How to win a Kaggle competition

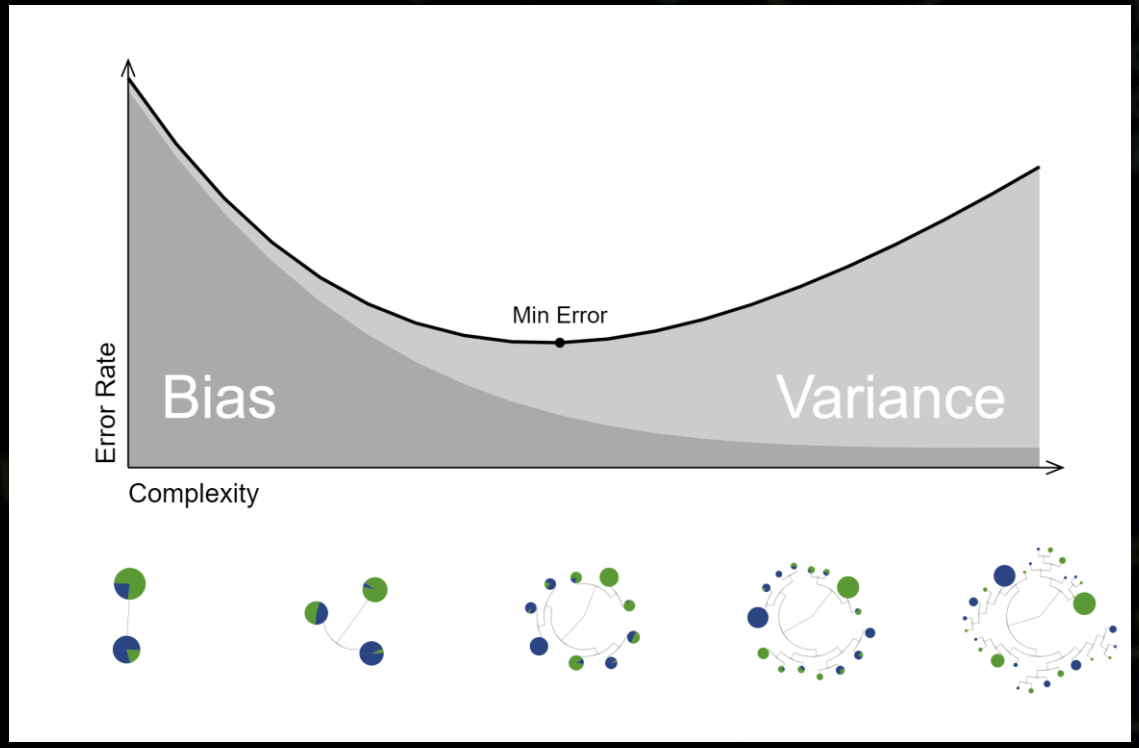
Part 4: model ensemble and other tricks

Chris Chen@Calgary Data Science Academy

Essentially all models are wrong, but some are useful.

George E.P. Box

Bias-variance trade off



To control the balance between bias-variance, we can

- Add regularizations that decrease bias by reducing model complexity.
- Reduce variance by averaging predictions from strong models (low bias, high variance).
  - Bagging: averaging predictions from models trained with the same algorithm but bootstrap-sampled versions of the training data.
- Reduce bias by keep evolving week models (high bias, high variance), this is also called boosting.

http://www.r2d3.us/visual-intro-to-machine-learning-part-2/

Why averaging reduces variance?

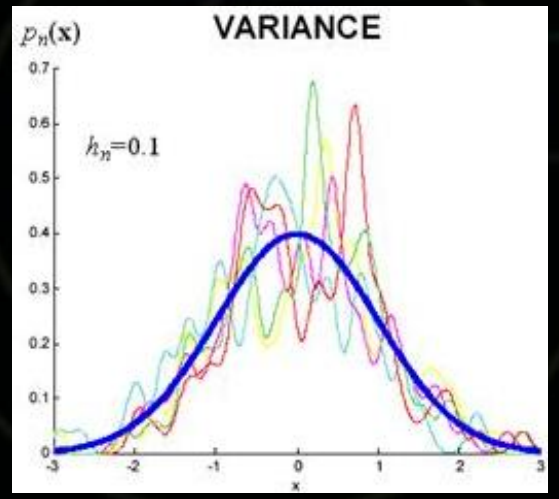
Let $\hat{y}_i$, ... $\hat{y}_N$ be predictions from N models and assume they are i.i.d., then:

$$Var\left(\frac{1}{N}\sum_i \hat{y}_i\right) = \frac{1}{N}Var(\hat{y}_i)$$

If models are correlated, let σ^2 be the std of $\hat{y}_i$, ... $\hat{y}_N$ and ρ be correlation of two models, then

$$Var\left(\frac{1}{N}\sum_i \hat{y}_i\right) = \rho * \sigma^2 + \frac{(1-\rho)}{N}$$

As we can see, the correlation of models plays an important role in model ensemble. In another word, the more diversified the models are, the better the ensemble will be.



Predictions from each model is centered around the true density, but is overly complicated (low bias, high variance). By averaging them out, we get a smoothed version of them (low variance), still centered around the true density (low bias). (Retrieved from https://www.quora.com/What-does-Bagging-reduces-the-variance-while-retaining-the-bias-mean)

* More details on bagging, boosting and how bagging reduces variance can be found from "The Elements of Statistical Learning"

What to consider for model ensemble

- Diversity algorithms
  - GBDT (XGBoost, LightGBM) + NN models have been found to be the best duos.
  - Random Forest/ Extra trees/ XGBoost w/ linear booster can be good complement .
  - FM/FFM/DeepFM work very well for CTR problems as well as there are lots of categorical features.
- Diversify training data or even test data
  - Random Forest and GBDT automates the process with bootstrap sampling.
  - Sub-modelling is a useful trick to enhance training data diversity
    - Train a global model
    - Train sub-models with training data specific to a sub-scenario, e.g. train housing models by state, county or city, make predictions accordingly.
    - Ensemble predictions from global model and sub-models.
  - Data augmentation
    - TTA(test-time-augmentation)
- Diversity training parameters
  - Train the same algorithms with different hyper parameters
  - Or even train the same algorithms with same hyper parameters but different seeds or initializations.

How to perform model ensemble

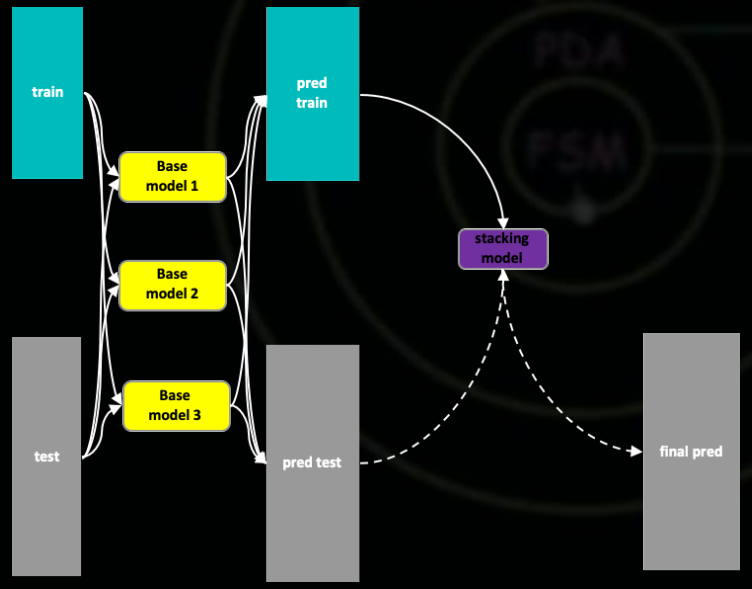
- Averaging
- Weighted averaging
- Ranked averaging
  - Works for imbalanced binary classification problems measured by ROC-AUC because ROC-AUC concerns the relative values across the population rather than the absolution values.
- Voted averaging
- Harmonic averaging
  - Works for regression problems where the target is always positive.

pred1	pred2	rank1	rank2	rank1+rank2	ranked_averaging (scaled to [0,1])
0.1	0.3	1	3	4	0
0.2	0.2	2	2	4	0
0.4	0.1	4	1	5	0.2
0.2	0.6	2	5	7	0.6
0.5	0.4	5	4	9	1

Example of ranked averaging

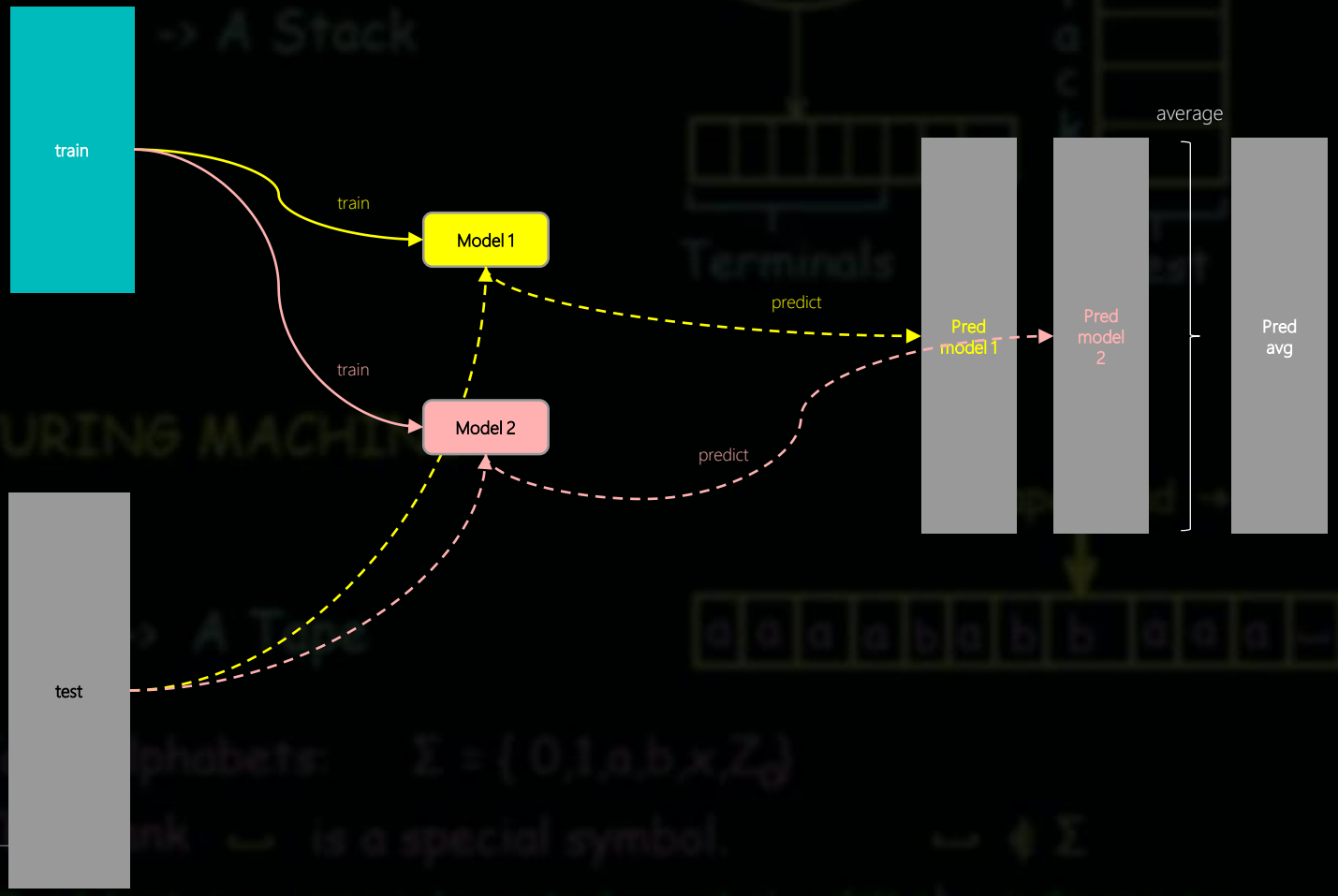
Stacking – the ultimate ensemble

- How to effectively synergize many models, e.g. find out the appropriate weights for 100 base models?
- The natural thought is to utilize supervised learning algorithms
- However, the challenge is that for supervised learning algorithms to work we have to have a labeled dataset for training. How can we do that on predictions from base models where there're no labels? Or in another word, how to create predictions on training data?
- Inspired by cross validation, the key idea of stacking is to create predictions on test data and out-of-fold predictions on training data.



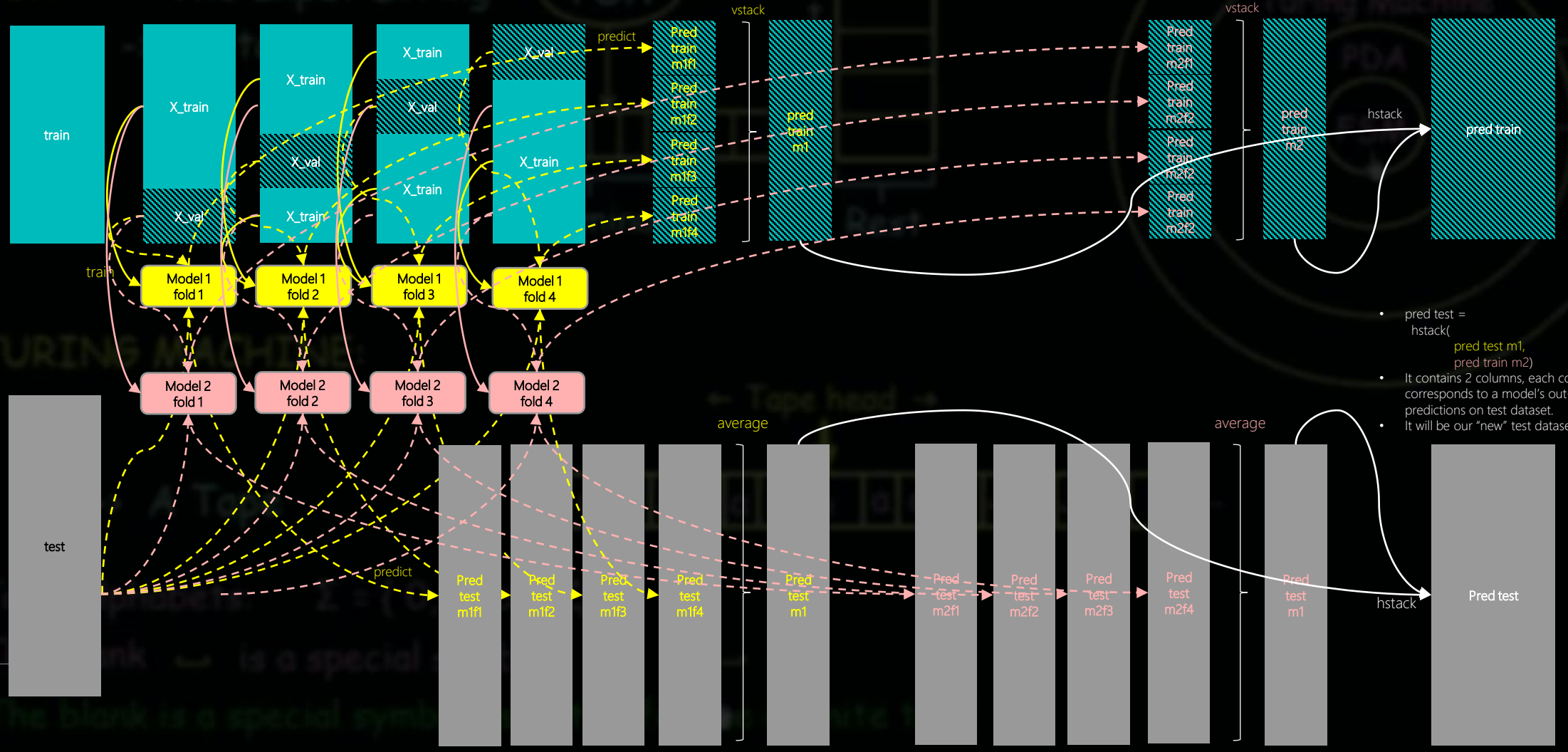
Stacking in a nut shell

Model ensemble – simple case



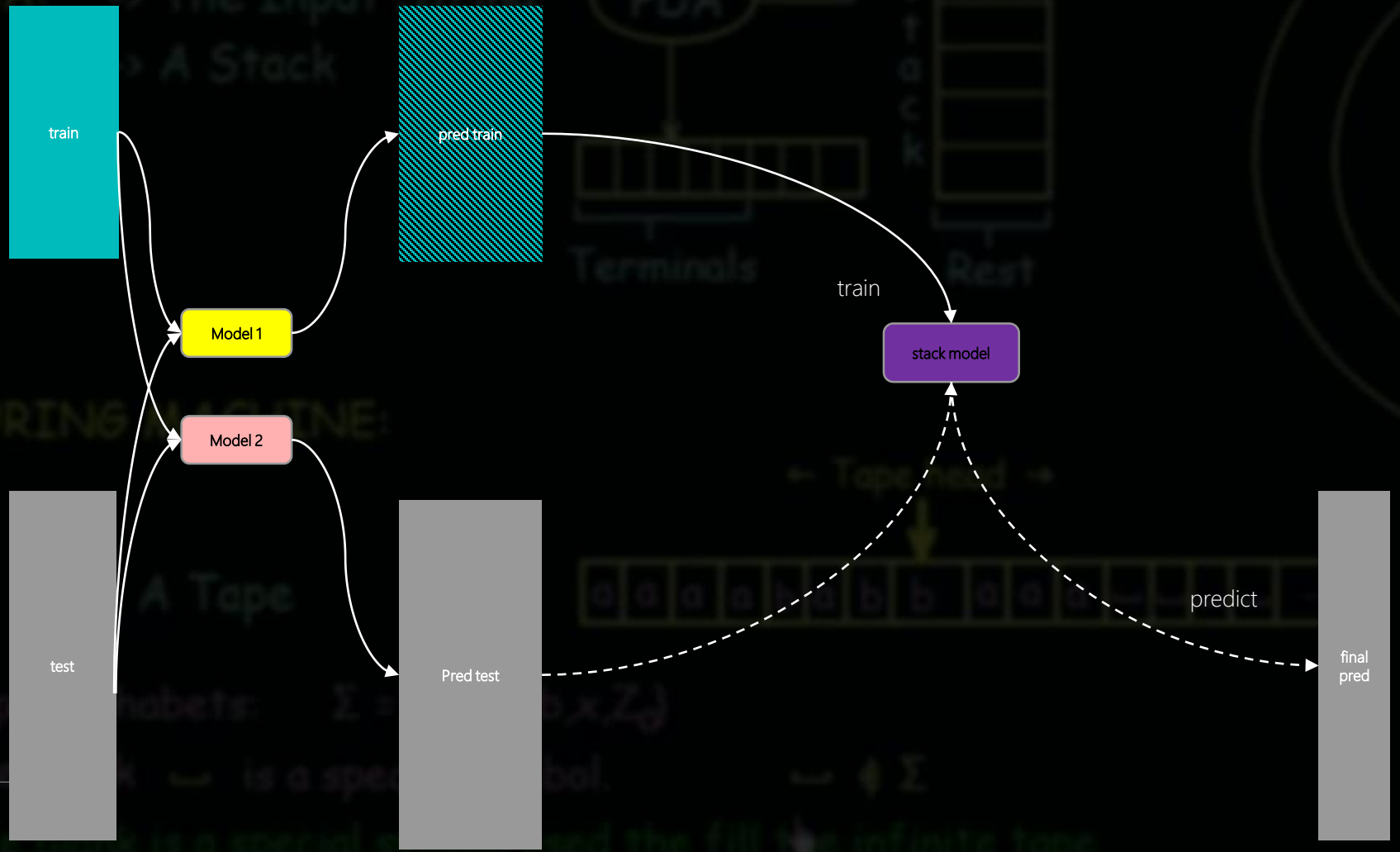
Model ensemble – stacking – level 1

- pred train =
  hstack(
  pred train m1,
  pred train m2)
- It contains 2 columns, each column corresponds to a model's out-of-fold predictions on train dataset.
- It will be our "new" train dataset

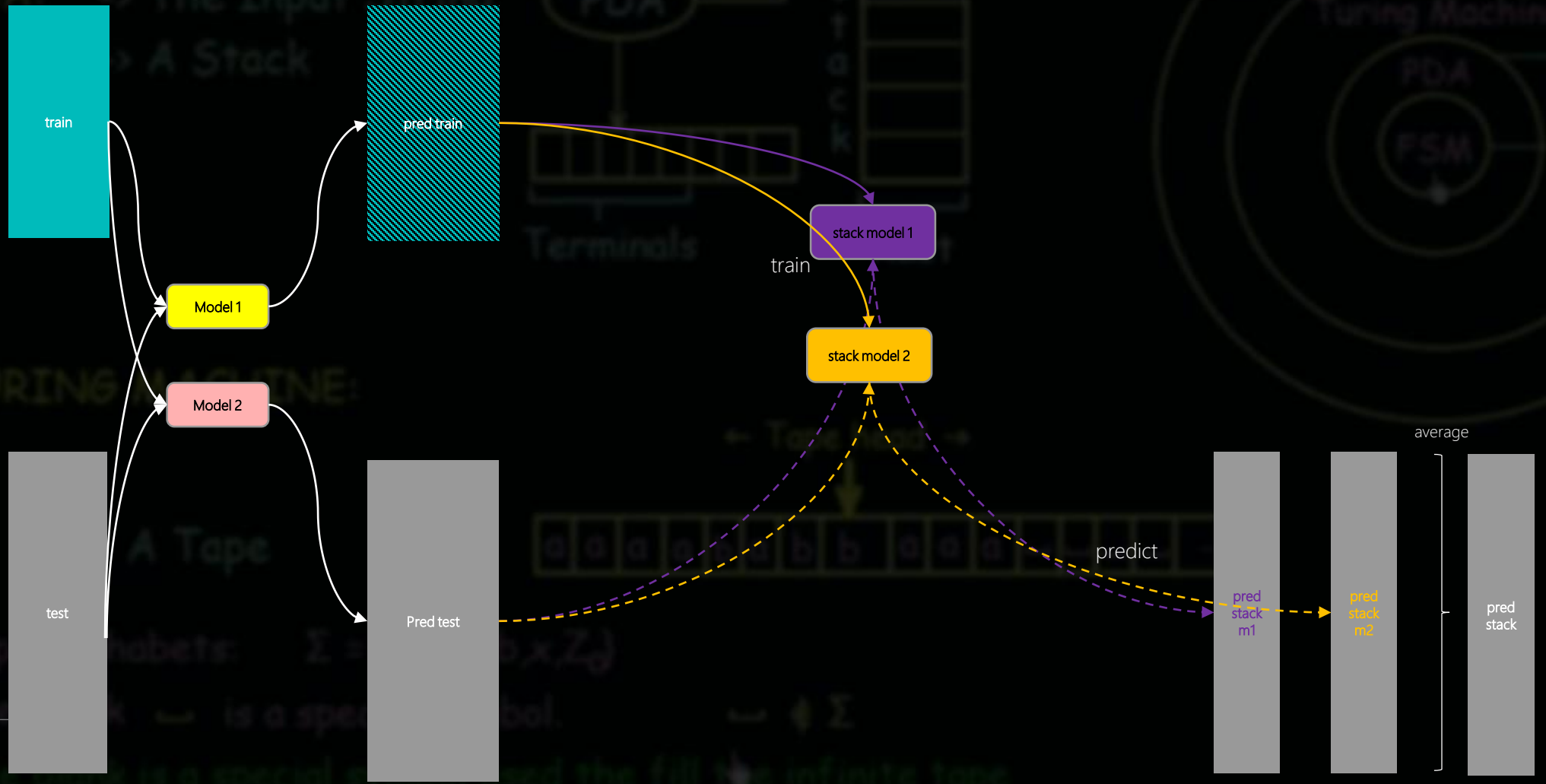


- pred test =
  hstack(
  pred test m1,
  pred train m2)
- It contains 2 columns, each column corresponds to a model's out-of-fold predictions on test dataset.
- It will be our "new" test dataset

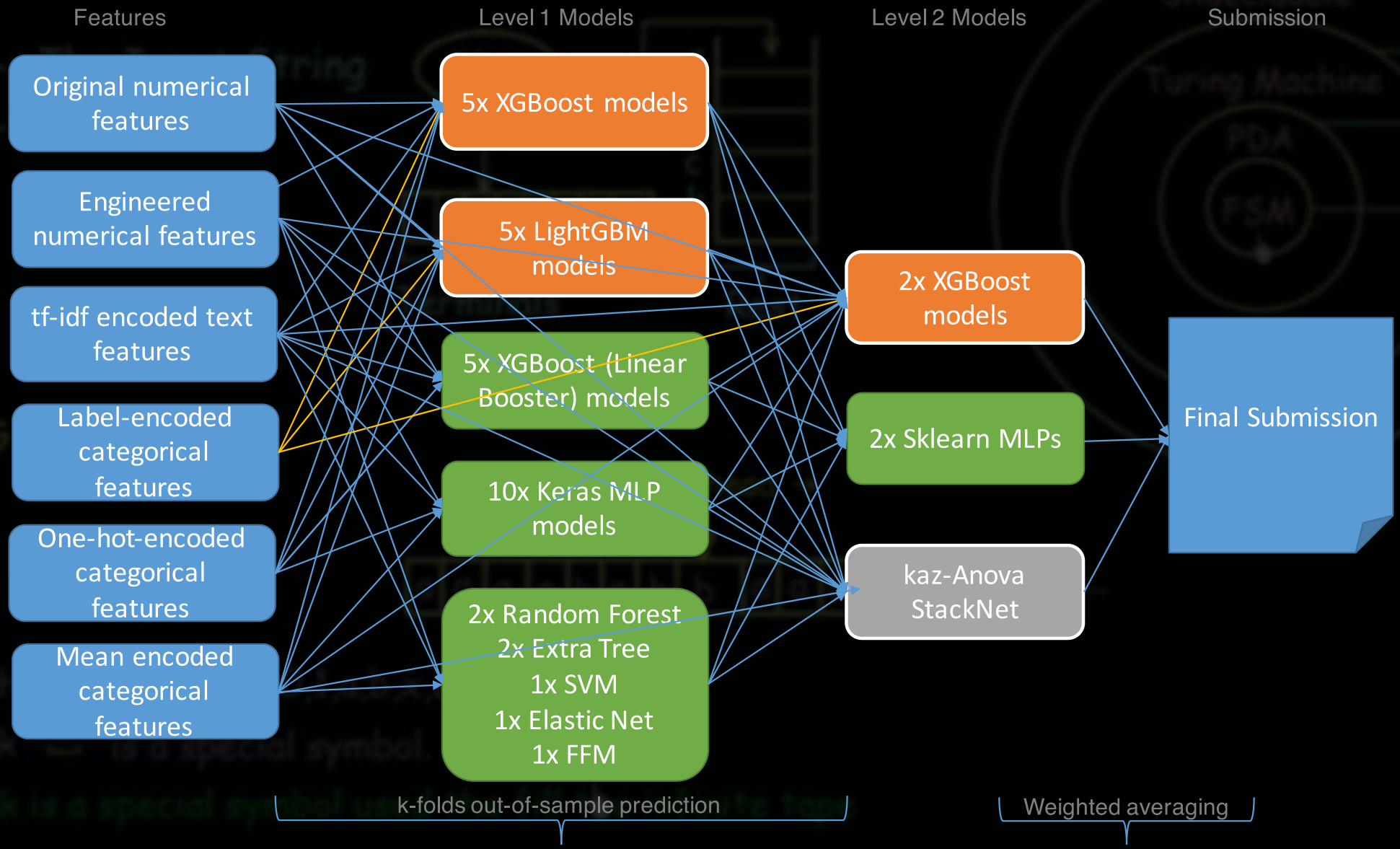
Model ensemble – stacking – level 2



Model ensemble – stacking – level 2



Stacking solution for TwoSigma Renthop



Adversarial validation: train vs. test

- One of the most common practical problems in predictive modeling is how to validate if the training and test datasets are of the same or similar distribution. However, this is typically challenging especially when working with large datasets.
- Adversarial validation is a useful trick that allows DS to effective tell if test data is different from training data and on which variables the difference takes place.

- Vertically stack train and test data together
- Create a new variable and set the value = 1 where samples are from training and 0 where samples are from test.
- Shuffle and split the new dataset into two datasets: AV_train and AV_val.
- Train a binary classification model on AV_train and validate the model agsint AV_val, measured by ROC-AUC.
- If the ROC_AUC is close to 1, it means the model is able to perfectly identify original training and test datasets. If ROC_AUC is close to 0.5, means the model is not able to differentiate training from test.
- The importance of each variables can be used to identify problematic variables – stronger variables are more suspicious. Kaggle Grand Master Bojan Tunguz developed a nice notebook to demonstrate how it works" https://www.kaggle.com/tunguz/adversarial-ieee

Pseudo labeling

Just two weeks ago, researchers from Google Brain and Carnegie Mellon University released models trained with a semi-supervised learning method called "Noisy Student" that achieve 88.4 percent top-1 accuracy on ImageNet. It turns out the similar trick (pseudo labelling) has been used by Kagglers for quite a while for winning competitions (e.g. Santander customer transaction prediction, Instant Gratification). The idea is actually quite straightforward but works amazingly well.

1. Train a teacher model on labeled data (training data).
2. Use the teacher model to generate pseudo labels on unlabeled data (test data).
3. Append pseudo labeled data with high confidence to the original training data.
4. Train a student model on the combined data (training + pseudo labeled).
5. Make predictions on the rest of unlabeled data.

Chris Deotte has put together a notebook explaining how and why pseudo labelling work along with demo code.

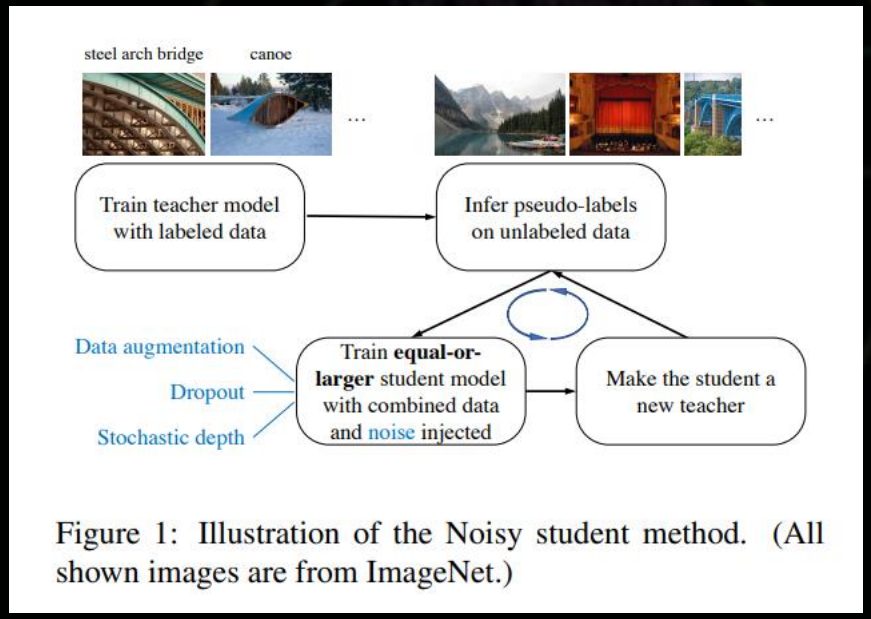


Figure 1: Illustration of the Noisy student method. (All shown images are from ImageNet.)

Self-training with Noisy Student improves ImageNet classification