Early Warnings for Bank Failure

Andrew Nguyen

Federal Reserve Board of Governors, Federal Reserve Bank of Chicago (Gutierrez, Luo, Tewolde)

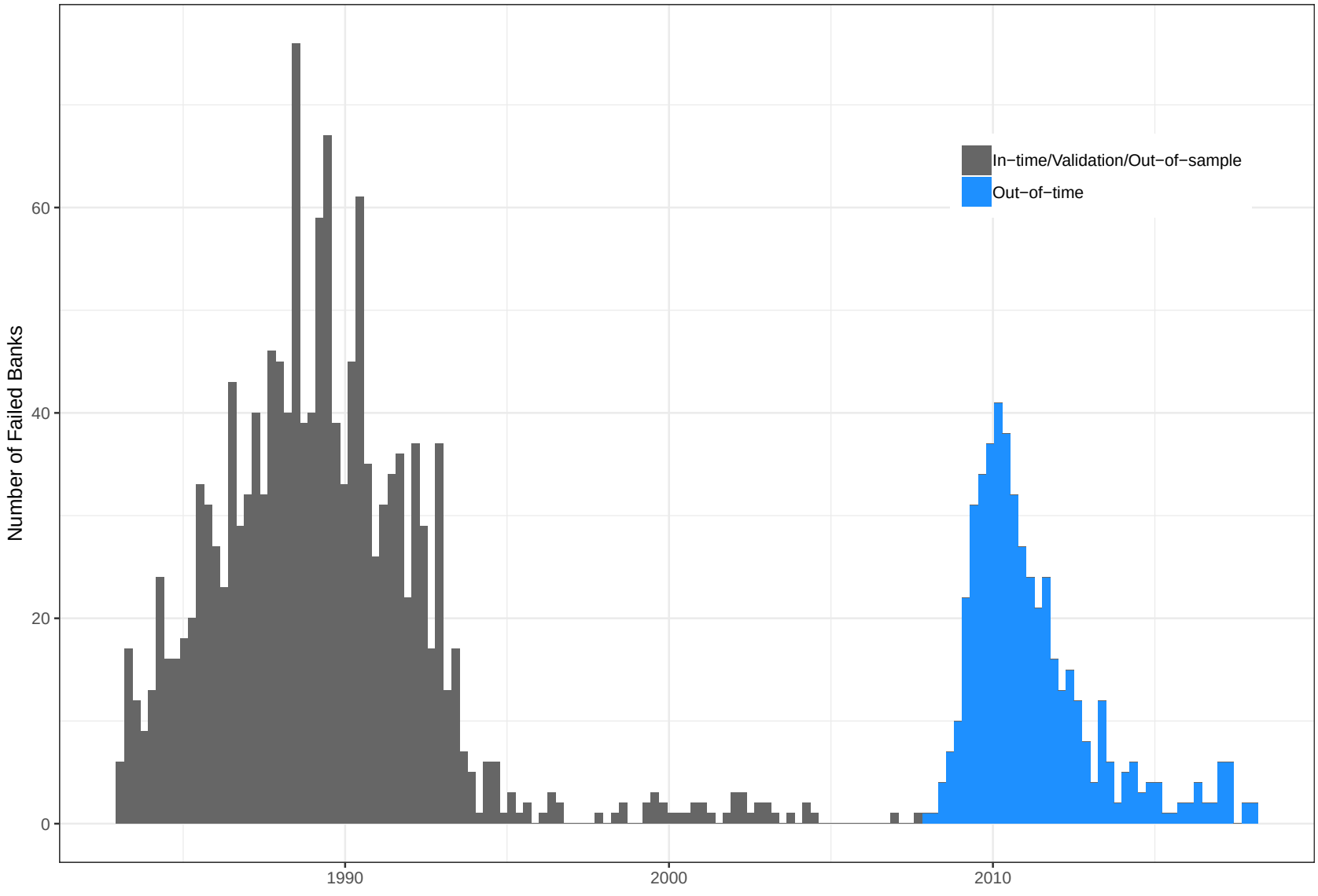
Background

- The Federal Reserve acts as bank supervisor and regulator
- Bank failure forecasting is an extremely non-linear problem
- We compare logistic regression vs machine learning models
- Use a broad array of explanatory variables:
    1. Balance sheet and income statement information (call-report)
    2. State-level leading indicators
    3. Aggregate failure rate
    4. Market information

Main Considerations

- What is the best model?
  - Data modelers find this interesting
- We also want to explain results from each model
  - Modelers and policymakers care

Data


In−time/Validation/Out−of−sample
Out−of−time
Number of Failed Banks
0
20
40
60
1990
2000
2010

Results: Model AUC

Sample/ Model	Failure Forecasting Horizon			
Training	**1 Quarter**	**2 Quarters**	**1 Year**	**2 Years**
Logistic				0.8840
RF				**0.9970**
GB				0.8386
SVM				0.8021
DEEP				0.9300
Validation	**1 Quarter**	**2 Quarters**	**1 Year**	**2 Years**
Logistic				0.8460
RF				0.8734
GB				0.8432
SVM				0.8095
DEEP				**0.8900**
Out of Sample	**1 Quarter**	**2 Quarters**	**1 Year**	**2 Years**
Logistic	0.9480	**0.9660**	0.9270	0.8770
RF	0.9120	0.9210	0.9130	0.8740
GB	0.9068	0.9260	0.9169	0.8521
SVM	0.9335	0.9299	0.9104	0.8307
DEEP	**0.9500**	**0.9500**	**0.9400**	**0.8900**
Out of Time	**1 Quarter**	**2 Quarters**	**1 Year**	**2 Years**
Logistic	**0.9700**	**0.9560**	0.9390	**0.9070**
RF	0.9330	0.9189	0.8820	0.8010
GB	0.9278	0.9148	0.8967	0.8136
SVM	0.9305	0.9214	0.9066	0.8645
DEEP	0.9600	0.9500	**0.9500**	0.9000

Linear Probability Models

$$P(y) = \alpha + \beta_1 x_1 + \beta_2 x_2$$

- Really easy to explain - we always know β_1 explains a one unit change in x_1

Non-linear Models

Consider the logistic regression model

$$P(y) = \frac{1}{1 + e^{-(\alpha+\beta_1 x_1+\beta_2 x_2)}}$$

- Not easy to explain - we know that $\frac{\beta_1 e^{\alpha+\beta_1 x_1+\beta_2 x_2}}{1+e^{\alpha+\beta_1 x_1+\beta_2 x_2}^2}$ explains a one unit change in x_1

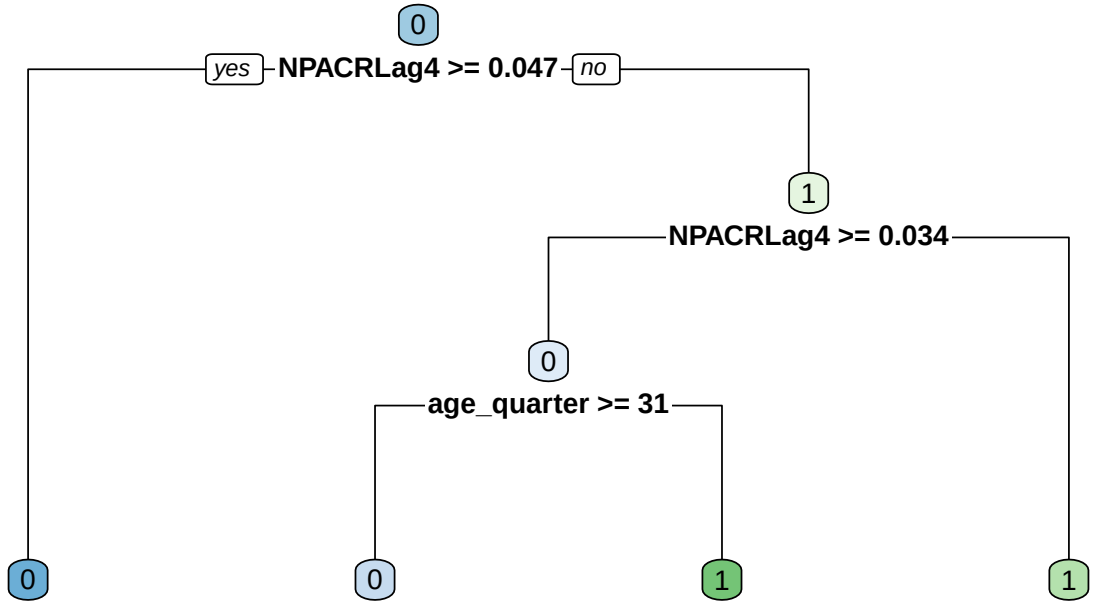
Problematic because this relies on the level of x_1, x_2

Non-linear Models

- No global way to say what effect x_1 has on $P(y)$
- Logistic regression is our simplest non-linear model, what about something even more complex?
- We have some options!

A Single Tree

- Minimize Gini-impurity
  $G = p_{fail} * (1 - p_{fail}) + p_{notFailed} * (1 - p_{notFailed})$
- How often a randomly chosen element would be incorrectly labeled if it was randomly labeled according to the model?



Random Forests (RF)

- Build a bunch of weakly-correlated trees
- The average based on those many trees will significantly reduce the variance over any single tree

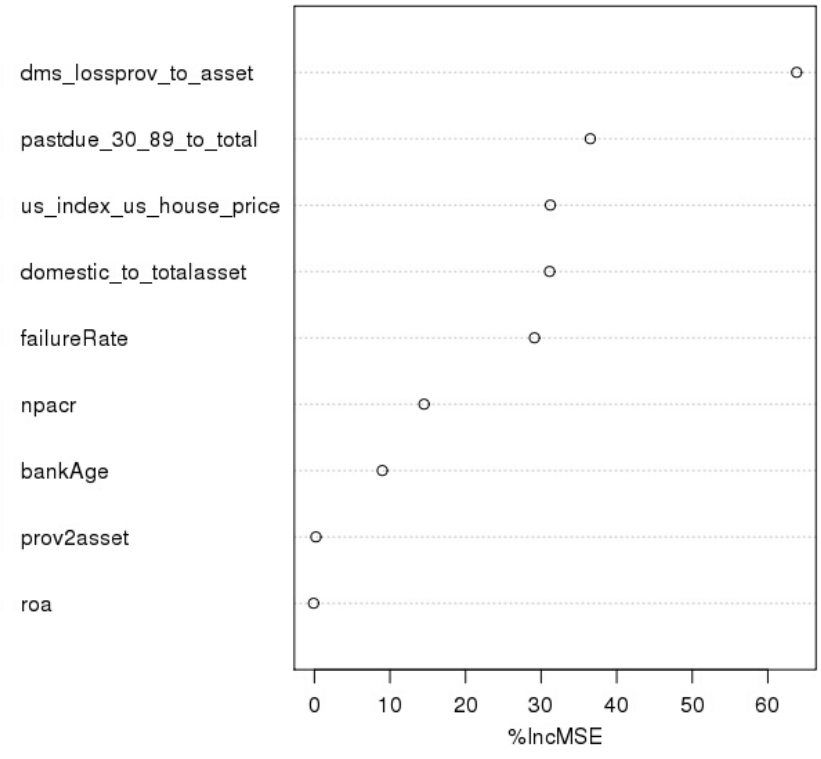
1. Randomly draw a sample with replacement from the original sample
2. Randomly draw a subset of predictors for each tree
3. Average the prediction over all trees

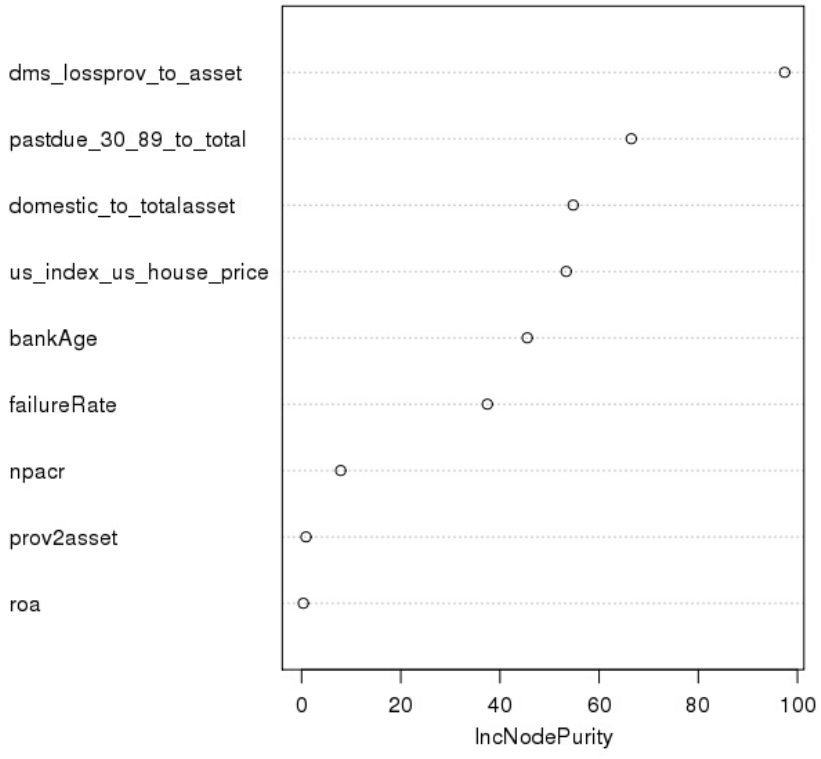
- How can we do inference on this tree?

Permutation Importance

- Basic idea: drop one feature and re-train, how do our fit statistics perform now?
- Re-training over and over is prohibitively expensive
- Instead: replace the original feature with noise, and re-run through prediction

Permutation Importance Plot

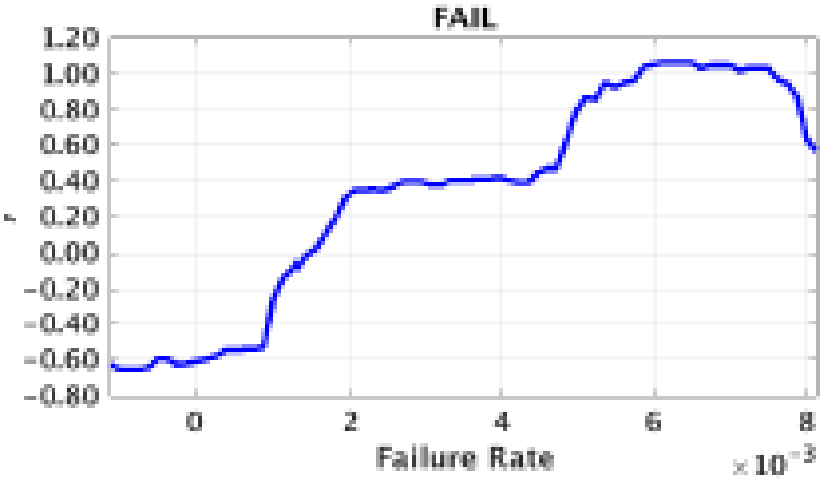


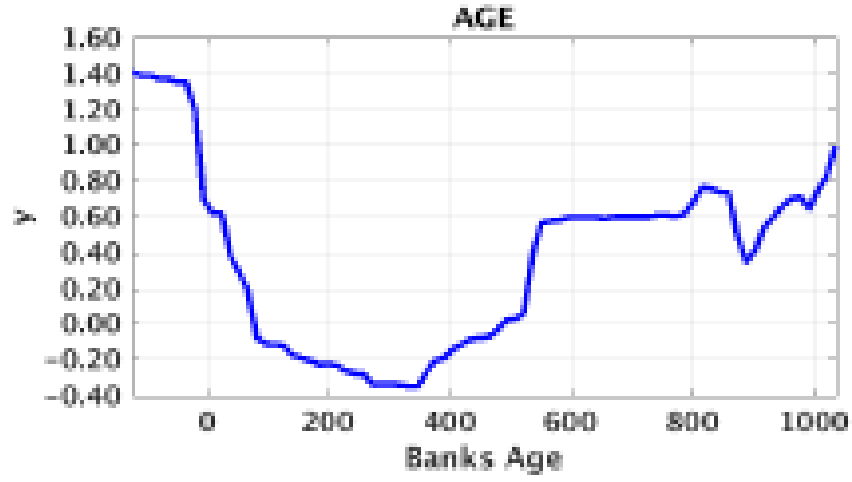


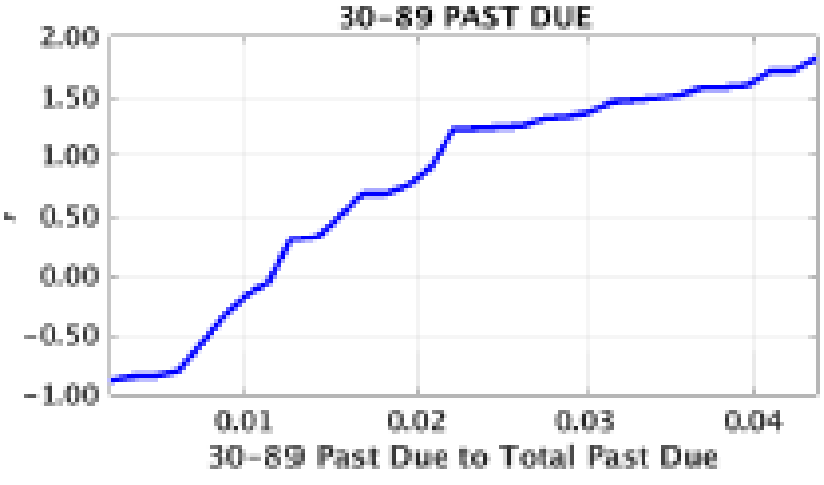
Partial Dependence

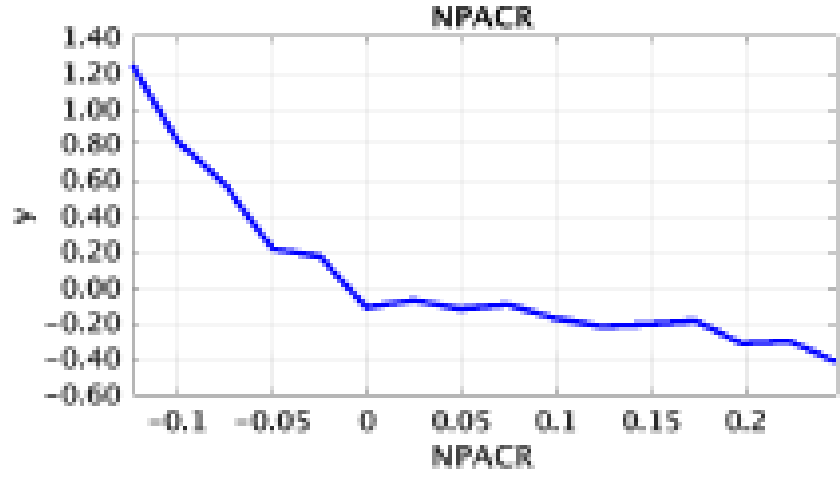
- For a single observation, hold all but one variable x_1 constant
- Record how much P_{fail} changes as a results of x_1
- Repeat this for a sample of observations
- Interpreted as the average partial effect, $\frac{\Delta \bar{P}_{fail}}{\Delta x_1}$

Partial Dependence Plot









LIME

- Local Interpretable Model-Agnostic Explanations (LIME)
- Uses a series of small linear approximations to approximate the complex decision function of a neural network
- A model of a model for a *single observation*

LIME

- For bank i by randomly perturbing the inputs x_i
- Estimate a LASSO regression from the perturbed inputs

$$P(failure_i) = \sum_{n=1}^{k} \tilde{x}_{i,n} \beta_{i,n} \omega_{i,n}$$

- k is chosen by the researcher, and $\tilde{x}_{i,n}$ is chosen by the LASSO procedure
- The weight $\omega_{i,n}$ is the distance from the original obsevation

LIME Outputs

- For a single observation
  - k most important features
  - Predictions of NNet probabilities `lime_prob`
  - Contribution of features to `lime_prob`
- Bank 1

nnet_prob	lime_prob	intercept	equity2assets	npacr	over89_to_pastdue	us_index_us_house_price
0.76	0.61	0.21	0.21	0.27	-0.01	-0.08

- Bank 2

nnet_prob	lime_prob	intercept	equity2assets	npacr	failureRate	ncloan_to_loan
0.85	0.82	0.15	0.21	0.28	-0.01	0.2

LIME: Estimating A NN

```
library(parsnip)
library(keras)

keras_fit <- logistic_reg() %>%
    set_engine("keras"
               , epochs = 1000
               , batch_size = 32
               , act = 'relu'
               , hidden_units = 10) %>%
    fit(fail ~ ., data = train_data)
```

LIME: Evaluating Fit

```
library(lime)

explainer <- lime(
  , x = train_data %>% select(-fail)
  , model = keras_fit
  , bin_continuous = FALSE)


explanation_df <- explain(
  x = test_data %>% select(-fail)
  , explainer = explainer
  , n_labels = 1 # explaining a single
  , n_features = 4 # returns top four features
  )
```

Machine Learning Interpretations

Bad:

- None of these techniques is as easy to explain as linear-regression
- Yet another set of hyperparameters to choose

Good:

- All of the shown techniques are model agnostic
- A maturing software ecosystem