Predictive Analytics Lecture 5

Adam Kapelner

Stat 422/722
at The Wharton School of the University of Pennsylvania

February 14 & 15, 2017

The Scourge of Modeling

Missingness is the "scourge" of data analysis and modeling. It's a big nuisance which has to be dealt with in order to get to inference and prediction. What is missingness? Show JMP class project prediction dataframe.

Simply put, missingness is the absence of the measurement for an observation (a hole in the matrix!) I will cover some theory of missingness, of which you will not need to know. But you will need to know the eventual takeaway messages.

We will denote the full data matrix as $\boldsymbol{X} := [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_p]$ where each $\boldsymbol{x}$ has length n. We will denote the observed data matrix as $\boldsymbol{X}_{obs} := [\boldsymbol{x}_{1,obs}, \ldots, \boldsymbol{x}_{p,obs}]$ as well as the missingness features as $\boldsymbol{M} = [\boldsymbol{m}_1, \ldots, \boldsymbol{m}_p]$.

Previously the goal was to fit $f(x_1, \ldots, x_p)$ using $\boldsymbol{X}$ now we only have $\boldsymbol{X}_{obs}$ and $\boldsymbol{M}$.

Types of Missingness Models

There are two ways to view response models with missingness. Given the entire dataset is seen we have:

1. **Selection** Models. The response is independent of missingness. Imagine we are predicting whether or not a person will default on their mortgage. The person fills out an application for a mortgage in a bank. One of the fields on the paper form elicits their age. If a person forgot to fill out their age (there are many fields and this one just slipped by them), age will be missing on the final form. This will indeed impact the ability of the built model to predict loan payback (we will discuss workarounds soon), but there is no reason to suspect that there will be a separate model for those who forget to fill it out and those who don't.
2. **Pattern Mixture** Models. The response is not independent of missingness. Same modeling context but here we have the age filled out by a computerized background check which scours public and FBI records. If age goes missing here... that means... the person may have some legal issues going on. It would make sense to fit two models: $f(x, m = 0)$ and $f(x, m = 1)$. Likely, $\hat{p}$ will be higher in the latter!

Default to Pattern Mixture Model

In the latter, missingness itself affects the response. Unless you have a good reason to believe otherwise (you would need to think about how each variable went missing), you should default to thinking about it as a pattern mixture model.

In the mortgage application example, the fact that age went missing needs to become a feature in its own right. This is handled (usually) by including the dummy variables $\boldsymbol{m}_1, \ldots, \boldsymbol{m}_p$ and then... interacting these dummies with other features to allow for differential response models based on missingness.

Night and Day Strategies to Fill in the Holes

The dichotomy of selection vs. pattern mixture is one piece of the puzzle. Now you have the practical piece of the puzzle — your dataframe has holes!! What to do?

1. **Listwise Deletion** — simply put, delete all rows with missingness. Why is this bad? Selection Bias!!! This bias will lead to poor future predictive performance (bad generalization). Also, even if there is no selection bias, ... you are throwing out precious information! Your model could have been better if you figure out a principled means to make use of the measurements that you *do* have on those observations.
2. **Imputation** — "fill in" the holes with "good guesses". This essentially means you must create a model for holes in each measurement. What is the simplest imputation strategy (i.e. what is the simplest model)? The average!

By default, when producing a model, JMP will use listwise deletion (don't do this!!). I you are less lazy, you can generate a dummy columns, i.e. $\boldsymbol{m}$, and impute by using $\bar{x}$ (better than nothing). Even better is to create a model to do the imputation by treating the x as the response (meta dude!) and the other $x's$ as the predictors. There are other ideas beyond the scope of the course!

Missingness Mechanisms

Good imputation requires you to know what's going on i.e. how measurements wound up missing. Statisticians split this space up into three categories called **missingness data mechanisms** (MDM): (1) Missing Completely at Random (MCAR) (2) Missing at Random (MAR) and (3) Not Missing at Random (NMAR) and they are defined in the following way:

MDM	$\mathbb{P}\left(\boldsymbol{M}_j \mid X_{j,\text{miss}}, \boldsymbol{X}_{-j,\text{miss}}, \boldsymbol{X}_{-j,\text{obs}}, \boldsymbol{Z}, \gamma\right) = \ldots$
MCAR	$\mathbb{P}\left(\boldsymbol{M}_j \mid \gamma\right)$
MAR	$\mathbb{P}\left(\boldsymbol{M}_j \mid \boldsymbol{X}_{-j,\text{miss}}, \boldsymbol{X}_{-j,\text{obs}}, \gamma\right)$
NMAR	(does not simplify) Note: $\boldsymbol{Z}$ denotes unobserved variables.

Let's take missing the age field as an example to explore these three conceptually.

Conceptual Understanding of the MDM's

1. MCAR means the missingness has nothing to do with any of the characteristics of the person. Is this likely? No. You can think of MCAR as a completely random computer glitch that causes holes (MCAR scenario is usually rare). Are MCAR holes imputable? No. Strategy? Use $\bar{x}$.
2. MAR means the missingness is based on other *known* predictors. Thus, imputation is essentially a regression problem (or classification problem if the x that went missing is categorical) where y is one of the features (meta!). MAR holes are the most imputable! Could you think of a reason why that would be in this example? Likely not... Can you think of another example? MAR is imputable!
3. NMAR means the missingness is based on *unknown features* and/or *the value of this feature itself that we don't know*. Is that the case here? Likely... older people tend to forget to fill in fields... also... maybe the person is hiding something (in which case we are in the pattern mixture scenario). NMAR is likely the most common and it is not so imputable... but people still pretend it's MAR and attempt to impute... likely you're doing better than just using $\bar{x}$.

Missingness in Future Measurements

As we see in the prediction dataset, there is missingness in the x^*'s i.e. the observations you wish to predict on in the future. If you are building an imputation model based on the other features, you can row-join both the historical dataframe and the x^* data frame to build the models. Also, make sure you create the $\boldsymbol{m}$ dummy variables for the same variables that had $\boldsymbol{m}$ dummy variables in the historical dataframe!

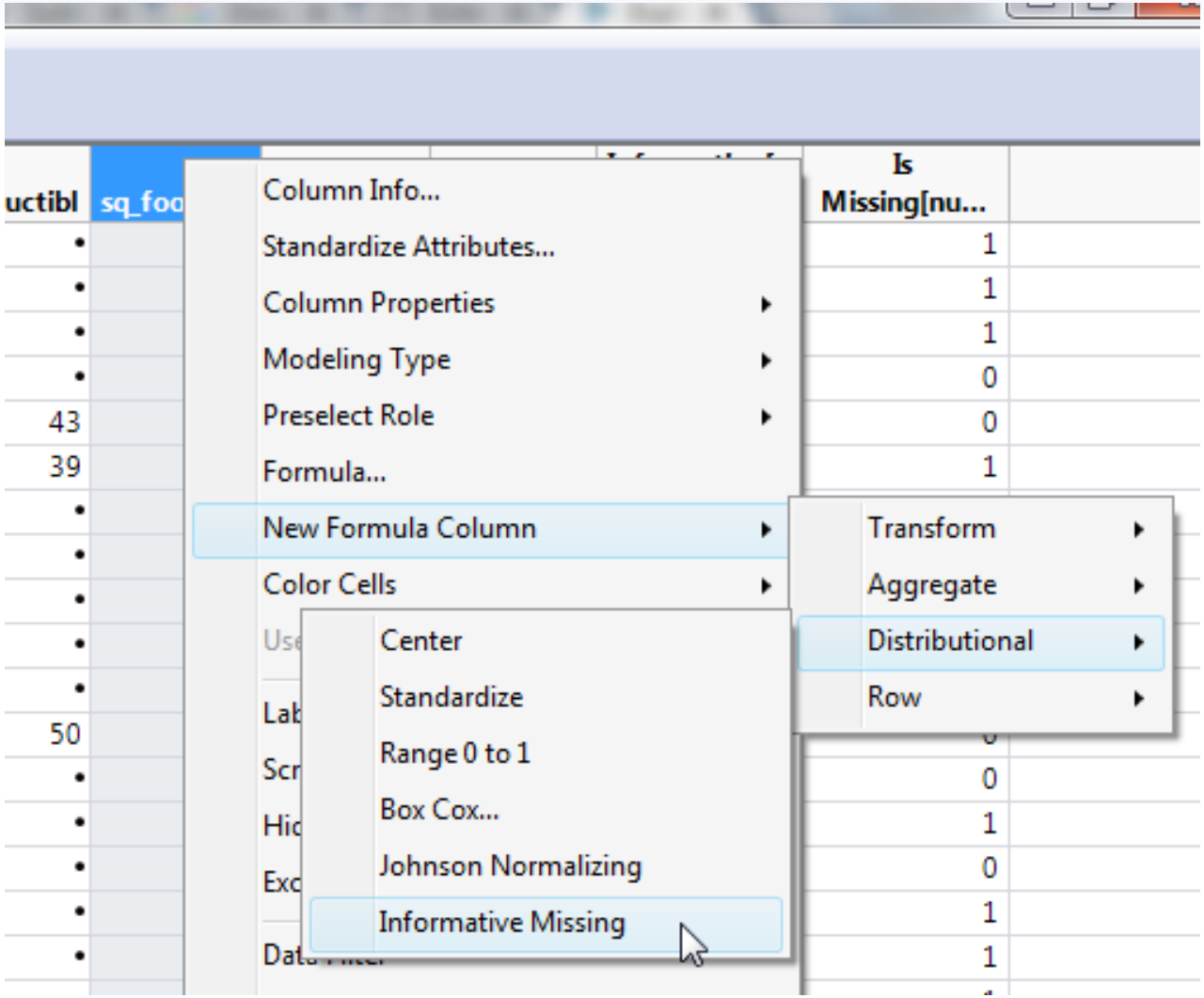
Also... obvious but must mention it... listwise deletion is not an option for the $\boldsymbol{x}^*$'s! You cannot say, "sorry I don't want to predict for this observation"!! You need to impute — it's your job!

Missingness Takeaways

Studying best ways to handle missingness can take a whole semester (some people have spent their entire careers on it), but for the purposes of this class we have a few takeaways:

- Create a missingness dummy column $\boldsymbol{m}$ and use it in models with appropriate interactions to hedge against pattern mixture models.
- Attempt to reason through missingness for each measurement separately.
- If MCAR, use $\bar{x}$ to impute holes; if MAR / NMAR, use a prediction model to impute holes (if you are supremely lazy, use $\bar{x}$).
- Delete observations that have missingness at your peril!!

How to create m and impute $\bar{x}$ in JMP



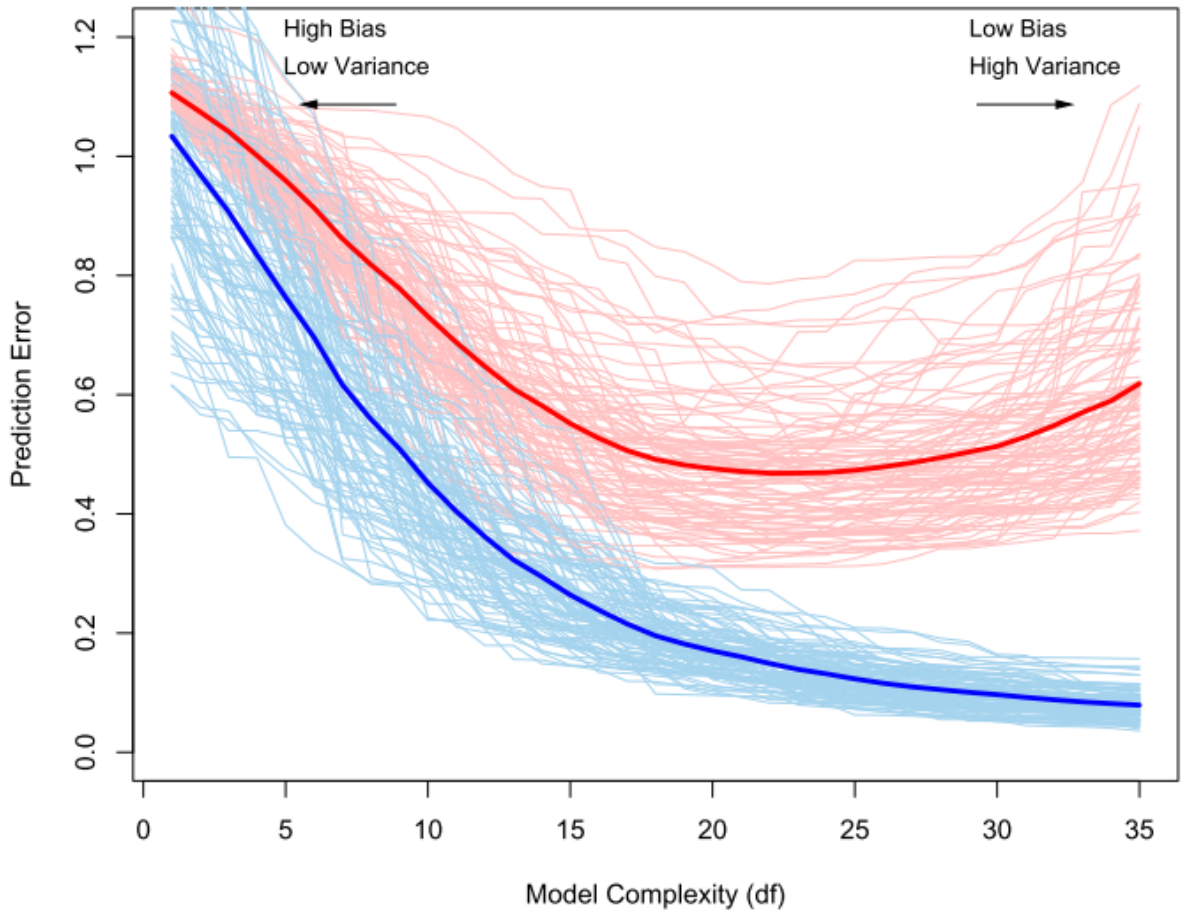
Underfitting & Overfitting

$$Y = f(x_1, \ldots, x_p) + \mathcal{E}$$

Goal of machine learning: fit f as best as possible. When we build models, we do one (or both of the following)

- We fall **short** by underfitting (usually due to too little degrees of freedom and inflexible bases). For example: if the f is a curve and we fit a line, we underfit (recall medicorp sales vs bonus regression).
- We can shoot too **long** by encroaching on and fitting / optimizing to the $\mathcal{E}$. Since $\mathcal{E}$ is independent of $x_1, \ldots, x_p$, this part of $\hat{f}$ is essentially a random fit and it is the opposite of the "data-driven approach".

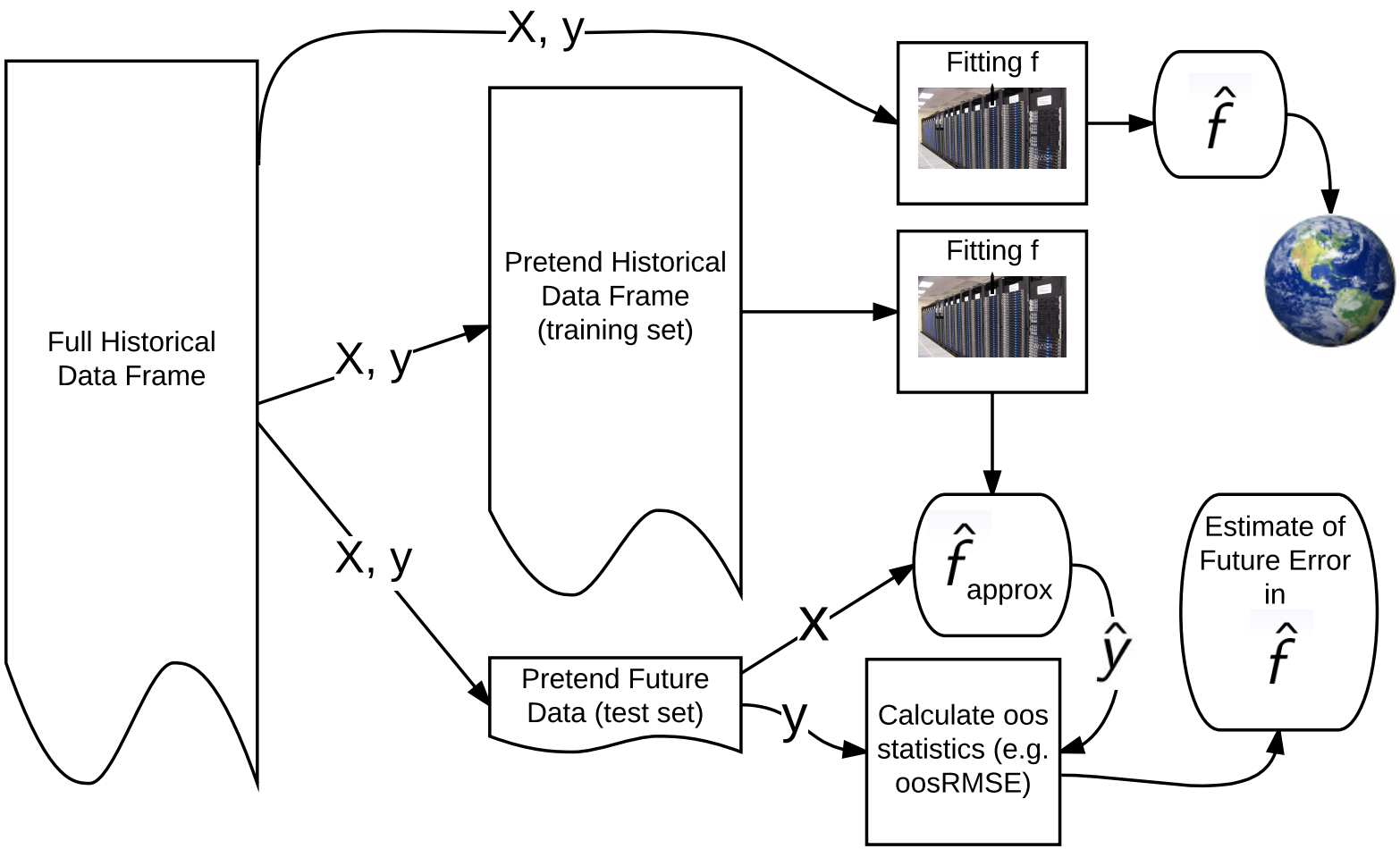
Complexity-Fit Tradeoff



Blue is in-sample fit metric and red is oos fit metric. This is Fig 7.1 from Hastie and Tibsharani (2009).

Assessment: OOS Validation

But knowing where you are on that y-axis would involve knowing the truth. We need to estimate this, so we use oos validation:



Assumptions and Tradeoffs when Splitting

We have a choice to split our dataframe into two pieces. Assuming each data point is independent (the running assumption), you should do this completely randomly. When would this assumption not be true? For example, a time series.

Additionally, we need to assume a non-stationary model relationship. So,

$$Y = f(x_1, \ldots, x_p) + \mathcal{E} \quad \text{and not} \quad Y = f_t(x_1, \ldots, x_p) + \mathcal{E}$$

where f changes with time. In essence non-stationarity is a lack of generalization and when predicting, it is a form of extrapolation.

How large should the test set be? Usual sizes are 10-30%. What's the tradeoff? If the test set is larger, then ...

1. the more accurate the assessment of generalization error would be (less variance) and
2. the less accurate the model will be since it's fitting with less data (more bias)

If the test set is smaller then, vice versa. Note: the in-sample and oos statistics are statistics! Thus, they are random!

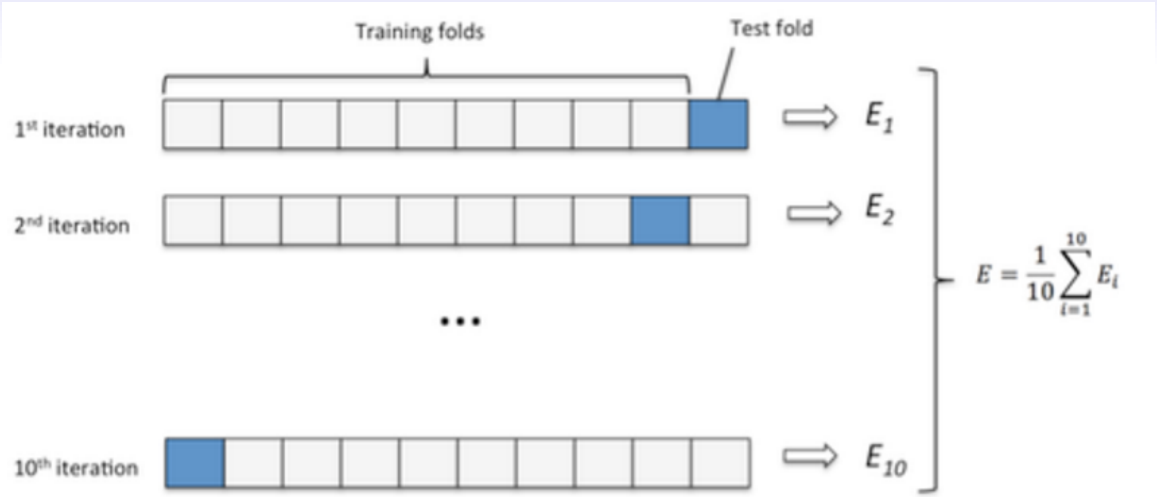
Less Randomness in the OOS Statistics

If we change the observations in the training/test splits, we will get different models and different estimates of future error. Thus, our oosRMSE was really oosRMSE conditional on the idiosyncratic split we happened to get!

We can at the very least... get rid of this idiosyncratic error by ... averaging over all training-test splits. If we have $n = 100$ and the test set is 10%, that means we only have $\binom{100}{10} = 1.73 \times 10^{13}$ split configurations to average over!

We can approximate the averaging over all splits by just taking $\frac{100\%}{10\%} = 10$ random but unique splits called **folds**. Thus, each observation is represented in the test set once (leading to a more stable estimate). This is known as **K-fold cross validation (CV)** where here $K = \frac{100\%}{10\%} = 10$ (and this procedure seems to be the industry standard).

10-fold CV



$K = 10$ is arbitrary (but remember where it was based on: the 10-30% test set recommendation). In practice, I've only used 5 or 10 fold CV.

This does not really solve any of our big problems but gives us a little boost in terms of a reduction in standard error of our generalization error estimate. That's OK; we can take all the help we can get if it's costless!!

Note 1: If $K = n$ then we the test set as one sample; this is known as "leave on out CV" (LOOCV) and it not recommended — high variance and computational cost! Note 2: bleeding edge of stats — find CI's for generalization error. Note 3: this is not the only way to reduce the variance in oos statistics but it's the one we will use in this class.

What does K-fold CV estimate?

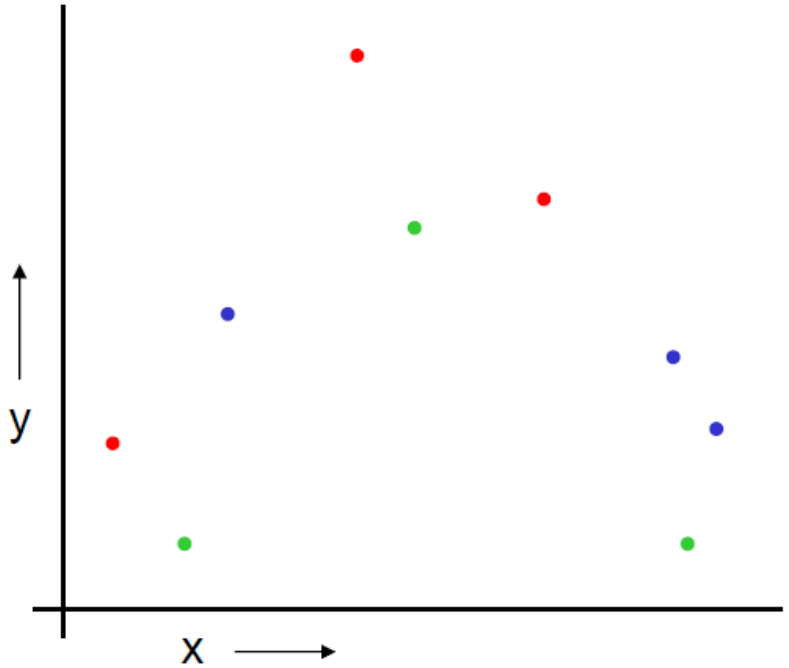
Remember... there will be different models built on each training set. So ...

- without K-fold CV, you are estimating the generalization of a model defined as the functional form and specific parameter estimates (the model and the fit).
- with K-fold CV, you are estimating the generalization of a model defined as just the functional form (the model).

I guess it depends on how you define "model". Usually, it's the latter... this is frequently ill-defined. This is a subtle point... and I won't be testing you on it!

$K = 3$-fold CV on a Linear Model Ex. 1/5

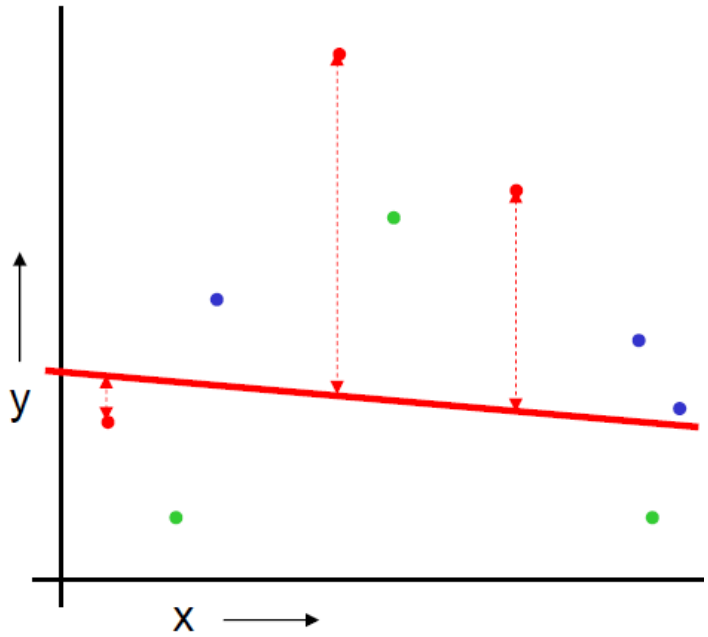
Imagine the following data $n = 9$ where we are fitting a response by one feature (ignore the colors):



Imagine we choose a linear model.

$K = 3$-fold CV on a Linear Model Ex. 2/5

In the first fold, the red is left out and thus we fit a line to the blue and green points:

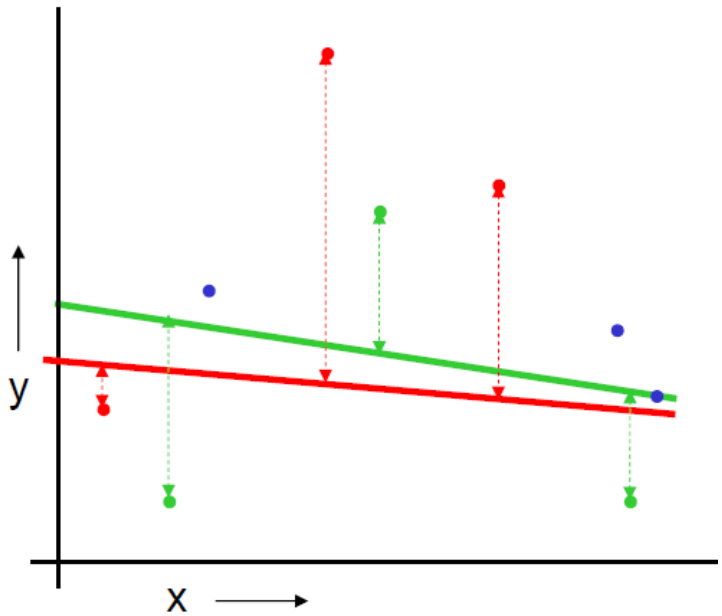


Then we calculate the residuals to the red points (the test set in this fold) and calculate

$$SSE = (2 - 2.2)^2 + (3.8 - 2.1)^2 + (3.5 - 2.05)^2 = 5.03$$

$K = 3$-fold CV on a Linear Model Ex. 3/5

In the second fold, the green is left out and thus we fit a line to the blue and red points:

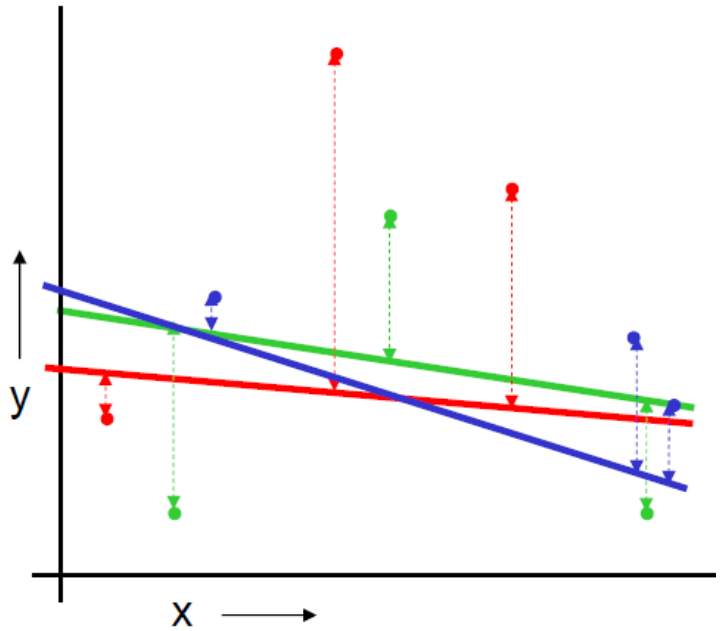


Then we calculate the residuals to the red points (the test set in this fold) and calculate

$$SSE = (1.2 - 2.3)^2 + (3.4 - 2.25)^2 + (1.3 - 2.2)^2 = 3.34$$

$K = 3$-fold CV on a Linear Model Ex. 4/5

In the third (last) fold, the blue is left out and thus we fit a line to the green and red points:



Then we calculate the residuals to the red points (the test set in this fold) and calculate

$$SSE = (2.35 - 2.3)^2 + (2.4 - 1.5)^2 + (2.2 - 1.4)^2 = 1.45$$

$K = 3$-fold CV on a Linear Model Ex. 1/5

Then we aggregate all oos results together (SSE's are additive) and we can compute a final oos statistics e.g. the oosRMSE:

$$oosRMSE = \sqrt{\frac{5.03 + 3.34 + 1.45}{9}} = 1.045$$

Limits of JMP / Intro to R's MLR Package

And... JMP can't do K-fold CV! (Except in one limited case which doesn't help us right now). But of course R can do it... [R Demo with MLR]

Validating Multiple Models

Let's look at a few models for the White Wine data with no validation (but no cross-validation). Here the response is wine quality as measured by professional raters and features are 11 features (e.g. acidity, sugar, pH and alcohol content).

A plain linear model

B 3-degree polynomials for all features

C 3° polynomials and all 1st order interactions

D 3° polynomials and all interactions up to 3rd degree (AKA 2nd order)

E 3° polynomials and all interactions up to 4th degree (AKA 3rd order)

F 3° polynomials and all interactions up to 11th order

[JMP col validation... fit all models with validation ... save prediction formula cols... analyze model... model comparison... complexity tradeoff illustration] Conclusions? Model C looks the best. Note: another popular assessment metric besides oosRMSE is oosAAE which is just average absoute value difference. Strange ... given that linear models optimize for squared error. Show Demo with MLR w/ 10-fold CV What *precisely* did I do that wasn't legal?

A Possible Spin on Validation

Recall the proposal from last class:

1. Split dataframe into training and test.
2. Build model A on training.
3. Predict using the test set.
4. Calculate estimate of future generalization error of model 1.
5. Build a different model B on training.
6. Predict using the test set.
7. Calculate estimate of future generalization error of model 2.
8. ... steps 5-7 for model 3
9. ... steps 5-7 for model 4
10. ...
11. ... steps 5-7 for model M
12. Pick whichever model has better generalization error.

This is a form of **Model Selection**. What was wrong with it?

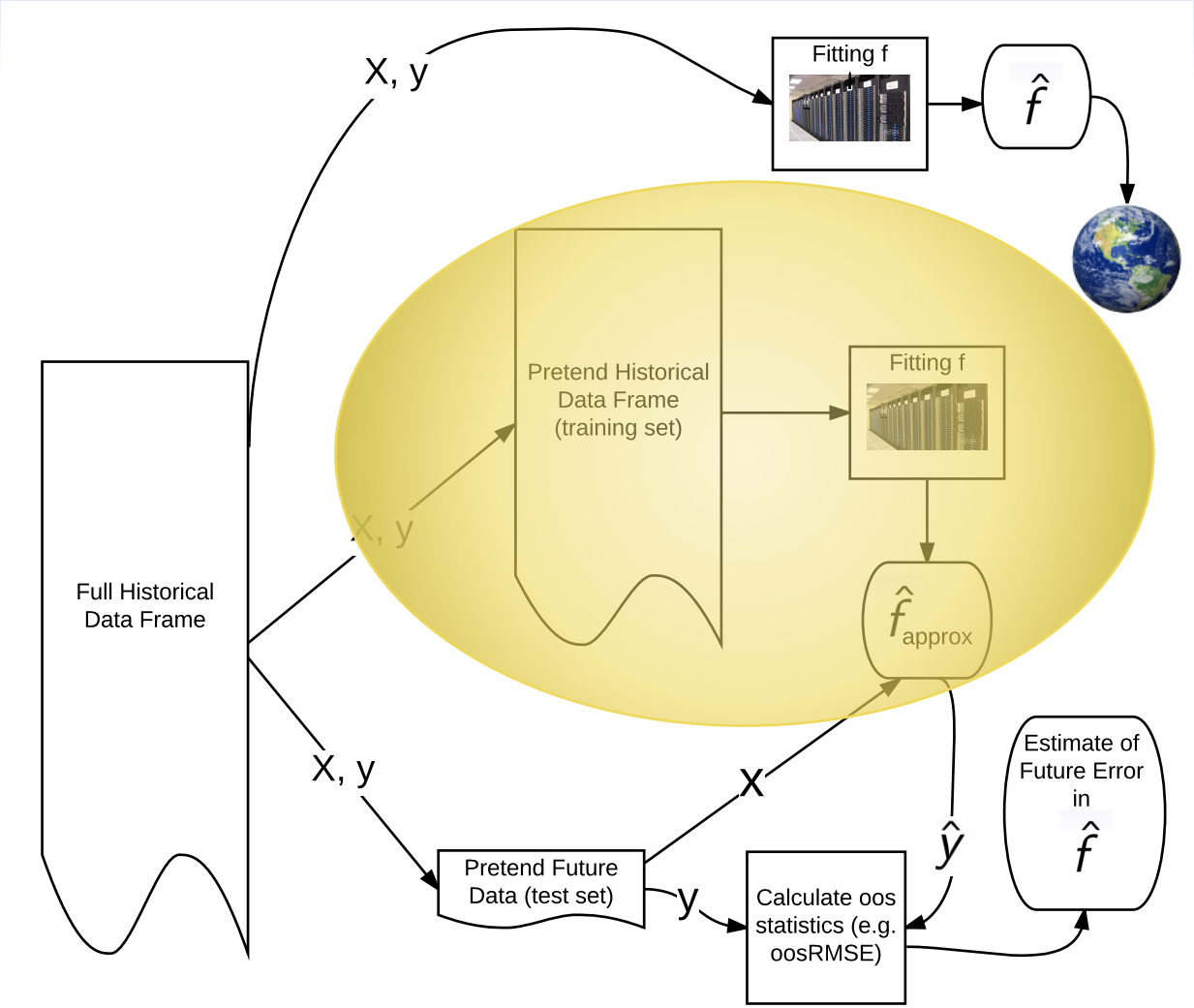
Looking into the Future is Not Legal

The oos validation is only valid if...

you treat the test set as a lockbox. Once you open it up, that's it! And we opened it up M times!

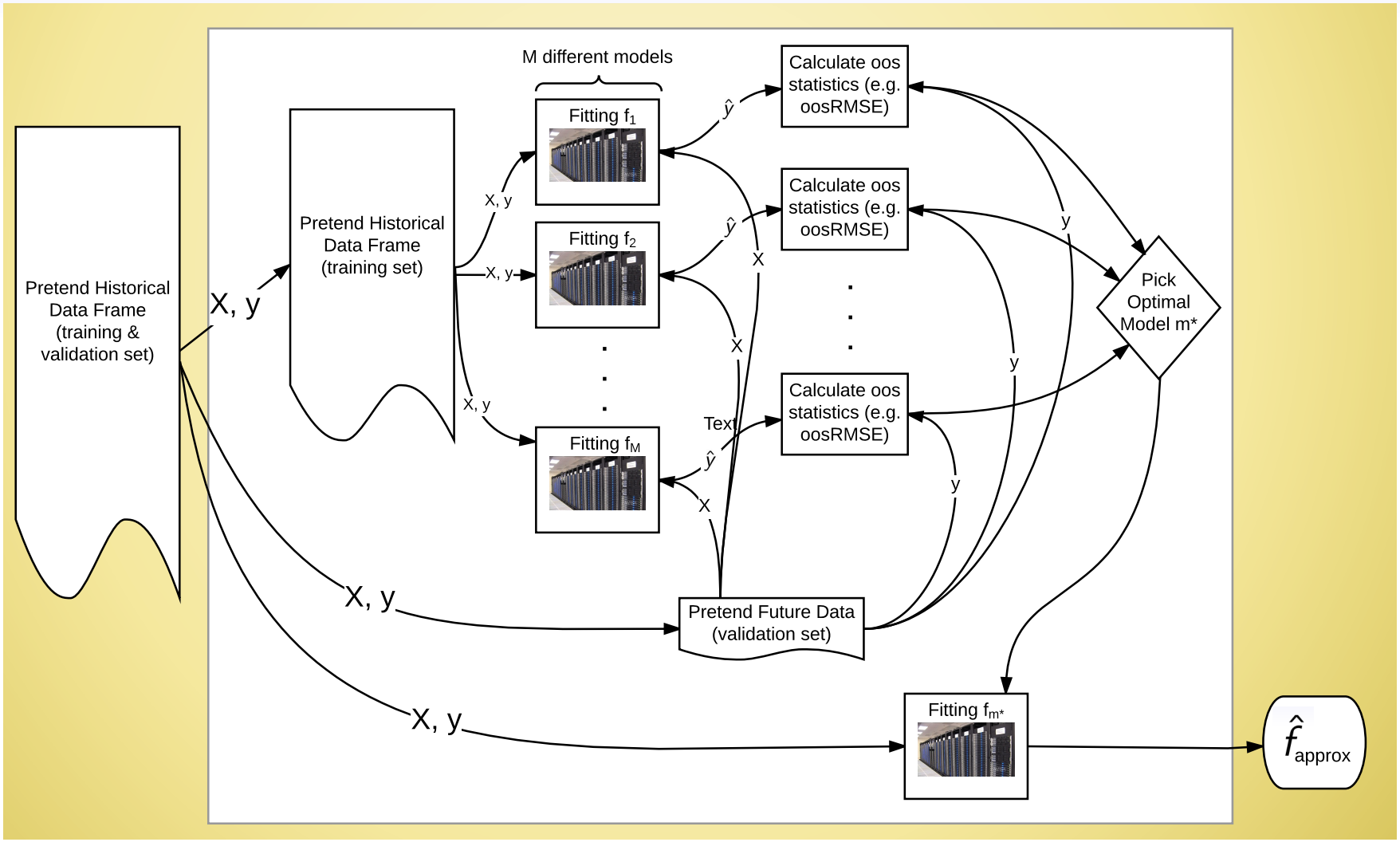
This is indeed a **Model Selection** procedure but ... our estimate of future generalization error is invalid. How can we do both?

In One Fold Let's Focus on the Training Set

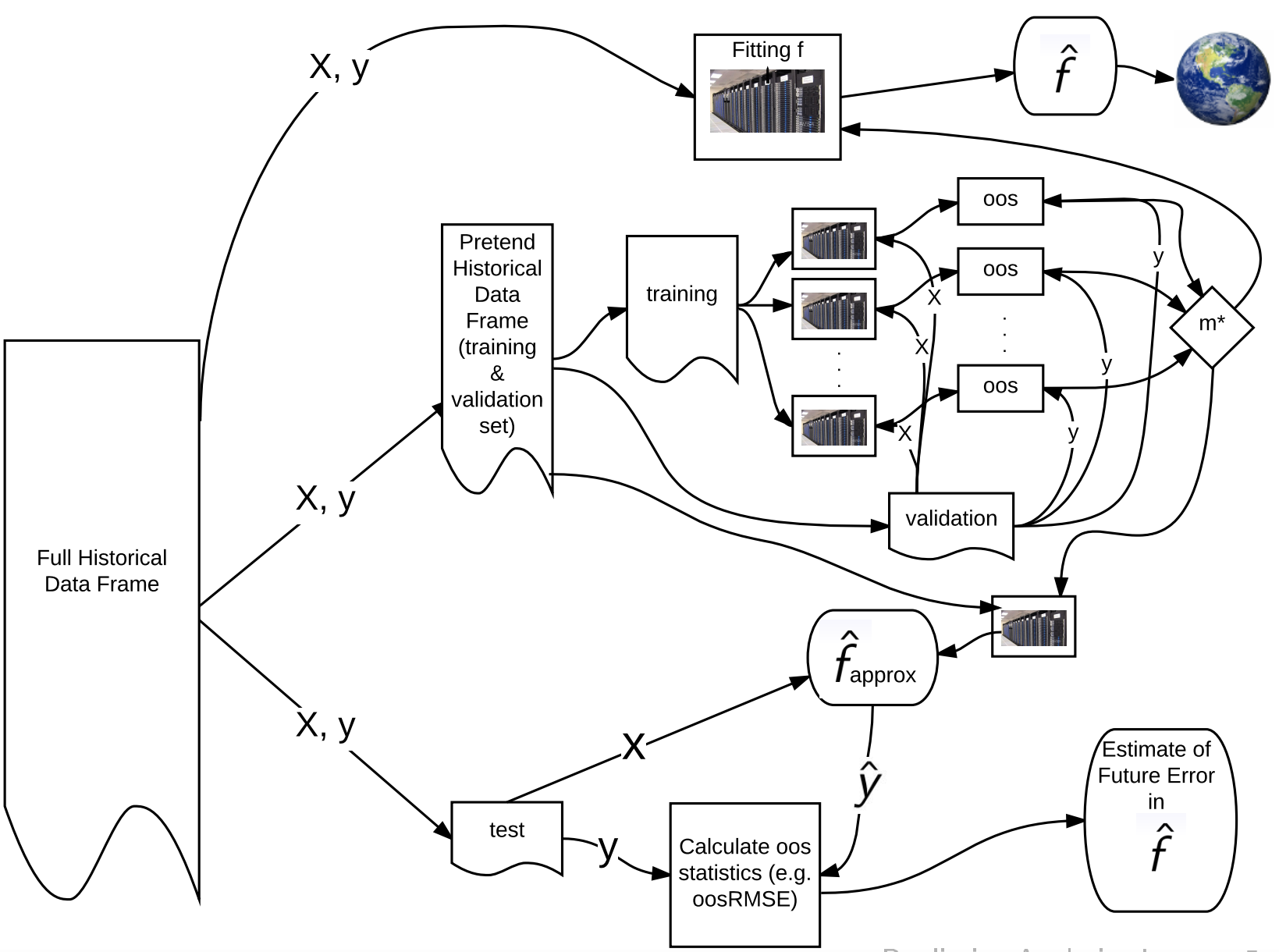


This procedure was completely valid as long as we did not touch the test set, right? As long as we operate only within the training set... we're OK!

Training ⇒ Training & Validation



3-way Splitting: The Full Picture



Why Training-Validation-Test Splitting?

- Training set: provides fits for many models where overfitting is "okay"
- Validation set: provides out-of-sample validation for each of the models. If the models are overfit, they will get wrecked at this stage.
- Test set: this lockbox provides a layer of security against overfitting within the training-validation union set.

Just like in the previous 2-way training-test split, you can overfit the training, get killed on the test set and be stuck. How could you similaryl overfit here? Be careful of optimizing to the validation set. Models $1, 2, \ldots, M$ should still be reasonable thought-through models.

Sizes of Training-Validation-Test Splits

Previously, we had the test set being 10-30%. This recommendation remains. Thus, the training-validation sets together should make up 70-90% of which a portion is the validation set (which is like a test set) and should be 10-30% of the total. Thus we arrive at proportions like 50-25-25 or 50-30-20 or 70-20-10. There is no exact guidance here.

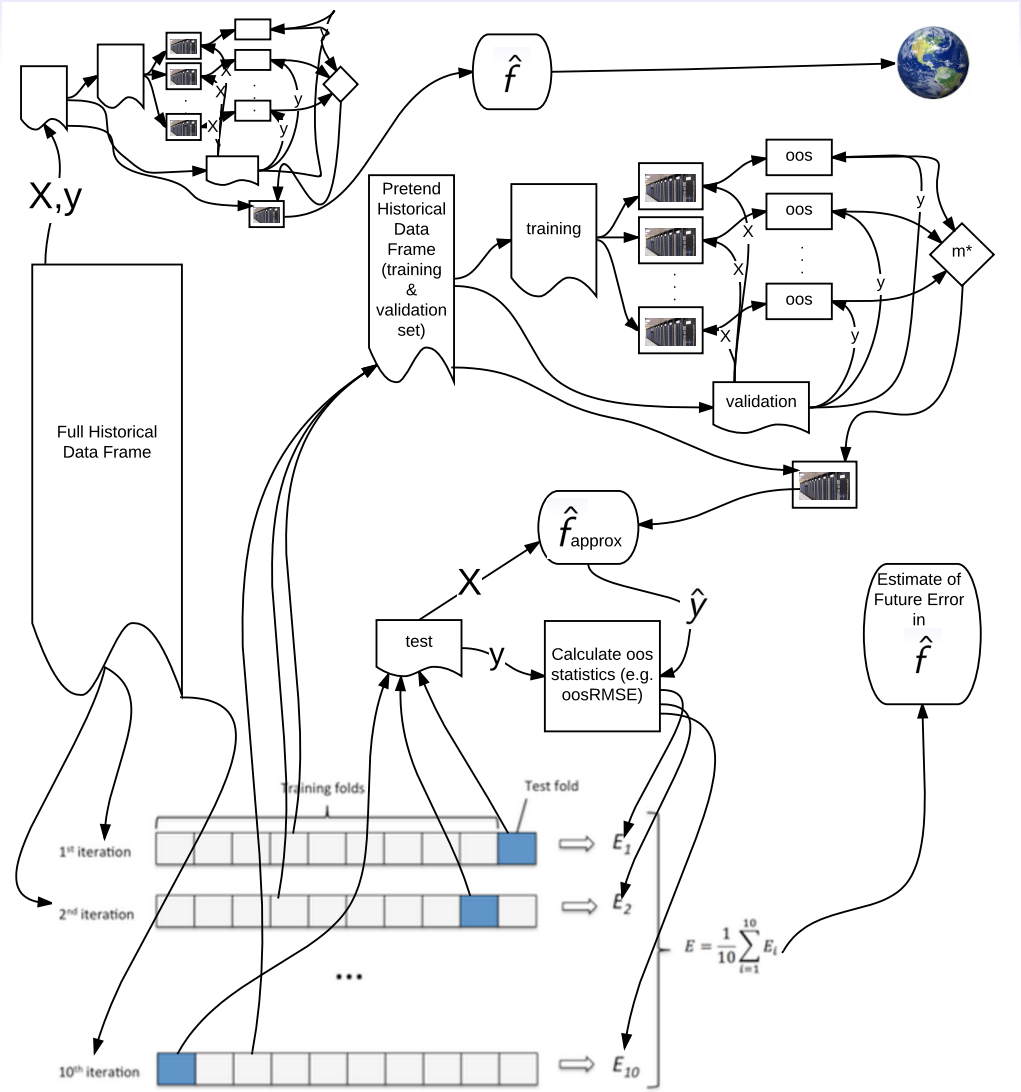
The same tradeoffs apply to the test set size but now we have new tradeoffs for the training set size vs. the validation set size:

- The larger the training set, the better the fit of the model (less bias) but the more variance in its assessment versus its peers
- The smaller the training set, the worse the fit of the model (more bias) but the less variance in its assessment versus its peers

Back to Wine...

We can do a single 3-partition split in JMP using the validation column... then data filter... then fit the 6 models again... then use model comparison to select best model ... then undo the filter ... then use model comparison again to find our test set error (the guess of the generalization error)... then build the full model for public consumption.

Can you CV this 3-Split Procedure?



Yes. It is called "nested resampling" and $K = 10$ fold CV is illustrated here. But... what are you now evaluating?

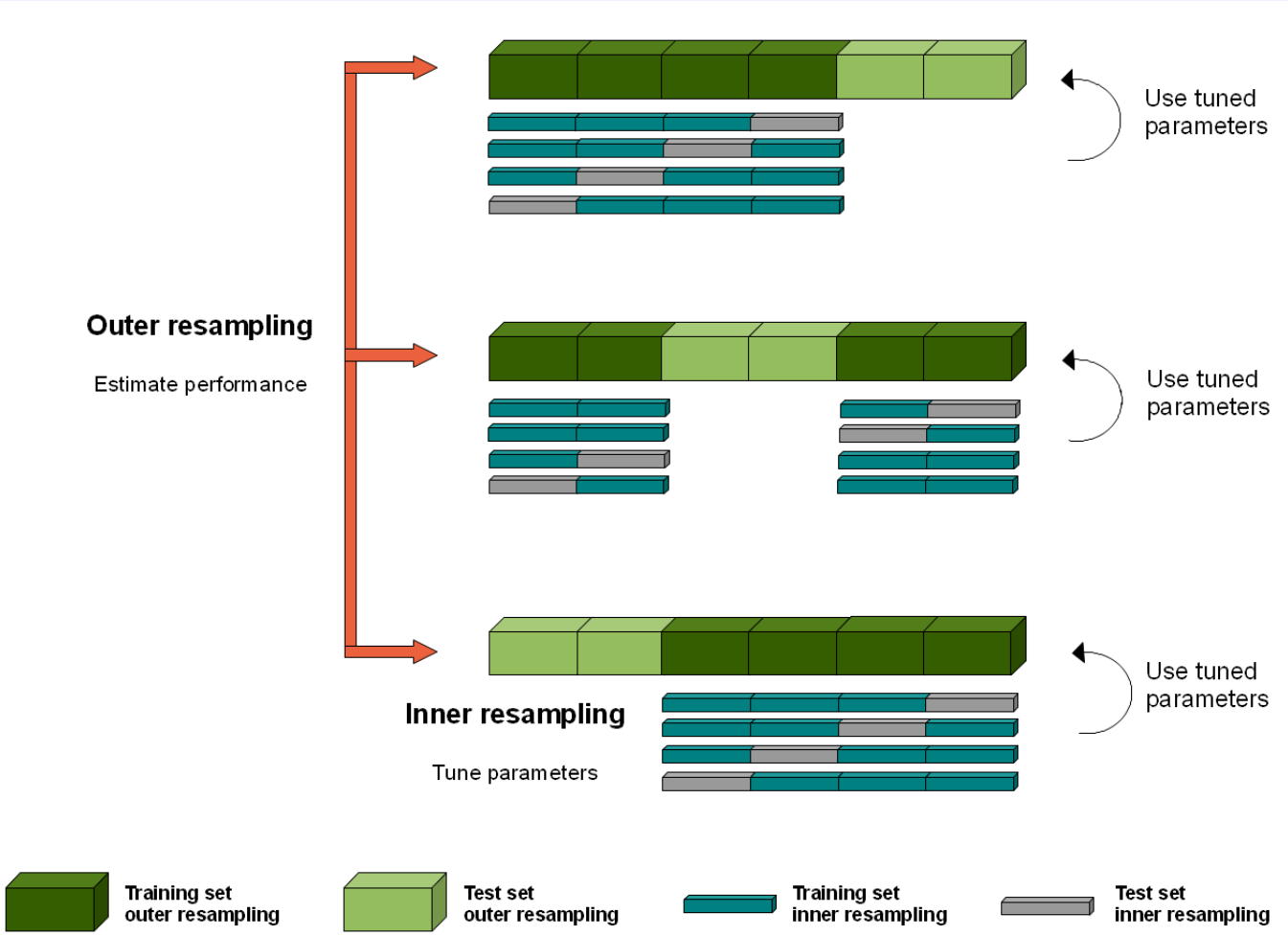
Without CV, it was just model m^*. But here m^* varies with the 10 folds! You are testing the entire procedure i.e. given models $1, 2, \ldots, M$, pick the best one and ship it. How well you do in the future is estimated by the CV test set.

Not the First Means for Model Selection

CV with nested resampling is not generally done in the way it was described here as a means to evaluate a model selection procedure. Beyond scope of course: it is usually done to compare tuning settings in a non-parametric machine learning algorithm. We will see what this means next class.

Neither R nor JMP (to my knowledge) can do this out-of-the-box. In R, even with MLR, you have to program it (MLR uses it for tuning an algorithm). Maybe I will write to them?

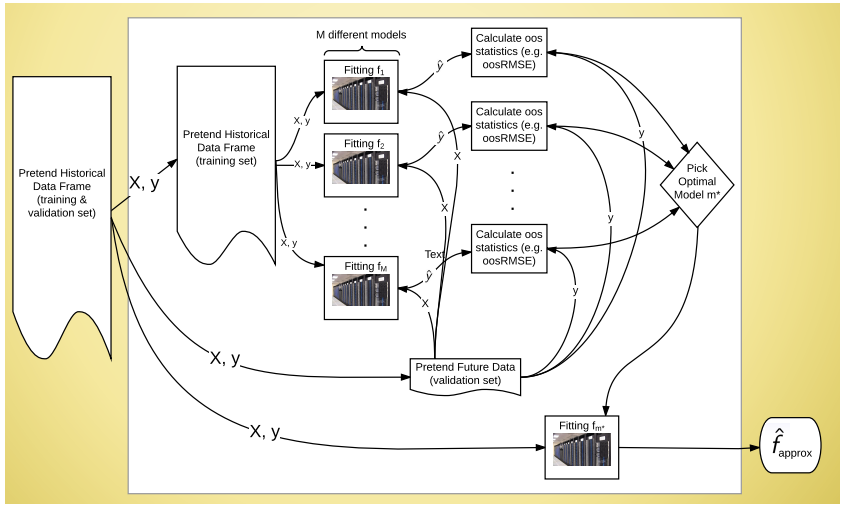
Nested Resampling for Tuning



(from MLR's tutorial website).

3-way splits for model selection & evaluation

Forget CV for a moment since it complicates things ... the "inner split" consisted of the training-validation. We used this to "select" a model based on lowest oos error on validation set (RMSE, or highest R^2).



What's the cost? We had to use only 70-90% of the data (save the test set) to build the model. But what's the bigger cost?? You had to provide models $1, 2, \ldots, M$!! This is no trivial task. So far I've been making them up! I've been using elebaorate linear models which pivoted from parametric to non-parametric with the addition of curves and interactions. Forget finding the best model... how do we know we even employed good candidates?? We don't!

Modeling Framework Refresher

Recall the general regression model:

$$Y = f(x_1, \ldots, x_p) + \mathcal{E}$$

A couple lectures ago, we made the parametric assumption that:

$$Y = s(x_1, \ldots, x_p; \theta_1, \ldots, \theta_\ell) + \tilde{\mathcal{E}}$$

where the $\tilde{\mathcal{E}}$ term now includes the previous $\mathcal{E}$ plus $f - s$, the misspecification error. The parametric model s we employed was the linear model and the θ's we called β's:

$$Y = \beta_0 + \beta_1 x_1 + \ldots + \beta_p x_p + \tilde{\mathcal{E}}$$

Last lecture, we started adding interactions and polynomials (as well as other transformations e.g. log which we did not cover). This was a means of "expanding" the feature set "visible" to the model using "derived" features:

$\{x_1, \ldots, x_p\} \Rightarrow \{x'_1, \ldots, x'_{p'}\}$ where $p' > p$ and maybe much, much greater.

"Non-parametric" Linear Regression

Once we expand this feature set, we can now fit a larger linear model:

$$Y = \beta_0 + \beta_1 x_1' + \ldots\ldots\ldots\ldots\ldots\ldots + \beta_p x_{p'}' + \tilde{\mathcal{E}}$$

Given more degrees of freedom with this expanded feature set allows the linear model to fit more complicated real-world functions. This is essentially a means of doing non-parametric parametric modeling (it's oxymoronic). It's technically parametric but conceptually it's non-parametric since we don't have our parametric benefits: parsimony, inference nor interpretation. Hopefully $\tilde{\mathcal{E}}$ will be close to $\mathcal{E}$, the irreducible noise.

Back to our problem... we can curb overfitting by ... using 3-way split oos validation but we need to select good models... how to do so? One approach is termed **subset selection methods**.

Stepwise Regression

First we expand the feature set from $\{x_1, \ldots, x_p\} \Rightarrow \left\{x'_1, \ldots, x'_{p'}\right\}$. Then we attempt to find the "best" model consisting of a subset of these features. However there are $2^{p'}$ possible models. For $p' = 20$ that's about 1,000,000. So we try to find a model *close* to the optimal using a "heuristic" (a rule of thumb that seems to generally be useful).

That heuristic is called **stepwise** model construction. We begin with **forward stepwise** model construction:

1. Find the "best" feature from the list of expanded features.
2. Find the "next best" feature from the remaining expanded features.
3. Repeat step 2 until you believe you are overfitting.

Estimating Overfitting (again)

If you choose the feature to give you the best in-sample R^2, you will eventually take all the features (until $n = p + 1$) and you will get $R^2 = 100\%$. We need a metric to tell us when we may be overfitting and halt at that moment. Here are a few:

1. oosRMSE (keep a holdout set and quit when this starts increasing)
2. Only include a variable if its t stat (or partial F stat) is significant
3. Use $AICc$.

$$-AIC = 2\ell\left(\hat{\beta}; \boldsymbol{y}, \boldsymbol{x}\right) - 2p$$

The first component (the log-likelihood) represents in-sample fit. $\ell()$ is like R^2 though... as the fit gets closer to the points, the likelihood goes to 1 (and the log likelihood goes to 0). The $2p$ term is a reality check. If you have more features, you are going to overfit. So each additional feature must be justified in terms of the increase in log-likelihood. Thus, good models maximize $-AIC$ (i.e. minimize AIC).