Programming with Data

Session 9: LASSO and Tree Ensembles for Fraud Detection

Dr. Wang Jiwei
Master of Professional Accounting

Corporate/Securities Fraud

Typical accounting fraud

- A company is underperforming, motivated to inflate earnings or reduce expenses
  - Worldcom (1999-2002)
    - Capitalizing costs (should be expensed)
  - Olympus (late 1980s-2011): Hide losses in a separate entity
    - "Tobashi scheme"
  - Wells Fargo (2011-2018?)
    - Fake/duplicate customers and transactions
    - "improperly encouraging" to actively trade
- A company is overperforming, motivated to save for future rainy days (smoothing of earning)
  - Dell (2002-2007)
    - Cookie jar reserves
  - Brystol-Myers Squibb (2000-2001)
    - Cookie jar reserves

Other accounting fraud types

- Apple (2001)
  - *Options backdating*
- Commerce Group Corp (2003)
  - Using an auditor that *isn't registered*
- Cardiff International (2017)
  - Releasing 10Q fin. statements that were *not reviewed by an auditor*
- China North East Petroleum
  - *Related party transactions* (transferring funds to family members)
- *Insufficient internal controls*
  - Citigroup (2008-2014) via Banamex
  - Asia Pacific Breweries
- *Bribery*
  - Keppel O&M (2001-2014): US$55M to Brazilian officials
  - Baker Hughes (2001, 2007: Payments to officials in Indonesia, and possibly to Brazil/India (2001), Angola/Indonesia/Nigeria/Russia/Uzbekistan (2007)
- ZZZZ Best (1982-1987): *Fake the whole company*
  - Also faked a real project to get a clean audit to take the company public

More interesting fraud types

- Bernard Madoff: Ponzi scheme
  1. Get money from individuals for "investments"
  2. Pretend as though the money was invested
  3. Use new investors' money to pay back anyone withdrawing
- Applied Wellness Corporation (2008)
  - Failed to file annual and quarterly reports
- Tesla (2018): Misleading statements on Twitter
  - Settled the charge in one week
- Am-Pac International (1997)
  - Auditors lacked independence: maintain books, generate FS, then audit their own work
- Keppel Club (2014)
  - Employees created 1,280 fake memberships, sold them, and retained all profits (SG$37.5M)

What will we look at today?

SMU SINGAPORE MANAGEMENT UNIVERSITY
School of Accountancy

> Misstatements: non-conmpliance with accounting standards, maybe an unintentional error but in many cases seemingly intentionally done by management or other employees at the firm (ie, fraud).

How do misstatements come to light?

1. The company/management admits to it publicly
2. A government entity forces the company to disclose
   - In more egregious cases, government agencies may disclose the fraud publicly as well
3. Investors sue the firm, forcing disclosure

Where are these disclosed? (US)

By the company:

1. 10-K/A filings (10-K: annual report, /A: amendment)
   - Big R: material corrections to previously issued financial statements
   - Note: not all 10-K/A filings are caused by fraud!
     - Audit Analytic's write-up on this
2. In a note inside a 10-K filing
   - Little r: immaterial corrections
3. In a press release, which is later filed with the US SEC as an 8-K
   - 8-Ks are filed for many other reasons too though

By the regulator:

1. By the SEC through the Section 13(b)
   - Section 13(b) of the Securities Exchange Act, commonly called the "books and records" provision, requires issuers to "make and keep books, records, and accounts, which, in reasonable detail, accurately and fairly reflect the transactions and dispositions of the assets of the issuer."
2. SEC AAERs: Accounting and Auditing Enforcement Releases
   - Generally highlight larger or more important cases

AAERs

- Today we will examine these AAERs
  - Using a proprietary data set of > 1,000 such releases
  - The data is available here

> To get a sense of the data we're working with, read the *Summary* section (starting on page 2) of this AAER against Sanofi SEC against Sanofi

Predicting Fraud

Main question

> How can we *detect* if a firm *is* involved in a major instance of missreporting?

- This is a pure forensic analytics question
- "Major instance of misreporting" will be implemented using AAERs

> Unfortunately we don't have an "Altman Z" type of measure from the academia yet, partly due to the complexity of fraudulent activities.

Approaches

- In these slides, we'll walk through the primary detection methods since the 1990s, up to currently used methods
- 1990s: Financials and financial ratios
  - Follow up in 2011
- Late 2000s/early 2010s: Characteristics of firm's disclosures
- mid 2010s: More holistic text-based measures of disclosures
  - This will tie to a future topic where we will explore how to work with text

> All of these are discussed in Section 2.1 of Brown, Crowley and Elliott (2020) -- we'll refer to the research paper as **BCE** for short

The data

- Preprocessed data from **BCE(2020)**
- It contains 401 variables from Compustat, CRSP, and SEC
  - Many precalculated measures including:
    - Firm characteristics, such as auditor type (**bigNaudit**)
    - Financial measures, such as total accruals (**rsst_acc**)
    - Financial ratios, such as ROA (**ni_at**)
    - AR characteristics, such as mean sentence length (**sentlen_u**)
    - ML based content analysis (everything with **Topic_** prepended)
- Testing data (2004) and training data (1999-2003) by variable **Test**
  - Testing == 1: testing data
  - Testing == 0: training data

> Reminder: data provided in the course are mainly through the WRDS subscription. You are prohibited by law from sharing the data with people outside of SMU community.

Event frequency

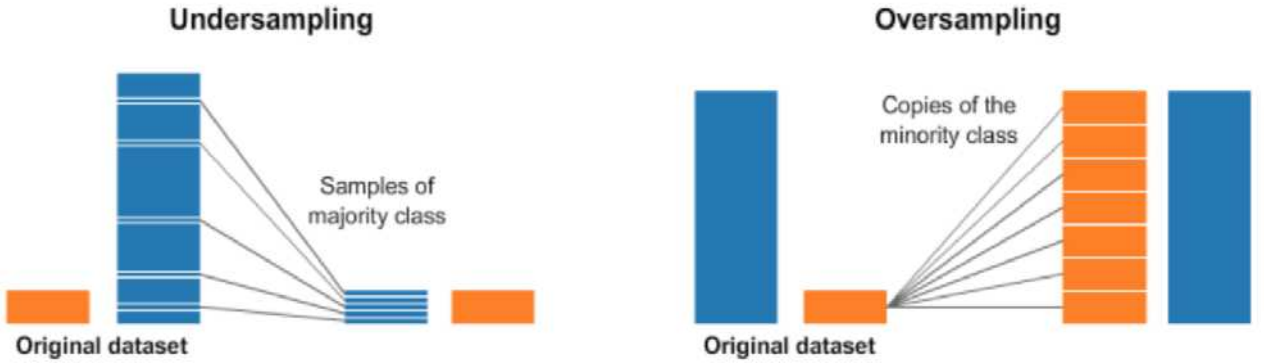
- Very low event frequencies (<10%) can make things tricky

```
df %>%
  group_by(year) %>%
  mutate(total_AAERS = sum(AAER), total_observations = n(),
         percent = scales::percent(total_AAERS/total_observations,
                                   accuracy = 0.01)) %>%
  slice(1) %>%  ungroup() %>%
  select(year, total_AAERS, total_observations, percent) %>% html_df
```

year	total_AAERS	total_observations	percent
1999	46	2195	2.10%
2000	50	2041	2.45%
2001	43	2021	2.13%
2002	50	2391	2.09%
2003	57	2936	1.94%
2004	49	2843	1.72%

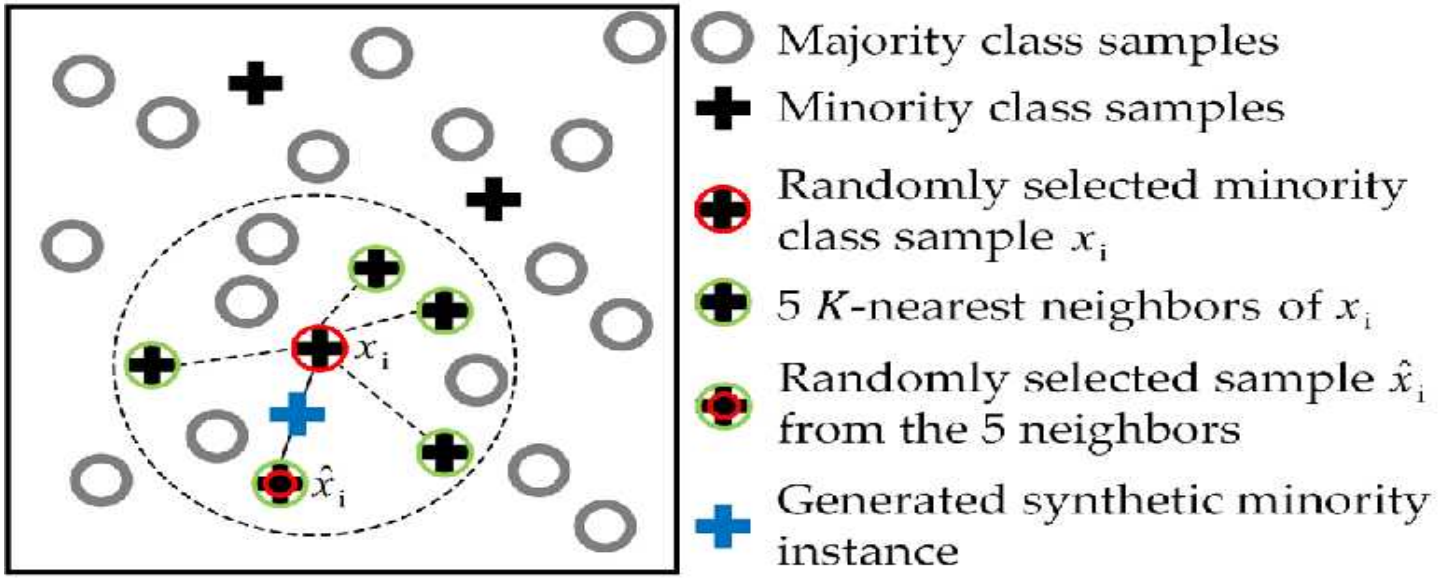
Dealing with imbalanced data

- imbalanced dataset: the number of obs belonging to one class is significantly lower than those belonging to the other classes.
- A few ways to handling imblanced dataset:
  - (1). Resampling to a balanced dataset (packages incl. `package:imbalance`, `package:ROSE`, `package:DMwR`, etc)



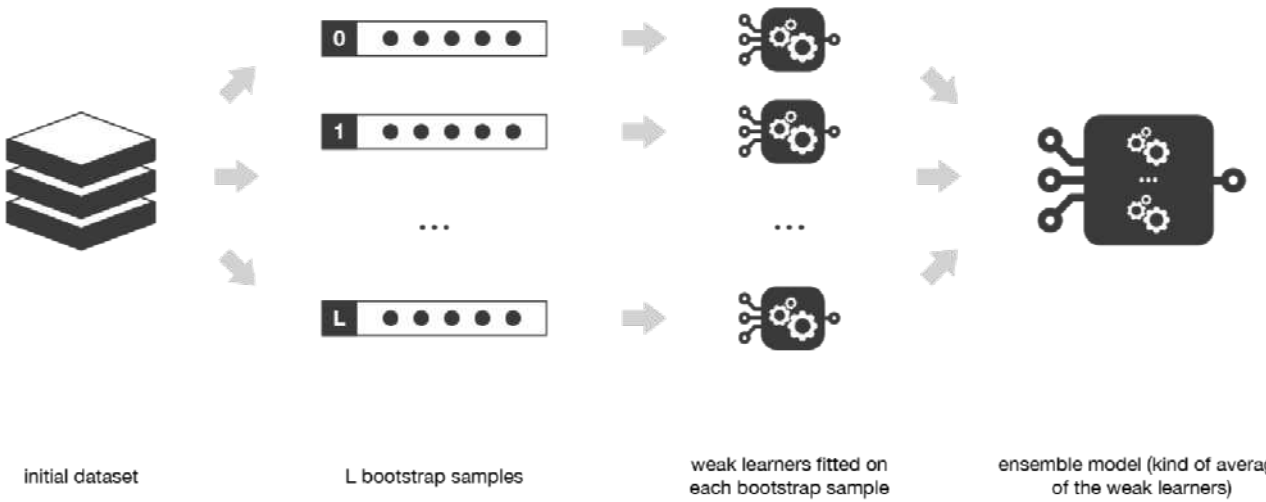
Oversampling by SMOTE

- Not good to use undersampling alone as it reduces data points.
- SMOTE: Synthetic Minority Over-sampling Technique
  - Combine oversampling minority and undersampling majority
  - Introducing synthetic examples along the line joining all of the **k nearest neighbors** of minority class.
  - `package:DMwR` and `package:smotefamily` have the *SMOTE()* function



Dealing with imbalanced data

- imbalanced dataset: the number of obs belonging to one class is significantly lower than those belonging to the other classes.
- A few ways to handling imblanced dataset:
  - (2). Algorithmic ensemble techniques such as bagging (eg, Random Forest) and boosting (eg, XGBoost)
    - we will try the ensembling techniques



1990s approach

The 1990s model

- Many financial measures and ratios can help to predict fraud

- EBIT
- Earnings / revenue
- ROA
- Log of liabilities
- liabilities / equity
- liabilities / assets
- quick ratio
- Working capital / assets
- Inventory / revenue
- inventory / assets
- earnings / PP&E
- A/R / revenue
- Change in revenue
- Change in A/R + 1
- $>$ 10% change in A/R
- Change in gross profit + 1
- $>$ 10% change in gross profit
- Gross profit / assets
- Revenue minus gross profit
- Cash / assets
- Log of assets
- PP&E / assets
- Working capital

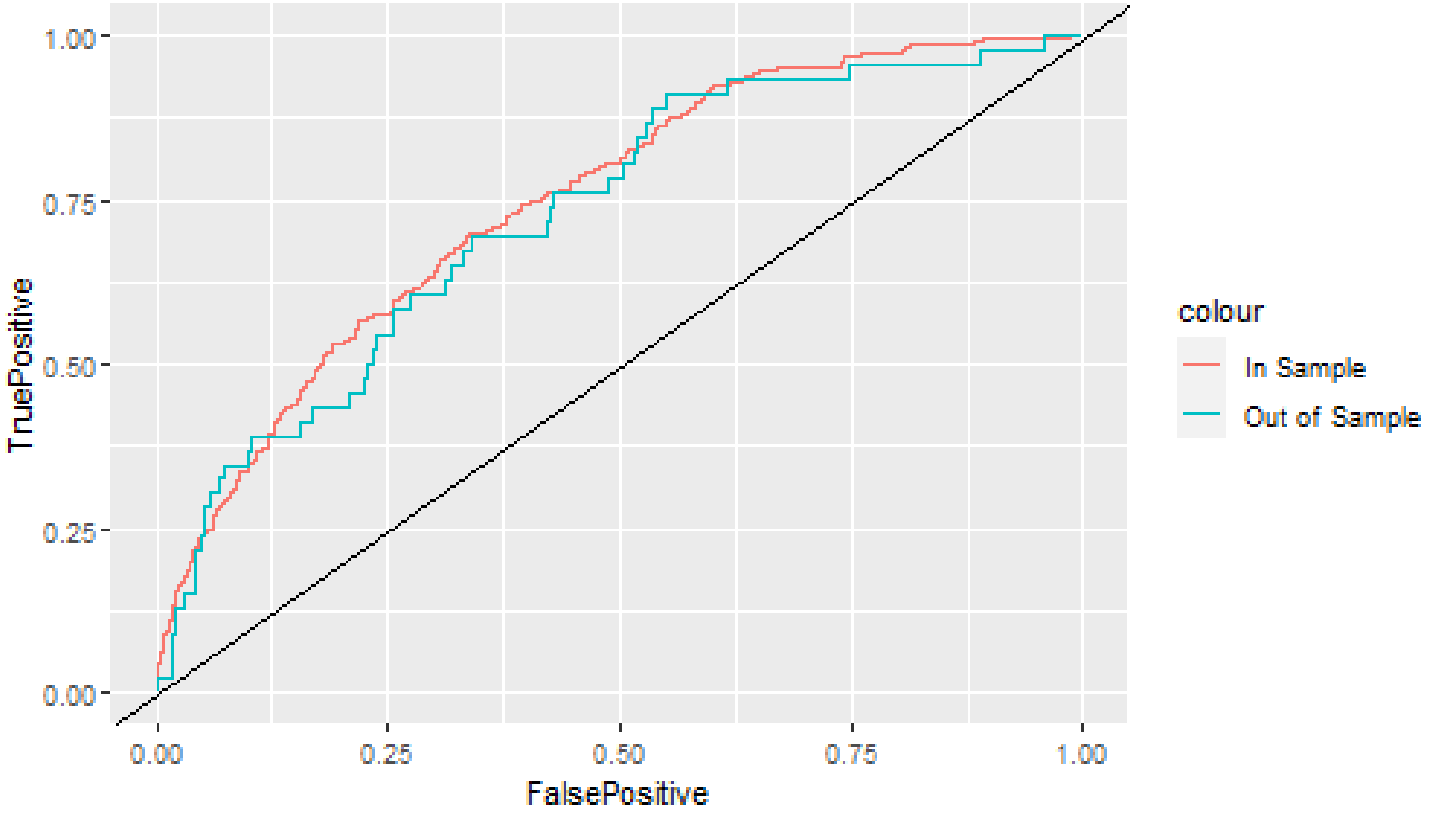
- Misreporting firms' financials should be different from expected
  - odd financials

Approach

```
fit_1990s <- glm(AAER ~ ebit + ni_revt + ni_at + log_lt + ltl_at + lt_seq +
                   lt_at + act_lct + aq_lct + wcap_at + invt_revt + invt_at +
                   ni_ppent + rect_revt + revt_at + d_revt + b_rect + b_rect +
                   r_gp + b_gp + gp_at + revt_m_gp + ch_at + log_at +
                   ppent_at + wcap,
                 data = df[df$Test == 0, ], #Test=0 is the training data
                 family = binomial)
tidy(fit_1990s)
```

```
## # A tibble: 26 x 5
##    term          estimate std.error statistic      p.value
##    <chr>            <dbl>     <dbl>     <dbl>        <dbl>
##  1 (Intercept) -4.66       0.834      -5.59   0.0000000226
##  2 ebit        -0.000356   0.000109   -3.26   0.00112
##  3 ni_revt      0.0366     0.0306      1.20   0.231
##  4 ni_at       -0.320      0.233      -1.37   0.169
##  5 log_lt       0.149      0.341       0.438  0.661
##  6 ltl_at      -0.231      0.707      -0.326  0.744
##  7 lt_seq      -0.0000283  0.000457   -0.0619 0.951
##  8 lt_at       -0.856      0.927      -0.923  0.356
##  9 act_lct      0.140      0.0701      2.00   0.0455
## 10 aq_lct      -0.175      0.0916     -1.91   0.0559
## # ... with 16 more rows
```

ROC




```
##     In sample AUC Out of sample AUC
##         0.7483132         0.7292981
```

Code for last slide's curve

```r
library(ROCR)
# Have to remove all NA in prediction() with ROCR 1.0-11 version
# You may install the older version which does not require so
# https://cran.r-project.org/src/contrib/Archive/ROCR/ROCR_1.0-7.tar.gz
pred <- predict(fit_1990s, df, type="response")
ROCpred <- prediction(as.numeric(pred[df$Test == 0 & !is.na(pred)]),
                      as.numeric(df[df$Test == 0 & !is.na(pred), ]$AAER))
ROCpred_out <- prediction(as.numeric(pred[df$Test == 1 & !is.na(pred)]),
                          as.numeric(df[df$Test==1 & !is.na(pred), ]$AAER))
ROCperf <- performance(ROCpred, 'tpr', 'fpr')
ROCperf_out <- performance(ROCpred_out, 'tpr', 'fpr')
df_ROC_1990s <- data.frame(FalsePositive = c(ROCperf@x.values[[1]]),
                           TruePositive = c(ROCperf@y.values[[1]]))
df_ROC_out_1990s <- data.frame(FalsePositive = c(ROCperf_out@x.values[[1]]),
                               TruePositive = c(ROCperf_out@y.values[[1]]))

ggplot() +
  geom_line(data = df_ROC_1990s, aes(x = FalsePositive, y = TruePositive,
                                     color = "In Sample")) +
  geom_line(data = df_ROC_out_1990s, aes(x = FalsePositive, y = TruePositive,
                                         color = "Out of Sample")) +
  geom_abline(slope = 1)

auc <- performance(ROCpred, measure = "auc")
auc_out <- performance(ROCpred_out, measure = "auc")
aucs_1990s <- c(auc@y.values[[1]], auc_out@y.values[[1]])
names(aucs_1990s) <- c("In sample AUC", "Out of sample AUC")
aucs_1990s
```

The 2011 follow up

The 2011 model

- Log of assets
- Total accruals
- % change in A/R
- % change in inventory
- % soft assets
- % change in sales from cash
- % change in ROA
- Indicator for stock/bond issuance
- Indicator for operating leases
- BV equity / MV equity
- Lag of stock return minus value weighted market return
- **Below are BCE's additions**
- Indicator for mergers
- Indicator for Big N auditor
- Indicator for medium size auditor
- Total financing raised
- Net amount of new capital raised
- Indicator for restructuring

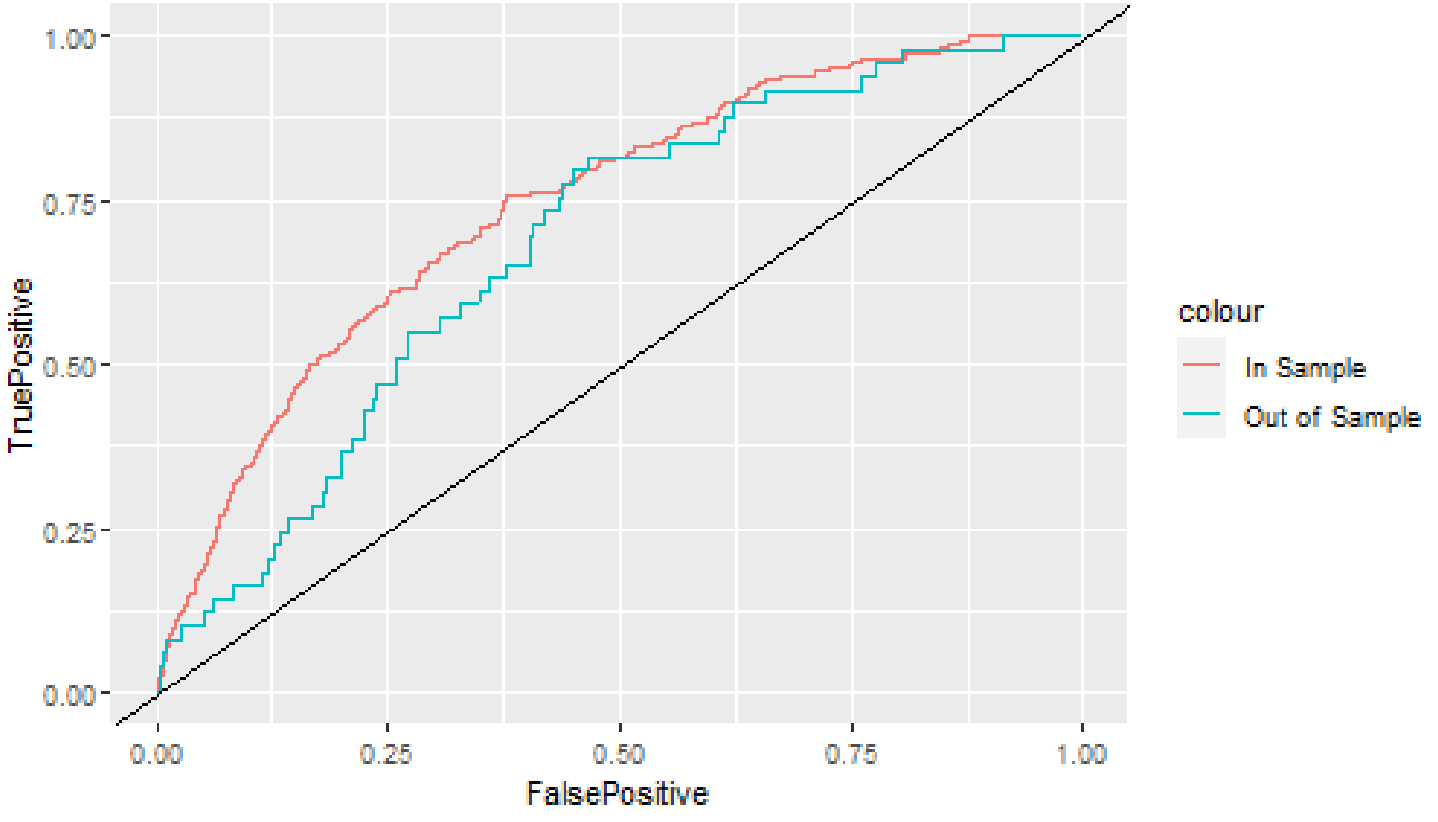
> Based on Dechow, Ge, Larson and Sloan (2011)

The model

```
fit_2011 <- glm(AAER ~ logtotasset + rsst_acc + chg_recv + chg_inv +
                  soft_assets + pct_chg_cashsales + chg_roa + issuance +
                  oplease_dum + book_mkt + lag_sdvol + merger + bigNaudit +
                  midNaudit + cffin + exfin + restruct,
                data = df[df$Test == 0, ],  family = binomial)
tidy(fit_2011)
```

```
## # A tibble: 18 x 5
##    term                estimate std.error statistic  p.value
##    <chr>                  <dbl>     <dbl>     <dbl>    <dbl>
##  1 (Intercept)        -7.15        0.534   -13.4    6.82e-41
##  2 logtotasset         0.321       0.0355    9.04   1.53e-19
##  3 rsst_acc           -0.219       0.301    -0.728  4.67e- 1
##  4 chg_recv            1.10        1.06      1.04   2.98e- 1
##  5 chg_inv             0.0390      1.25      0.0311 9.75e- 1
##  6 soft_assets         2.31        0.333     6.94   3.81e-12
##  7 pct_chg_cashsales -0.000691     0.0109   -0.0635 9.49e- 1
##  8 chg_roa            -0.270       0.255    -1.06   2.91e- 1
##  9 issuance            0.144       0.319     0.453  6.51e- 1
## 10 oplease_dum        -0.203       0.197    -1.03   3.03e- 1
## 11 book_mkt            0.0150      0.0111    1.36   1.75e- 1
## 12 lag_sdvol           0.0517      0.0555    0.932  3.52e- 1
## 13 merger              0.353       0.151     2.33   1.96e- 2
## 14 bigNaudit          -0.200       0.360    -0.555  5.79e- 1
## 15 midNaudit          -0.489       0.512    -0.956  3.39e- 1
## 16 cffin               0.456       0.344     1.33   1.85e- 1
## 17 exfin              -0.00536     0.0393   -0.136  8.92e- 1
## 18 restruct            0.383       0.147     2.60   9.30e- 3
```

ROC




```
##     In sample AUC Out of sample AUC
##         0.7445378         0.6849225
```

Code for last slide's curve

```
library(ROCR)
pred <- predict(fit_2011, df, type = "response")
ROCpred <- prediction(as.numeric(pred[df$Test == 0 & !is.na(pred)]),
                      as.numeric(df[df$Test==0 & !is.na(pred), ]$AAER))
ROCpred_out <- prediction(as.numeric(pred[df$Test == 1 & !is.na(pred)]),
                          as.numeric(df[df$Test == 1 & !is.na(pred), ]$AAER))
ROCperf <- performance(ROCpred, 'tpr', 'fpr')
ROCperf_out <- performance(ROCpred_out, 'tpr', 'fpr')
df_ROC_2011 <- data.frame(FalsePositive = c(ROCperf@x.values[[1]]),
                          TruePositive=c(ROCperf@y.values[[1]]))
df_ROC_out_2011 <- data.frame(FalsePositive = c(ROCperf_out@x.values[[1]]),
                              TruePositive = c(ROCperf_out@y.values[[1]]))

ggplot() +
  geom_line(data = df_ROC_2011, aes(x = FalsePositive, y = TruePositive,
                                    color = "In Sample")) +
  geom_line(data = df_ROC_out_2011, aes(x = FalsePositive, y = TruePositive,
                                        color = "Out of Sample")) +
  geom_abline(slope = 1)

auc <- performance(ROCpred, measure = "auc")
auc_out <- performance(ROCpred_out, measure = "auc")
aucs_2011 <- c(auc@y.values[[1]], auc_out@y.values[[1]])
names(aucs_2011) <- c("In sample AUC", "Out of sample AUC")
aucs_2011
```

Late 2000s/early 2010s approach

The late 2000s/early 2010s model

SMU SINGAPORE MANAGEMENT UNIVERSITY School of Accountancy

- Log of # of bullet points + 1
- # of characters in file header
- # of excess newlines
- Amount of html tags
- Length of cleaned file, characters
- Mean sentence length, words
- S.D. of word length
- S.D. of paragraph length (sentences)
- Word choice variation
- Readability
  - Coleman Liau Index
  - Fog Index
- % active voice sentences
- % passive voice sentences
- # of all cap words
- # of exclamation mark !
- # of question mark ?

> From a variety of papers

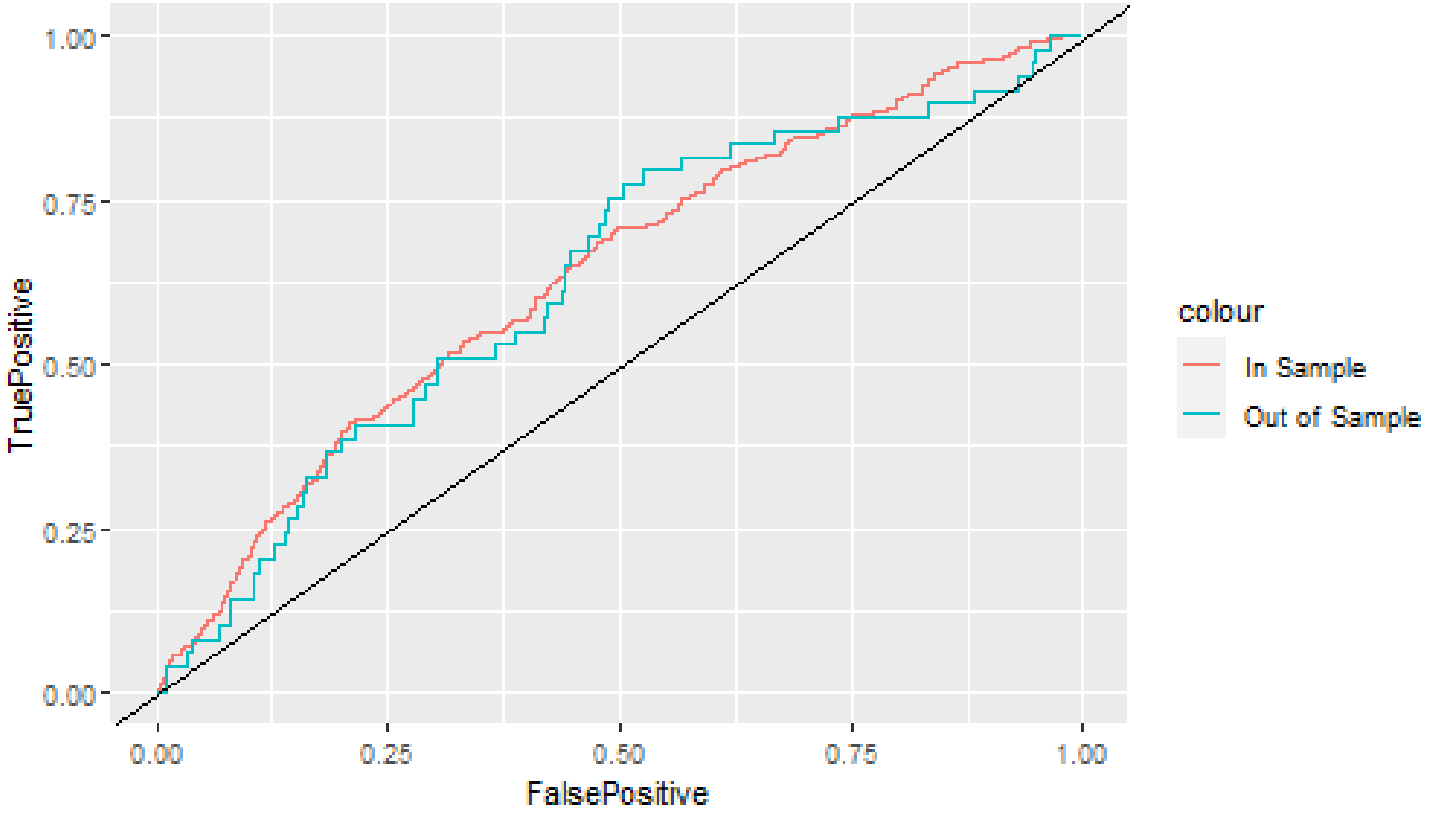
- Generally pulled from the communications literature
  - Sometimes ad hoc
- The main idea:
  - Companies that are misreporting probably write their annual report differently

The late 2000s/early 2010s model

```
fit_2000s <- glm(AAER ~ bullets + headerlen + newlines + alltags +
                   processedsize + sentlen_u + wordlen_s + paralen_s +
                   repetitious_p + sentlen_s + typetoken + clindex + fog +
                   active_p + passive_p + lm_negative_p + lm_positive_p +
                   allcaps + exclamationpoints + questionmarks,
                 data=df[df$Test == 0, ], family = binomial)
tidy(fit_2000s)
```

```
## # A tibble: 21 x 5
##    term               estimate   std.error statistic    p.value
##    <chr>                 <dbl>       <dbl>     <dbl>      <dbl>
##  1 (Intercept)   -5.66         3.14           -1.80  0.0716
##  2 bullets       -0.0000263    0.0000263      -1.00  0.316
##  3 headerlen     -0.000294     0.000348       -0.846 0.397
##  4 newlines      -0.0000482    0.000122       -0.395 0.693
##  5 alltags        0.0000000506 0.000000257     0.197 0.844
##  6 processedsize  0.00000571   0.00000129      4.44  0.00000919
##  7 sentlen_u     -0.0379       0.0690         -0.550 0.583
##  8 wordlen_s      0.128        1.20            0.107 0.915
##  9 paralen_s     -0.0481       0.0305         -1.58  0.115
## 10 repetitious_p -1.67         1.67           -1.00  0.315
## # ... with 11 more rows
```

ROC




```
##     In sample AUC Out of sample AUC
##         0.6377783         0.6295414
```

Code for last slide's curve

```
library(ROCR)
pred <- predict(fit_2000s, df, type = "response")
ROCpred <- prediction(as.numeric(pred[df$Test == 0 & !is.na(pred)]),
                      as.numeric(df[df$Test == 0 & !is.na(pred), ]$AAER))
ROCpred_out <- prediction(as.numeric(pred[df$Test == 1 & !is.na(pred)]),
                          as.numeric(df[df$Test == 1 & !is.na(pred), ]$AAER))
ROCperf <- performance(ROCpred, 'tpr', 'fpr')
ROCperf_out <- performance(ROCpred_out, 'tpr', 'fpr')
df_ROC_2000s <- data.frame(FalsePositive = c(ROCperf@x.values[[1]]),
                           TruePositive = c(ROCperf@y.values[[1]]))
df_ROC_out_2000s <- data.frame(FalsePositive = c(ROCperf_out@x.values[[1]]),
                               TruePositive = c(ROCperf_out@y.values[[1]]))

ggplot() +
  geom_line(data = df_ROC_2000s, aes(x = FalsePositive, y = TruePositive,
                                     color = "In Sample")) +
  geom_line(data = df_ROC_out_2000s, aes(x = FalsePositive, y = TruePositive,
                                         color = "Out of Sample")) +
  geom_abline(slope = 1)

auc <- performance(ROCpred, measure = "auc")
auc_out <- performance(ROCpred_out, measure = "auc")
aucs_2000s <- c(auc@y.values[[1]], auc_out@y.values[[1]])
names(aucs_2000s) <- c("In sample AUC", "Out of sample AUC")
aucs_2000s
```

Combining the 2000s and 2011 models

> Why is it appropriate to combine the 2011 model with the 2000s model?

- 2011 model: Parsimonious financial model
- 2000s model: Textual characteristics

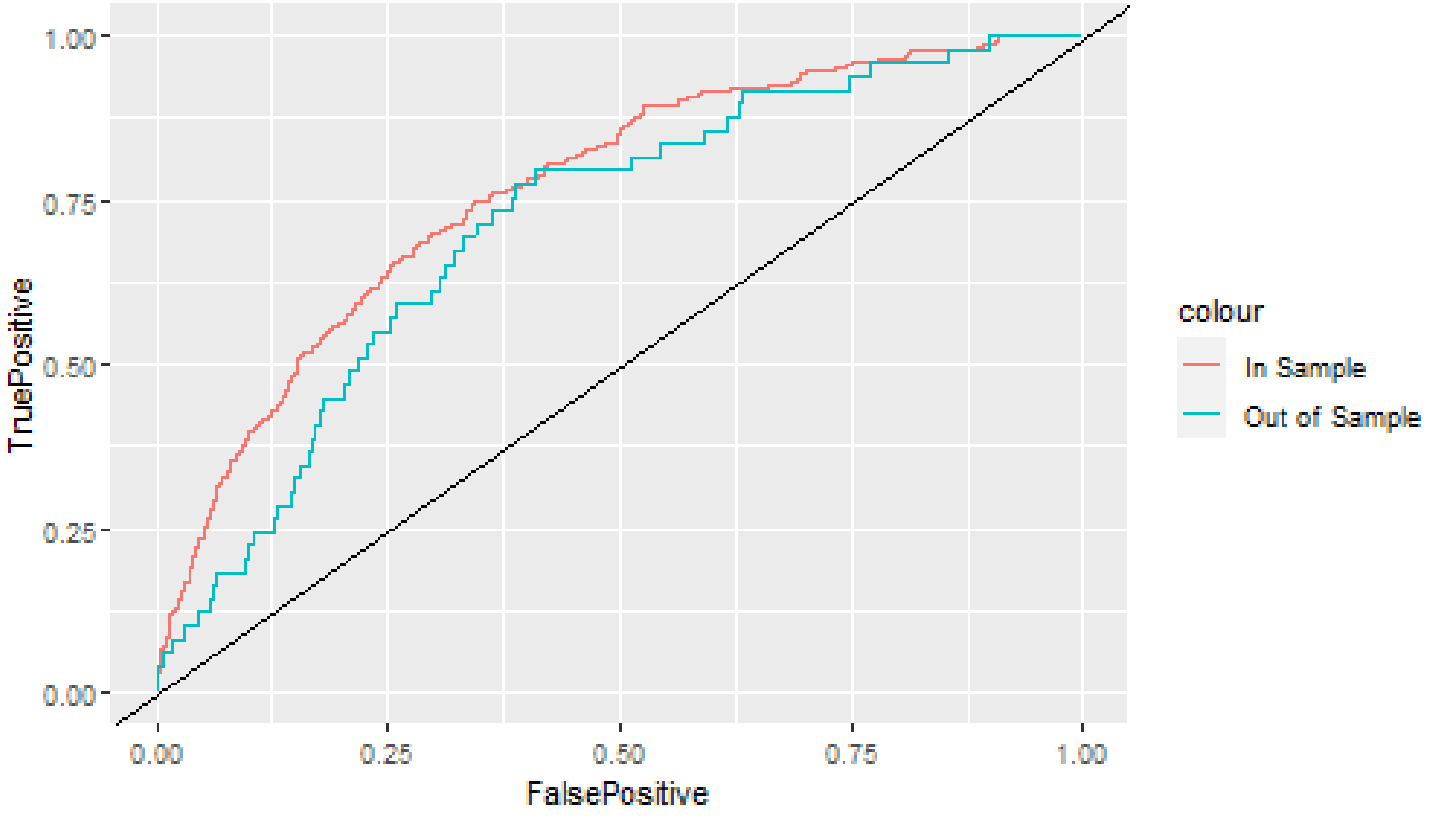
> Little theoretical overlap
>
> Limited multicollinearity across measures

The model

```
fit_2000f <- glm(AAER ~ logtotasset + rsst_acc + chg_recv + chg_inv +
                   soft_assets + pct_chg_cashsales + chg_roa + issuance +
                   oplease_dum + book_mkt + lag_sdvol + merger + bigNaudit +
                   midNaudit + cffin + exfin + restruct + bullets + headerlen +
                   newlines + alltags + processedsize + sentlen_u + wordlen_s +
                   paralen_s + repetitious_p + sentlen_s + typetoken +
                   clindex + fog + active_p + passive_p + lm_negative_p +
                   lm_positive_p + allcaps + exclamationpoints + questionmarks,
                 data = df[df$Test == 0, ], family = binomial)
tidy(fit_2000f)
```

```
## # A tibble: 38 x 5
##    term              estimate std.error statistic  p.value
##    <chr>                <dbl>     <dbl>     <dbl>    <dbl>
##  1 (Intercept)       -1.63       3.41     -0.479  6.32e- 1
##  2 logtotasset        0.344      0.0392    8.77   1.86e-18
##  3 rsst_acc          -0.212      0.299    -0.709  4.78e- 1
##  4 chg_recv           1.02       1.05      0.969  3.33e- 1
##  5 chg_inv           -0.0946     1.23     -0.0772 9.38e- 1
##  6 soft_assets        2.57       0.339     7.60   3.01e-14
##  7 pct_chg_cashsales -0.00113    0.0110   -0.103  9.18e- 1
##  8 chg_roa           -0.270      0.247    -1.09   2.75e- 1
##  9 issuance           0.147      0.322     0.457  6.48e- 1
## 10 oplease_dum       -0.286      0.202    -1.42   1.57e- 1
## # ... with 28 more rows
```

ROC




```
##     In sample AUC Out of sample AUC
##         0.7664115         0.7147021
```

The BCE model

The BCE approach

- Retain the variables from the other regressions
- Add in a machine-learning based measure of *topics*
  - Every document is a mixture of topics. It may contain words from several topics in particular proportions. Eg, we could say "Document 1 is 90% topic A and 10% topic B, while Document 2 is 30% topic A and 70% topic B."
  - Every topic is a mixture of words. Eg, we imagine a two-topic model of annual reports, "decrease in income" and "bad debt." The most common words in the income decreasing topic might be "income decreased" and "company expects", while the bad debt topic may be made up of words such as "accounts receivable" and "allowance for doubtful". Importantly, words can be shared between topics.

> Don't worry, we will cover this in a future topic

What the topics look like

Topic 6

Topic 11

Topic 21

Topic 30

Topic 2

Topic 9

Topic 12

Topic 26

Topic 8

Topic 19

Theory behind the BCE model

SMU SINGAPORE MANAGEMENT UNIVERSITY School of Accountancy

> Why use document content?

- From communications and psychology:
  - When people are trying to deceive others, what they say is carefully picked
    - Topics chosen are intentional
- Putting this in a business context:
  - If you are manipulating inventory, you don't talk about it

The model

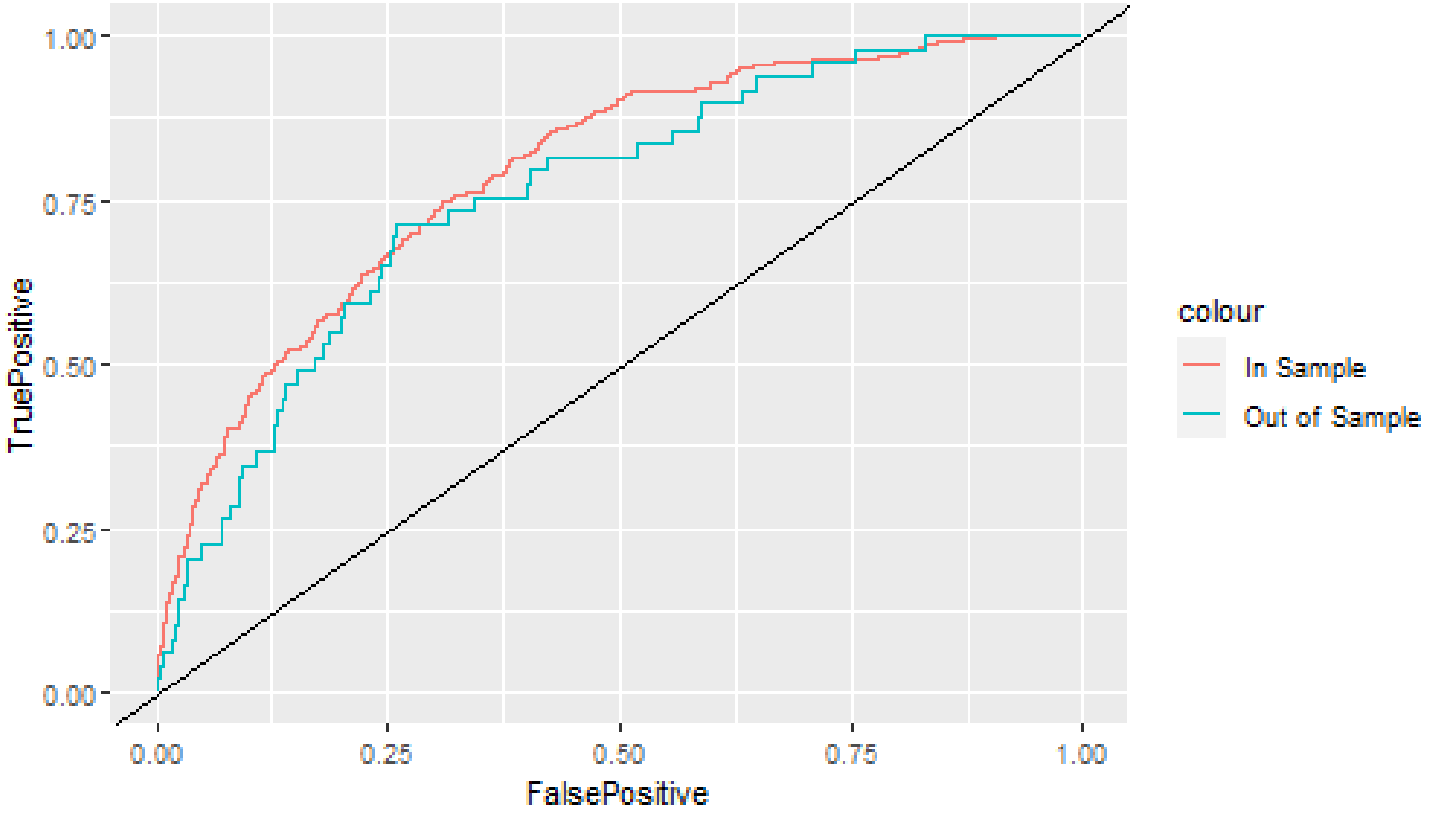
```
BCE_eq = as.formula(paste("AAER ~ logtotasset + rsst_acc + chg_recv + chg_inv +
                          soft_assets + pct_chg_cashsales + chg_roa + issuance +
                          oplease_dum + book_mkt + lag_sdvol + merger +
                          bigNaudit + midNaudit + cffin + exfin + restruct +
                          bullets + headerlen + newlines + alltags +
                          processedsize + sentlen_u + wordlen_s + paralen_s +
                          repetitious_p + sentlen_s + typetoken + clindex + fog +
                          active_p + passive_p + lm_negative_p + lm_positive_p +
                          allcaps + exclamationpoints + questionmarks +",
                          paste(paste0("Topic_", 1:30, "_n_oI"), collapse=" + "),
                          collapse = "")) # Topic_1_n_oI is the topic variable

fit_BCE <- glm(BCE_eq, data = df[df$Test == 0, ], family = binomial)
```

BCE model output

```
tidy(fit_BCE) # Topic_1_n_oI is the topic variable
```

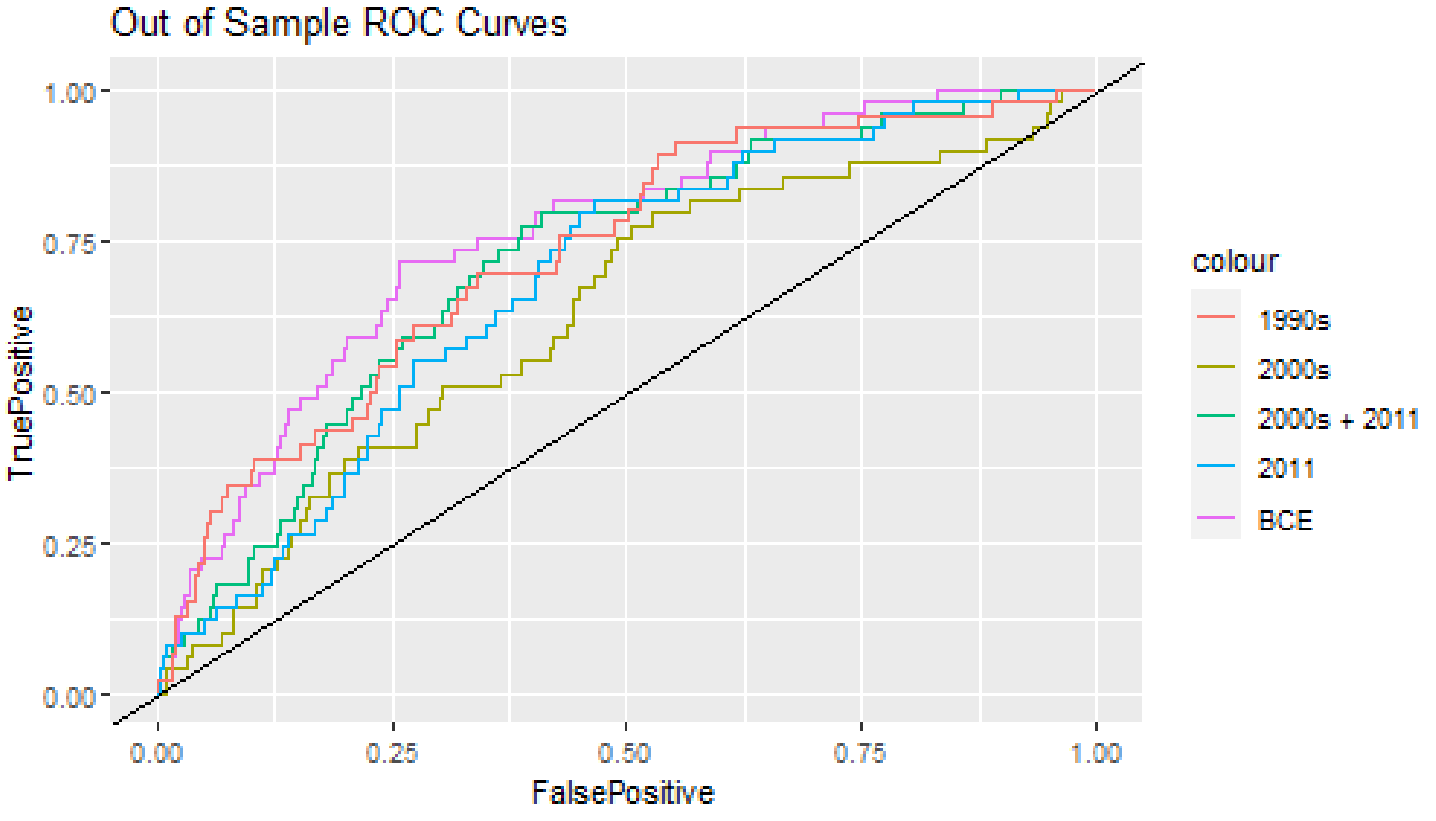
```
## # A tibble: 68 x 5
##    term              estimate std.error statistic  p.value
##    <chr>                <dbl>     <dbl>     <dbl>    <dbl>
##  1 (Intercept)       -8.03      3.87       -2.07  3.81e- 2
##  2 logtotasset        0.388     0.0455      8.52  1.62e-17
##  3 rsst_acc          -0.194     0.306      -0.634 5.26e- 1
##  4 chg_recv           0.858     1.07        0.801 4.23e- 1
##  5 chg_inv           -0.261     1.22       -0.213 8.31e- 1
##  6 soft_assets        2.55      0.380       6.73  1.70e-11
##  7 pct_chg_cashsales -0.00198   0.00700    -0.282 7.78e- 1
##  8 chg_roa           -0.253     0.279      -0.909 3.64e- 1
##  9 issuance           0.0969    0.327       0.296 7.67e- 1
## 10 oplease_dum       -0.345     0.210      -1.65  9.99e- 2
## # ... with 58 more rows
```

ROC




```
##     In sample AUC Out of sample AUC
##         0.7941841         0.7599594
```

Comparison across all models




```
##       1990s        2011       2000s 2000s + 2011         BCE
##   0.7292981   0.6849225   0.6295414   0.7147021   0.7599594
```

Simplifying models with LASSO

What is lasso?

What is LASSO?

- Least Absolute Shrinkage and Selection Operator
  - Least absolute: uses an error term like $|\varepsilon|$
  - Shrinkage: it will make coefficients smaller
    - Less sensitive and less overfitting issues
  - Selection: it will completely remove some variables
    - Less variables and less overfitting issues
- Sometimes called L^1 regularization
  - L^1 means 1 dimensional distance, i.e., $|\varepsilon|$

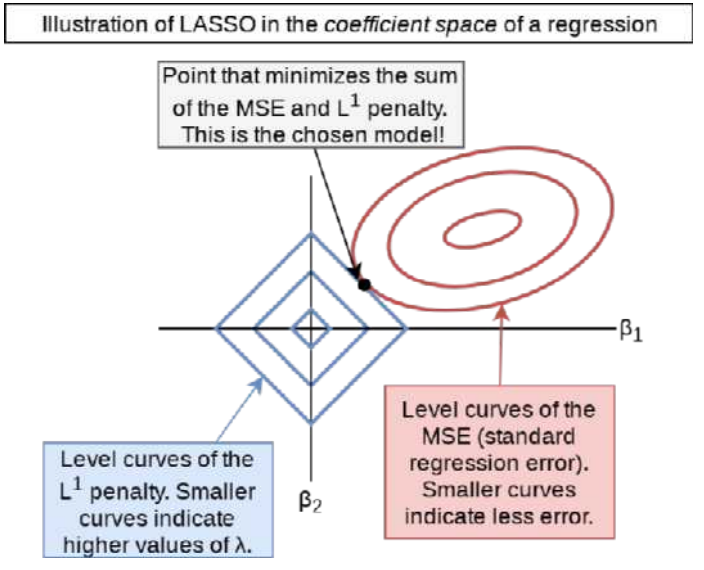
> Great if you have way too many inputs in your model

- This is how we can, in theory, put more variables in our model than data points

How does it work?

$$\min_{\beta \in \mathbb{R}^p} \left\{ \frac{1}{N} \|y - X\beta\|_2^2 + \lambda \|\beta\|_1 \right\}$$

- where $\|\beta\|_p = \left(\sum_{i=1}^{N} |\beta_i|^p \right)^{1/p}$
- to minimize training loss and regularization term
- Add an additional penalty term that is increasing in the absolute value of each β
  - Incentivizes lower βs, *shrinking* them



Why we use it?

- A better balance of **bias** (low) and **variance** (low)
  - **bias**: the inability for a ML method to capture the true relationship (intuitively, the prediction error in the training dataset)
  - **variance**: the difference in fits between datasets (intuitively, the prediction error in the testing dataset)
  - mitigate the overfitting (high variance) and underfiting (high bias) problems
  - regularization, bagging and boosting are used to achieve the balance
- Model selection by the machine
  - β could be 0, ie, selection of features to best fit the model
  - discourages learning a more complex model
- Automate the feature selection process

Package for LASSO

- `package:glmnet`, pronounced glimnet

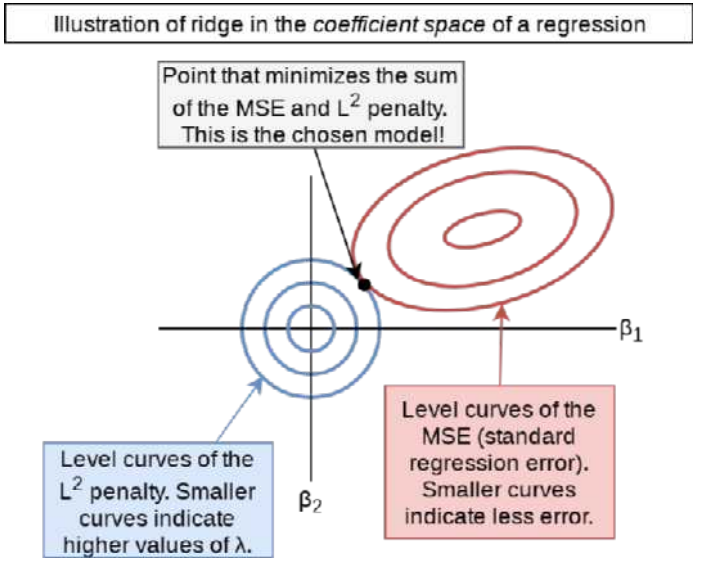
1. For all regression commands, they expect a **y** vector and an **x** matrix instead of our usual `y ~ x` formula
   - R has a helper function to convert a formula to a matrix: `model.matrix()`
     - Supply it the right hand side of the equation, starting with `~`, and your data
     - It outputs the matrix `x`
   - Alternatively, use `as.matrix()` on a data frame of your input variables
2. Its family argument should be specified in quotes, i.e., `"binomial"` instead of `binomial`

What else can the package do?

SMU SINGAPORE MANAGEMENT UNIVERSITY School of Accountancy

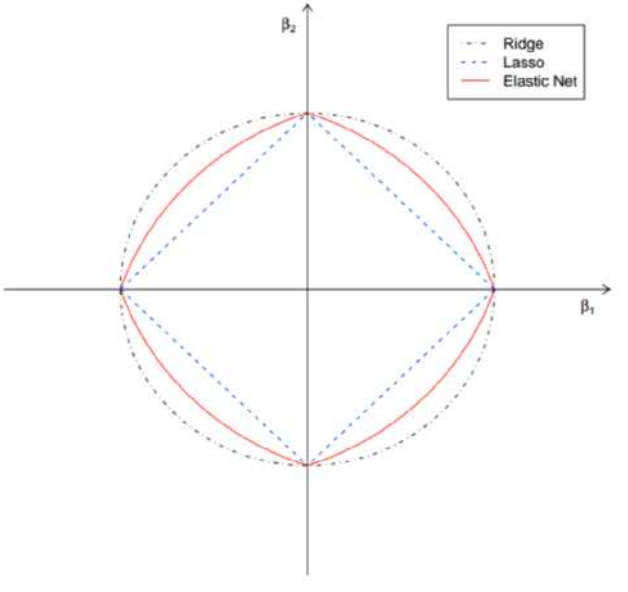
Ridge regression

- Similar to LASSO, but with an L^2 penalty (Euclidean norm)



Elastic net regression

- Hybrid of LASSO and Ridge
- Below image by Jared Lander



How to run a LASSO

- To run a simple LASSO model, use `glmnet()`

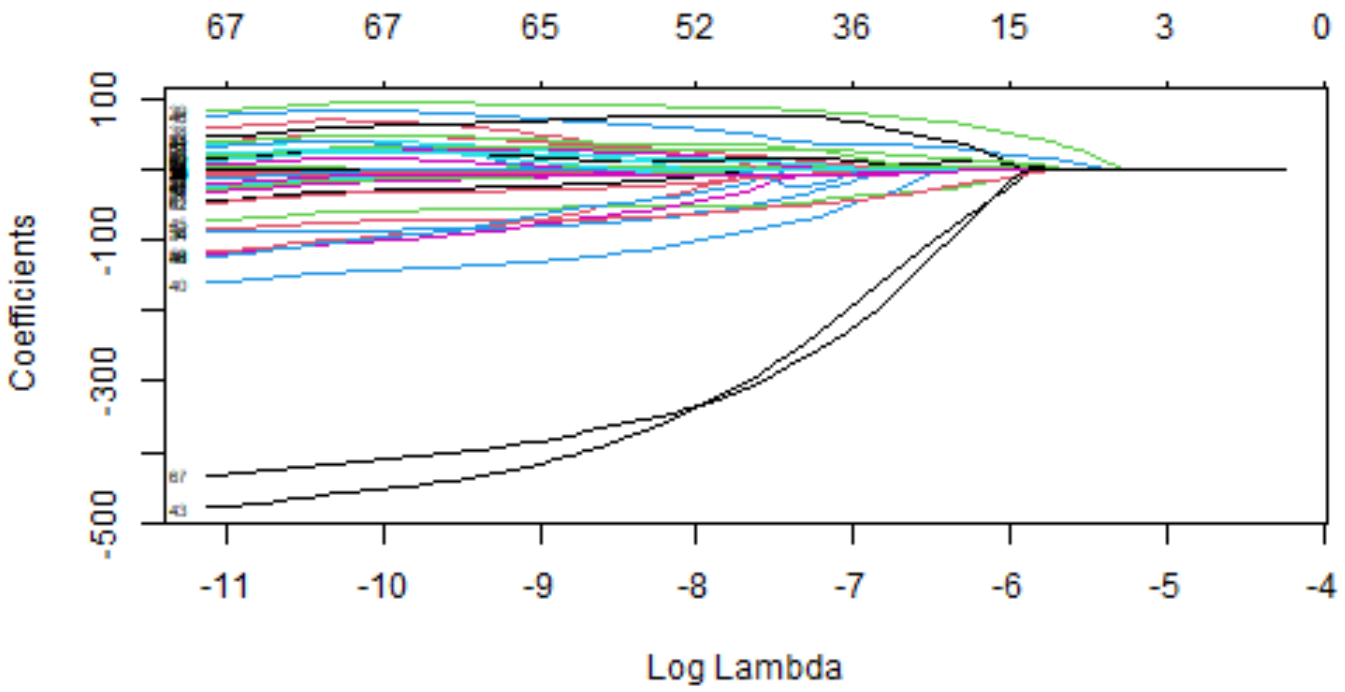
$$\min_{\beta_0,\beta} \frac{1}{N} \sum_{i=1}^{N} w_i l(y_i, \beta_0 + \beta^T x_i) + \lambda \left[(1-\alpha)||\beta||_2^2/2 + \alpha||\beta||_1 \right]$$

```
library(glmnet)
x <- model.matrix(BCE_eq, data = df[df$Test == 0, ])[ , -1] # remove intercept
y <- model.frame(BCE_eq, data = df[df$Test == 0, ])[ , "AAER"]
fit_LASSO <- glmnet(x = x, y = y,
                    family = "binomial", # "gaussian" for least squares (default)
                    alpha = 1  # LASSO, the default.  alpha = 0 is ridge
                               # alpha between 0 and 1: elastic net
                    )
```

- Note: the **x** and **y** can be more elegantly done using the `package:useful`, see here for an example
  - `useful::build.x(formula, data)` and `useful::build.y(formual, data)`

Visualizing Lasso

```
plot(fit_LASSO, xvar = 'lambda', label = TRUE)
```



- the top x-axis is the number of degrees of freedom of the model, i.e., the number of variables **minus 1**.

Interactive visualizing Lasso

```
coefpath(fit_LASSO)
```

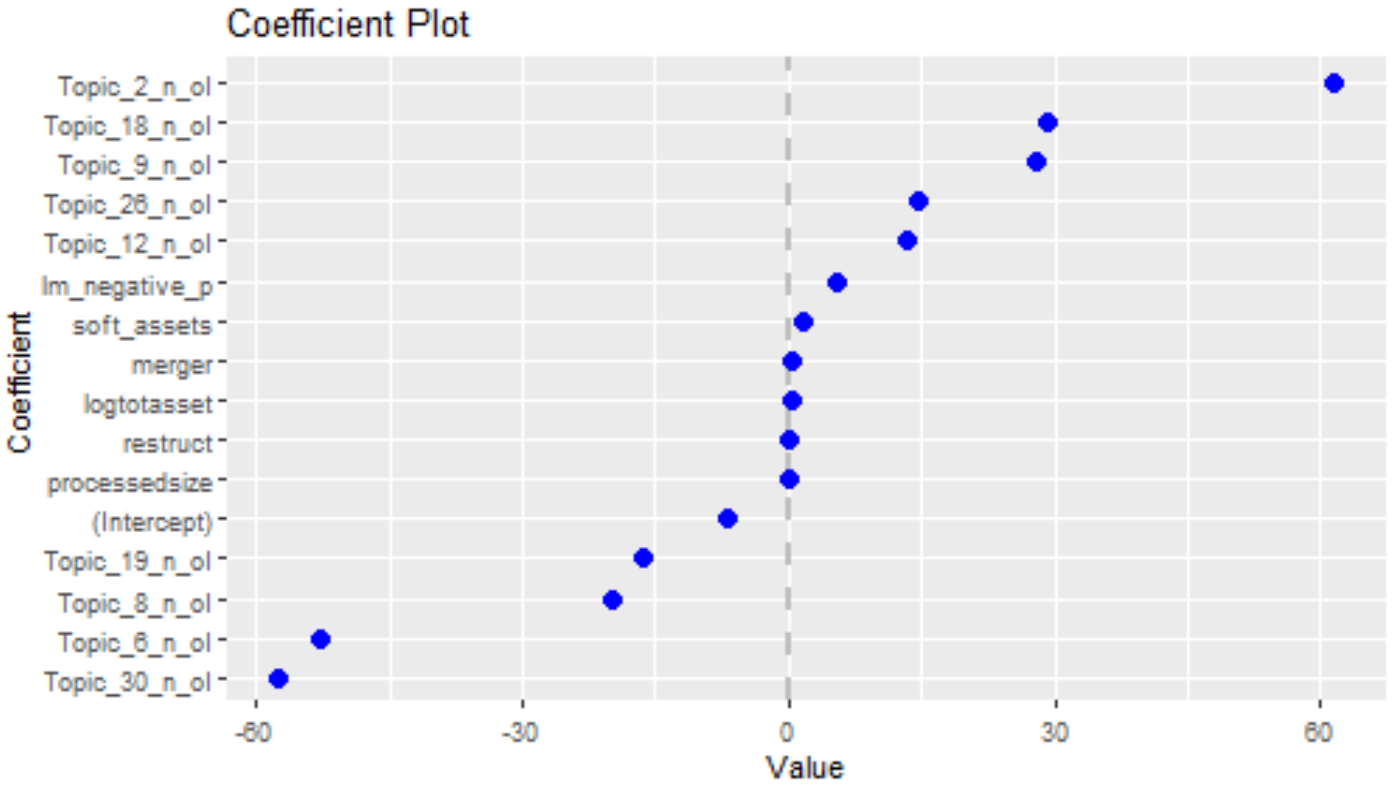
What's under the hood?

```
# It shows from left to right the number of nonzero coefficients (Df),
# the percent of deviance explained (%dev) and the value of Lambda.
# Deviance is the stat for maximum likelihood estimation
# Although by default glmnet calls for 100 values of lambda the program stops
# early if %dev doesn't change sufficiently from one lambda to the next.
print(fit_LASSO)
```

```
##
## Call:  glmnet(x = x, y = y, family = "binomial", alpha = 1)
##
##    Df  %Dev    Lambda
## 1   0  0.00 0.0143300
## 2   1  0.81 0.0130500
## 3   1  1.46 0.0118900
## 4   1  2.00 0.0108400
## 5   2  2.47 0.0098740
## 6   2  3.22 0.0089970
## 7   2  3.85 0.0081970
## 8   2  4.37 0.0074690
## 9   2  4.81 0.0068060
## 10  3  5.22 0.0062010
## 11  3  5.59 0.0056500
## 12  4  5.91 0.0051480
## 13  4  6.25 0.0046910
## 14  5  6.57 0.0042740
## 15  7  6.89 0.0038940
## 16  8  7.22 0.0035480
## 17 10  7.52 0.0032330
## 18 12  7.83 0.0029460
```

One of the 100 models

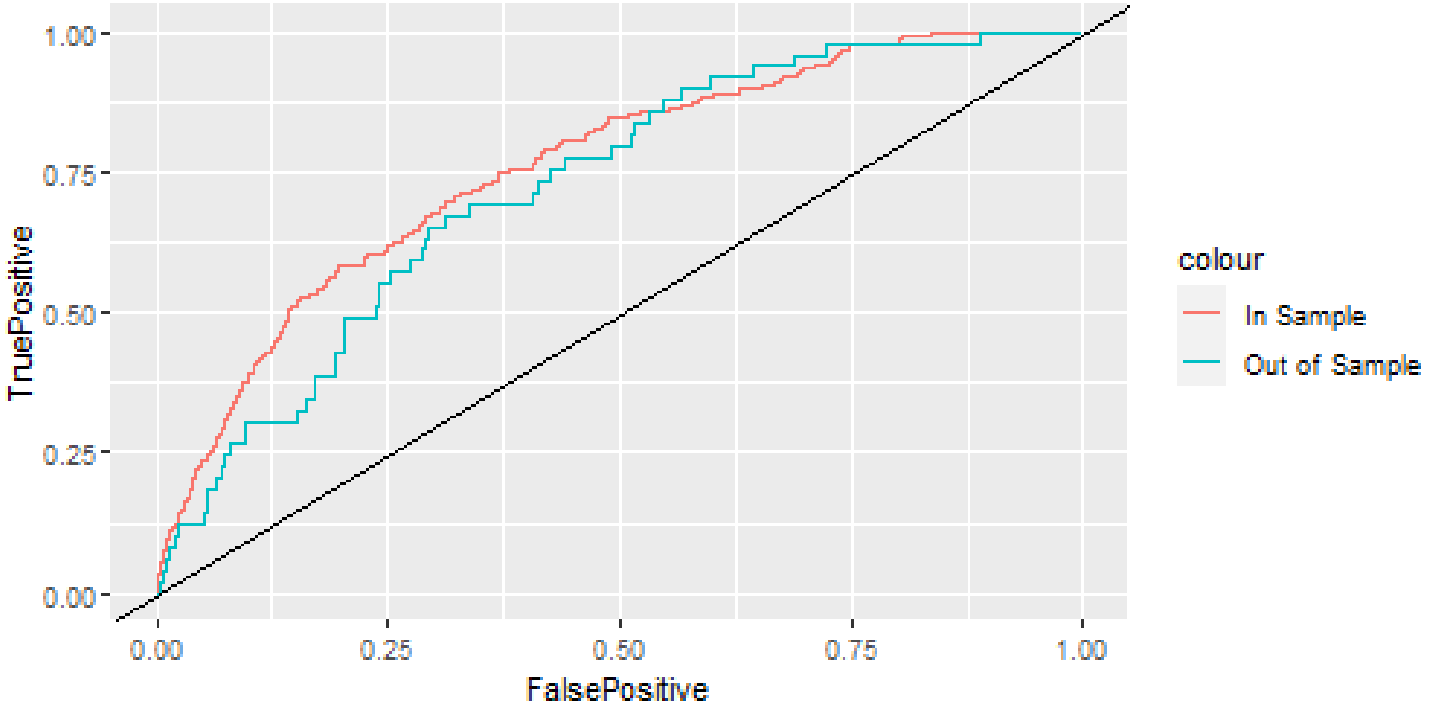
```
# I randomly pick a lambda and show the features
# You may try a different lambda from the previous slide
coefplot(fit_LASSO, lambda = 0.002031, sort = 'magnitude')
```




```
# coef(fit_LASSO, s=0.002031) #to check the magnitude of coefficients
```

How does this perform?

```
# na.pass has model.matrix retain NA values (so the # of rows is constant)
xp <- model.matrix(BCE_eq, data = df, na.action = 'na.pass')[ , -1]
# s= specifies the version of the model to use
pred <- predict(fit_LASSO, xp, type = "response", s = 0.002031)
```




```
##      In sample AUC Out of sample AUC
##          0.7593828          0.7239785
```

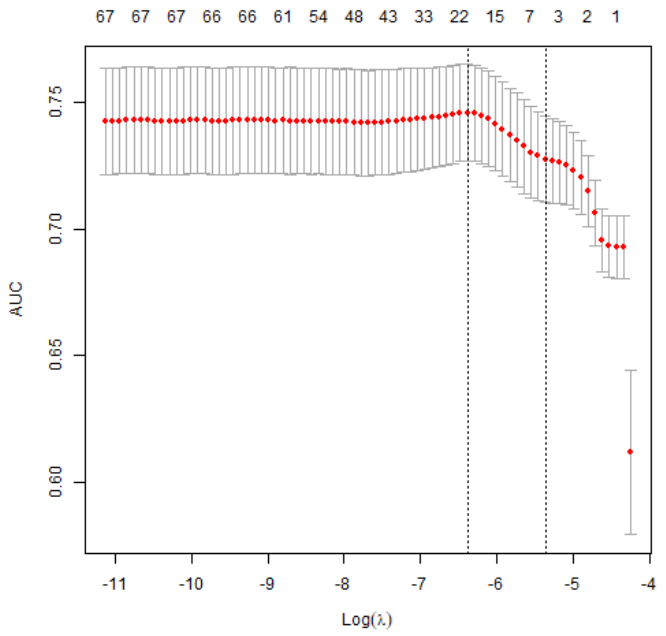
Automating model selection

- LASSO seems nice, but picking between the 100 models is tough!
- It also contains a method of *k-fold cross validation* (default, $k = 10$)
  1. Randomly split the data into k groups
  2. Hold out one group and run model on the rest data
  3. Determine the best model using the hold out group
  4. Repeat steps 2 and 3 $k - 1$ more times
  5. Uses the best overall model across all k hold out samples
- It gives 2 model options:
  - `"lambda.min"`: The best performing model that gives minimum mean cross-validated error.
  - `"lambda.1se"`: the most regularized model such that error is within one standard error of the `"lambda.min"`
    - This is the better choice if you are concerned about overfitting

Running a cross validated model

```
# Cross validation
set.seed(2021)  #for reproducibility
cvfit = cv.glmnet(x = x, y = y,family = "binomial", alpha = 1,
                  type.measure = "auc")
```

```
plot(cvfit)
```




```
cvfit$lambda.min
```

```
## [1] 0.001685798
```

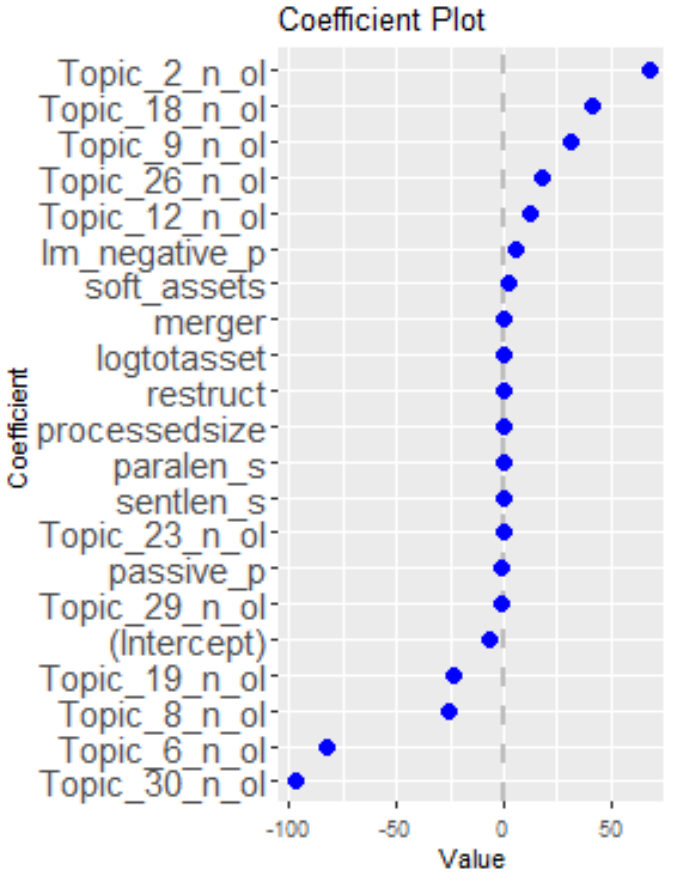
```
cvfit$lambda.1se
```

```
## [1] 0.004690834
```

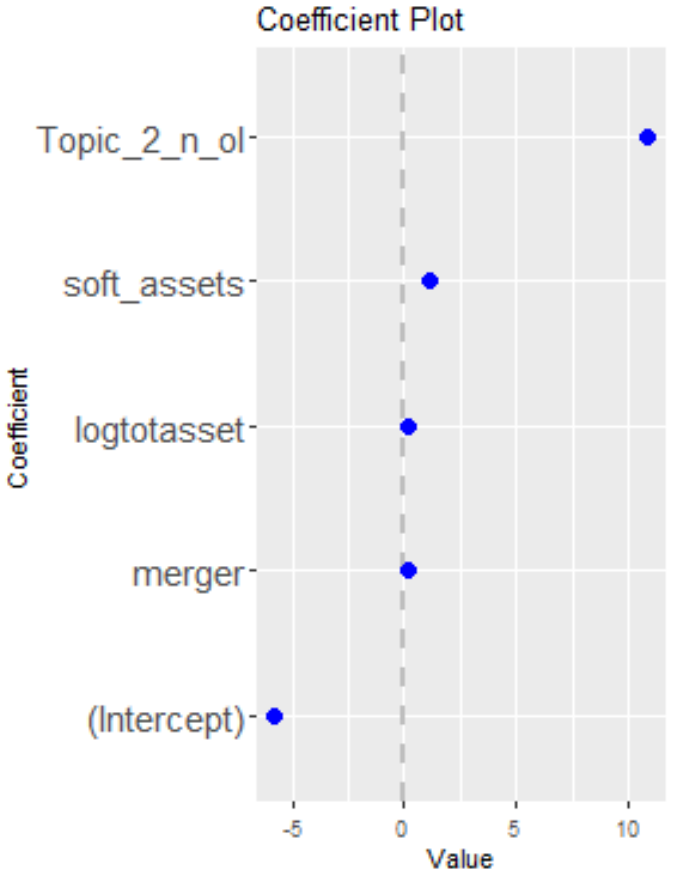
- The left figure also shows the upper and lower standard deviation of AUC

Models

`lambda.min`

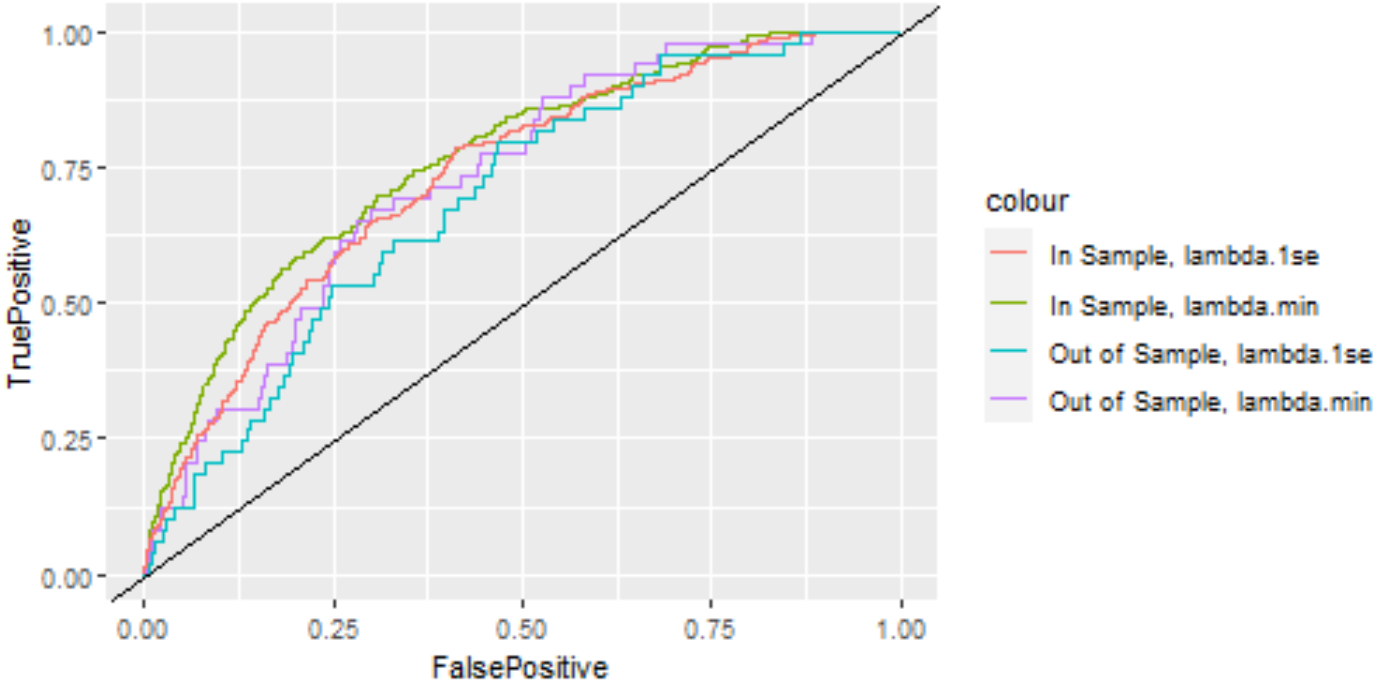


`lambda.1se`



CV LASSO performance

```
# s= specifies the version of the model to use
pred <- predict(cvfit, xp, type = "response", s = "lambda.min")
pred2 <- predict(cvfit, xp, type = "response", s = "lambda.1se")
```




```
##     In sample AUC, lambda.min Out of sample AUC, lambda.min
##                       0.7631710                       0.7290185
##     In sample AUC, lambda.1se Out of sample AUC, lambda.1se
##                       0.7337026                       0.6906418
```

Drawbacks of LASSO

1. No p-values on coefficients
   - Simple solution -- run the resulting model with `glm()`
   - Solution only if using `family = "gaussian"`:
     - Run the lasso use the `package:lars` package
       - `m <- lars(x = x, y = y, type = "lasso")`
     - Then test coefficients using the `package:covTest` package
       - `covTest(m, x, y)`
2. Generally worse in sample performance
3. Sometimes worse out of sample performance (short run)
   - BUT: predictions will be more stable

Logistic with lambda.min variables

```
fit_glm_lambda.min <- glm(AAER ~ logtotasset + soft_assets + merger + restruct +
                  processedsize + paralen_s + sentlen_s + passive_p +
                  lm_negative_p + Topic_2_n_oI + Topic_6_n_oI +
                  Topic_8_n_oI + Topic_9_n_oI + Topic_12_n_oI + Topic_18_n_oI +
                  Topic_19_n_oI + Topic_23_n_oI + Topic_26_n_oI +
                  Topic_29_n_oI + Topic_30_n_oI,
                data = df[df$Test == 0, ], family = binomial)
tidy(fit_glm_lambda.min)
```

```
## # A tibble: 21 x 5
##    term              estimate    std.error statistic  p.value
##    <chr>                <dbl>        <dbl>     <dbl>    <dbl>
##  1 (Intercept)   -7.51         0.455         -16.5   2.49e-61
##  2 logtotasset    0.379        0.0376         10.1   5.61e-24
##  3 soft_assets    2.43         0.359           6.77  1.26e-11
##  4 merger         0.391        0.144           2.72  6.53e- 3
##  5 restruct       0.175        0.151           1.16  2.46e- 1
##  6 processedsize  0.000000957  0.000000854     1.12  2.62e- 1
##  7 paralen_s     -0.0369       0.0244         -1.51  1.31e- 1
##  8 sentlen_s     -0.0139       0.0155         -0.897 3.70e- 1
##  9 passive_p     -7.17         3.95           -1.81  6.96e- 2
## 10 lm_negative_p  8.48        11.8             0.716 4.74e- 1
## # ... with 11 more rows
```

Logistic with lambda.1se variables

```
fit_glm_lambda.1se <- glm(AAER ~ logtotasset + soft_assets + merger
                          + Topic_2_n_oI,
                          data = df[df$Test == 0, ], family = binomial)
tidy(fit_glm_lambda.1se)
```

```
## # A tibble: 5 x 5
##   term         estimate std.error statistic   p.value
##   <chr>           <dbl>     <dbl>     <dbl>     <dbl>
## 1 (Intercept)    -7.22     0.316     -22.8  1.51e-115
## 2 logtotasset     0.315    0.0327      9.64 5.35e- 22
## 3 soft_assets     2.26     0.324       6.99 2.72e- 12
## 4 merger          0.421    0.141       2.98 2.89e-  3
## 5 Topic_2_n_oI   82.6     23.1         3.58 3.42e-  4
```

Compare with Ridge and Elastic Net

SMU Singapore Management University School of Accountancy

```
# It takes 3 minutes on my Xeon W-2133 32G workstation
list.of.fits <- list() # To store the results
for (i in 0:10) {
  ## First, make a variable name that we can use later to refer
  ## to the model optimized for a specific alpha.
  ## For example, when alpha = 0, we will be able to refer to
  ## that model with the variable name "alpha0".
  fit.name <- paste0("alpha", i/10)

  ## Now fit a model (i.e. optimize lambda) and store it in a list that
  ## uses the variable name we just created as the reference.
  list.of.fits[[fit.name]] <-
    cv.glmnet(x = x, y = y, family = "binomial", alpha = i/10,
              type.measure = "auc")
}
```

Which alpha?

```
## Now we see which alpha (0, 0.1, ... , 0.9, 1) does the best job
## predicting the values in the Testing dataset.
results <- data.frame()
for (i in 0:10) {
  fit.name <- paste0("alpha", i/10)

  ## Use each model to predict 'y' given the Testing dataset
  pred <-
    predict(list.of.fits[[fit.name]], xp, type = "response",
            s = list.of.fits[[fit.name]]$lambda.1se)

  ROCpred_out <- prediction(as.numeric(pred[df$Test == 1 & !is.na(pred)]),
                            as.numeric(df[df$Test == 1 & !is.na(pred), ]$AAER))
  auc_out <- performance(ROCpred_out, measure = "auc")

  ## Store the results
  temp <- data.frame(alpha=i/10, auc = auc_out@y.values[[1]],
                     fit.name = fit.name)
  results <- rbind(results, temp)
}
```

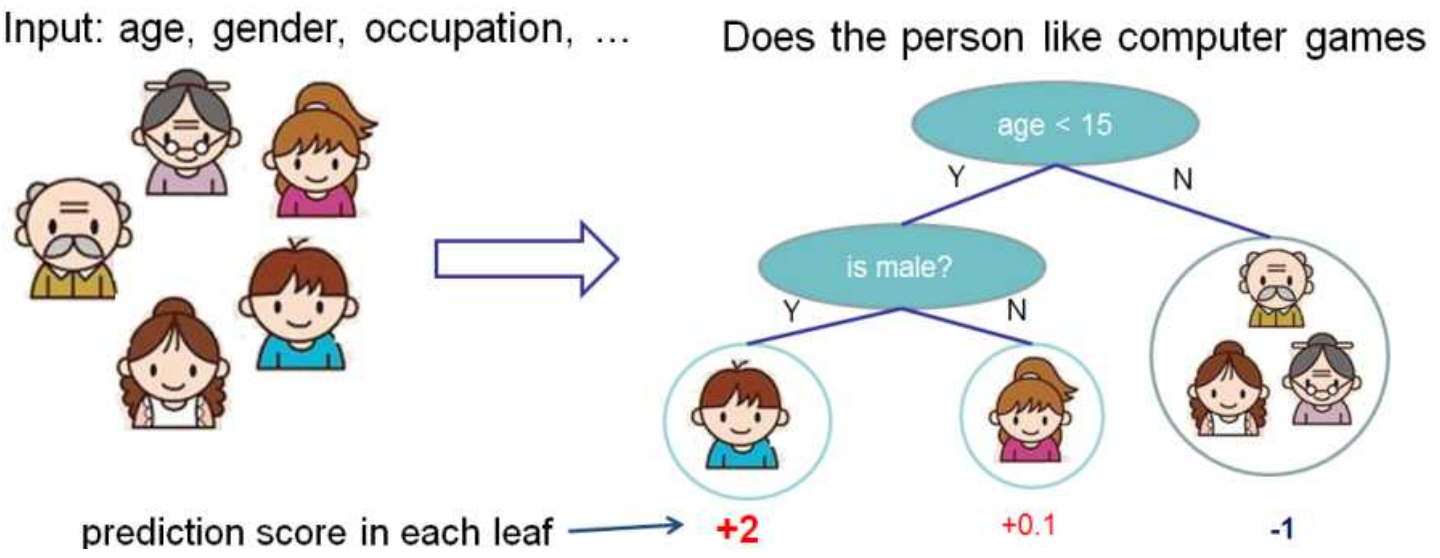
Which alpha?

alpha	auc	fit.name
0.0	0.7263743	alpha0
0.1	0.7145268	alpha0.1
0.2	0.7139278	alpha0.2
0.3	0.6957328	alpha0.3
0.4	0.6988006	alpha0.4
0.5	0.6904372	alpha0.5
0.6	0.7123501	alpha0.6
0.7	0.7122478	alpha0.7
0.8	0.7123574	alpha0.8
0.9	0.7169080	alpha0.9
1.0	0.7013864	alpha1

Supervised Learning with Boosted Trees

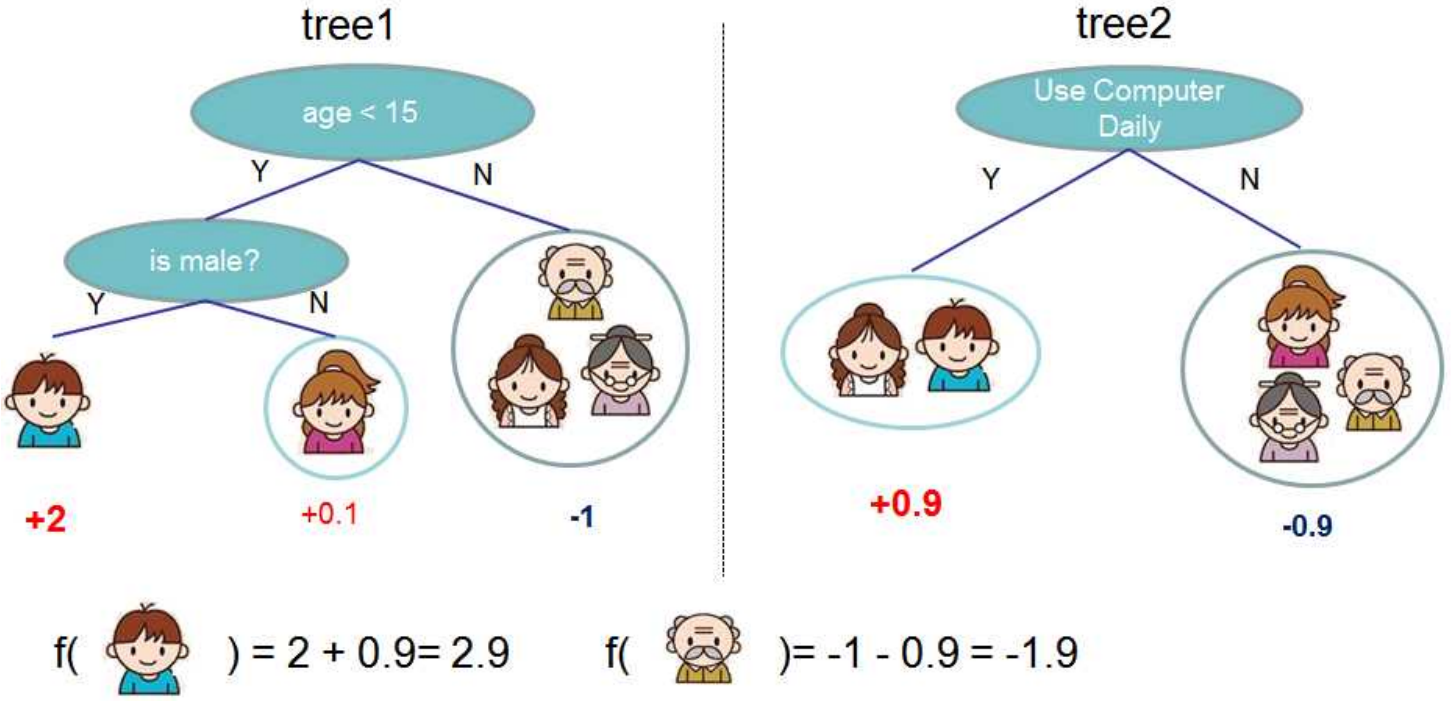
Decision trees

- We can make prediction through decision trees.
- Here's an example of a decision tree which classifies whether someone will like computer games.
- Decision tree is prone to overfitting (high variance) as it can keep growing until it has exactly one leaf node for every single observation (low bias).



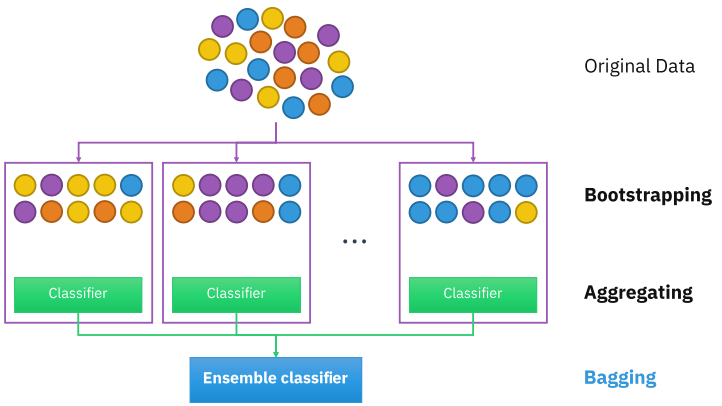
Decision tree ensembles

- To mitigate the overfitting, we can limit the tree depth and creat more trees (of course resulting in higher bias).
- Here is an example of a tree ensemble of two trees. The prediction scores of each individual tree are summed up to get the final score. An important benefit is that the two trees try to complement each other.



Random Forest

- A popular tree ensembles is the random forest
  - Random sampling of training data points when building trees
  - Random subsets of features considered when splitting nodes
- The random forest combines hundreds or thousands of decision trees, trains each one on a slightly different set of the observations, splitting nodes in each tree considering a limited number of the features.
  - It grows trees independently in parallel
- The final predictions of the random forest are made by averaging the predictions of each individual trees.
- This process is known as bootstrap aggregating (or bagging)
- The benefit is to reduce variance, ie, mitigate overfitting problem

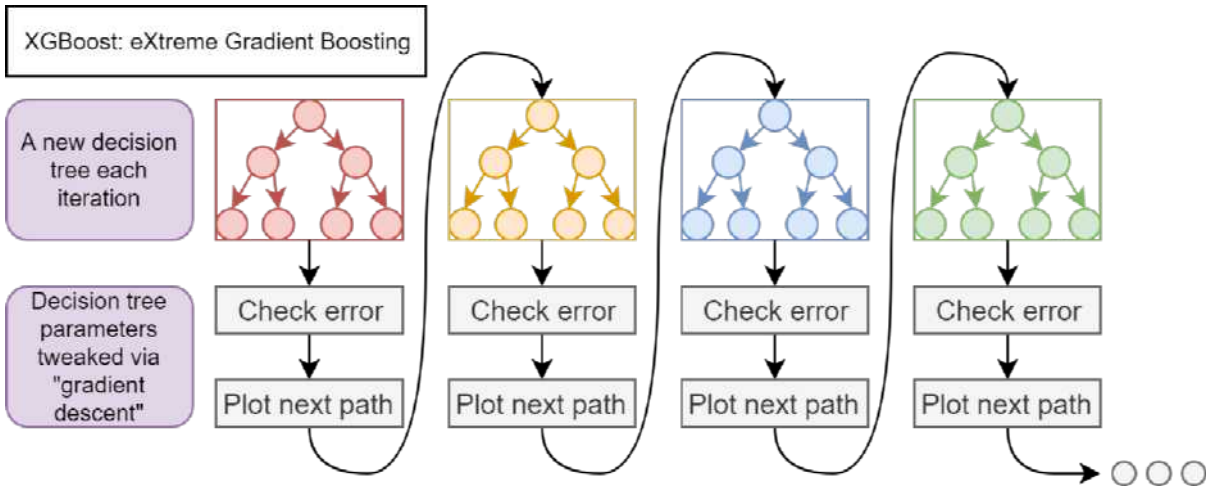


Gradient boosting through XGBoost

- *Boosting*: we use an additive strategy: fix what we have learned, and add one new tree at a time (ie, grow trees sequentially).
  - lower variance as we use multiple models (bagging)
  - lower bias by training subsequent trees using what errors the previous trees made (boosting).
- There are two main algorithms:
  - *Adaboost*: original, subsequent trees to punish more heavily observations mistaken by previous trees
  - *Gradient boosting*: train each subsequent tree using residuals
- `package:xgboost` is a library which provides a gradient boosting framework for C++, Java, Python, R, and Julia.

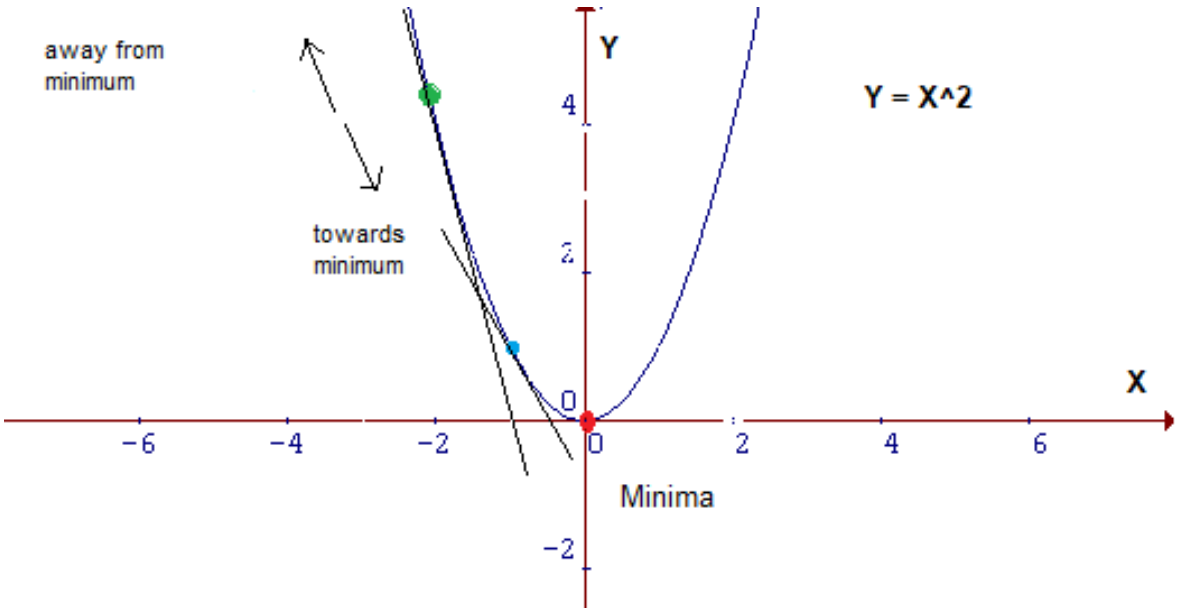
What is XGBoost

- e**X**treme **G**radient **B**oosting
- A simple explanation:
  1. Start with 1 or more decision trees & check error
  2. Make more decision trees & check error
  3. Use the difference in error to guess another model
  4. Repeat #2 and #3 until the model's error is stable



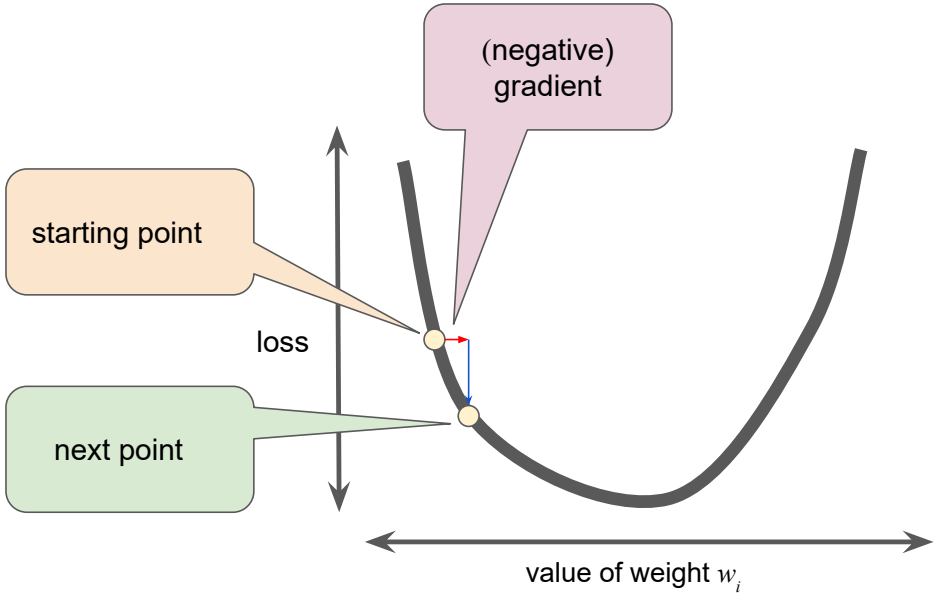
What is Gradient Descent

- Gradient descent algorithm is an iterative process that takes us to the minimum of a loss function (such as the SSE in a linear model)
  - min: slope of the tangent line (gradient or partial derivatives) = 0
  - in fact OLS is to estimate coeffcients which equal PD of SSE to zero
- Then why not OLS directly (or why iterate the gradient process?)
  - Many loss functions have no analytical solutions (ie, cannot solve PD = 0 equation such as $\frac{dy}{dx} = \frac{1}{x}$)
  - Have to rely on numerical solutions (by giving input values)



What is Learning Rate

- Gradient descent algorithms multiply the gradient by a scalar known as the learning rate (also called step size) to determine the next point.
  - eg, if the gradient magnitude is 2.5 and the learning rate is 0.01, then the gradient descent algorithm will pick the next point 0.025 away from the previous point.
  - an optimal learning rate can increase the model efficiency to converge, try the game



Steps of gradient descent

1. Take the derivative (gradient) of the loss function
2. Pick a random value of the parameters
3. Plug parameter values and calcuate gradient values (slope)
4. Step size = slope * learning rate
5. New parameter = old parameter - step size
6. Repeat 3-5 until step size is very small or reach max number of steps

> Stochastic Gradient Descent: use random subset at every step rather than the full dataset (to save time)

XGBoost model setup

```
# XGBoost Model Setup
library(xgboost)
set.seed(2021)

# These params take some work to pin down
params <- list(max_depth = 5, # Maximum depth of a tree
               eta = 0.2, # learning rate or step size for each boosting
               gamma = 10, # Min loss reduction to make a further partition
               min_child_weight = 20, # min obs needed to be in each node
               objective = "binary:logistic") # "reg:linear" for simple OLS
# The cross-validation function of xgboost
xgbCV <- xgb.cv(params = params, data = x, label = y,
                nrounds = 100, # The number of rounds/iterations for boosting
                eval_metric = "auc",
                # randomly partitioned into nfold equal size subsamples
                nfold = 10,
                # stratified sampling by the values of outcome labels
                # ie, same proportion of outcomes for subsamples
                stratified = TRUE)
# Boost at minimum number of iterations with the max AUC
# which(): position of the elements in a logical vector which are TRUE
numRounds <- min(which(xgbCV$evaluation_log$test_auc_mean ==
                       max(xgbCV$evaluation_log$test_auc_mean)))
```

Run XGBoost

```
fit_XGB <- xgboost(params = params,
                   data = x,
                   label = y,
                   nrounds = numRounds,
                   eval_metric = "auc")
```

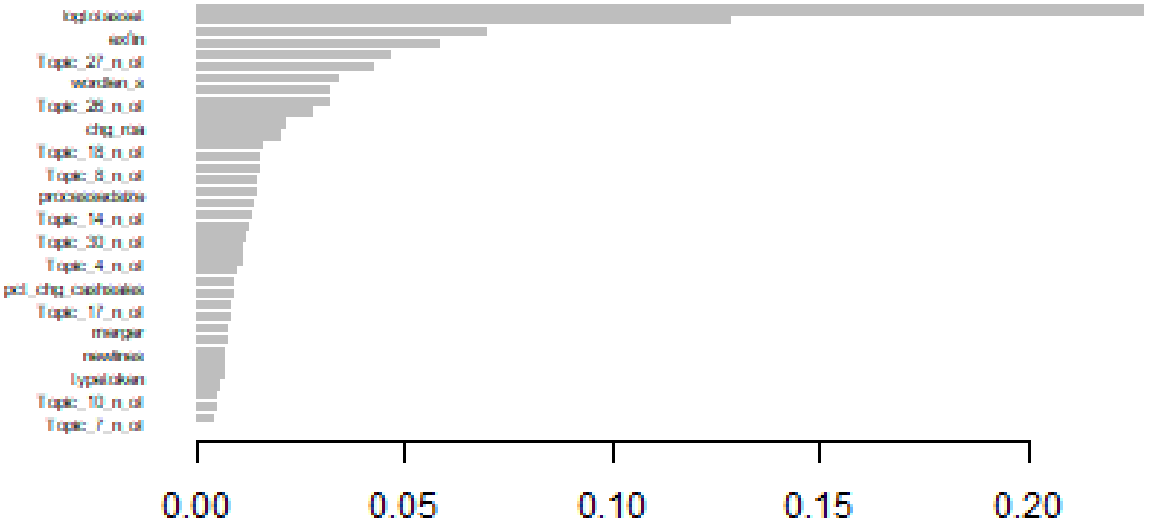
- Refer to the XGBoost Parameters Manual
- Refer to the xgb.cv manual
- Refer to the Assignment for parameter tuning
- You can plot the trees from your model using `xgb.plot.tree()`

Relative importance of variables

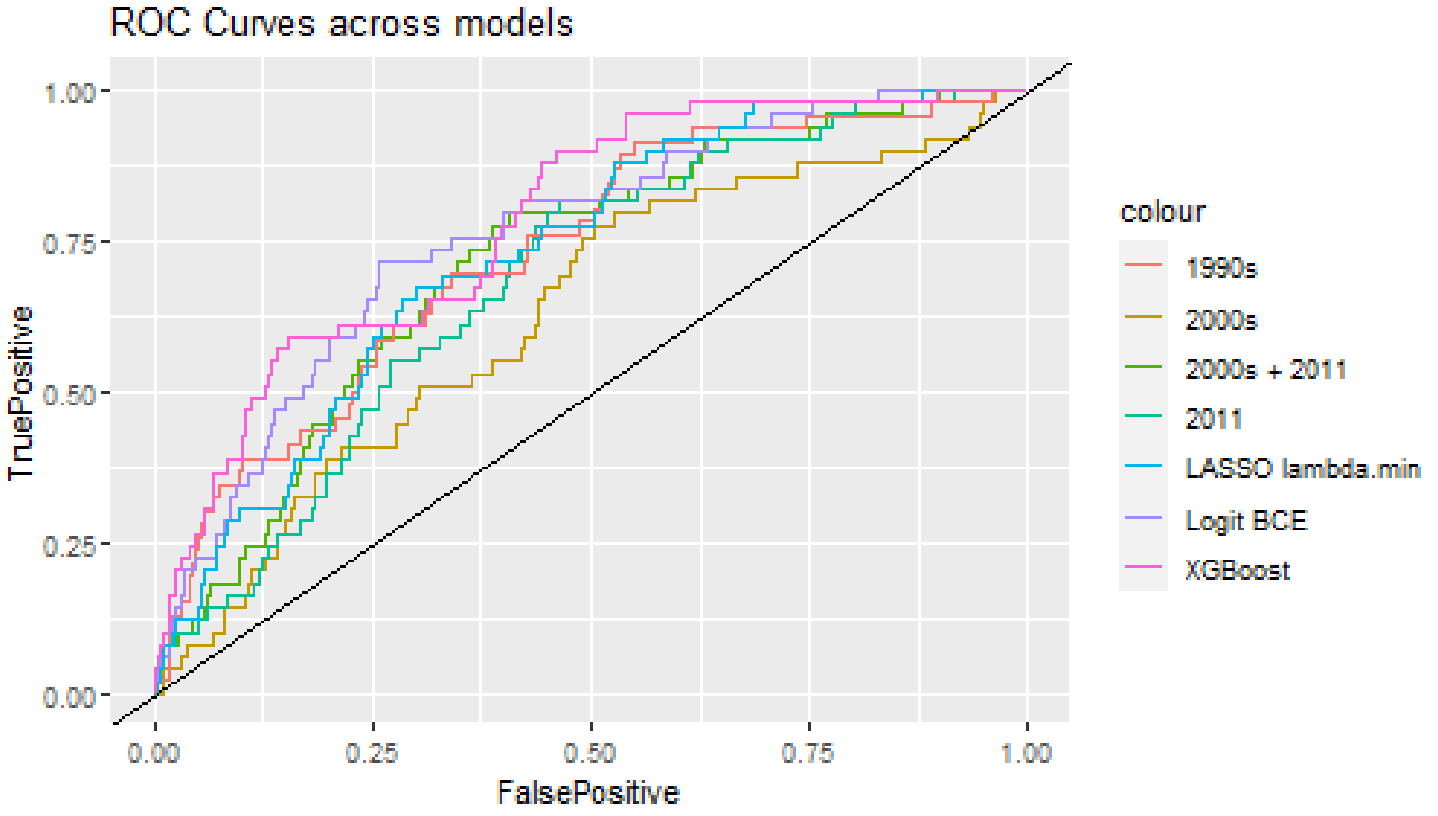
```
# Display relative importance of variables for prediction
xtest <- model.matrix(BCE_eq, data = df[df$Test == 1, ])[ , -1]
ytest <- model.frame(BCE_eq, data = df[df$Test == 1, ])[ , "AAER"]
xgb.train.data = xgb.DMatrix(x, label = y, missing = NA)
xgb.test.data = xgb.DMatrix(xtest, label = ytest, missing = NA)
col_names = attr(xgb.train.data, ".Dimnames")[[2]]
imp = xgb.importance(col_names, fit_XGB)
```

Plot relative importance of variables

```
# the bar represents the improvement in accuracy brought by a feature
xgb.plot.importance(imp)
```



Model performance




```
##           1990s            2011           2000s  2000s + 2011
##       0.7292981       0.6849225       0.6295414       0.7147021
##             BCE LASSO, lambda.min     XGBoost AUC
##       0.7599594       0.7290185       0.7817700
```

Explaining XGBoost

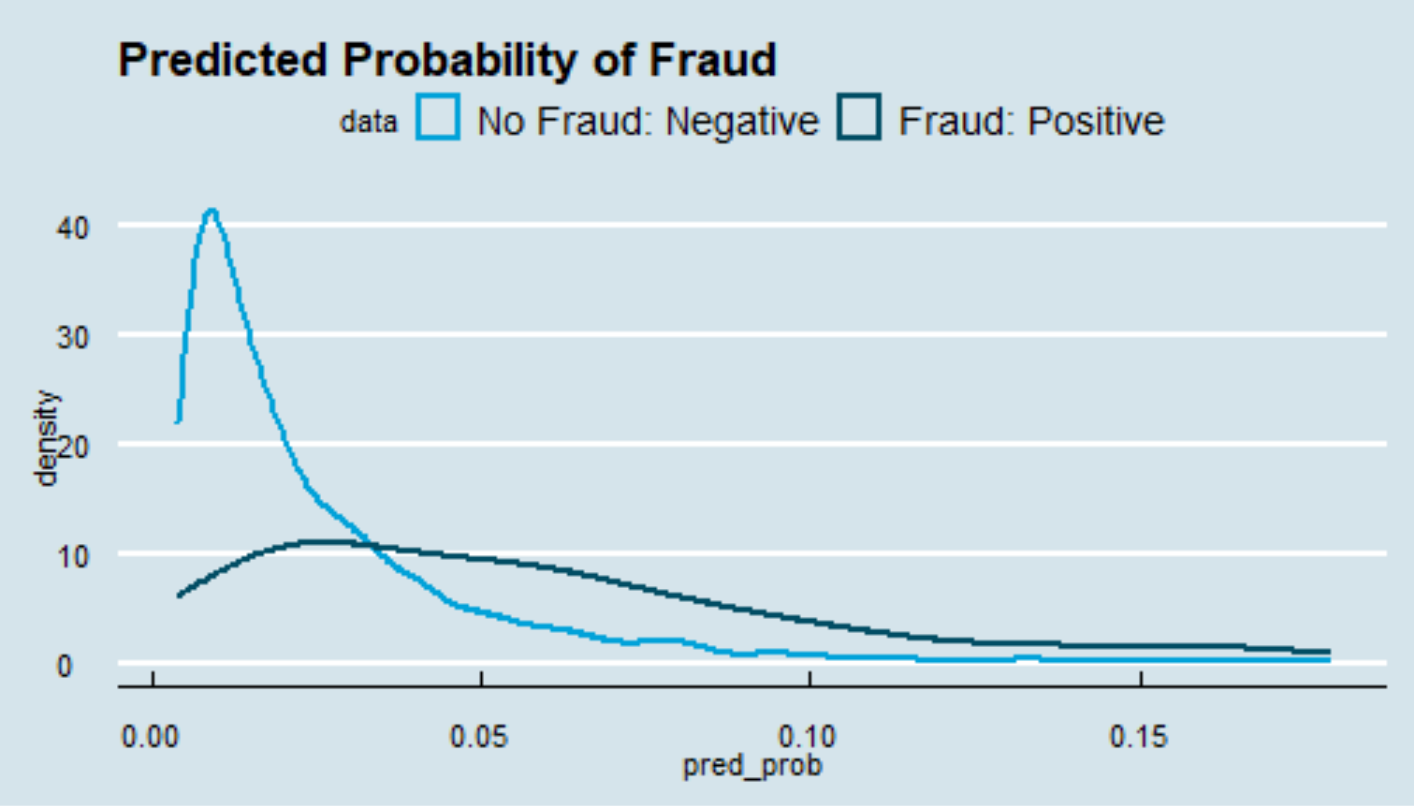
```r
# Use the following code to install the xgboostExplainer package
# You may need to launch the RStudio "as an administrator"
# It is also advised to launch a fresh RStudio without any startup files
# install.packages("devtools")
# library(devtools)
# install_github("AppliedDataSciencePartners/xgboostExplainer", force = T)
library(xgboostExplainer)
explainer = buildExplainer(fit_XGB, xgb.train.data, type = "binary",
                           base_score = 0.5, trees_idx = NULL)
pred.breakdown = explainPredictions(fit_XGB, explainer, xgb.test.data)
```

How about Enron in 2000

```
# See what XGBoost is thinking of Enron Corp (gvkey 6127) in year 2000
# enron_row is the position of record for Enron (row number 2237 in the data)
# the prediction is -2.9 (the last black bar), so how much is the probability?
enron_row = 2237
showWaterfall(fit_XGB, explainer, xgb.test.data, data.matrix(df[df$Test == 1, ])
              enron_row, type = "binary") +
  ggtitle("Enron Corp in 2000 -- what's probability to have fraudulent report?")
```

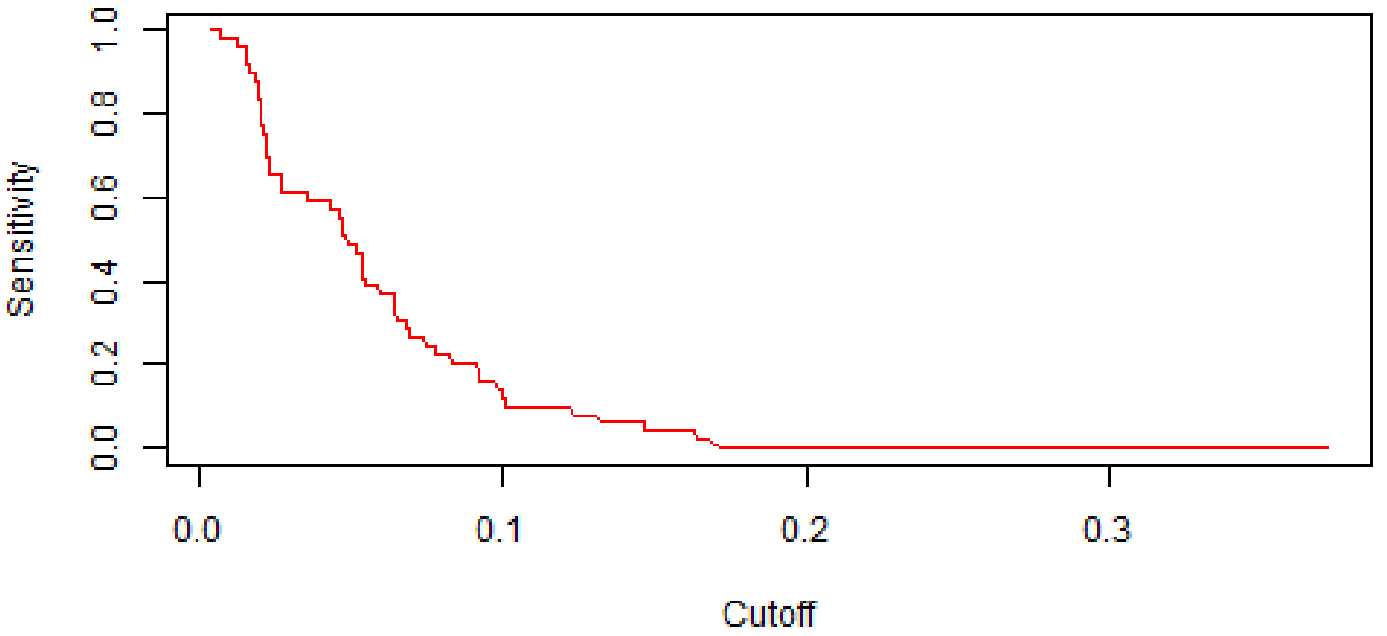
The cutoff

- Note that logistic regression does not predict 1 or 0 directly
- Pick a cutoff value, ie, fraud (positive) classification if >= cutoff%
- The optimal cutoff depends on data and context
- Check a more detailed explanation

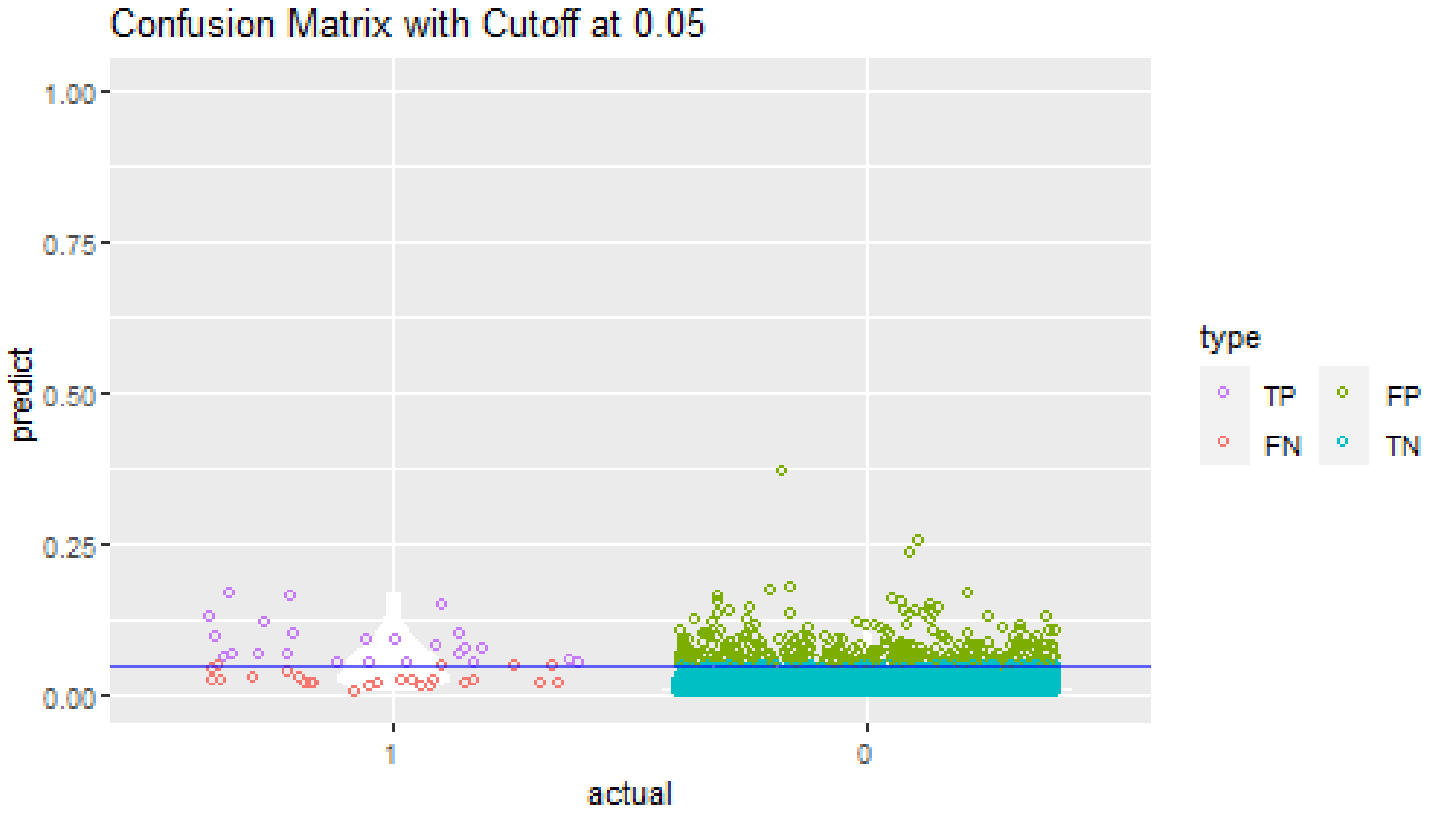


Sesentivity vs. Cutoff

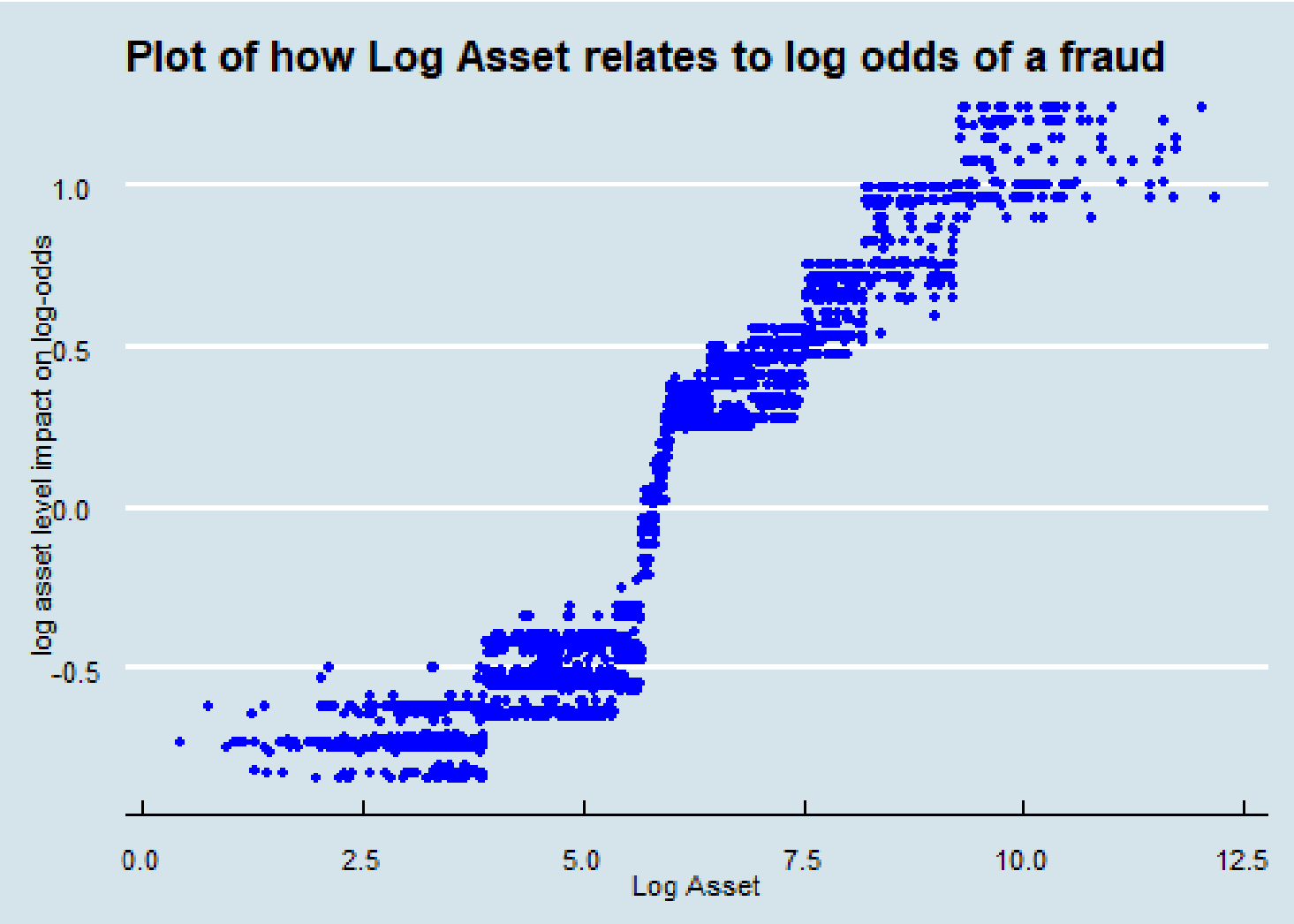
- For imbalanced data, we care more about Sensitivity (TP), rather than Accuracy (TP + TN)
- The following plot shows the relation between Sensitivity and Cutoff values



Confusion Matrix at Cutoff 0.05



The impact on logodds



Summary of Session 9

For next week

- Try to replicate the code
- Continue your Datacamp career track
- Try the new models with your project data
- Submit project by the dateline and prepare for presentation

Supplementary readings

SMU SINGAPORE MANAGEMENT UNIVERSITY
School of Accountancy

- Many Ways to Lasso
- Tune XGBoost with tidymodels
- Using XGBoost with Tidymodels
- Choosing Cutoff Value for Unbalanced Dataset
- XGBoost

R Coding Style Guide

Style is subjective and arbitrary but it is important to follow a generally accepted style if you want to share code with others. I suggest the The tidyverse style guide which is also adopted by Google with some modification

- Highlights of **the tidyverse style guide**:
  - *File names*: end with .R
  - *Identifiers*: variable_name, function_name, try not to use "." as it is reserved by Base R's S3 objects
  - *Line length*: 80 characters
  - *Indentation*: two spaces, no tabs (RStudio by default converts tabs to spaces and you may change under global options)
  - *Spacing*: x = 0, not x=0, no space before a comma, but always place one after a comma
  - *Curly braces {}*: first on same line, last on own line
  - *Assignment*: use `<-`, not `=` nor `->`
  - *Semicolon(;)*: don't use, I used once for the interest of space
  - *return()*: Use explicit returns in functions: default function return is the last evaluated expression
  - *File paths*: use relative file path "../../filename.csv" rather than absolute path "C:/mydata/filename.csv". Backslash needs `\\`

R packages used in this slide

This slide was prepared on 2021-09-08 from Session_9s.Rmd with R version 4.1.1 (2021-08-10) Kick Things on Windows 10 x64 build 18362 😀.

The attached packages used in this slide are:

```
##       data.table         ggthemes xgboostExplainer          xgboost
##         "1.14.0"          "4.2.4"             "0.1"        "1.4.1.1"
##           glmnet           Matrix              ROCR         coefplot
##          "4.1-2"          "1.3-4"         "1.0-11"          "1.2.7"
##            broom          forcats           stringr            dplyr
##          "0.7.9"          "0.5.1"           "1.4.0"          "1.0.7"
##            purrr            readr             tidyr           tibble
##          "0.3.4"          "2.0.1"           "1.1.3"          "3.1.3"
##          ggplot2        tidyverse       kableExtra            knitr
##          "3.3.5"          "1.3.1"           "1.3.4"           "1.33"
```