Model assessment

How good is my model and how do I properly test it?

Terence Parr
MSDS program
University of San Francisco

UNIVERSITY OF SAN FRANCISCO

Terminology: Loss function vs metric

- *Loss function*: minimized to train a model (if appropriate)
  E.g., gradient descent uses loss to train regularize linear model
- *Metric*: evaluate accuracy of predictions compared to known results (the business perspective)
- Both are functions of y and $\hat{y}$, but also possibly model parameters…
- Examples:
  - MSE loss & MSE metric
  - MSE loss & MAE metric
  - Gini index & misclassification or FP/FN metric
- If metric is applied to validation or test set, informs on generality and quality of your model

See also stackoverflow post by Chstiros Tsatsoulis: https://goo.gl/T5AmrT

Train, validate, test

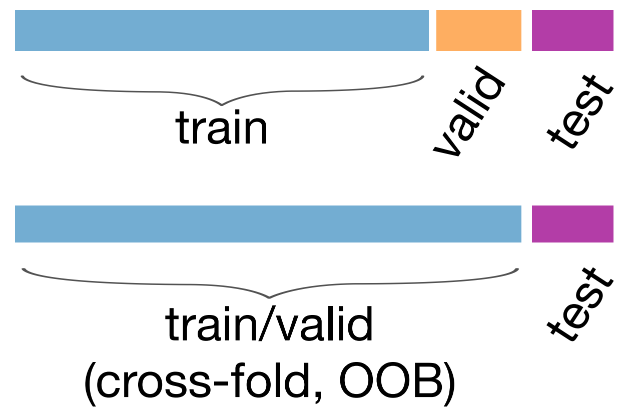
- *This might be the most important slide of entire class!*
- We always need 3 data sets with known answers:
  - training
  - validation (as shorthand you'll hear me / others call this test set)
  - testing (put in a vault and **don't peek**!!)
- Validation set: used to evaluate, tune models and features
  - Any changes you make to model tailor it to this specific validation set
- Test set: used exactly **once** after you think you have best model
  - The only true measure of model's generality, how it'll perform in production
  - Never use test set to tune model
- Production: recombine all sets back into a big training set again, retrain model but don't change it according to test set metrics

Key question: is your data time-sensitive?

- Time series: temperature, stock prices, sales, inflation, city population, …
- You can try to detrend the data to flatten average *y* etc…
- Almost all data sets are time sensitive in some way (boo!)
- Some data sets are skewed over time even if no date column; e.g., new users to facebook are different over time
- Try to find things that are less time dependent; e.g., air conditioning sales appear to fluctuate over time but these sales are driven more by temperature and humidity than date

How to extract validation, test sets

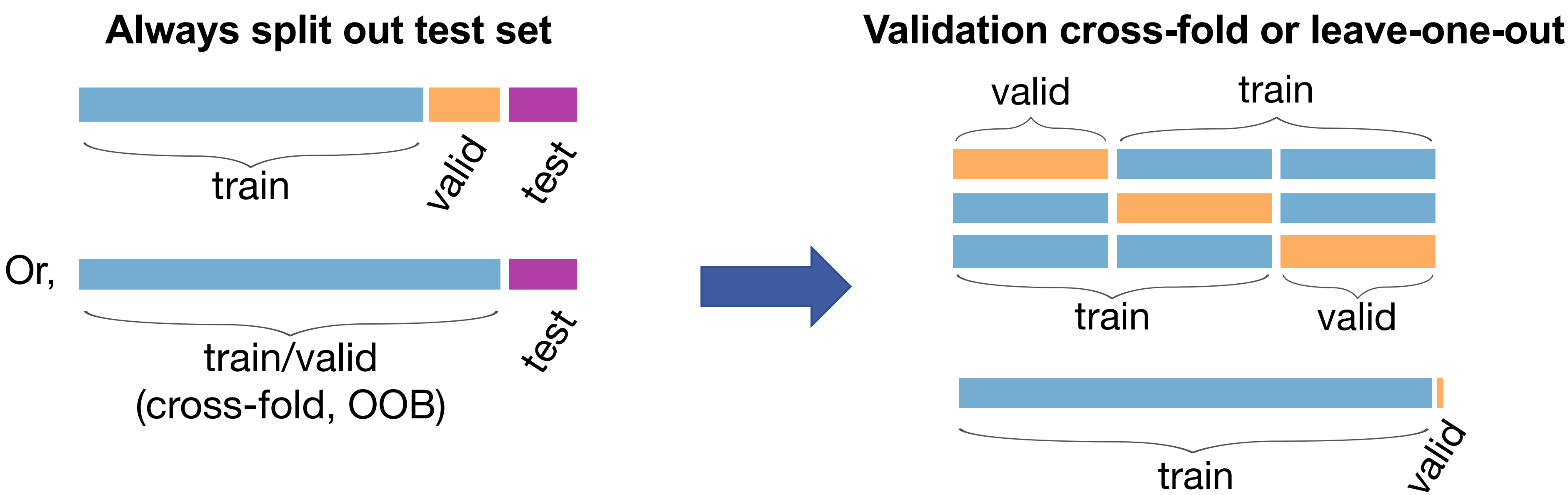
- Extract random subsets; perhaps 75%/15%/10%; can shuffle then chop
- Or, grab 15% test set (and hide it away) & use cross-fold on remainder for train/valid
- For RF, we can start with out-of-bag score
- Ensure validation set has same properties as test set (e.g., size, time, …):
  - if 10k samples in test, make 10k sample validation set
  - if test is last 2 months, validation must be last 2 of remaining data



*If time-sensitive data set, do **not** shuffle, sort by date, valid/test are newest rows (means OOB not useful here)*

UNIVERSITY OF SAN FRANCISCO

Testing strategies



RF Out-of-bag (OOB)

- RFs have a major advantage over other models: OOB metrics
- Each tree is trained on 63% of data, leaving 27% OOB
- OOB subsamples used by the trees are different
- The OOB metric for tree T is computed using T's OOB samples and averaged to get overall OOB metric
- It's an excellent estimate of the validation error
- Stick with OOB unless default sklearn metric not what you want (not having to process training and validation sets separately is a huge productivity win)

OOB continued

- OOB error will slightly underestimate test set error. Why?
  - At least one of the trees in the forest is trained on the OOB samples
- OOB metrics don't affect training, just gives metric
- Compare OOB with validation set:
  - If we add predictive feature and OOB isn't worse but the validation set is, the validation set is not good; e.g., different distribution or time sensitive
- OOB not to be used with time-sensitive data sets

Metrics interpretation

- Basic idea: for each test record, compute error from y and $\hat{y}$; the metric is then usually the average of these errors
- **Perspective**: Is 99% vs 99.5% accuracy difference 2x or 0.5%? What about 80% vs 90%? 10% diff but also 2x
- As we approach 100%, getting better is tougher and tougher
- Is 90% accuracy or R^2=0.8 good? Maybe. What are the lower bounds from a simpler or trivial mode?
- Classifiers must beat *a priori* probabilities
  - If 90% of email are spam, your model must beat 90% accuracy
- Regressors should beat “mean model” and linear model

Training score

- Training score not really useful by itself because, for example, we can get good fit for random data X->y with 1000 x 4 data X data, R^2=.85
- Actually, if training score is low, model is too weak
- Or, dataset is missing vars like "we had a sale that day" or "closed on holidays"

```
X_train = np.random.random((1000,4))
y_train = np.random.random(1000)
rf = RandomForestRegressor(n_estimators=100)
rf.fit(X_train, y_train)
rf.score(X_train, y_train)
```

```
0.8474731797281314
```

WOW!

What if training score is good but validation is low?

- Bad validation set
  (didn’t extract random or time-sorted set or just given bad set?)
- Time-sensitive data set diverging? (try detrending data)
- Overfit model?
- Not properly applying feature transforms from training to validation set?
- Bug?

Comparing training / validation sets

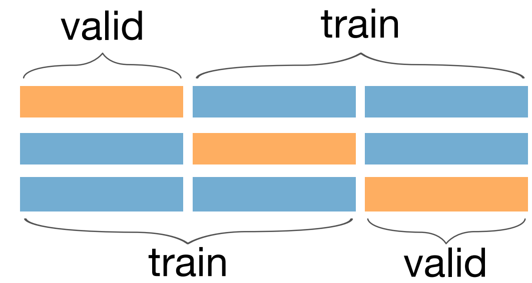
Awesome but not well-known trick

- Process to see if valid (or test) set is distinguishable from train set
  1. Combine both X and y from train & valid sets into single data set
  2. Create new target column called **istest**
  3. Train model on the combined set
  4. Assess metric
- If train/valid not different, should get 0 metric (can't distinguish them)
- But if you get, say, 0.95 metric, train/valid sets are very different

If train/test are easily distinguishable…

- You might find that an ID or date that is totally different in train vs valid set or maybe all y's are bigger in the validation set
- Drop that feature and see how the validation score changes
- Look at the importances of original and istest models. Features that are important in both are the problem
  - If it's not important in original model, we don't care about it: it's not predictive of target
  - If it's not important in istest model, it's not causing confusion between the data sets

Stability of metric values



- Getting a single validation metric is usually not enough because scores can vary from run to run because of outliers and anomalies (even with k-fold)
- Consider score fluctuations in NYC rent data (before cleaning)

	OOB	R^2	MAE	MSE
0	0.582	0.002	1,150.354	2,402,630,918.659
1	0.027	0.050	878.512	2,053,053,487.605
2	-0.309	0.323	408.299	3,852,177.529
3	-0.152	-44.812	527.185	265,146,585.910
4	-0.155	-0.105	404.700	7,275,725.459

Outliers cause mismatch between train/valid sets; k-fold, random subsets will see high variability

See https://github.com/parrt/msds621/blob/master/notebooks/assessment/metrics.ipynb

Regressor metrics

Common regressor metrics

- Mean squared error
  Range 0..∞, units(y)^2, symmetric

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

- Mean absolute value
  Range 0..∞, units(y), symmetric

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

- Mean absolute percentage error
  Range 0..∞, unitless, **asymmetric**
  undefined if y=0

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right|$$

R^2

- How well our model does compared to “mean model”

$$R^2 = 1 - \frac{\text{Squared error}}{\text{Variation from mean}} = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(\hat{y}_i - \overline{y})^2}$$

- Range of possible values: (-∞,1)
- Our model could be really bad, giving large negative numbers
- For OLS linear models, R^2 in [0,1]
- R^2 is default regressor metric for sklearn

Which metric should I use?

- That depends what we care about for business reasons
- For prices, we usually care more about the percentage error than the absolute amount
- $500 off for a $1,000 apartment is 0.5x or 1.5x off and a big deal but $500 off for a $1 million apartment is a trivial difference
- For things like body temperature that will most likely all be within a small range, the mean absolute error (MAE) is a good and interpretable measure
- The percentage error can be interpretable but there are problems with it: asymmetry

Symmetry in metrics

- Most metrics use $y - \hat{y}$ but $y/\hat{y}$ is often better
- Most ratio-based metrics are asymmetric however which is bad!
  - If y=100, $\hat{y}$=0.01: MAPE = 0.9999
  - If y=0.01, $\hat{y}$=100: MAPE = 9999
- So I like MMAR
  (mean max abs ratio)
  Range 1..∞, unitless, symmetric,
  undefined at y=0 or $\hat{y}$ =0
  - If y=100, $\hat{y}$=0.01: MAPE = 10000
  - If y=0.01, $\hat{y}$=100: MAPE = 10000

$$MMAR = \frac{1}{n}\sum_{i=1}^{n} max(\left|\frac{y_i}{\hat{y}_i}\right|, \left|\frac{\hat{y}_i}{y_i}\right|)$$

Classifier metrics

Common classifier metrics

(Ugh; much more complicated than for regressors)

- TP = true positive, TN = true negative
- FP = false positive, FN = false negative
- Confusion matrix for binary classification

		Predicted F	*Predicted* T
Actual	**F**	35	3
Actual	**T**	1	75

- Accuracy = correct classification rate = (TN+TP)/n
- , F, AUC, prec, recall, FP, TP, conf matrix, log loss

Confusion matrix

Log loss

Diff in oob / validation

- when OOB score is better than validation score, it indicates one of three things:
  - Sometimes the validation score is better or worse than the OOB score, due to random fluctuations caused by the inherent randomness of RF construction.
  - The validation set is drawn from a different distribution than the training set.
  - The model is overfit to the data in the training set, focusing on relationships that are not relevant to the test set

Dealing with unbalanced datasets

- balancing by oversampling. You can make copies if model doesn't handle oversampling. Or, can use RF class weights in scikit learn.
- Upsample **AFTER** train/test split! Otherwise, test data leaks into the training set!
- order that you oversample / get test set matters hugely
- ROC not good; use PR curve