Don’t Get Kicked!

Data Science Challenge

21 August 2022

The Problem

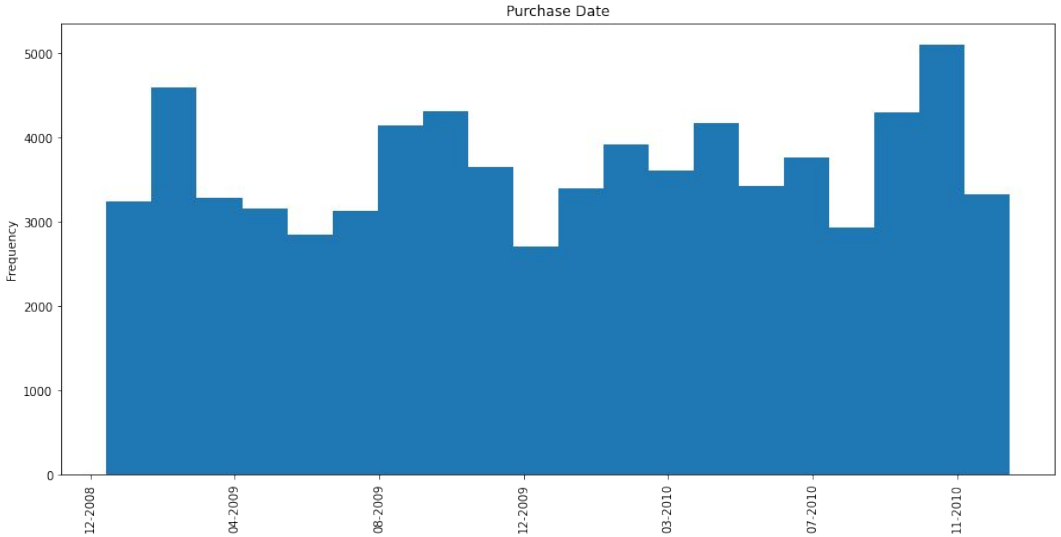
Can we predict whether a car purchased at auction is a lemon?

What's a Lemon?

Vehicle "kicked" back to auction after purchase due to unexpected issues.

The Data

- Purchase-level auction data from Jan 2009 to Dec 2010
- Binary outcome for whether a purchase was an “avoidable bad buy”
- 32 Independent Variables, not all useful



Sampling Strategy

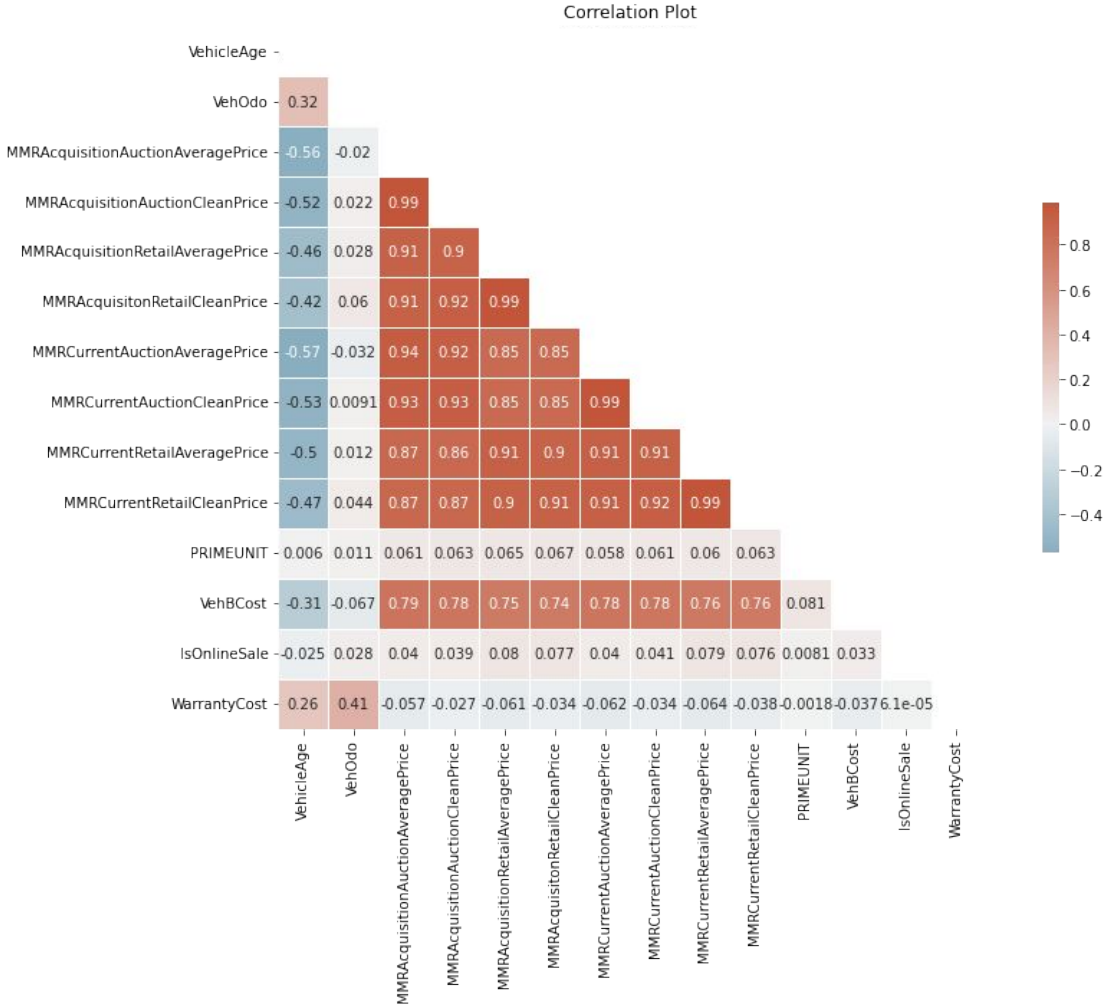
Data split into separate training, validation, and test sets at a 70-15-15 split, stratifying on target

Split	n False	n True
Train	44,803	6,283
Validation	9,601	1,347
Test	9,602	1,346

Numerical Features

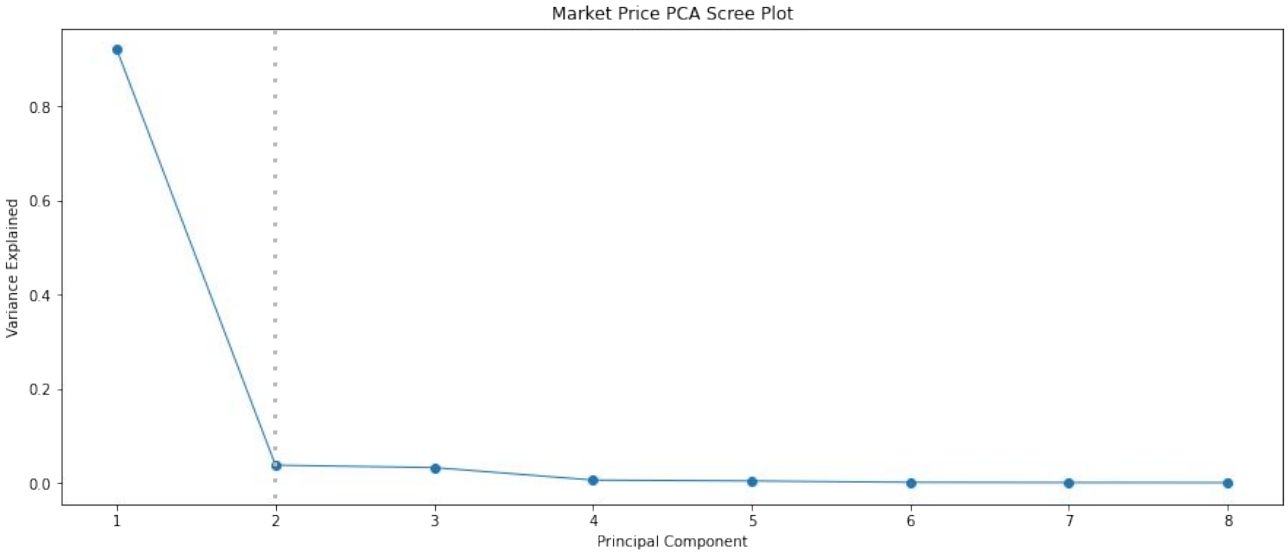
We have quite a few numerical features, many of which are highly correlated with one another.

Market prices, especially, seem to be coupled tightly.



Solution: PCA

To address this multicollinearity, we will reduce our market features using Principal Component Analysis.



Categorical Features

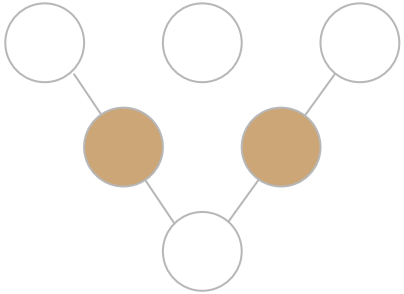
Categorical features present their own issues. Many express high cardinality.

This causes issues with generalized encoding.

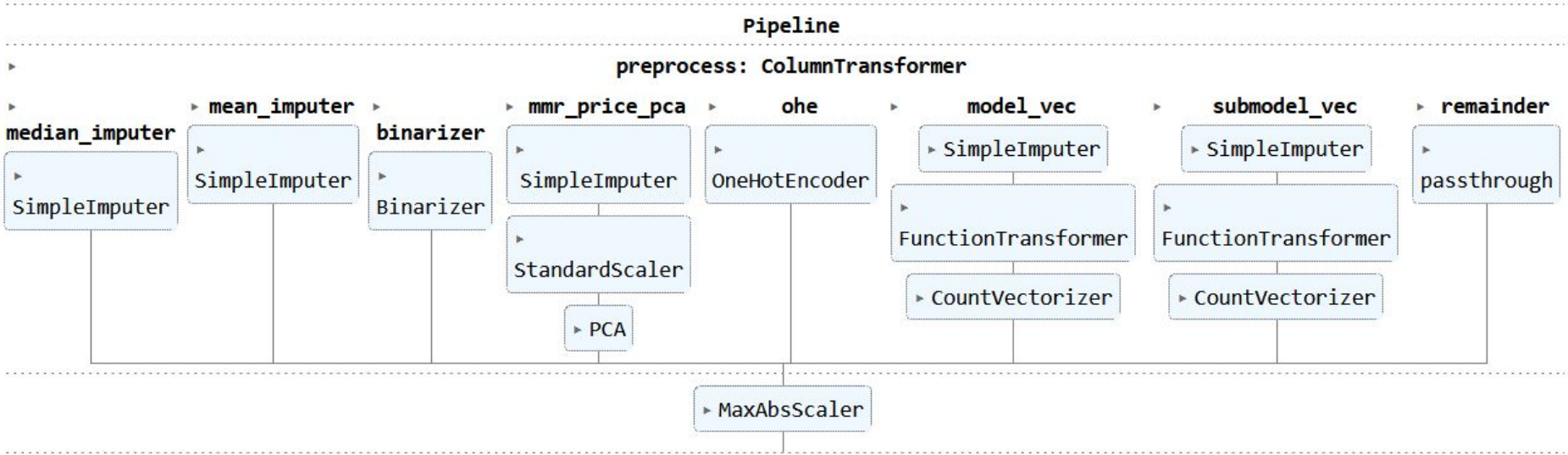
	index	cardinality
0	Auction	3
1	Make	33
2	Model	1008
3	Trim	131
4	SubModel	797
5	Color	16
6	Transmission	2
7	WheelType	3
8	Nationality	4
9	Size	12
10	TopThreeAmericanName	4
11	AUCGUART	2
12	BYRNO	73
13	VNZIP1	152
14	VNST	37
15	PurchYear	2
16	PurchMonth	12

Solution: Smoothing

1. Limit categorical features to max 50 total levels and min 30 train observations
2. Interpret Model, SubModel as strings, reduce to words and limit vocabulary to top 50
3. Apply downstream regularization

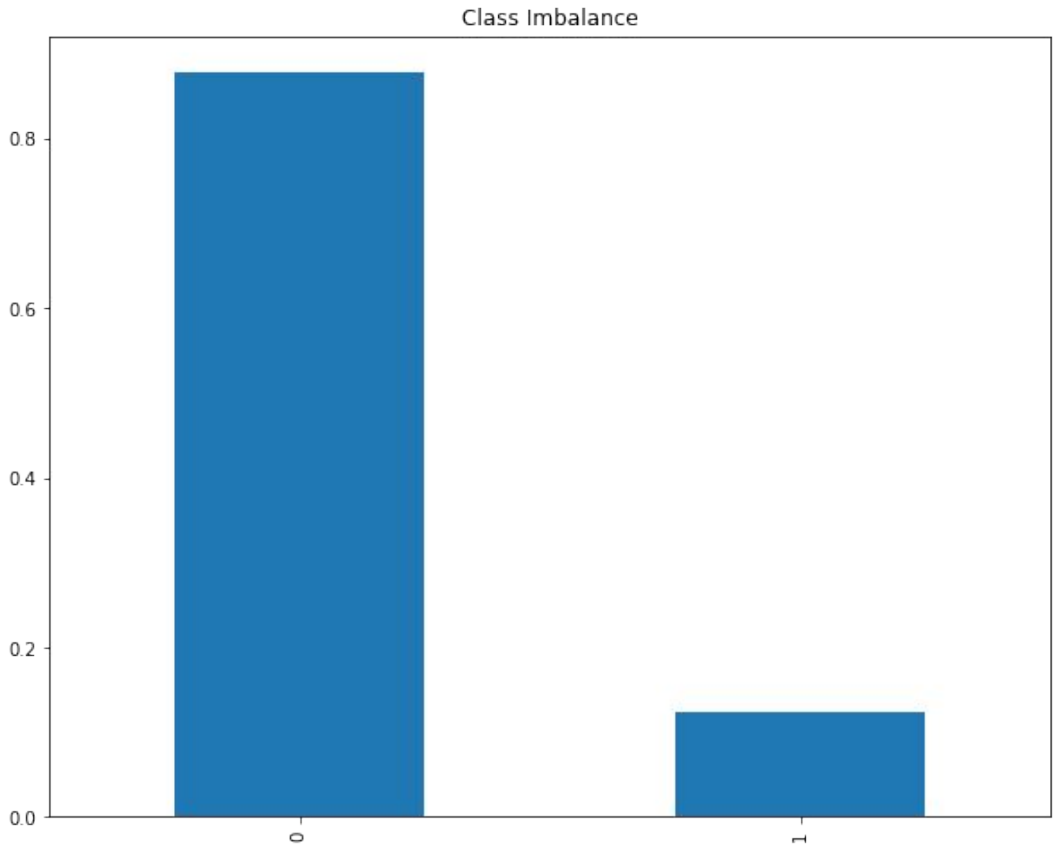


Full Pipeline



Class Imbalance

Our data contain mostly negative labels, which could bias our model towards the negative case.



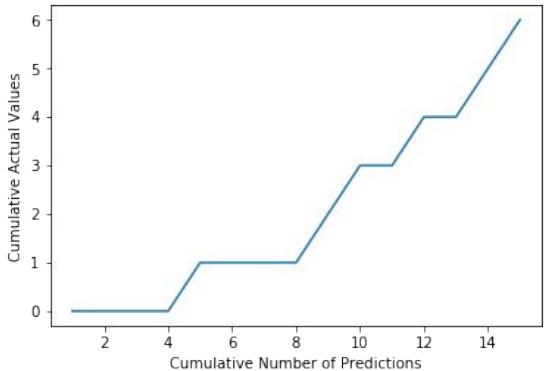
Solution: Class Weights

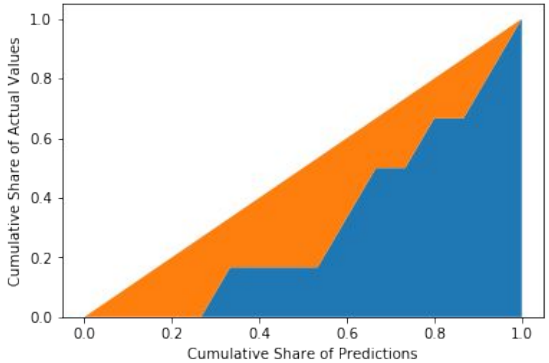
Weigh our loss by inverse class occurrence during training, scaled to 1 to accommodate XGB.

Label	Weight
1	7.13
0	1

Evaluation Metrics

1. Binary Accuracy: what % do we correctly predict?
2. Precision: what % of our predicted lemons are actually lemons?
3. Recall: what % true lemons do we accurately predict?
4. AUC-ROC: captures trade-off between recall and false positives
5. Avg. Precision (AUC-PR): captures trade-off between recall and precision
6. Gini Index: Kaggle's scoring measure; how well can we discriminate lemons?





Source: https://www.kaggle.com/code/batzner/gini-coefficient-an-intuitive-explanation

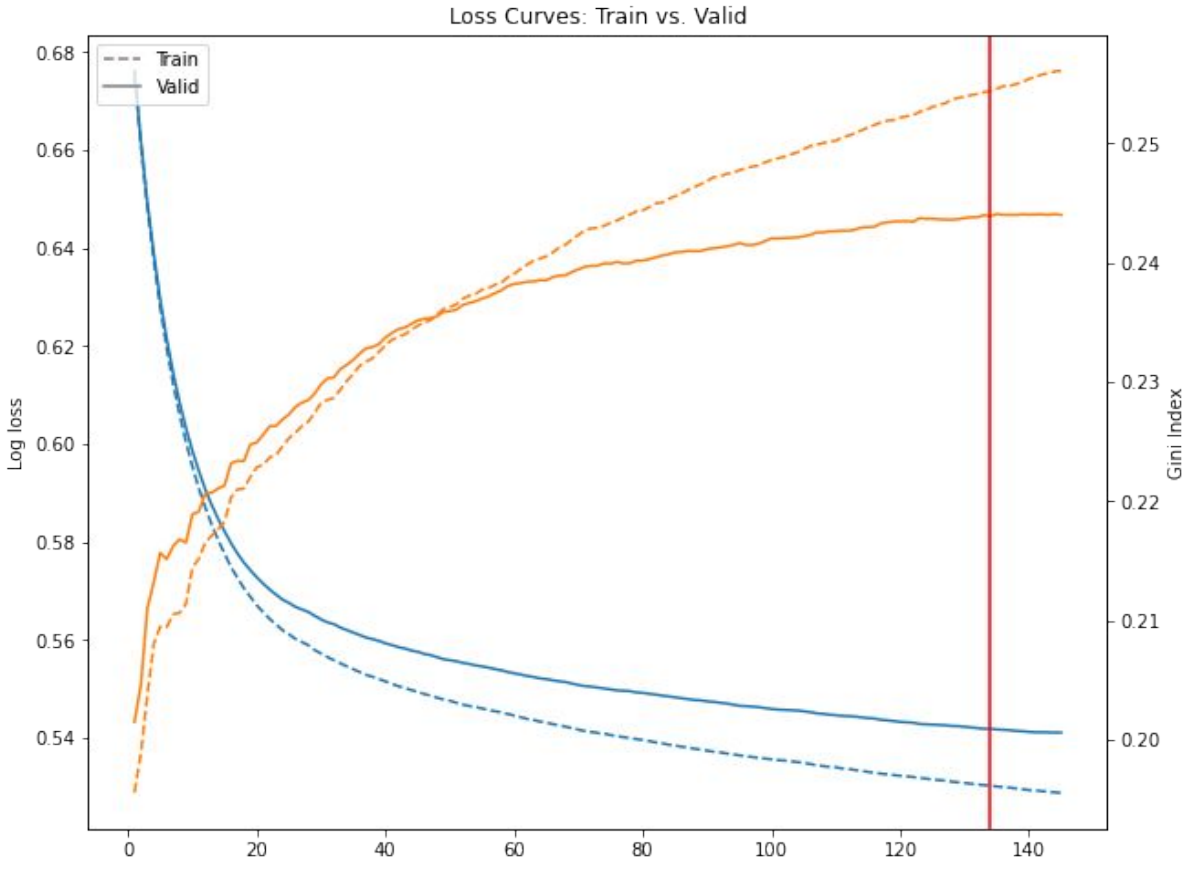
Time to Model

1. Logistic Regression
2. K Nearest Neighbors (KNN)
3. Random Forest
4. Extreme Gradient Boosting (XGB)

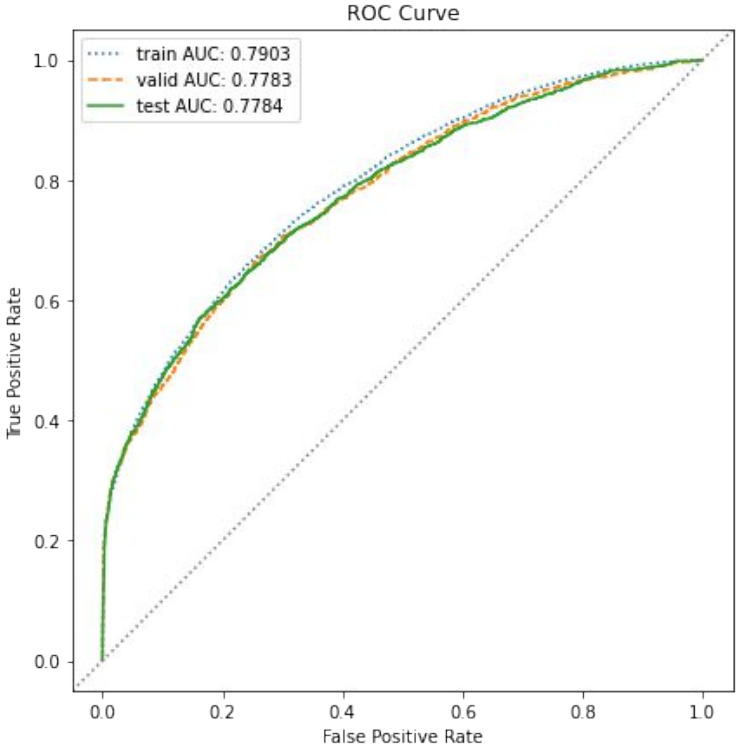
	algorithm	train accuracy	valid accuracy	train precision	valid precision	train recall	valid recall	train auc roc	valid auc roc	train avg. precision	valid avg. precision	train gini index	valid gini index	exec. time
0	logistic regression	0.7438	0.7379	0.2727	0.2640	0.6498	0.6318	0.7810	0.7703	0.4467	0.4533	0.2464	0.2371	0.964842
1	knn	0.9045	0.8603	0.7691	0.3064	0.3196	0.1069	0.9307	0.5847	0.5507	0.1663	0.3777	0.0781	171.512491
2	rf	0.6986	0.6854	0.2253	0.2178	0.5949	0.6006	0.7233	0.7217	0.3655	0.3717	0.1955	0.1947	1.331460
3	xgb	0.7567	0.7463	0.2839	0.2741	0.6430	0.6444	0.7903	0.7783	0.4912	0.4802	0.2546	0.2440	4.551598

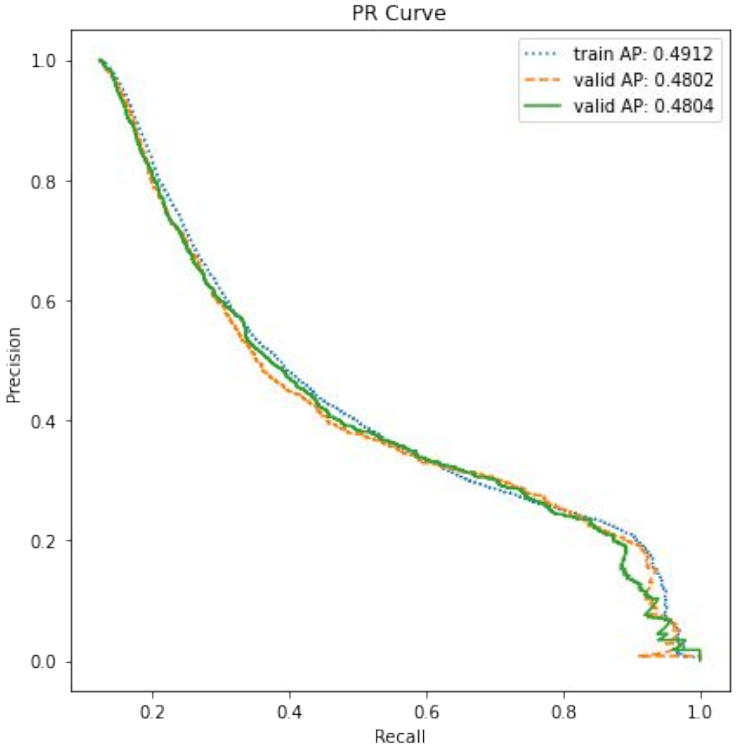
XGBoost in Depth

1. Learning rate: 0.1
2. Max depth: 3
3. Regularization: early stopping on gini index, patience of 10 epochs, tolerance of 0.01

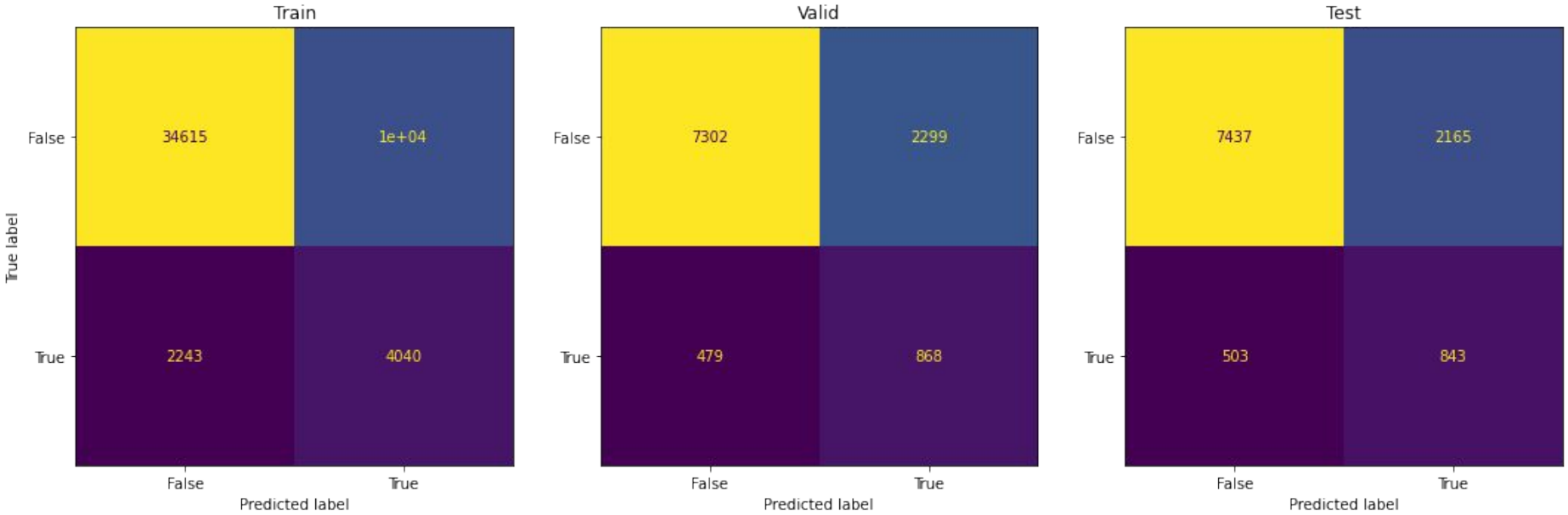


Evaluation: Does our Model Generalize?





Summing it Up: Classification Ability



	label	accuracy	precision	recall	auc roc	avg. precision	gini index
0	xgb test perf.	0.7563	0.2803	0.6263	0.7784	0.4804	0.2441

This Gini Index would place us in the:

86th Percentile

Prior to hypertuning

Next Steps

Hyperparameter Tuning: further optimize performance over a wider search space

Feature Importance Extraction: use SHAP, etc. to better explain predictions

Ablation Study: systematically eliminate preprocessing/modeling elements to improve parsimony

Cost Analysis: reduce predictions to $ costs