Winning Data Science Competitions

Some (hopefully) useful pointers

Owen Zhang
Data Scientist

DataRobot

A plug for myself

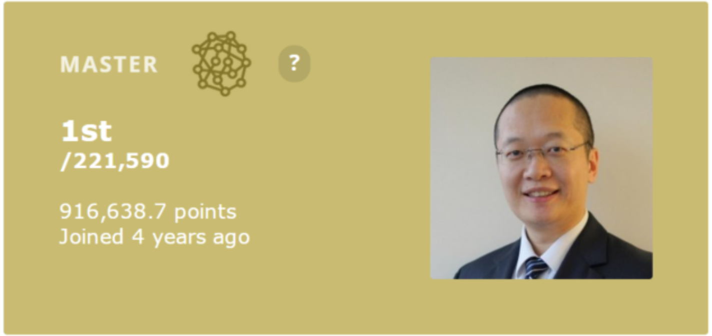
Owen

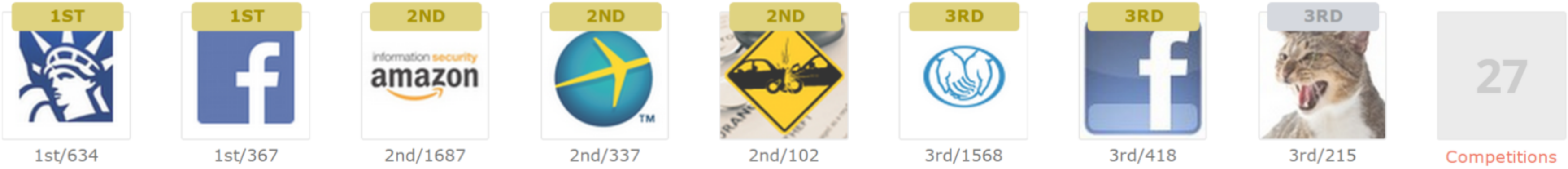
Current

- Chief Product Officer

Previous

- VP, Science





A plug for myself

Owen

Current

- Chief Product Officer — DataRobot

Previous

- VP, Science — AIG

MASTER

1st ~~/221,590~~ 1st / 330,336

~~916,638.7 points~~ 176,181 points

Joined 4 years ago

1ST	1ST	2ND	2ND	2ND	2ND	3RD	3RD	33
1st/634	1st/367	2nd/1687	2nd/1604	2nd/337	2nd/102	3rd/1568	3rd/418	Competitions

DataRobot

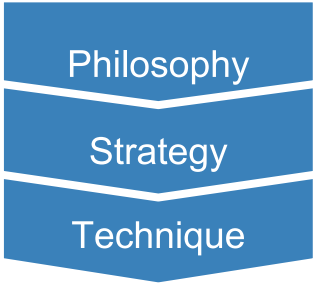
Agenda

- Structure of a Data Science Competition

- Philosophical considerations
- Sources of competitive advantage
- Some tools/techniques

- Three cases -- Amazon Allstate LM

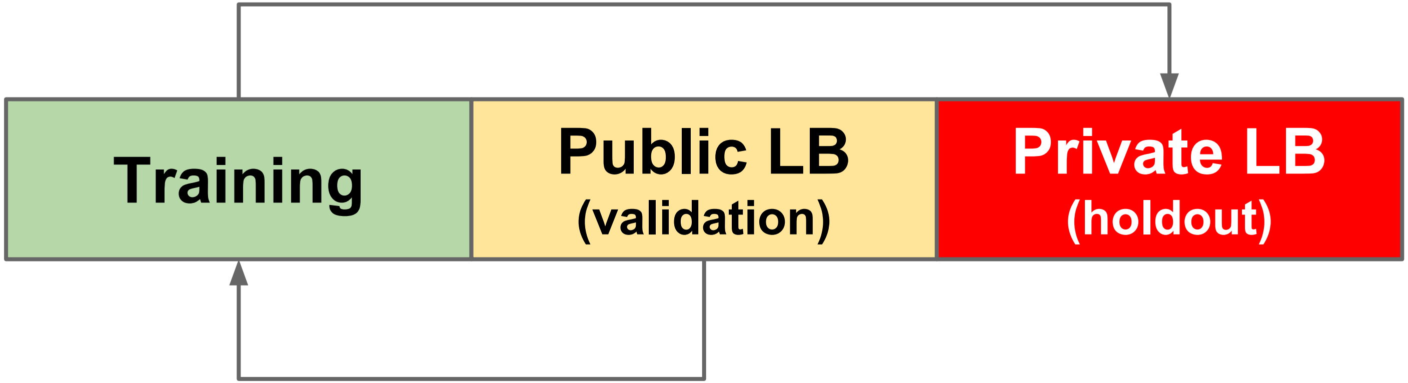
- Apply what we learn out of competitions



DataRobot

Structure of a Data Science Competition

Build model using Training Data to
predict outcomes on Private LB Data



Quick but sometimes misleading feedback

Data Science Competitions remind us that the purpose of a predictive model is to predict on data that we have ***NOT*** *seen.*

DataRobot

A little “philosophy”

- There are many ways to overfit

- Beware of “multiple comparison fallacy”
  - There is a cost in “peeking at the answer”,
  - Usually the first idea (if it works) is the best

“Think” more, “try” less

Sources of Competitive Advantage (the Secret Sauce)

- Luck
- Discipline (once bitten twice shy)
    - Proper validation framework
- Effort
- (Some) Domain knowledge
- Feature engineering
- The “right” model structure
- Machine/statistical learning packages
- Coding/data manipulation efficiency

The right tool is very important

Be Disciplined
+
Work Hard
+
Learn from everyone
+
Luck

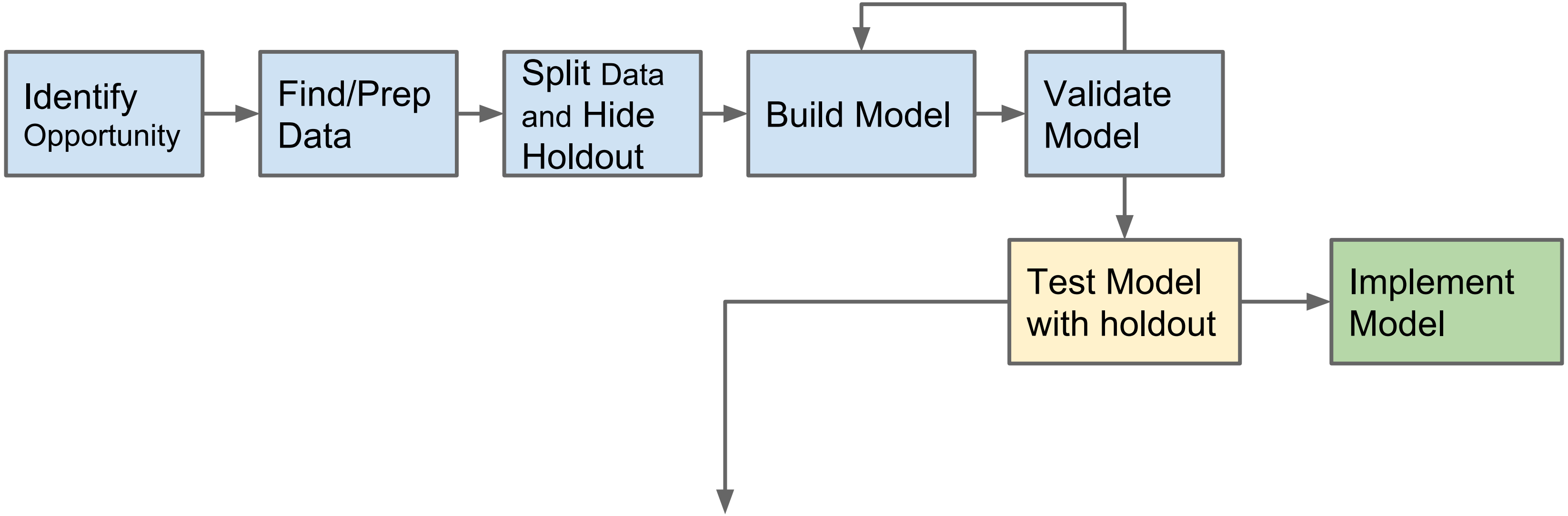
DataRobot

Good Validation is MORE IMPORTANT than Good Model

- Simple Training/Validation split is NOT enough
  - When you looked at your validation result for the Nth time, you are training models on it

- If possible, have “holdout” dataset that you do not touch at all during model building process
  - This includes feature extraction, etc.

A Typical Modeling Project



- What if holdout result is bad?
    - Be brave and scrap the project

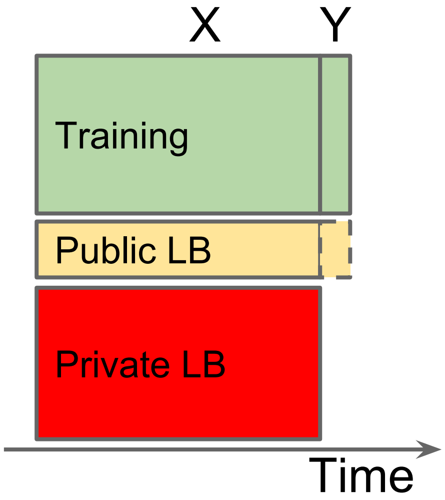
Make Validation Dataset as Realistic as Possible

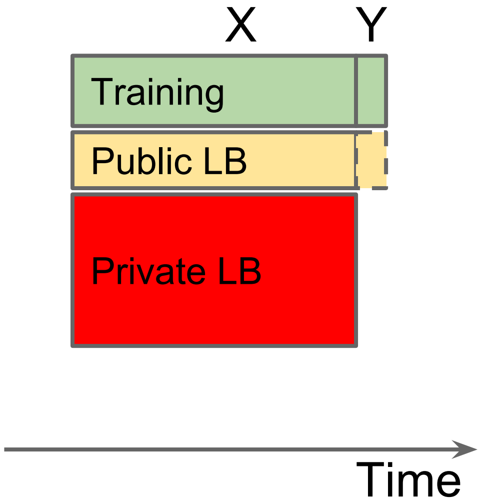
- Usually this means “out-of-time” validation.
    - You are free to use “in-time” random split to build models, tune parameters, etc
    - But hold out data should be out-of-time

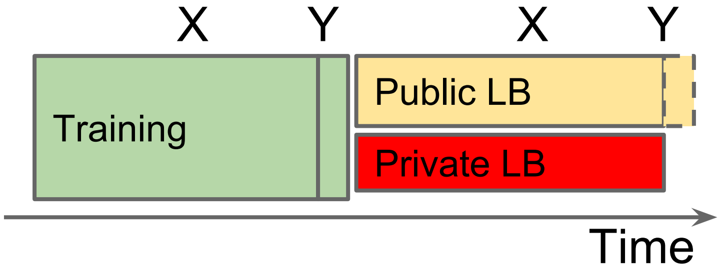
- Exception to the rule: cross validation when data extremely small
    - But keep in mind that your model won’t perform as well in reality
    - The more times you “tweak” your model, the bigger the gap.

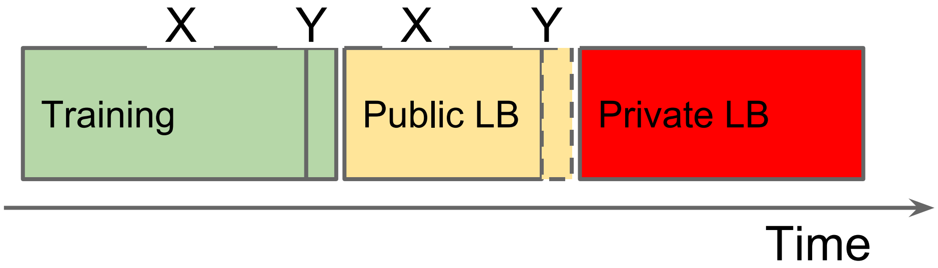
DataRobot

Kaggle Competitions -- Typical Data Partitioning



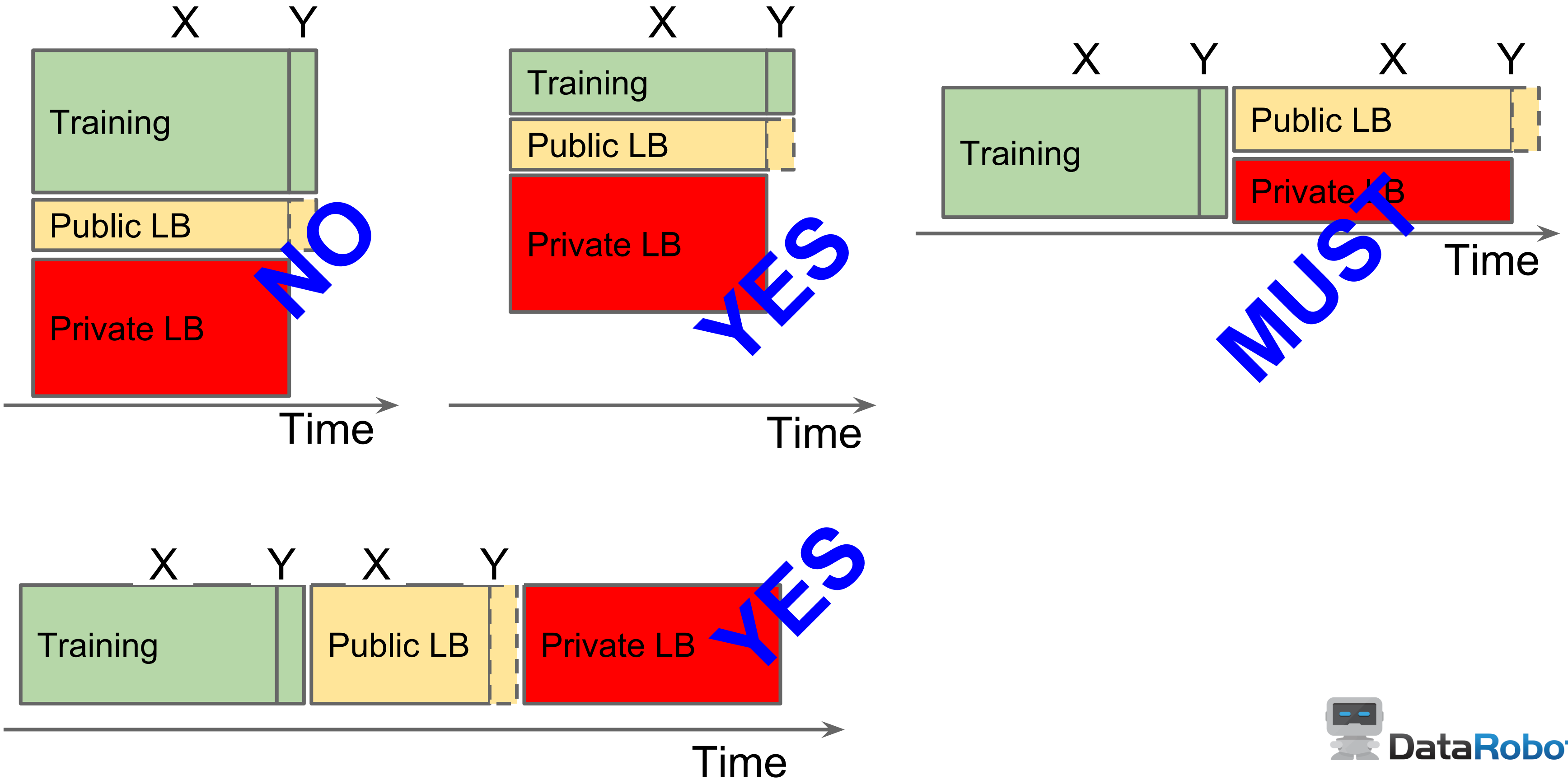






- When should we use Public LB feedback to tune our models?

Kaggle Competitions -- Use PLB as Training?



DataRobot

Tools/techniques -- GBM

- My confession: I (over)use GBM
  - When in doubt, use GBM
- GBM automatically approximate
  - Non-linear transformations
  - Subtle and deep interactions
- GBM gracefully treats missing values
- GBM is invariant to monotonic transformation of features

GBDT Hyper Parameter Tuning

Hyper Parameter	Tuning Approach	Range	Note
# of Trees	Fixed value	100-1000	Depending on datasize
Learning Rate	Fixed => Fine Tune	[2 - 10] / # of Trees	Depending on # trees
Row Sampling	Grid Search	[.5, .75, 1.0]	
Column Sampling	Grid Search	[.4, .6, .8, 1.0]	
Min Leaf Weight	Fixed => Fine Tune	3/(% of rare events)	Rule of thumb
Max Tree Depth	Grid Search	[4, 6, 8, 10]	
Min Split Gain	Fixed	0	Keep it 0

Best GBDT implementation today: https://github.com/tqchen/xgboost
by **Tianqi Chen** (U of Washington)

DataRobot

Tools/techniques -- data preprocessing for GBDT

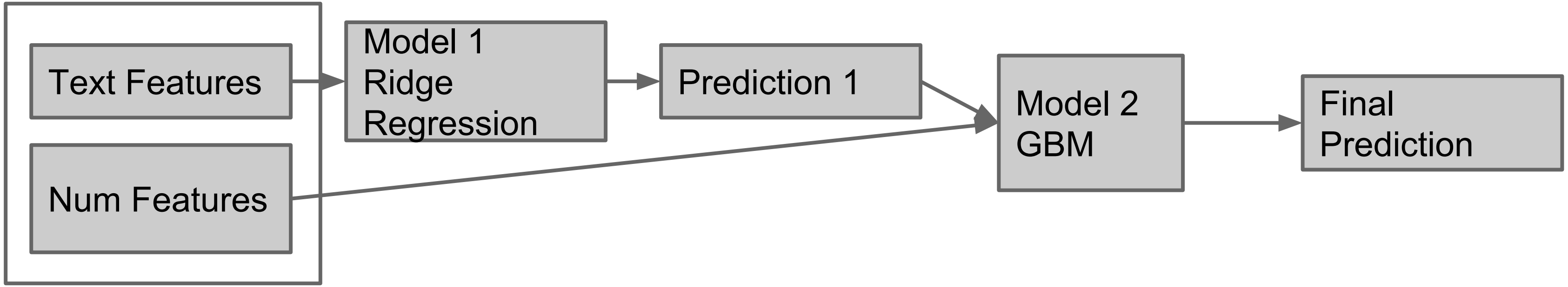
- High cardinality features
    - These are very commonly encountered -- zip code, injury type, ICD9, text, etc.
    - Convert into numerical with preprocessing -- out-of-fold average, counts, etc.
    - Use Ridge regression (or similar) and
        - use out-of-fold prediction as input to GBM
        - or blend
    - Be brave, use N-way interactions
        - I used 7-way interaction in the Amazon competition.

- GBM with out-of-fold treatment of high-cardinality feature performs very well

DataRobot

Technical Tricks -- Stacking

- Basic idea -- use one model's output as the next model's input



- It is NOT a good idea to use in sample prediction for stacking
    - The problem is over-fitting
    - The more "over-fit" prediction1 is , the more weight it will get in Model 2

Technical Tricks -- Stacking -- OOS / CV

- Use out of sample predictions
  - Take half of the training data to build model 1
  - Apply model 1 on the rest of the training data, use the output as input to model 2

- Use cross-validation partitioning when data limited
  - Partition training data into K partitions
  - For each of the K partition, compute “prediction 1” by building a model with OTHER partitions

DataRobot

Technical Tricks -- feature engineering in GBM

- GBM only APPROXIMATE interactions and non-linear transformations
- Strong interactions benefit from being explicitly defined
    - Especially ratios/sums/differences among features
- GBM cannot capture complex features such as "average sales in the previous period for this type of product"

DataRobot

Technical Tricks -- Glmnet

- From a methodology perspective, the opposite of GBM
- Captures (log/logistic) linear relationship
- Work with very small # of rows (a few hundred or even less)
- Complements GBM very well in a blend
- Need a lot of more work
  - missing values, outliers, transformations (log?), interactions
- The sparsity assumption -- L1 vs L2

Technical Tricks -- Text mining

- tau package in R
- Python’s sklearn
- L2 penalty a must
- N-grams work well.
- Don’t forget the “trivial features”: length of text, number of words, etc.
- Many “text-mining” competitions on kaggle are actually dominated by structured fields -- KDD2014

Technical Tricks -- Blending

- All models are wrong, but some are useful (George Box)
  - The hope is that they are wrong in different ways
- When in doubt, use average blender
- Beware of temptation to overfit public leaderboard
  - Use public LB + training CV
- The strongest individual model does not necessarily make the best blend
  - Sometimes intentionally built weak models are good blending candidates -- Liberty Mutual Competition

DataRobot

Technical Tricks -- blending continued

- Try to build “diverse” models
  - Different tools -- GBM, Glmnet, RF, SVM, etc.
  - Different model specifications -- Linear, lognormal, poisson, 2 stage, etc.
  - Different subsets of features
  - Subsampled observations
  - Weighted/unweighted
  - …
- But, do not “peek at answers” (at least not too much)

Apply what we learn outside of competitions

- Competitions give us really good models, but we also need to
  - Select the right problem and structure it correctly
  - Find good (at least useful) data
  - Make sure models are used the right way

Competitions help us

- Understand how much “signal” exists in the data
- Identify flaws in data or data creation process
- Build ***generalizable*** models
- Broaden our technical horizon
- …

Case 1 -- Amazon User Access competition

- One of the most popular competitions on Kaggle to date
  - 1687 teams

- Use anonymized features to predict if employee access request would be granted or denied

- All categorical features
  - Resource ID / Mgr ID / User ID / Dept ID …
  - Many features have high cardinality

- But I want to use GBM

Case 1 -- Amazon User Access competition

- Encode categorical features using observation counts
  - This is even available for holdout data!
- Encode categorical features using average response
  - Average all but one (example on next slide)
  - Add noise to the training features

- Build different kind of trees + ENET
  - GBM + ERT + ENET + RF + GBM2 + ERT2
- I didn't know VW (or similar), otherwise might have got better results.

- https://github.com/owenzhang/Kaggle-AmazonChallenge2013

DataRobot

Case 1 -- Amazon User Access competition

"Leave-one-out" encoding of categorical features:

Split	User ID	Y	mean(Y)	random	Exp_UID
Training	A1	0	.667	1.05	0.70035
Training	A1	1	.333	.97	0.32301
Training	A1	1	.333	.98	0.32634
Training	A1	0	.667	1.02	0.68034
Test	A1	-	.5	1	.5
Test	A1	-	.5	1	.5
Training	A2	0			

DataRobot

Case 2 -- Allstate User Purchase Option Prediction

- Predict final purchased product options based on earlier transactions.
    - 7 correlated targets
- This turns out to be very difficult because:
    - The evaluation criteria is all-or-nothing: all 7 predictions need to be correct
    - The baseline “last quoted” is very hard to beat.
        - Last quoted 53.269%
        - #3 (me) : 53.713% (+0.444%)
        - #1 solution 53.743% (+0.474%)

- Key challenges -- capture correlation, and not to lose to baseline

DataRobot

Case 2 -- Allstate User Purchase Option Prediction

- Dependency -- Chained models
  - First build stand-alone model for F
  - Then model for G, given F
  - F => G => B => A => C => E => D
  - “Free models” first, “dependent” model later
  - In training time, use actual data
  - In prediction time, use most likely predicted value

- Not to lose to baseline -- 2 stage models
  - One model to predict which one to use: chained prediction, or baseline

DataRobot

Case 3 -- Liberty Mutual fire loss prediction

DATA OVERVIEW

- ~1 million insurance records
- 300 variables:

target : The transformed ratio of loss to total insured value

id : A unique identifier of the data set

dummy : Nuisance variable used to control the model, but not a predictor

var1 – var17 : A set of normalized variables representing policy characteristics

crimeVar1 – crimeVar9 : Normalized Crime Rate variables

geodemVar1 – geodemVar37 : Normalized geodemographic variables

weatherVar1 – weatherVar236 : Normalized weather station variables

Numeric Variable Name	Variable Type
target	Continuous
id	Discrete
var10	Continuous
var11	Continuous
var12	Continuous
var13	Continuous
var14	Continuous
var15	Continuous
var16	Continuous
var17	Continuous
crimeVar1 – crimeVar9	Continuous
geoDemVar1 – geoDemVar37	Continuous
weatherVar1 – weath	

Categorical Variable Name	Variable Type	Possible Values
var1	Ordinal	1, 2, 3, 4, 5, Z*
var2	Nominal	A, B, C, Z*
var3	Ordinal	1, 2, 3, 4, 5, 6, Z*
var4+	Nominal	A1, B1, C1, D1, D2, D3, D4, E1, E2, E3, E4, E5, E6, F1, G1, G2, H1, H2, H3, I1, J1, J2, J3, J4, J5, J6, K1, L1, M1, N1, O1, O2, P1, R1, R2, R3, R4, R5, R6, R7, R8, S1, Z*
var5	Nominal	A, B, C, D, E, F, Z*
var6	Nominal	A, B, C, Z*
var7	Ordinal	1, 2, 3, 4, 5, 6, 7, 8, Z*
var8	Ordinal	1, 2, 3, 4, 5, 6, Z*
var9	Nominal	A, B, Z*
dummy	Nominal	A, B

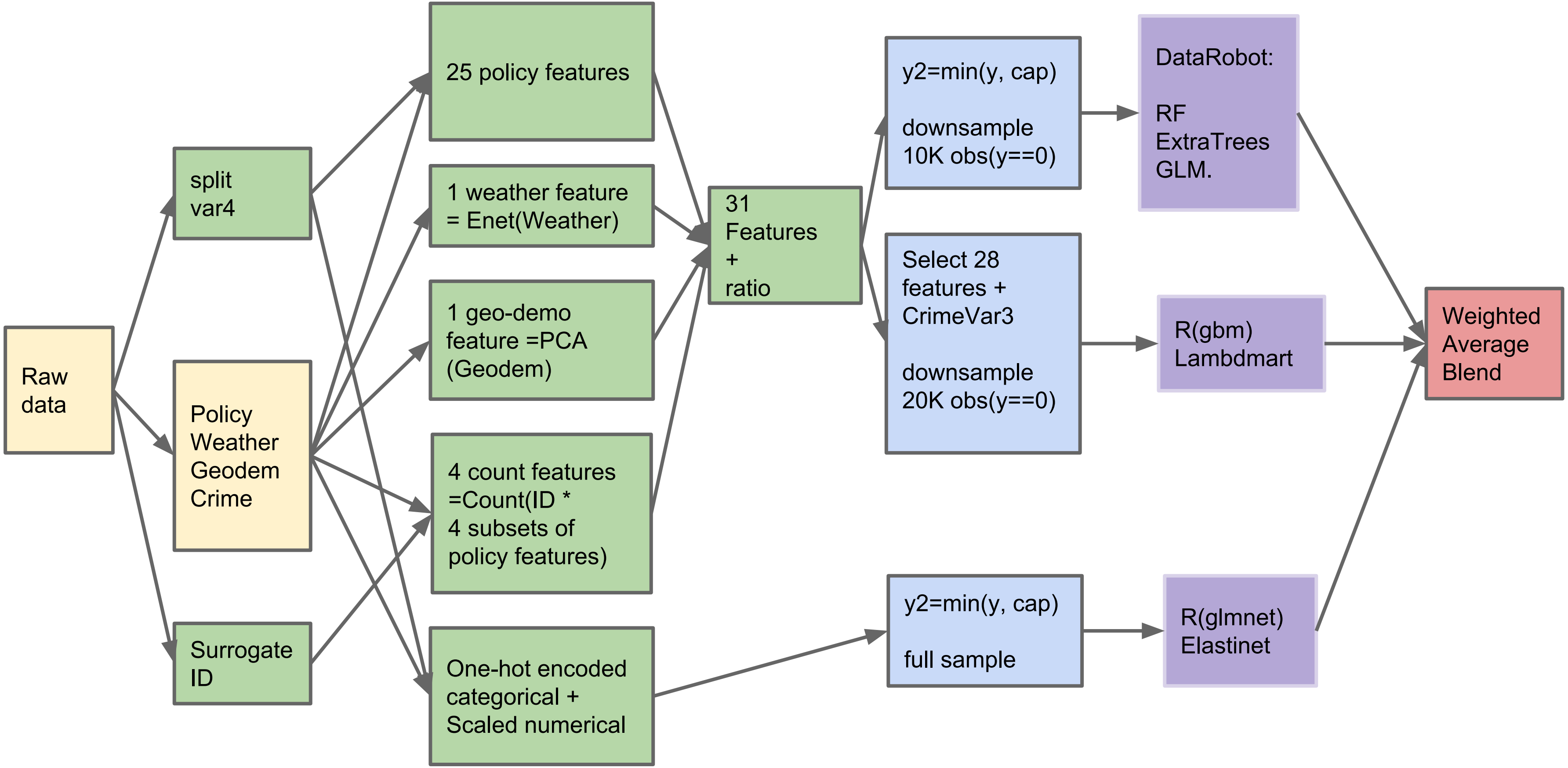
FEATURE ENGINEERING

- Broke feature set into 4 components
- Created surrogate ID based on identical crime, geodemographics and weather variables

Policy Characteristics	**Geodemographics**
<u>30 features</u>: - All policy characteristics features (17) - Split V4 into 2 levels (8) - Computed ratio of certain features - Combined surrogate ID and subsets of policy vars	<u>1 feature:</u> - Derived from PCA trained on scaled vars
Weather	**Crime Rate**
<u>1 feature:</u> - Derived from elasticnet trained on scaled variables	<u>0 features</u>

32 features

FINAL SOLUTION SUMMARY



Useful Resources

- http://www.kaggle.com/competitions
- http://www.kaggle.com/forums

- http://statweb.stanford.edu/~tibs/ElemStatLearn/

- http://scikit-learn.org/
- http://cran.r-project.org/
- https://github.com/JohnLangford/vowpal_wabbit/wiki
- ….