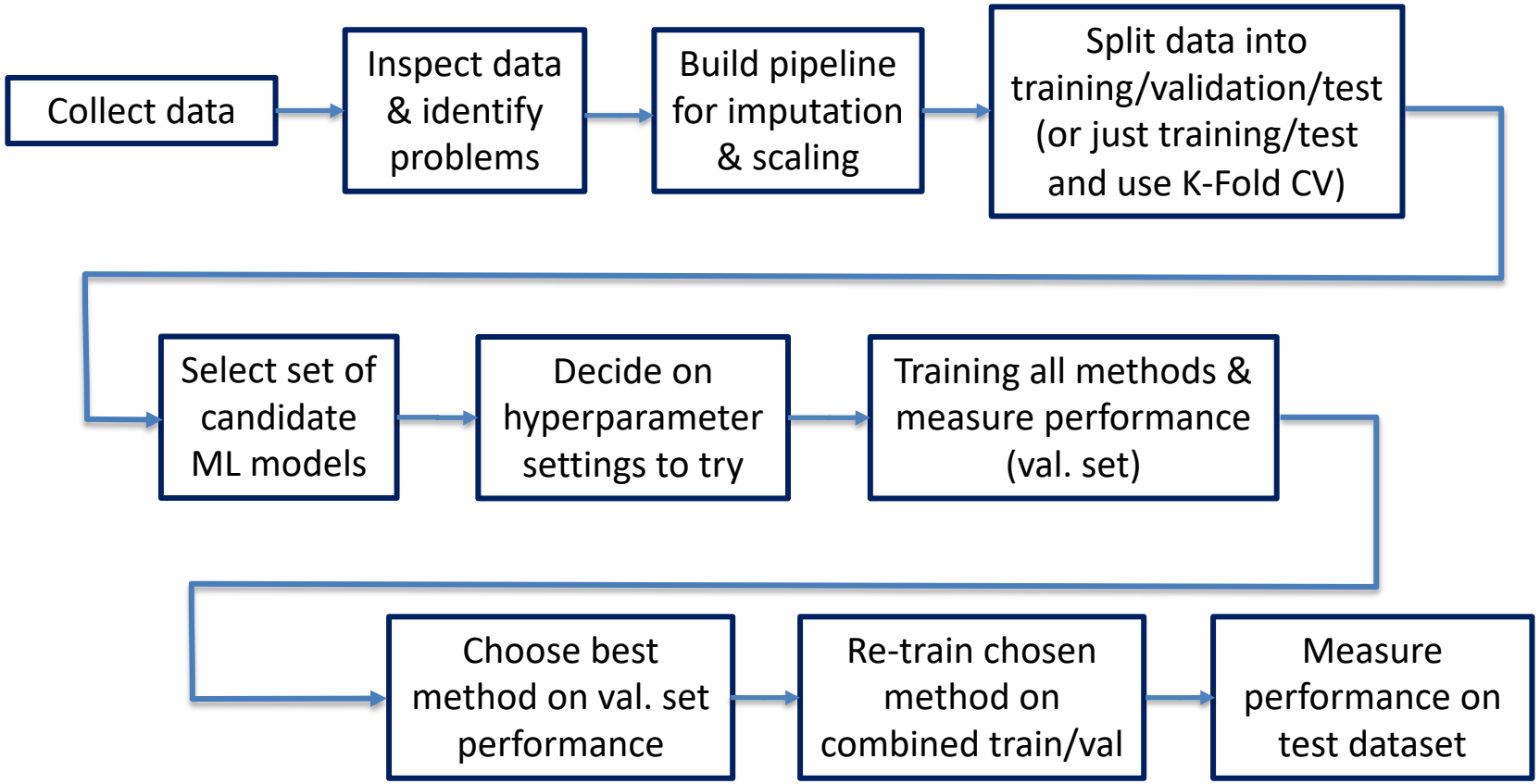
Summary: Standard (basic) workflow

Simplest form (least detail)



More detailed version



Bias

- A lot of what we do is to avoid biases
- Randomness is always a factor as we only have a limited set of samples (our dataset) out of all potential samples
- Statistical theory helps keep things fair / unbiased by:
  - Avoiding circularity
    - if a dataset is used to optimize anything (params or hyper-params/models) then it cannot be used again for that method without giving biased results (e.g. training results are always biased)
  - Understanding that all performance measures are noisy
    - For example, in workshop 3 a 5-fold CV gave me:
      - Validation rmse: 59662.5, 64107.2, 59657.9, 62793.9, 62193.4
      - Training rmse: 56188.7, 55252.4, 56264.3, 55634.2, 55866.4
      - Each fold has a different rmse based on different training sets
      - Training rmse is biased to be smaller (it is always less than validation here, but it does not have to be – both are noisy)
      - The smaller the sample set, the noisier the measurements
      - K-fold allows variance to be measured (helpful to understand how noisy the measurements are)

General Guidelines

- Use the right dataset in the right place: train – validation – test
- Don't run imputation/scaling/pre-processing on all data first (as this mixes between train/val/test) – instead put them in a pipeline (then their parameters are only ever fit using training data, regardless of CV choice)
- Larger datasets are better for training and better for stable validation/test performance measures, but we don't always have as much as we want
  - For smaller datasets K-fold CV can be very helpful in testing performance using all data, but in an unbiased way.
  - Hold-out validation sets are more common with bigger datasets.
  - Either way a fixed, holdout test set is needed and 80/20 split is common.
- Obtain performance measurements on both training and validation sets to assess overfitting/underfitting.
  - Remember to interpret high/low based on the context of data and task
- Choose best model based on validation set results (it is imperfect/noisy, but the best that we have)
- All performance measurements using a dataset are biased for any method optimized using that data (i.e. choosing best params/hyper-params/models/architecture/etc)
  - This dataset can still give unbiased measurements for methods that were not optimized using it (e.g. adding new methods should still use old validation set)
  - Another, unseen dataset (i.e. test) is needed for biased measurements on methods that were chosen based on performance on another dataset (i.e. validation)

Common Mistakes (to avoid!)

- Using the test dataset before a model is chosen / not using the validation set(s) to choose the best model.
- Not using the same validation set (or same fold for K-fold) for all methods being compared.
- Choosing a poor performance metric (this will be discussed in more detail later on).
- Using too small a range of hyperparameter settings (e.g. trying k=2 and k=3 only for K-NearestNeighbour Regression; or weights in the range of 1 to 10, when reasonable values of the weights might be anything from 1e-8 to 1e+8)
- Not thinking about the context to figure out what is big and what is small
  - For example, is 3.0 a small error?
    - If it relates to a house price around $400,000 then yes, $3 is small.
    - If it relates to a value that can only be between 0 and 4, then 3 is quite large.
    - If it relates to a percentage then 3% might be small in many cases, but if you want to get your performance over 99% then a 3% error is big!
    - So the data, task and desired performance all contribute to the overall context.
  - Statistical tests can also help to assess when a difference is "big" (important/repeatable) or "small" (unimportant/unrepeatable)
- Treating different model types (e.g. linear and KNN regression) differently from how you treat different hyperparameter settings of one model (e.g. k=2 or k=3).
  - What you *should* do: treat every variant of a model as a different model

Common Mistakes (to avoid!)

- Only thinking about one single, final summary result (e.g. the result from GridSearchCV) without considering other information.
  - What you *should* do instead: characterise the performance by looking at the range of errors (not just the mean error) and looking at the errors on different sets (training, validation and test) to see if things are going wrong somewhere and/or if easy fixes might be available (e.g. different scaling)
- Using results obtained on the test set for doing *comparisons!*
  - What you *should* do instead: use performance measures on the validation set for all comparisons and only use the test set to get unbiased, generalisation performance
  - What you *can* do: if you add new methods later on then use the *same* training and validation sets as before (use the same random state if you have any random splitting or CV calls) and if the new method is better than the others, *based on the validation results,* then you can use this as the new best choice and measure its performance on the test set. But, you *cannot* change your choice to the previous best method, even if the test set performance is better for the previous method – as the test set results *cannot* be used to perform selection!

A Note on ML Challenges

- In ML challenges the following is common:
  - a training set is made publicly available
  - you take this and use some cross-validation to optimise your own method
  - you send in your method to the organisers
  - they evaluate it on a hidden test set
  - results (on the hidden test set) are displayed for all methods
  - the ranking is based on these results.
- This is biased!
  - it is very likely that the top ranked method would perform worse on another dataset
  - this is because the ranking is based on this test set, so it cannot also give a fair, unbiased performance estimate
  - to be fair to the challenge organisers, there isn't a lot of choice as if they did use a separate test set to measure performance again, another method might have a better result and then there would be (valid) complaints about whether it is fair to rank using the first test set and not the second test set
- There is no good solution to this, but you should be aware that the results shown by most challenges are biased in this way. If you used the best method on another dataset it will almost certainly do worse than the performance shown on the challenge's test set.