Nested Models and Simulated Out-of-Sample Data

Nina Zumel
Win-Vector, LLC

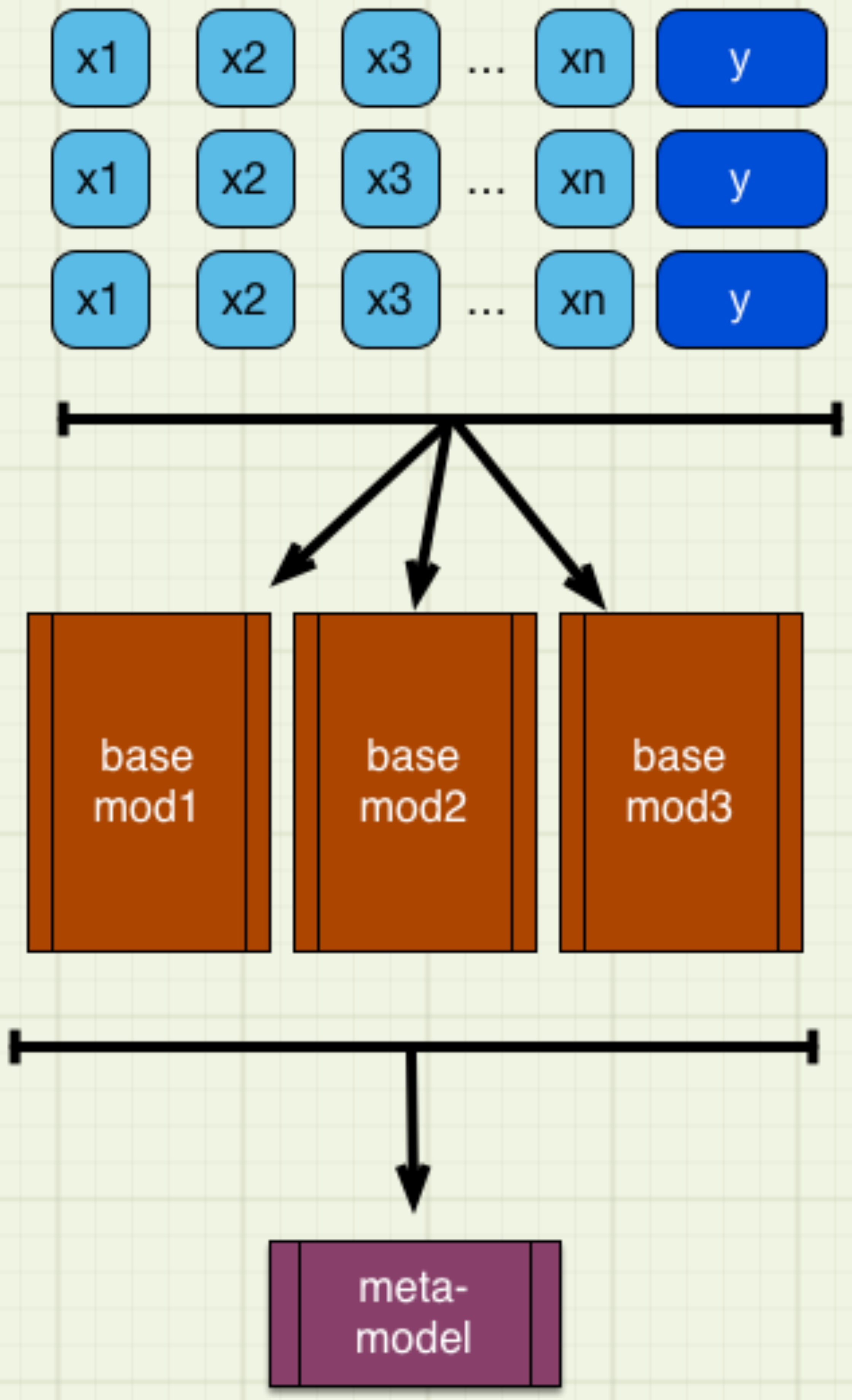
Outline

- What are Nested Models
  - Why do we care?
  - Types of Nested Models
- Examples of What Can Go Wrong
- Solutions
- Conclusion

What do We Mean by *Nested Models*?

- Use data to train several base models to predict outcome

- Use a "meta-model" to combine the base models' predictions into an overall prediction



Win-Vector LLC

Pro and Con

- **Pro**: Ensembles and Stacked models can improve performance over single models
    - Diversity of learning biases
- **Con**: They can also produce inferior overall models
    - We'll cover ways to avoid this in this talk

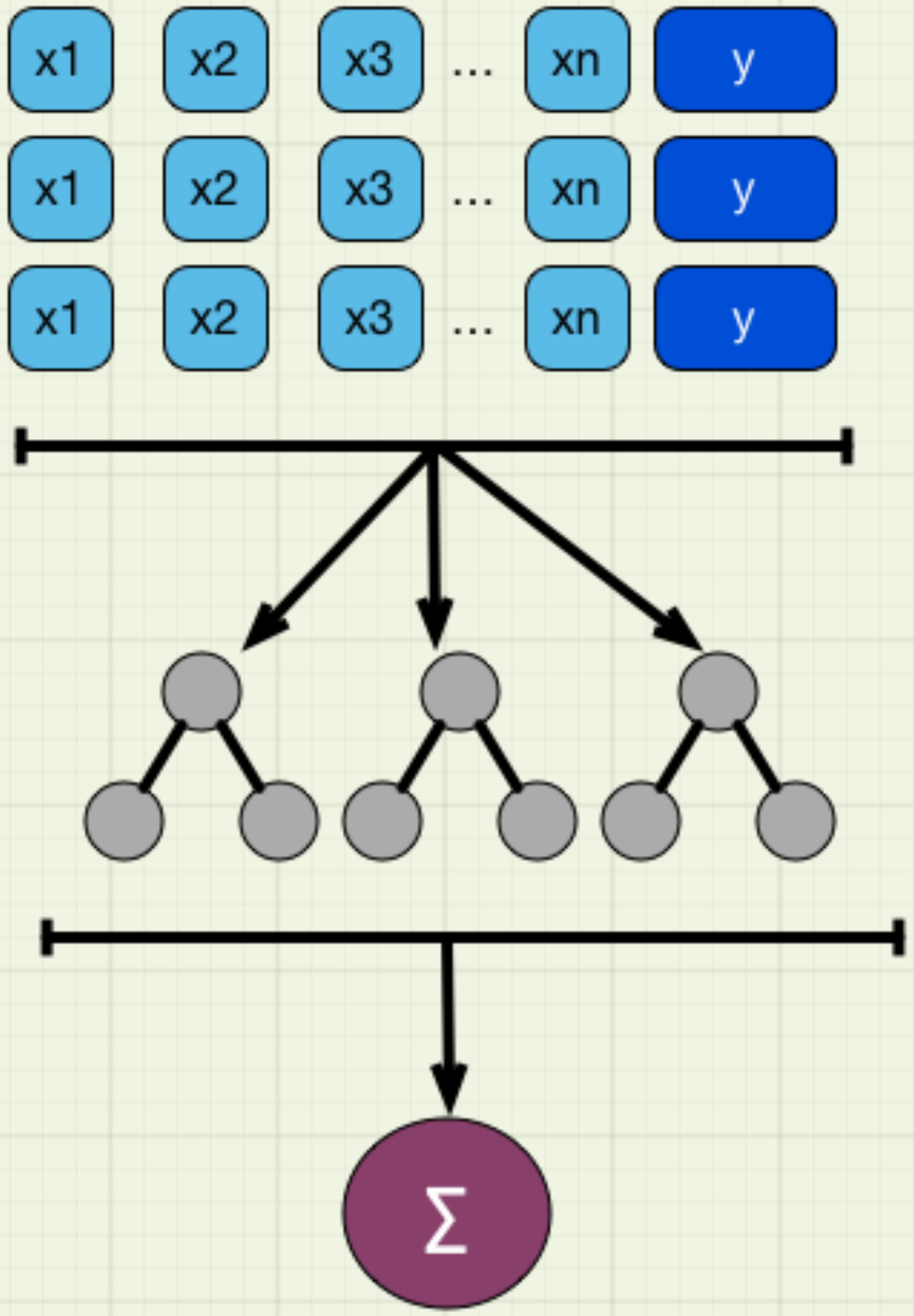
- Any nested model can introduce undesirable bias, to a greater or lesser degree
- But nested models are useful — and they are already everywhere
- This is basic statistical material, but it takes some care to properly apply it to modern data science workflows

Win-Vector LLC

Types of Nested Models

Ensemble Learning

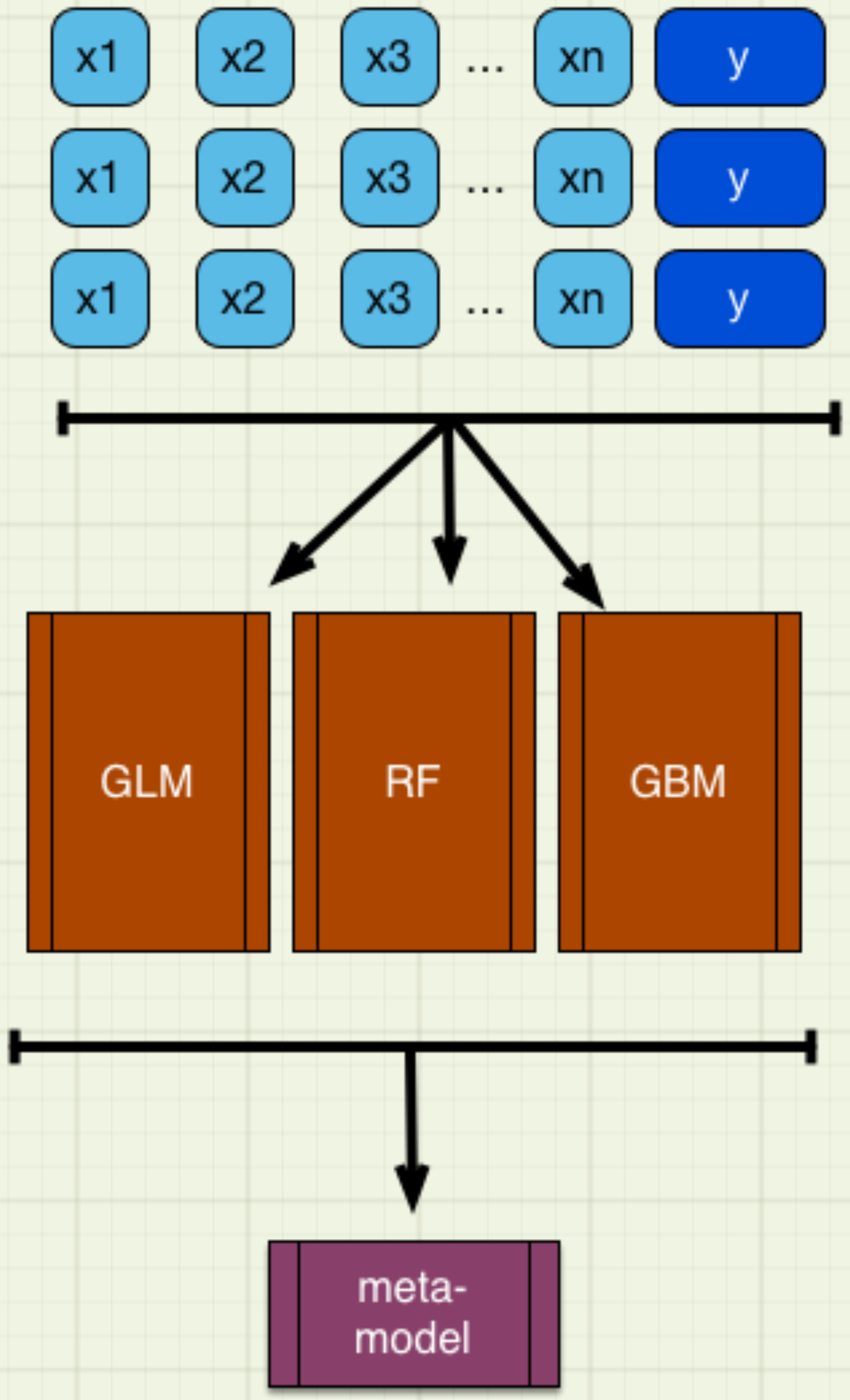
- Boosting (AdaBoost 1996, Gradient Boosting 1999)
- Bagging (1996)
- Random Forest (1995)
- Several diverse (usually) low-complexity learners that vote on outcome (sum, weighted sum)
- Different algorithms: different ways of getting diversity



Win-Vector LLC

Stacked Models

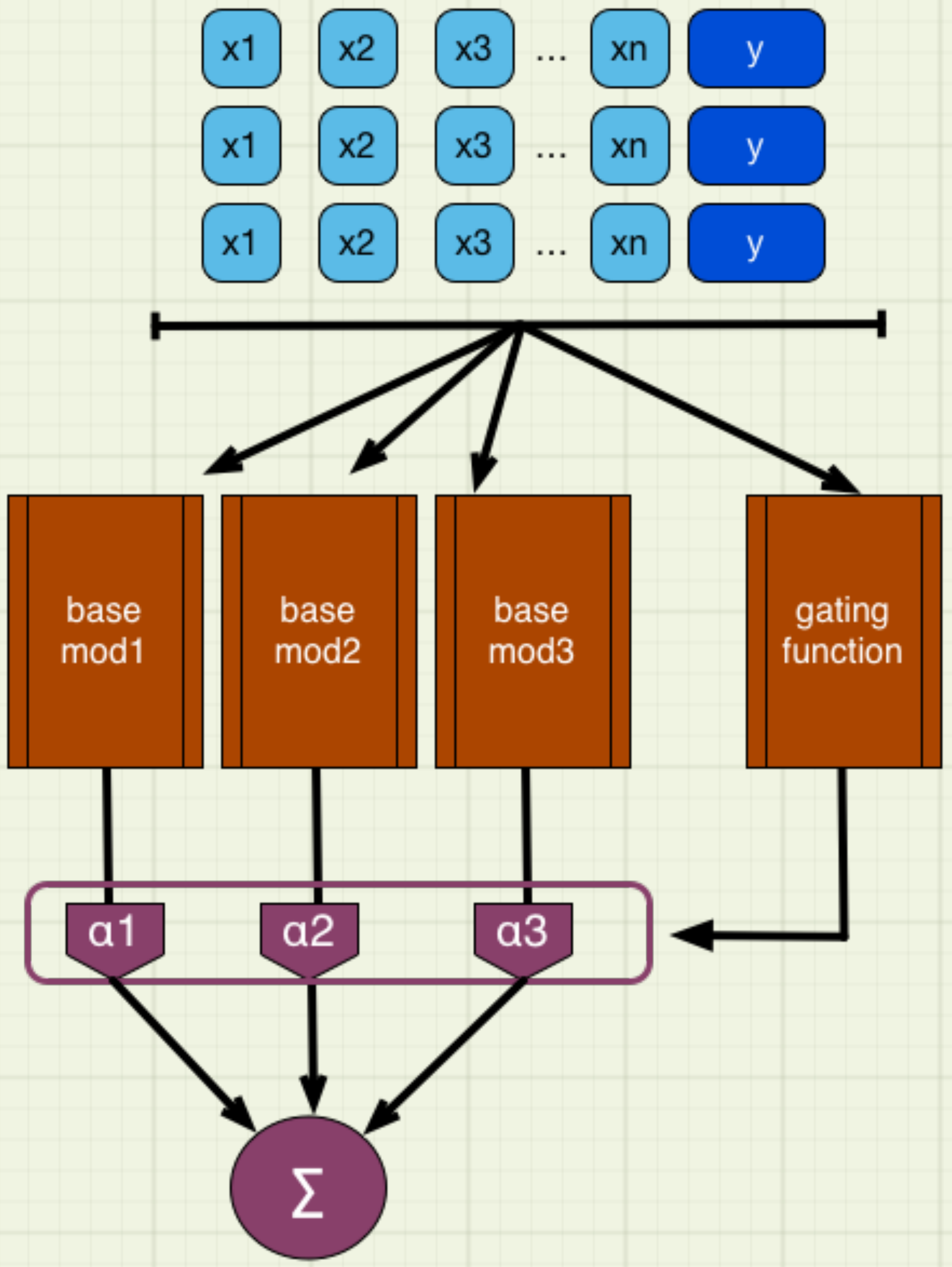
- Wolpert, 1992; Breiman, 1996
- Super Learner (van der Laan, 2007)
- Extension of hyper-parameter tuning
- Strong learners, higher-complexity meta-model



Win-Vector LLC

Mixture of Experts

- Jacobs, 1991
- Data fusion, Sensor fusion
- Base models combined by weighted sum
- Weights are per-example (gating function)
- Often neural-net based



Other "nested models"

Any pre-model-fitting task that uses knowledge of outcome is a nested model

- Variable treatment
- Hyperparameter tuning
- Variable selection/stepwise methods
  - https://CRAN.R-project.org/package=vtreat
- Y-aware dimension reduction
- Y-aware scaling
  - http://www.win-vector.com/blog/2016/05/pcr_part2_yaware/

Example 1: Stacking

- `H2OEnsemble` (R)
  - https://github.com/h2oai/h2o-3/tree/master/h2o-r/ensemble
- Manages training of base models and meta-model to combine them
- Alternatives: `SuperLearner`, `MLR` (both in CRAN)

(Image: H2OEnsemble github)

Win-Vector LLC

The Prediction Task

- Predict whether a seismic event will occur in the next time interval
- 2584 rows, 19 features, ~6.5% target class prevalence
- http://archive.ics.uci.edu/ml/datasets/seismic-bumps

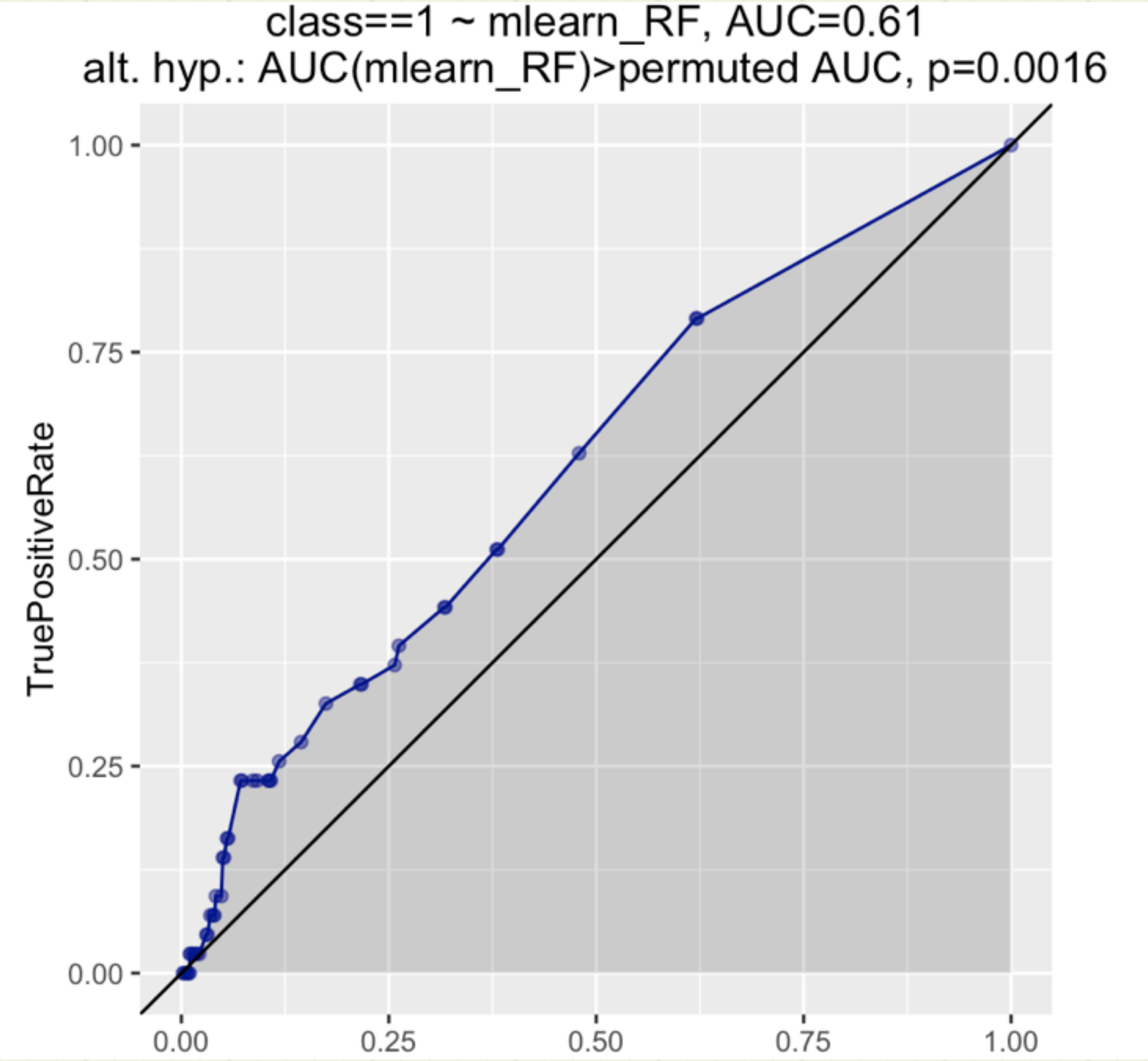
How It Can Fail

- Base learners:
  - Random Forest
  - Gradient Boosting
  - Logistic Regression (Elastic Net)
  - Deep Learning NN

- Meta-learner: Random Forest

- Base learners: AUC in 0.65-0.71 range (test set)

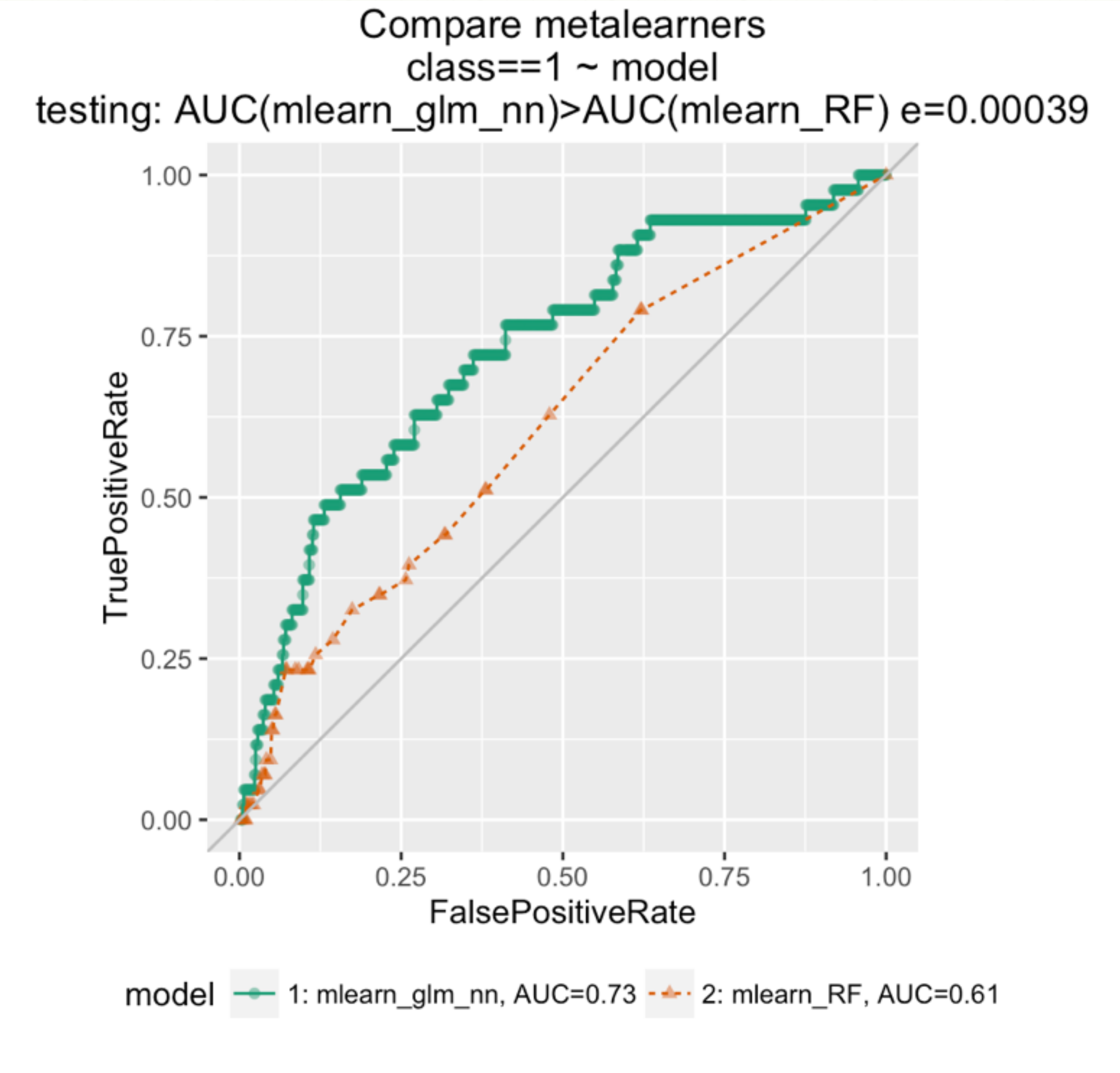
- Ensemble: AUC 0.61 on test



Win-Vector LLC

Issue: Meta-model complexity

- High-complexity meta-model can overfit
  - "Explain" unexplainable variance in training data
- Low-complexity meta-models have bounded excess generalization error
- Base models correlated
- Prefer non-negative weighted sum, or ridge regression
- AUC(RF) = 0.61; AUC(glm) = 0.73



Win-Vector LLC

Example 2: Effects/Impact Coding

- For categorical variables with many levels.
  - K levels = K-1 indicator vars
- Re-encode the categorical variable as a few numerical variables.
- Impact code: “base learner”

ZIP	PriceK	…	SoldInMon
94127	$1,122	…	Yes
94117	$1,296	…	No
94127	$1,370	…	Yes
94118	$1,325	…	No

Win-Vector LLC

The Prediction Task

- Predict whether a listed house will sell within a month
- Input: ZIP, list price
- 1000 houses, 100 zip codes
- P(sold) weakly dependent on price
- P(sold) independent of ZIP

Win-Vector LLC

Impact Model

ZIP	P(SoldIn Mon)	Impact
94117	0.6	0.2
94118	0.34	-0.06
94121	0.16	-0.24
94127	0.72	0.32
94132	**1.0**	**0.6**
…	…	…
Overall	0.4	0

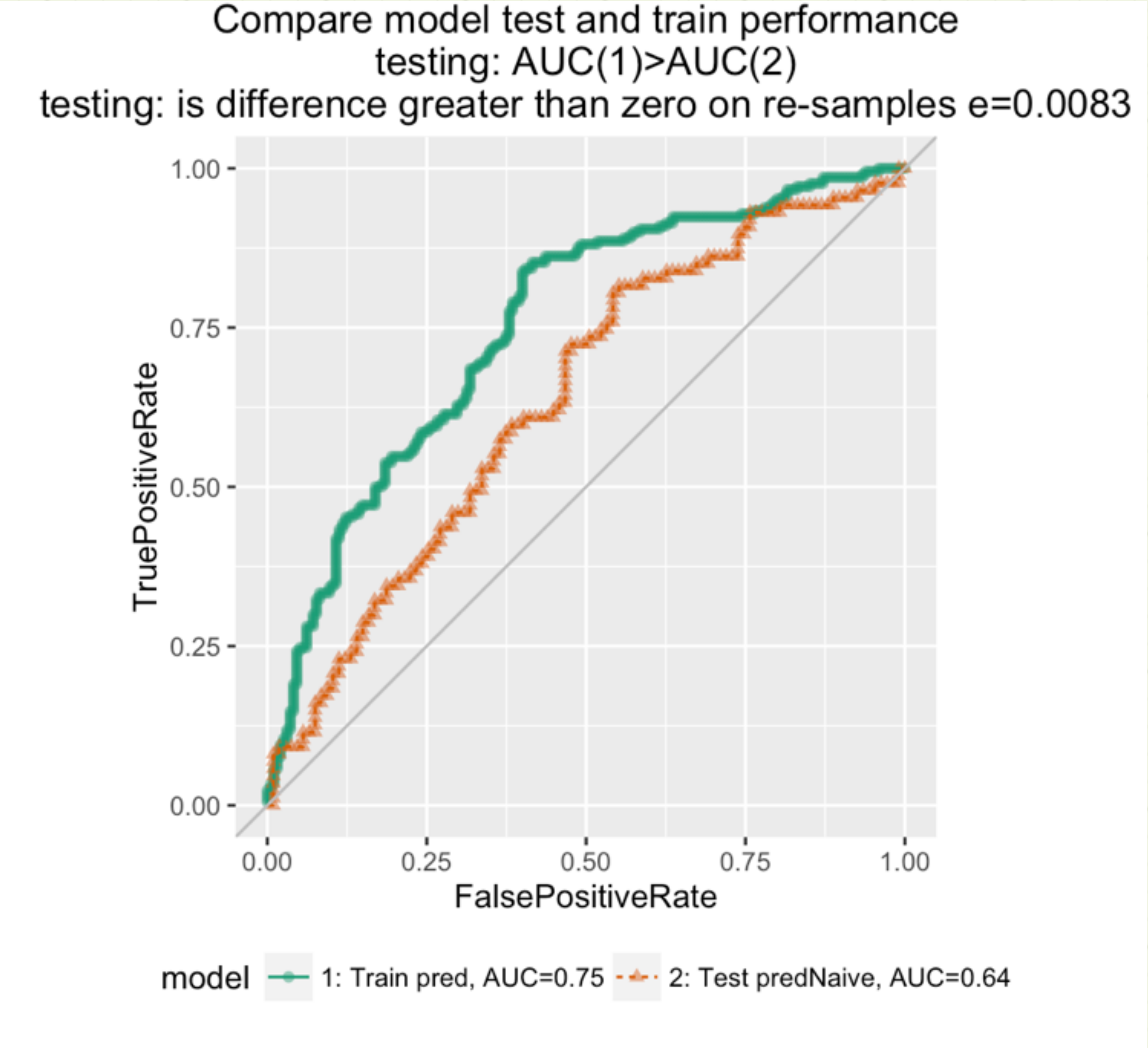
ZIP	N_SoldIn Month	N_NotSoldI nMonth	LogDiff	IsRare
94117	60	40	0.41	No
94118	68	132	-0.66	No
94121	8	42	-1.6	No
94127	108	42	0.94	No
94132	**1**	**0**	**1E+06**	**Yes**
…	…	…		

(Alternative model: count sold/not sold)

Win-Vector LLC

How It Can Fail

- Use training data to effects code
- Train a logistic regression model to predict
- AUC(Train) = 0.75
- AUC(Test) = 0.64
- Overfit



Issue: Nested Model Bias

```
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)   9.01213    1.06276   8.480  < 2e-16 ***
## log(price)   -0.73513    0.08693  -8.456  < 2e-16 ***
## zip_impact    0.98732    0.12683   7.785 6.99e-15 ***
```

- Effects model can memorize the training data
  - Rare levels
  - “Houses in 94132 always sell in a month”
- Full model may overestimate the value of effects-coded variable
  - Overfit

Hidden Degrees of Freedom

- High-complexity impact model posing as a low-complexity single variable.
- Same thing can happen when naively stacking models

True Complexity of ZIP variable

```
mod1 = glm(sold~zip, data=train,
family=binomial)
## [1] "Complexity (degrees of freedom): 96"
## [1] "% deviance explained: 20.8%"
## [1] "p-value on Chi^2 Test on model:
0.0725"
```

Apparent Complexity of ZIP (zip-impact)

```
mod2 = glm(sold~zip_impact, data=train,
family=binomial)
## [1] "Complexity (degrees of freedom): 1"
## [1] "% deviance explained: 20.8%"
## [1] "p-value on Chi^2 Test on model:
3.05e-27"
```

Why Do We Not Always Notice?

1. Problem not as obvious with ensembles
   - Lower complexity base models, simple model combination
   - Stacked learning: Higher-complexity models

2. We have noticed, we just ignore it.
   - Random Forests do tend to overfit

Solution 1: Train Base models on Different Data

- Split training set into train/ calibration

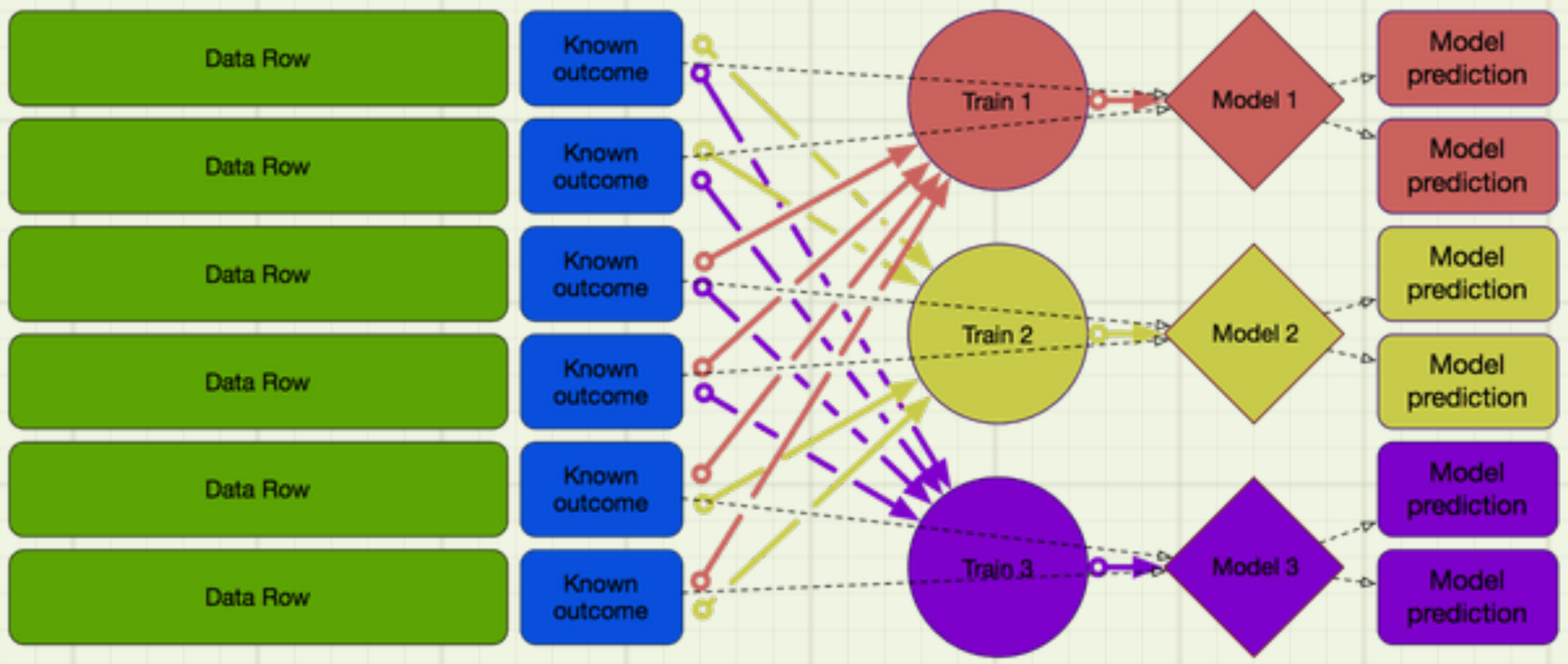
- Fit effects coding on calibration

- Train model on train

- AUC(Train) = 0.71, AUC(Test)=0.69
  - (vs 0.64 naive coding)

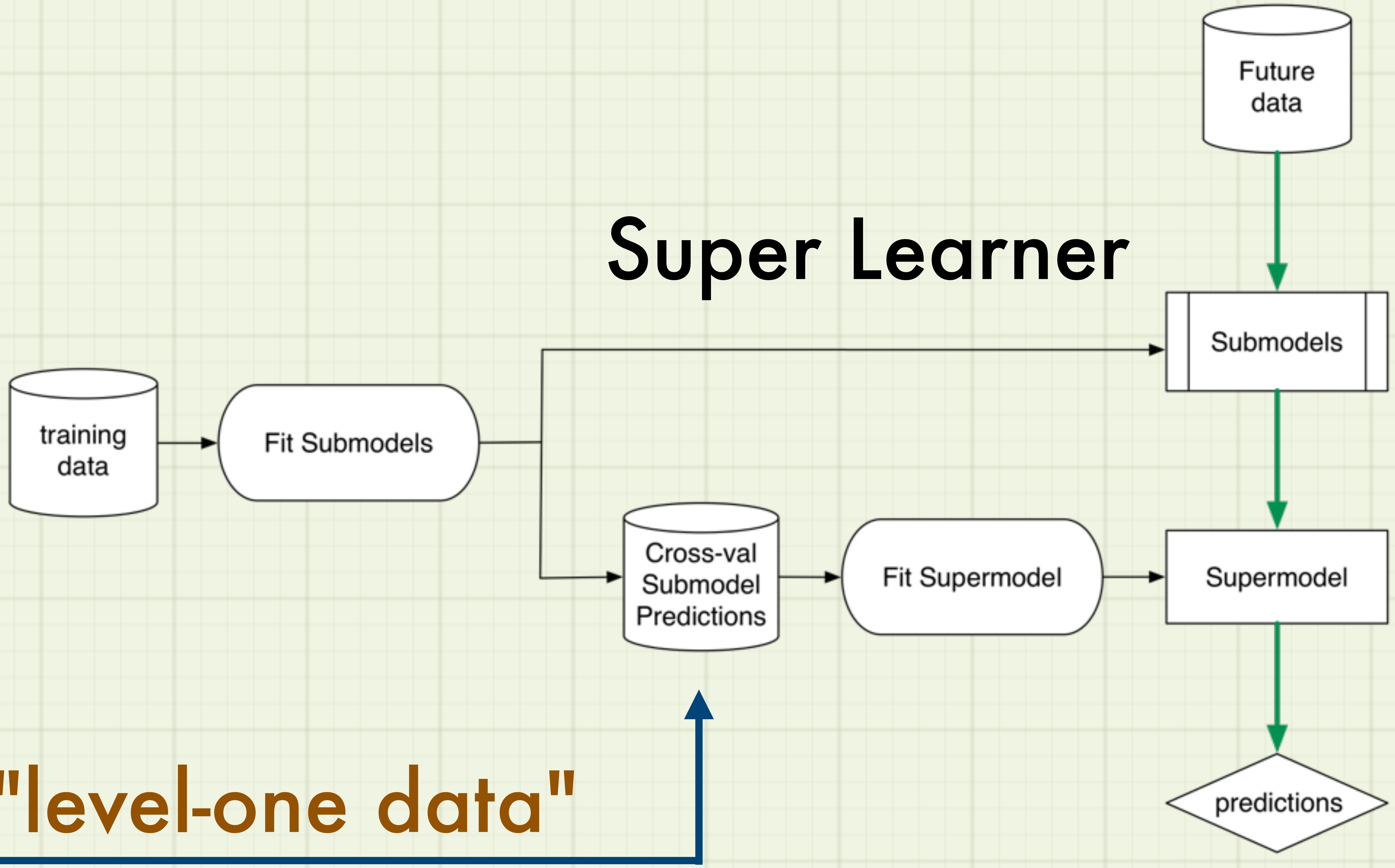
Win-Vector LLC

Solution 2: Simulate training models on different data



"level-one data"

Super Learner

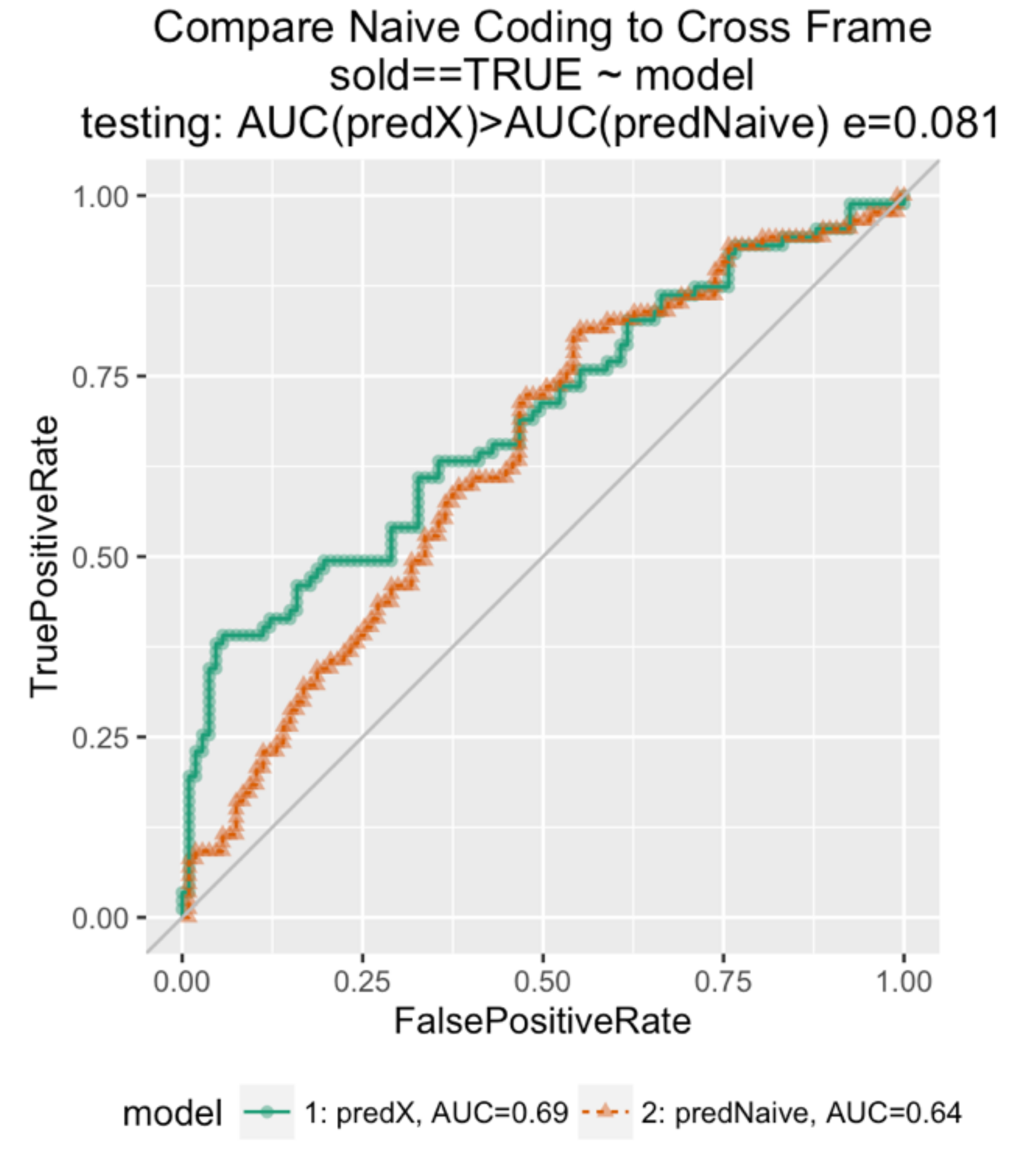


Win-Vector LLC

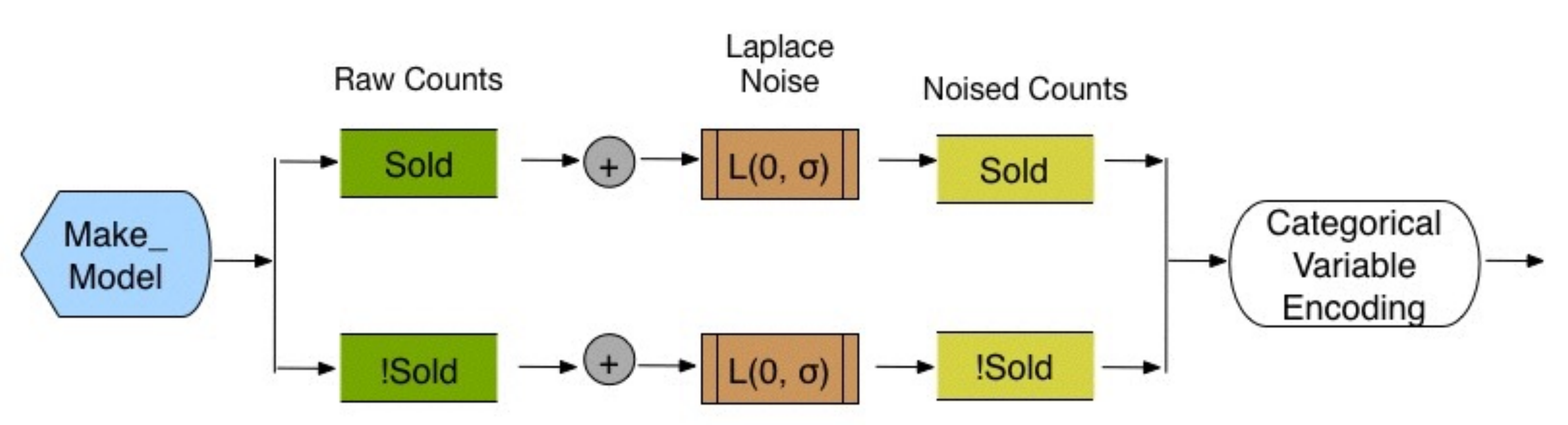
Impact Coding: CrossFrames

Data preparation package
`vtreat`

- Fits impact code on training data
- Creates "CrossFrame" impact coding for fitting model
- AUC(Train) = 0.7, AUC(Test)=0.69
  - (vs 0.64 naive coding)



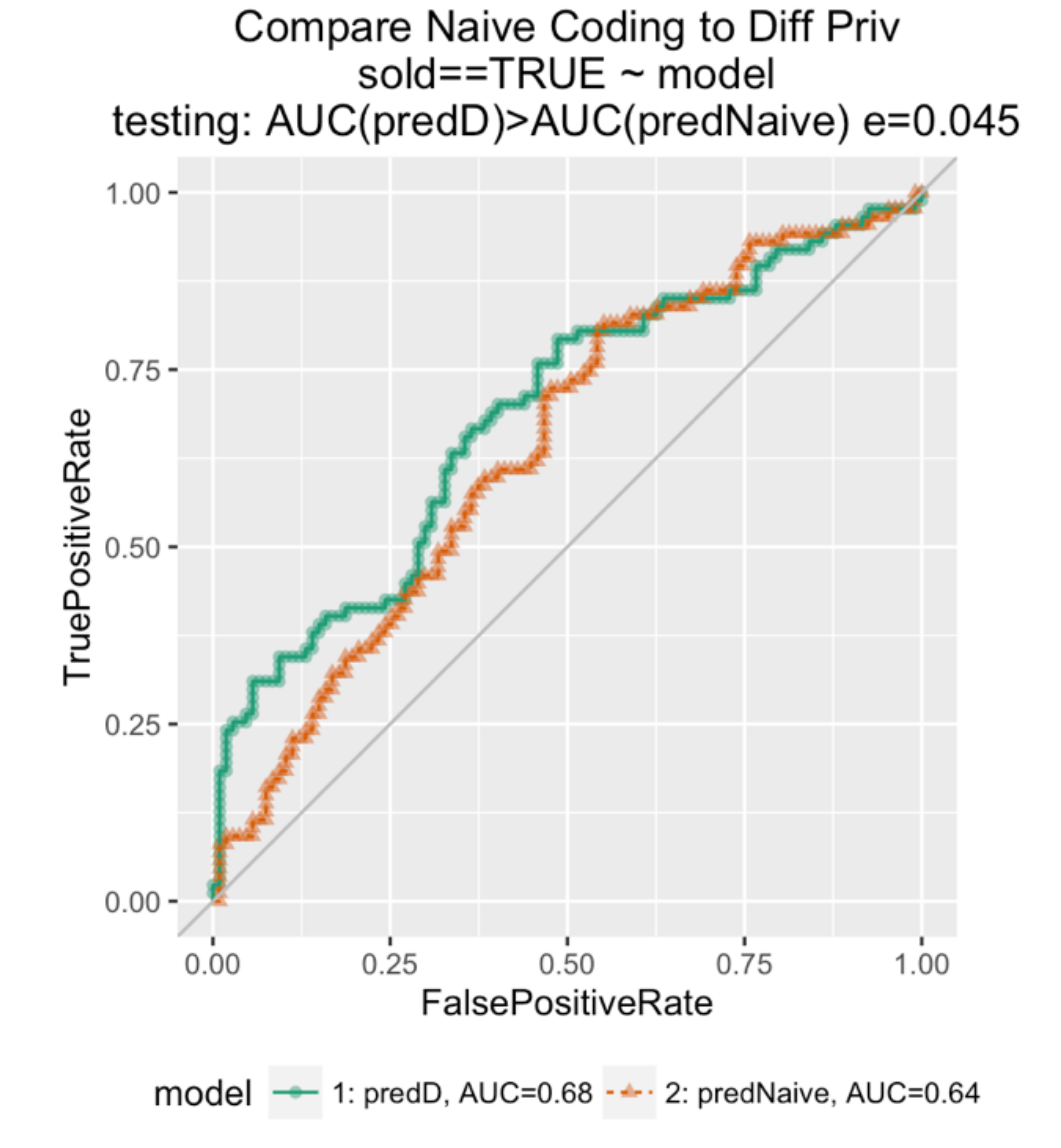
Solution 3: Noise the Observations



Add noise to training data before passing to effects coding

ZIP code example

- Add synthetic observations to training data
- Use noisy training data to effects code, clean training data to model
- AUC(Train) = 0.71, AUC(Test)=0.68
  - (vs 0.64 naive coding)



Win-Vector LLC

Differential Privacy

- Bilenko, Misha. “Big Learning made Easy — with Counts!” *Machine Learning Blog* http://blogs.technet.com/b/machinelearning/archive/2015/02/17/big-learning-made-easy-with-counts.aspx

- Dwork, Cynthia, *et.al*. “The reusable holdout: Preserving validity in adaptive data analysis”, *Science*, vol 349, no 6248 pp 636-638, August 2015.
    - Abstract: https://www.sciencemag.org/content/349/6248/636.abstract

- http://www.win-vector.com/blog/2015/11/our-differential-privacy-mini-series/

What about Smoothing/Regularization?

- Obscures rare values — should work, right?
- All three methods outperform Laplace smoothing/ regularization
- The meta-model can sometimes undo smoothing
    - Performs no better than naive effects coding

Win-Vector LLC

Summary

- Training data reuse increases nested model bias.
- High-complexity models (base or meta) increase nested model bias.
- Use (simulated) out-of-sample data for meta-model training.
- Prefer lower-complexity models when feasible.

Nested Model Workflow

1. Use training data to train base models
2. Create (simulated) out-of-sample base model features
   - Calibration set; cross-frame; Laplacian noise
3. Train meta-model on out-of-sample features

	Base Model	Meta Model
Ensembles	(Usually) Low comp.	Sum (oblivious/"no complexity")
Stacked models	Arbitrary (High comp.)	Arbitrary (prefer Lower comp.)
Impact Coding	Impact code (High comp) Numerical var (Low comp)	Arbitrary (High comp)

Win-Vector LLC

	Base Model	Meta Model
Hyperparameter tuning	Arbitrary w/ varying ρ (high comp.)	Pick best ρ
Variable selection (univariate models)	Numerical var (low comp.) Categorical var (high comp) Model algo (prefer lower comp)	Pick N best (low comp) Exceeds threshold (low comp)
Stepwise methods	Arbitrary (High comp., varies with #vars)	Pick best (multiple times) (Very high complexity)

More Information

- Examples from talk: https://github.com/WinVector/NestedModelsTalk
- Cross-frames for impact coding (`vtreat`)
  - http://winvector.github.io/vtreathtml/vtreatCrossFrames.html
  - `vtreat` on CRAN
- Breiman, Leo. "Stacked Regressions," *Machine Learning, 24* p. 49-64 (1996). http://statistics.berkeley.edu/sites/default/files/tech-reports/367.pdf
- van der Laan, Alan, et.al. *Super Learner*. http://biostats.bepress.com/ucbbiostat/paper222/

Thank You!