Exam

Table of contents

1 Introduction

The goal of this exam is to build and assess a predictive model to quantify the probability of corporate default **with a 1-year horizon**, using annual financial statement data reported by companies. The companies are listed on the US stock market.

Financial statement data relating to these companies is available in a data set. The companies have been anonymized and are designed by a unique index (column `company_id`).

The fiscal year to which the financial statements pertain is given (column `fiscal_year`). We assume that fiscal years run from January to December for all companies in the data set, and that financial statements are published and available the first day of next fiscal year.

When a corporate default (column `default`) occurs for a company during a given fiscal year, say `2011`, the observation (ie row in data set) of the preceding fiscal year (`2010`) for this company shows a value of `1` in the column `default`, `0` otherwise.

If an observation for fiscal year `2010` shows `1` in the column `default` it means that default has occurred sometimes in `2011`. To simplify and in the absence of information we assume that default event has occurred the last day `2011`, and consequently that the event has been recorded the first day of `2012`.

Your goal is to predict corporate default with a 1-year horizon. We assume that we make the predictions at the beginning of each year.

Given financial data up to a certain year, say `2011` (available the first day of `2012`), we do not yet know which companies will default by year end `2012`, but we want to predict it (hence the goal: predict corporate default with a 1-year horizon).

To make the predictions, we only know that some companies have defaulted by year end `2011` and before; this information being recorded in an observation with fiscal year `2010` and before. We also know the financial data for fiscal year `2011` and before.

We briefly describe the columns of the data set (Rows: 61,092 Columns: 23):

- `company_id`: a unique identifier for the company.
- `default`: 0 for a healthy company (ie no default during the next fiscal year), 1 for a company in default during the next fiscal year; to simplify it is assumed the default event occurs the last day of the next fiscal year.
- `default_will_occur`: 0 for a company that will not default in any of the next observed fiscal years, 1 for a company defaulting during one of the next observed fiscal years; when default occurs observations stop for the company.
- `fiscal_year`: a fiscal year is a twelve-month period chosen by a company to report its financial information; to simplify we assume here that it runs from January to December for each company, and financial statements are published the first day of next fiscal year.

- `fiscal_year_default`: if applicable the fiscal year at which a company has defaulted, NA otherwise.
- `cogs`: the cost of goods sold (COGS) is the sum of all direct costs associated with making a product. It appears on an income statement and typically includes money mainly spent on raw materials and labour. It does not include costs associated with marketing, sales or distribution.
- `current_asset`: the total value of a company's current assets; a current asset is any asset which can reasonably be expected to be sold, consumed, or exhausted through the normal operations of a business within the coming year.
- `da`: depreciation and amortization ; refers to the decline in value over time of tangible assets (depreciation: real estate, buildings, plants, equipment) or intangible assets (amortization: trade names, copyrights, patents, trademarks).
- `ebit`: earnings before interest, taxes (EBIT).
- `ebitda`: earnings before interest, taxes, depreciation, and amortization (EBITDA) is a widely used measure of core corporate profitability. EBITDA is calculated by adding interest, tax, depreciation, and amortization expenses to net income.
- `gp`: gross profit is the profit a company makes after deducting the costs associated with making and selling its products, or the costs associated with providing its services; calculated by subtracting the COGS from revenue/sales.
- `invt`: inventory or stock refers to the goods and materials that a business holds for the ultimate goal of resale, production or utilization.
- `mv`: market value, here market capitalization as companies are listed.
- `ni`: net income (also total comprehensive income, net earnings, net profit, bottom line, sales profit, or credit sales) company's income minus cost of goods sold, expenses, depreciation and amortization, interest, and taxes for an accounting period.
- `nsale`: sales refer to any operating revenues that a company earns through its business activities, such as selling goods, services, products; here net sales are provided ie sales minus applicable sales returns, allowances, and discounts.
- `opex`: operating expenses are expenses that a business incurs through its normal business operations; they include rent, equipment, inventory costs, marketing, payroll, insurance, step costs, and funds allocated for research and development.
- `re`: retained earnings are the cumulative net earnings or profits of a company after accounting for dividend payments, it shows the net income that a company has saved over time, and therefore has the ability to reinvest in the business or distribute to shareholders.
- `ta`: total asets, the complete collection of assets or valuable items owned by a company.

- `tcl`: total current liabilities is the sum of all current liabilities which are a company's short-term financial obligations that are due within one year (or within a normal operating cycle).
- `tl`: total liabilities are the combined debts and obligations that an individual or company owes to outside parties.
- `tltd`: total long term debt is any amount of outstanding debt a company holds that has a maturity of 12 months or longer.
- `tr`: total revenue (or sales, top line, gross income) is the money generated from normal business operations. It is the top line figure from which costs are subtracted to determine net income.
- `trec`: total receivable are the balance of money due to a firm for goods or services delivered or used but not yet paid for by customers.

Below we load the data set and display its structure:

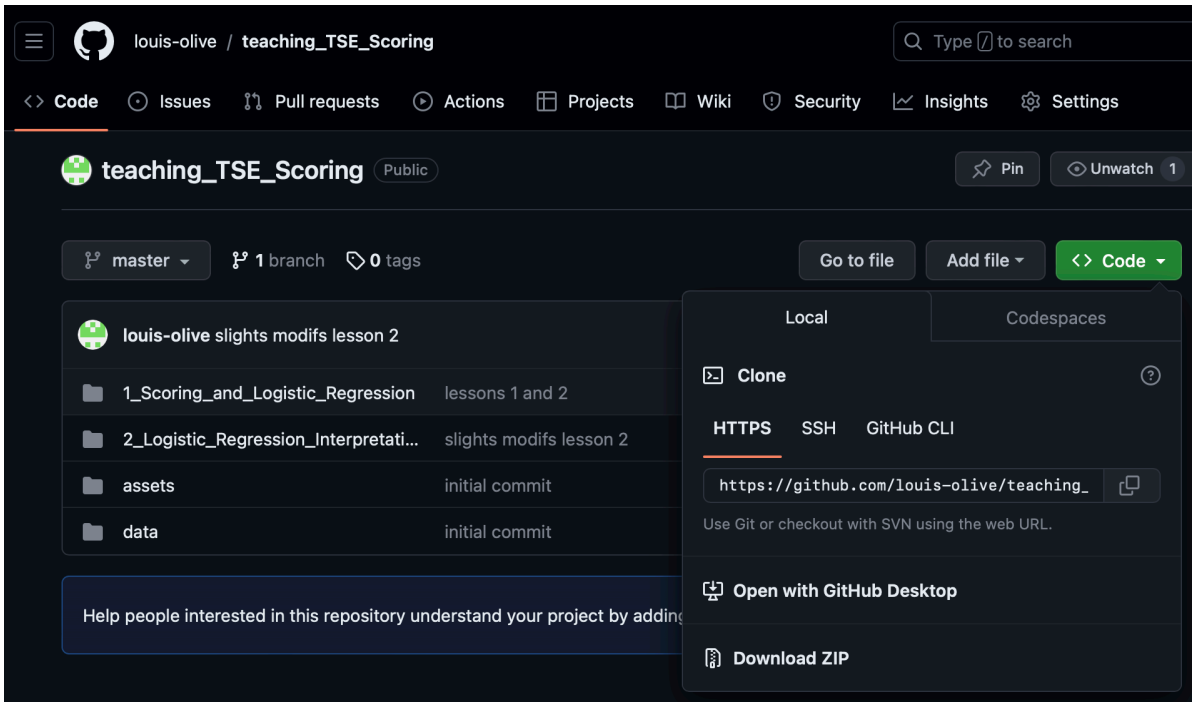
```
# loading and printing the structure of the dataset
default_dataset <- read_csv('data/default_dataset.csv')
glimpse(default_dataset)
```

```
Rows: 61,092
Columns: 23
$ company_id          <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2,~
$ default             <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
$ default_will_occur  <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
$ fiscal_year         <dbl> 2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007, 20~
$ fiscal_year_default <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
$ cogs                <dbl> 713.811, 526.477, 496.747, 523.302, 598.172, 704.0~
$ current_asset       <dbl> 485.856, 436.656, 396.412, 432.204, 474.542, 624.4~
$ da                  <dbl> 18.577, 22.496, 27.172, 26.680, 27.950, 29.222, 32~
$ ebit                <dbl> 45.790, 4.711, 3.573, 20.811, 33.824, 62.655, 86.7~
$ ebitda              <dbl> 64.367, 27.207, 30.745, 47.491, 61.774, 91.877, 11~
$ gp                  <dbl> 160.444, 112.244, 109.590, 128.656, 149.676, 193.2~
$ invt                <dbl> 320.590, 286.588, 259.954, 247.245, 255.477, 323.5~
$ mv                  <dbl> 377.1180, 364.5928, 143.3295, 308.9071, 522.6794, ~
$ ni                  <dbl> 18.531, -58.939, -12.410, 3.504, 15.453, 35.163, 5~
$ nsale               <dbl> 874.255, 638.721, 606.337, 651.958, 747.848, 897.2~
$ opex                <dbl> 809.888, 611.514, 575.592, 604.467, 686.074, 805.4~
$ re                  <dbl> 204.065, 139.603, 124.106, 131.884, 142.450, 183.5~
$ ta                  <dbl> 701.854, 710.199, 686.621, 709.292, 732.230, 978.8~
$ tcl                 <dbl> 125.392, 150.464, 203.575, 131.261, 160.025, 187.7~
```

```
$ tl                <dbl> 361.642, 399.964, 391.633, 407.608, 417.486, 556.1~
$ tltd              <dbl> 179.987, 217.699, 164.658, 248.666, 227.159, 318.5~
$ tr                <dbl> 874.255, 638.721, 606.337, 651.958, 747.848, 897.2~
$ trec              <dbl> 115.187, 77.528, 66.322, 104.661, 127.121, 136.272~
```

2 Instructions

- The exam is open documents, open browser. If copying large chunks of codes from the browser, give a reference (link to website, stackoverflow, stats.stackexchange etc). You can reuse code from the two first lessons hosted here https://github.com/louis-olive/teaching_TSE_Scoring/, click on Code/Download ZIP for the last version:



- The first parts of the exam are to be performed in-class (**TO DO in-class** in the exam document). The in-class exam will last two hours (10:30-12:30).
- You must use the R programming language, preferably through RStudio. Your code for analysis should use one of the following formats: preferably quarto Markdown (.qmd, as done in the course), but you might prefer R Markdown (.Rmd) or an R file (.R).
- Packages that you may need besides `base R` and `stats` (`glm()`, `step()`) that have been used in the course include:

`tidyverse`, `broom`, `ROCR`, `car`, `aod`, `rsample`, `bestglm`, `glmnet`, `glmnetUtils`, `splines`

You can install them using the following code (uncomment):

```
# # UNCOMMENT IF NEEDED
# # https://statsandr.com/blog/an-efficient-way-to-install-and-load-r-packages/
# # Package names
# packages <- c("tidyverse", "ROCR", "car", "aod", "broom", "rsample", "bestglm", "glmnet"
#
# # Install packages not yet installed
# installed_packages <- packages %in% rownames(installed.packages())
# if (any(installed_packages == FALSE)) {
#   install.packages(packages[!installed_packages])
# }
#
# # Packages loading
# invisible(lapply(packages, library, character.only = TRUE))
#
#
# # Additional packages used throughout the course but not needed for the analysis
# additional_packages <- c( "purrr", "pROC", "foreign", "patchwork", "class", "scales", "r
```

Check that it works on your computer before the course.

- For your report: either render a .html file of your analysis (when using .qmd/.Rmd) or provide a rich document with your text plus tables/plots of your analysis (.docx or .pdf). For the last solution again no recommendation but LibreOffice is free and runs cross-platforms.
- The code file and report's general readability will impact the grading as well as the richness of their content (do not hesitate to comment on your intents, assumptions, findings, conclusions, especially in the take home part).
- The .html file for report should be readable in a standard web browser (Chrome, Safari, Firefox, Edge …), alternatively .docx/.pdf should be valid. The .qmd/.Rmd/.R code file should run without errors (if something is not working as you wish, comment in the code with your intents). They should be posted before 12:30 on 27 September 2023 for the in-class part to my two email addresses `louis.olive@ut-capitole.fr`, `louis.olive@gmail.com` (in case the first one encounters issues) with subject `SCORING EXAM - YOUR NAME`. You can prepare your email in advance to save time at the end of exam.
- Allow yourself at least 15 minutes before the end to check your .qmd/.Rmd/.R file is running and you have a readable report. If you finish before the end, and are happy with the result post me your code/report and take a well deserved rest!

- If you are not happy with some or all parts of the in-class analysis, you might complete/correct/improve it at home, if it improves your grading a maximum of half of points will be given (for each relevant improvement).
- For the take-home part the deadline is 4 October 2023 00:00.
- Regarding the grading:
  - 30% of the total points for the in-class part, 70% for the at-home part
  - in-class: 10% of the total points for each parts: Data exploration, First step and Second step
  - at-home: 5% for the Third step, the rest will be detailed at the end of in-class exam.

You should follow the following plan in you report:

3 Data Exploration (10% total points)

3.0.1 TO DO in-class (~30-45 minutes) with first priority**

In the first part, you will need to explore carefully the data set, for example:

- providing relevant descriptive statistics,
- showing correlated features,
- showing/plotting individual features "interaction" with the response variable,
- also taking into account the time-series/temporal aspect of the data set.

Visualizations are expected.

You can remove observations from data if justified.

You might also need to remove some feature variables is perfectly co-linear or any other reason.

Also try to understand how you will use the data set to solve the problem.

3.0.2 Proposed solution

We first start by checking for each fiscal year, the number of "next-year" default occurring and computing the proportion of defaulting firms (note that we use the `default` variable):

```
# checking summary statistics with the article (its identical)
default_dataset %>%
```

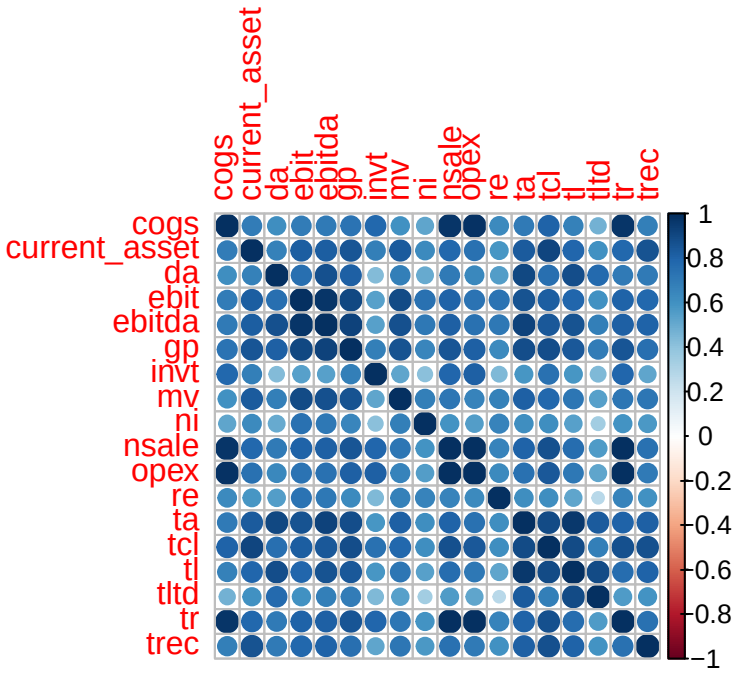
```
    group_by(fiscal_year) %>%
    summarise(total_firm = n_distinct(company_id),
              default = sum(default),
              pct_default = default / total_firm)
```

```
# A tibble: 15 x 4
   fiscal_year total_firm default pct_default
         <dbl>      <int>   <dbl>       <dbl>
 1        2000       5226       7     0.00134
 2        2001       4897      10     0.00204
 3        2002       4651      17     0.00366
 4        2003       4417      29     0.00657
 5        2004       4348      46     0.0106
 6        2005       4205      40     0.00951
 7        2006       4128      51     0.0124
 8        2007       4009      59     0.0147
 9        2008       3857      58     0.0150
10        2009       3743      23     0.00614
11        2010       3625      35     0.00966
12        2011       3513      25     0.00712
13        2012       3485      26     0.00746
14        2013       3484      28     0.00804
15        2014       3504      33     0.00942
```

We notice that the default event is a rare event, varying with vintage (picking up above 1% around the Great Financial Crisis)

We visualize correlations between predictors

```
cor_default <- cor(default_dataset %>% select(cogs:trec))

corrplot::corrplot(cor_default)
```



At first sight, we will have to investigate (ebit,ebitda, gp), (cogs, nsale, opex, tr) and (ta, tcl, tl) which seem strongly correlated.

We sort by the most positively correlated coefficients:

```
# http://www.sthda.com/english/wiki/correlation-matrix-a-quick-start-guide-to-analyze-form
# +++++++++++++++++++++++++++
# flattenCorrMatrix
# +++++++++++++++++++++++++++
# cormat : matrix of the correlation coefficients
flattenCorrMatrix <- function(cormat) {
  ut <- upper.tri(cormat)
  tibble(
    row = rownames(cormat)[row(cormat)[ut]],
    column = rownames(cormat)[col(cormat)[ut]],
    cor  =(cormat)[ut]
  )
}

flattenCorrMatrix(cor_default)
```

```
# A tibble: 153 x 3
```

```
   row           column          cor
   <chr>         <chr>         <dbl>
 1 cogs          current_asset 0.697
 2 cogs          da            0.611
 3 current_asset da            0.671
 4 cogs          ebit          0.699
 5 current_asset ebit          0.826
 6 da            ebit          0.758
 7 cogs          ebitda        0.710
 8 current_asset ebitda        0.823
 9 da            ebitda        0.873
10 ebit          ebitda        0.980
# i 143 more rows
```

```
flattenCorrMatrix(cor_default) %>%
  arrange(desc(cor)) %>%
  filter(cor > 0.90)
```

```
# A tibble: 13 x 3
   row           column   cor
   <chr>         <chr>  <dbl>
 1 nsale         tr     1
 2 nsale         opex   0.994
 3 opex          tr     0.994
 4 cogs          opex   0.989
 5 ebit          ebitda 0.980
 6 cogs          nsale  0.978
 7 cogs          tr     0.978
 8 ta            tl     0.969
 9 ebitda        gp     0.925
10 ebitda        ta     0.924
11 current_asset tcl    0.915
12 da            ta     0.909
13 ebit          gp     0.905
```

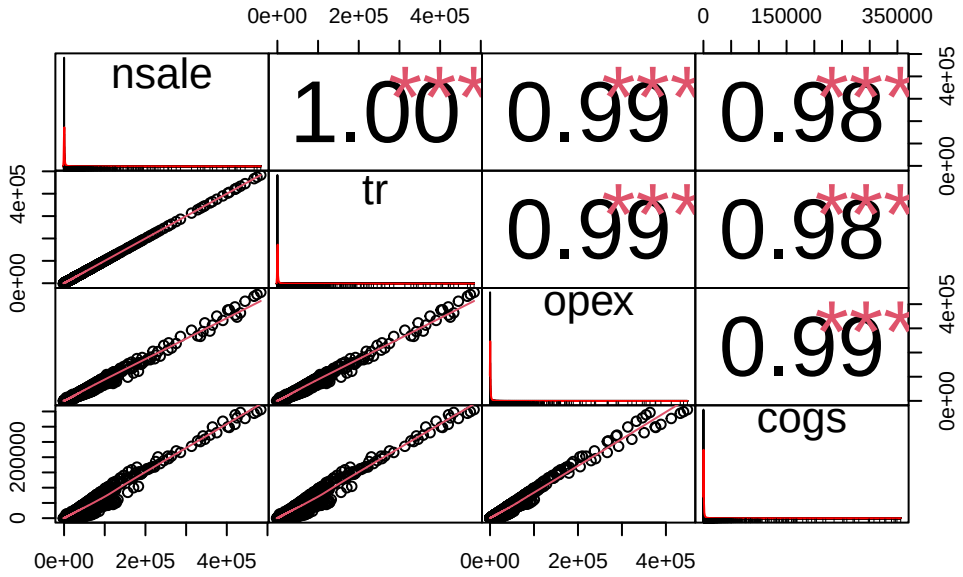
```
# # A tibble: 190 × 3
# row    column   cor
# <chr>  <chr>  <dbl>
# 1 nsale  tr     1
# 2 nsale  opex   0.994
# 3 opex   tr     0.994
```

```
# 4 cogs   opex   0.989
# 5 ebit   ebitda 0.980
# 6 cogs   nsale  0.978
# 7 cogs   tr     0.978
# 8 ta     tl     0.969
# 9 ebitda gp     0.925
# 10 ebitda ta     0.924
# #   180 more rows
# #   Use `print(n = ...)` to see more rows
```

nsale and tr are perfectly colinear and indeed share the same values, we remove tr from the predictors.

Operating expenses (opex) and cost of goods sold (cogs) see here are two expenditures incurred by businesses. Operating expenses refer to expenditures that are not directly tied to the production of goods or services, such as rent, utilities, office supplies, and legal costs. Cost of goods sold refers to expenses directly related to the production of a product, such as the materials needed to assemble a product and the transportation needed to bring goods from a distributor to a retailer. Both types of expenses are recorded as separate line items on a company's income statement. We could combine these two variables into a new variable expenditure. They are also strongly linked to the revenue/sales of the company. All of this suggest an important role of variable selection in the analysis.

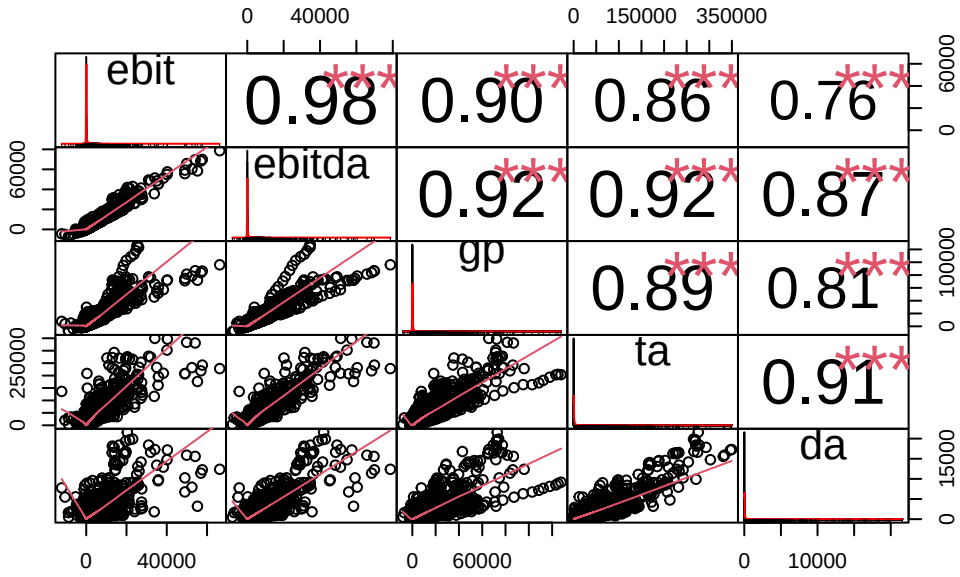
```
PerformanceAnalytics::chart.Correlation(default_dataset %>%
                                          select(nsale, tr, opex, cogs),
                                        histogram=TRUE, pch=19)
```




```
default_dataset %>%
  select(opex, cogs, nsale)
```

```
# A tibble: 61,092 x 3
    opex  cogs nsale
   <dbl> <dbl> <dbl>
 1  810.  714.  874.
 2  612.  526.  639.
 3  576.  497.  606.
 4  604.  523.  652.
 5  686.  598.  748.
 6  805.  704.  897.
 7  942.  837. 1061.
 8 1216. 1081. 1385.
 9 1258. 1111. 1424.
10 1218. 1066. 1352.
# i 61,082 more rows
```

```
PerformanceAnalytics::chart.Correlation(default_dataset %>%
                                          select(ebit, ebitda, gp, ta, da),
                                        histogram=TRUE, pch=19)
```



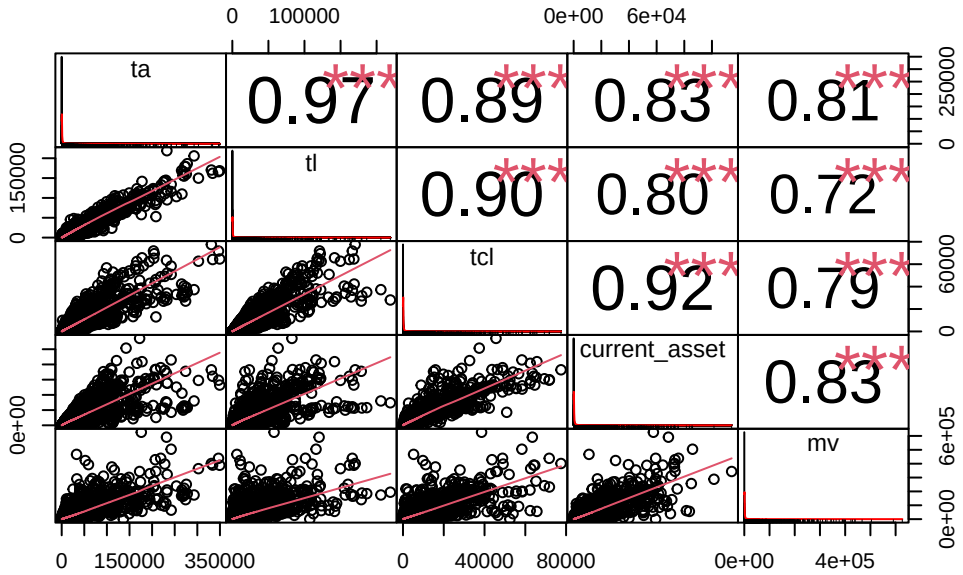
We can remove ebitda as ebitda is obtained as ebit + da (except for two observations):

```
default_dataset %>%
  mutate(ebitda_check = ebit + da) %>%
  select(ebit, da, ebitda_check, ebitda) %>%
  filter(abs(ebitda_check - ebitda)>1e-3)# A tibble: 2 × 4
```

```
# A tibble: 2 x 4
   ebit    da ebitda_check ebitda
  <dbl> <dbl>        <dbl>  <dbl>
1 2.00   2.38         4.38   5.56
2 0.992 10.9          11.9   12.1
```

```
# ebit    da ebitda_check ebitda
# <dbl> <dbl>        <dbl>  <dbl>
# 1 2.00   2.38         4.38   5.56
# 2 0.992 10.9          11.9   12.1
```

```
PerformanceAnalytics::chart.Correlation(default_dataset %>%
                                          select(ta, tl, tcl, current_asset, mv),
                                        histogram=TRUE, pch=19)
```



Also total assets (ta) and liabilities (tl) which are two sides of the balance sheet are strongly correlated as well as their sub-accounts.

Also total assets (ta), liabilities (tl) or market value (mv) would be natural scaling factors relating to the size of the balance sheet / company. It could lead to more comparable balance sheet or income statement figures.

For visualization and latter use we scale predictor by `ta` and also winsorize variables, we also remove perfect co-linear predictors:

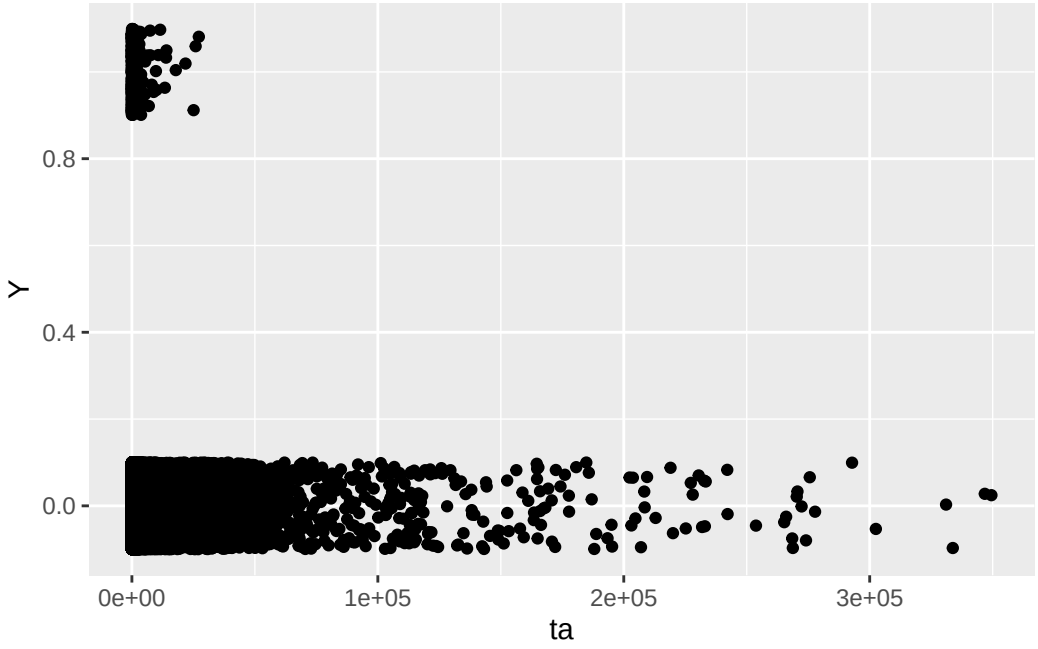
```
# -   showing/plotting individual features "interaction" with the response variable,

default_exam <- default_dataset %>%
  select(-ebitda,-tr,- fiscal_year_default) %>%
  mutate(Y = default,
         Y_alt = default_will_occur) %>%
  # scaling by `ta`
  mutate(across(.cols = cogs:trec,
                .fns = ~ . / ta,
                .names = "{.col}_/_ta")) %>%
  # winsorizing
  mutate(across(.cols = cogs:trec,
                .fns = ~ DescTools::Winsorize(.x , probs = c(0.05, 0.95)),
```

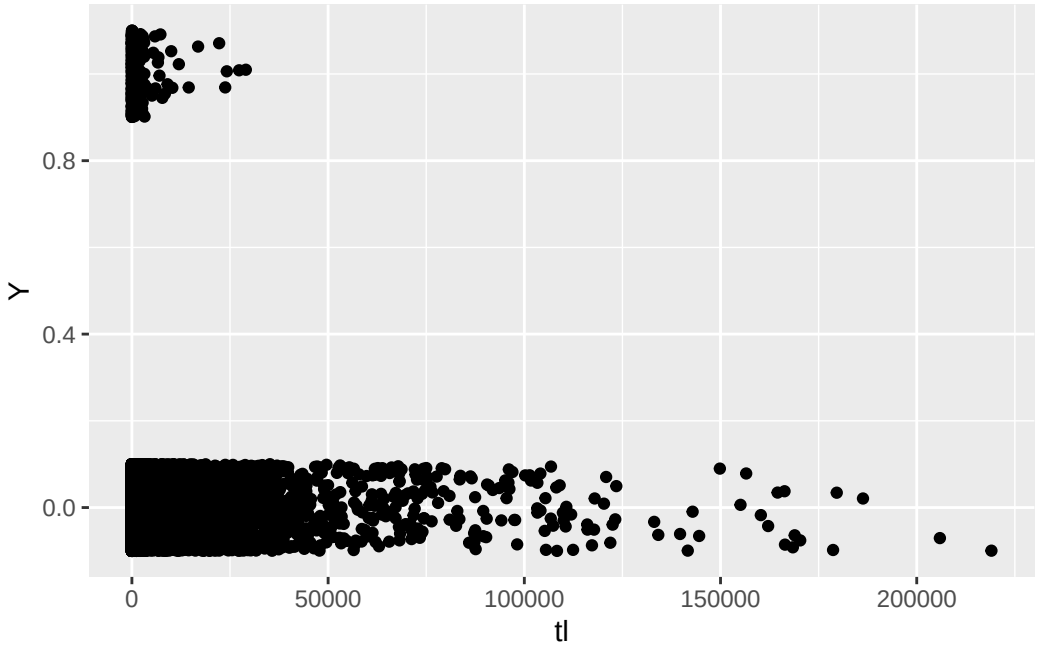
```
                .names = "{.col}_ws")) %>%
  # scaling by `ta` and winsorizing
  mutate(across(.cols = cogs:trec,
                .fns = ~ DescTools::Winsorize(. / ta, probs = c(0.05, 0.95)),
                .names = "{.col}_/_ta_ws"))
```

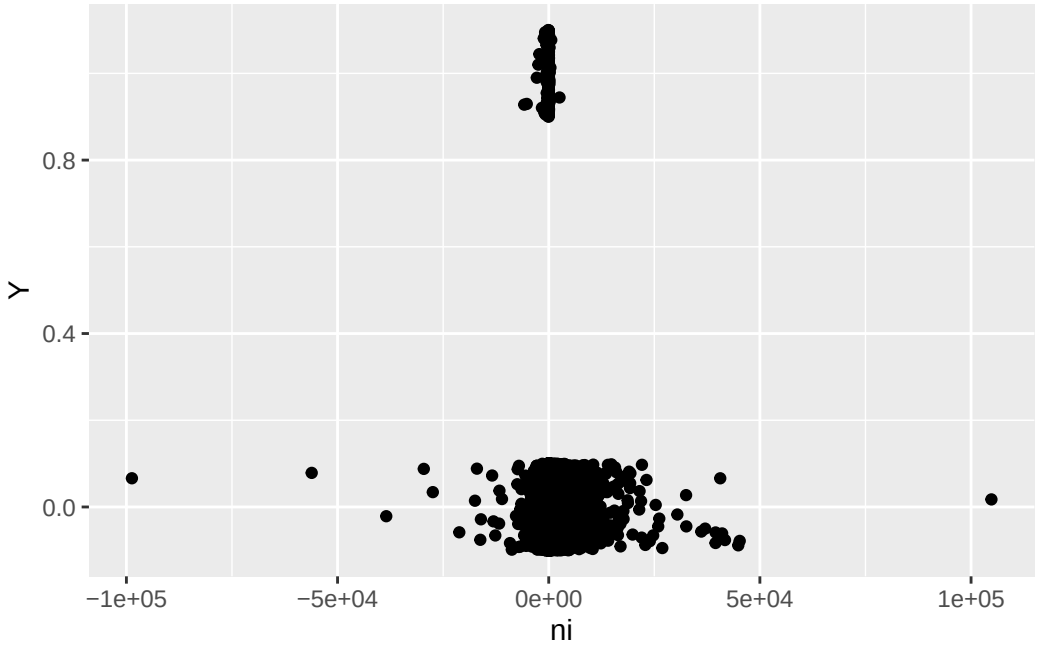
We first plot default variable with respect to unscaled predictors:

```
vars_quanti <- names(default_exam %>%
                       select_if(is.numeric) %>%
                       select(-Y_alt, -Y, -company_id, -default,
                              -default_will_occur, -fiscal_year) %>%
                       select(!matches("_/_ta|_ws")))
```

```
# for report only
vars_quanti <- c("ta", "tl", "ni")

for(var in vars_quanti){
  var <- as.name(var)
  print(ggplot(default_exam, aes(x=!! var,y = Y)) +
          geom_jitter(height = 0.1, width = 0))
}
```





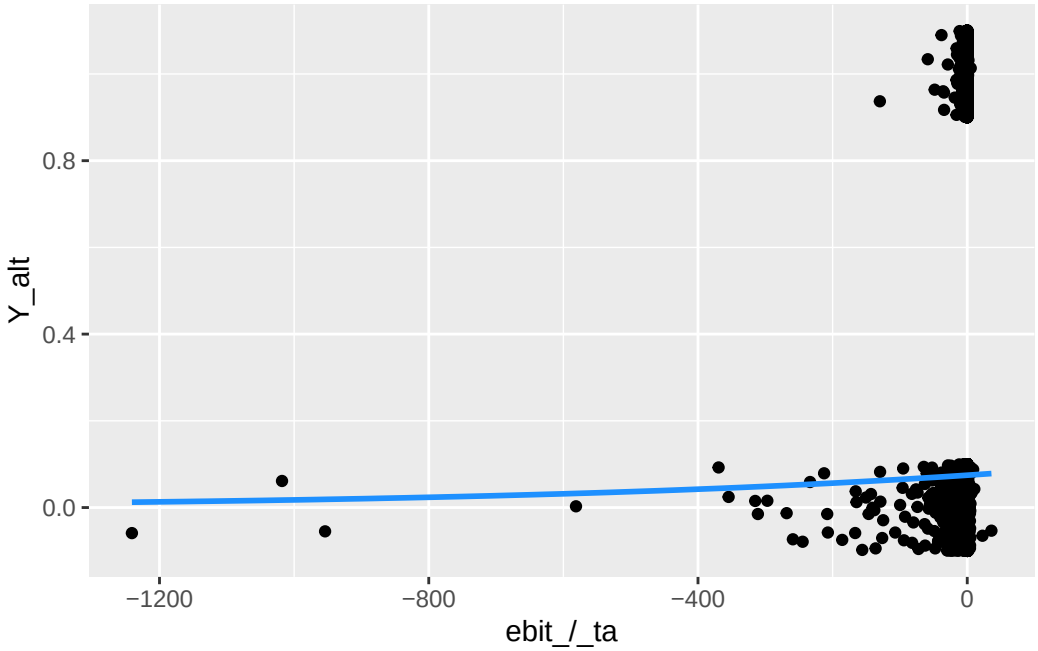


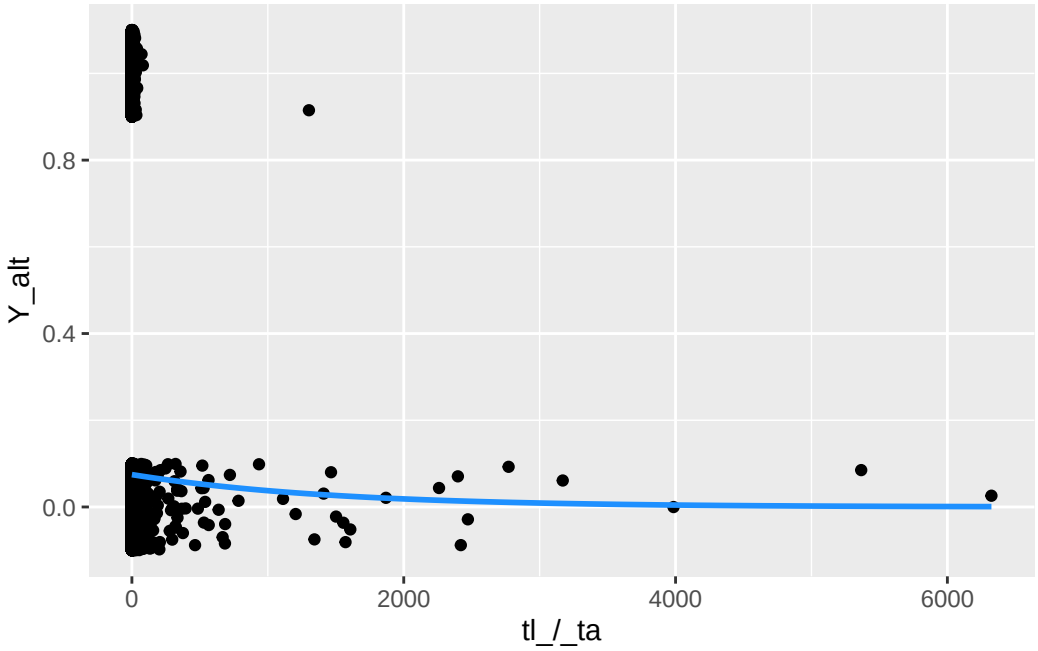
It is difficult to conclude anyting from these plots as financial accounts do not have the same scale (cmpanies total assets/liabilities vary a lot).

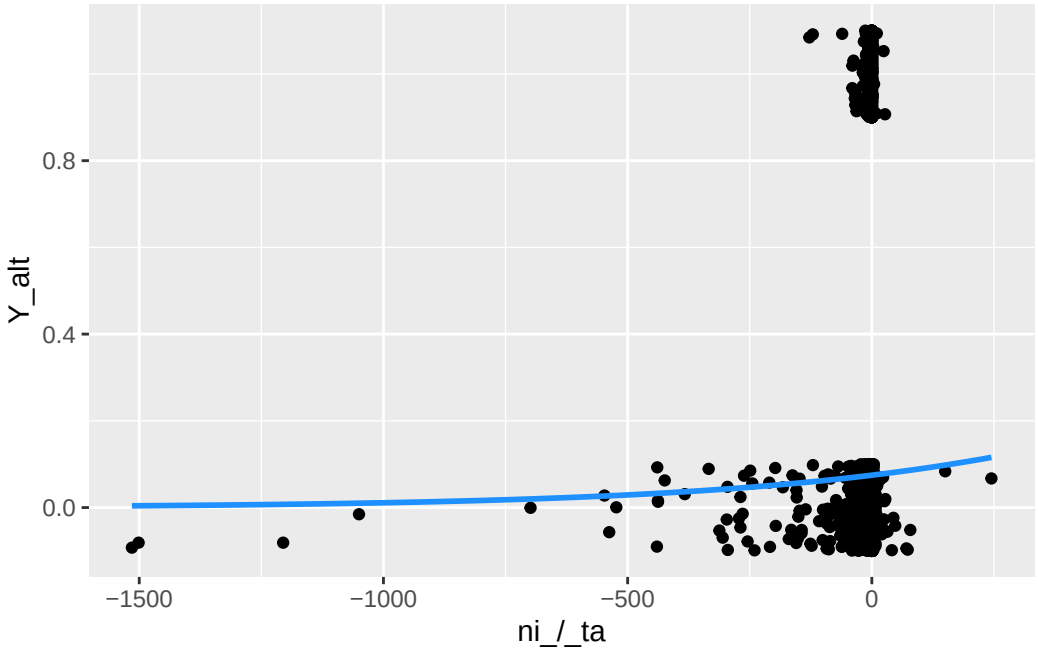
We plot default variable with respect to predictors rescaled using total assets (`ta`):

```
vars_quanti <- names(default_exam %>%
                       select_if(is.numeric) %>%
                       select(-Y_alt, -Y, -company_id, -default,
                              -default_will_occur, -fiscal_year) %>%
                       select(matches("_/_ta")) %>%
                       select(!matches("_/_ta_ws")) %>%
                       select(!matches("_ws")))
```

```
# for report only
vars_quanti <- c("ebit_/_ta", "tl_/_ta", "ni_/_ta")

for(var in vars_quanti){
  var <- as.name(var)
  print(ggplot(default_exam, aes(x=!! var, y = Y_alt)) +
          geom_jitter(height = 0.1, width = 0) +
          geom_smooth(method = "glm",
                      formula = y ~ x,
                      method.args = list(family = "binomial"),
                      se = FALSE,
                      col = "dodgerblue"))
}
```


0.8
0.4
0.0
Y_alt
−1200
−800
−400
0
ebit_/_ta


0.8
0.4
0.0
Y_alt
0
2000
4000
6000
tl_/_ta



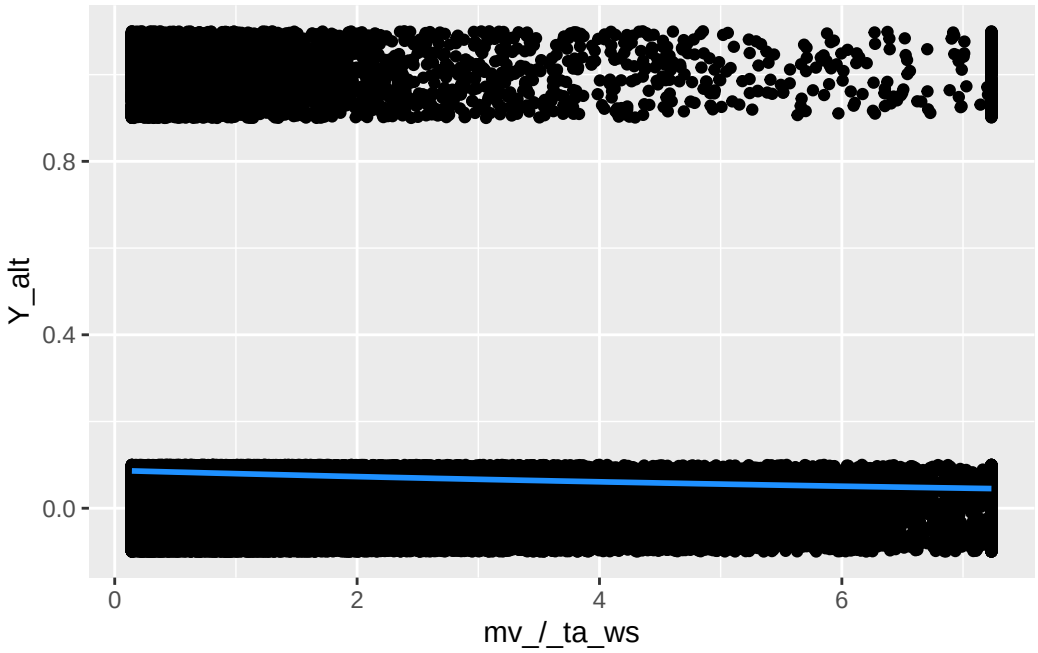
Outliers appear on these plots, in particular some companies (ca. 200) exhibit strongly negative ebit and/or near zero total assets without being classified as defaulting (most of them are nonetheless removed form the dat set, i.e. they do not live up to the maximum date). Since the companies are anonymized and we cannot investigate further we remove these outliers from the data set:

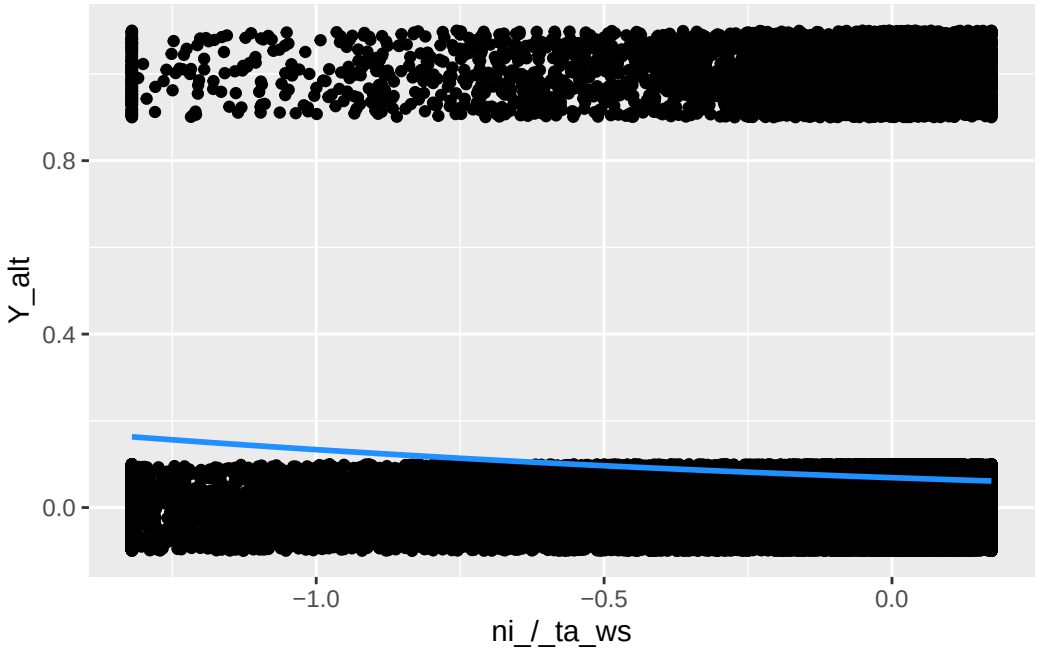
```
outliers_ebit <- default_exam %>%
  filter(`ebit_/_ta` < -10, default_will_occur == 0) %>%
  distinct(company_id) %>%
  pull(company_id)
```

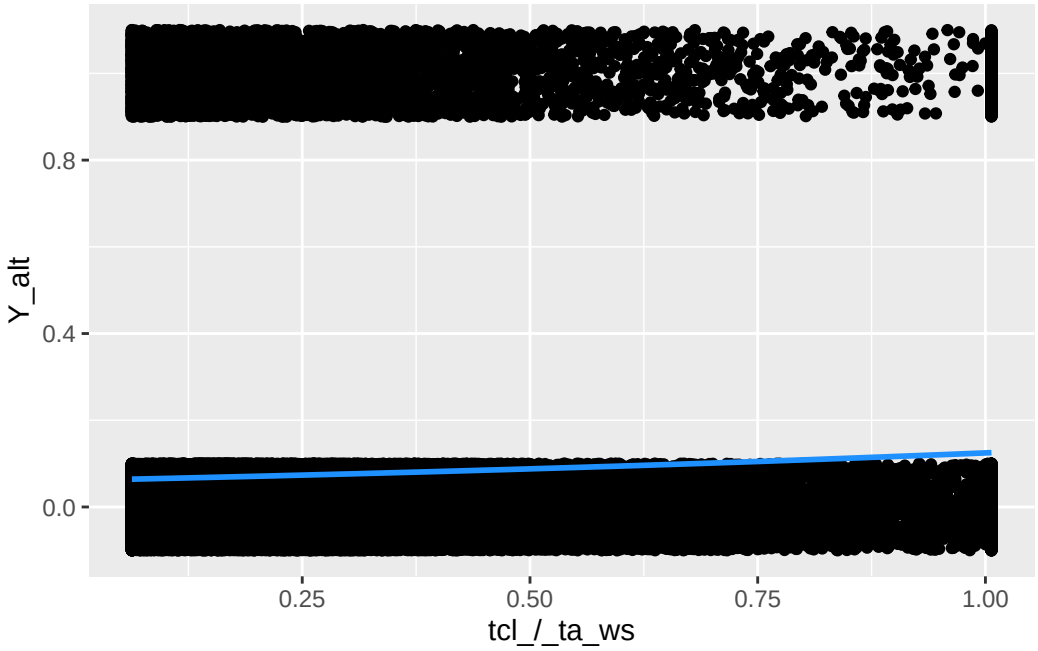
```
vars_quanti <- names(default_exam %>%
                       select_if(is.numeric) %>%
                       select(-Y_alt, -Y, -company_id, -default,
                              -default_will_occur, -fiscal_year) %>%
                       select(matches("_/_ta_ws")))
```

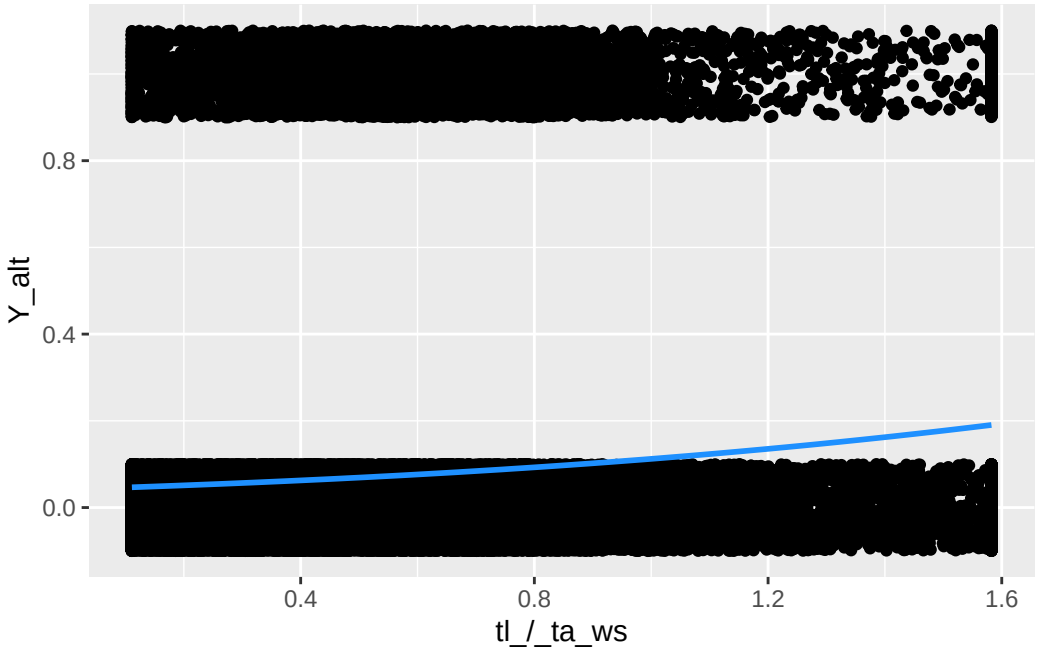
```
# for report only
vars_quanti <- c("mv_/_ta_ws", "ni_/_ta_ws", "tcl_/_ta_ws", "tl_/_ta_ws")

for(var in vars_quanti){
```

```
  var <- as.name(var)
  print(ggplot(default_exam %>% filter(!company_id %in% outliers_ebit),
               aes(x=!! var, y = Y_alt)) +
          geom_jitter(height = 0.1, width = 0) +
          geom_smooth(method = "glm",
                      formula = y ~ x,
                      method.args = list(family = "binomial"),
                      se = FALSE,
                      col = "dodgerblue"))
}
```




0.8
0.4
0.0
Y_alt
−1.0
−0.5
0.0
ni_/_ta_ws


0.8
0.4
0.0
Y_alt
0.25
0.50
0.75
1.00
tcl_/_ta_ws



4 Model building and assessment

4.1 First step (10% total points)

4.1.1 TO DO in-class (~45minutes-1h) with second priority

In a **first step**, using (i) the methods from the course, (ii) only the feature variables provided in the data set (`cogs:trec`) and (iii) using the full data set provided (for this first step do not use hold-out/cross-validation approaches), construct a baseline Logistic Regression model that you "believe" is "valid":

- For this use a method of your choice seen in course, either:
  - manual selection using hints from data exploration and statistical tests,
  - or automatic best subset (`bestglm`) / stepwise selection (`step`) with criterion of your choice,
  - or penalized lasso regression (`glmnet`/`glmnetUtils`), the predictors might need to be standardized if they have a different nature/scale, (for this, use the option `standardize=TRUE`, fitted coefficient will be automatically scaled back to the original scale of the data).

- For the selected model (**your_model**):
  - show how to interpret model coefficients for at least two variables (odds ratio, tests, confidence intervals)
  - compare using a statistical test of your choice to a full model (**full_model** using all the variables you retained in the Data Exploration part)
  - if you "fail" to construct a model that you "believe" is valid, choose as a best effort a model including a few variables (at least 4).

4.1.2 Proposed solution

We first fit the full model with all variables involved:

- using unscaled financial variables

```
model_full <- glm(Y~., data=default_exam %>%
                    filter(!company_id %in% outliers_ebit) %>%
                    select(Y, `cogs_/_ta`:`trec_/_ta`, -`ta_/_ta`), family="binomial")
summary(model_full)
```

```
Call:
glm(formula = Y ~ ., family = "binomial", data = default_exam %>%
    filter(!company_id %in% outliers_ebit) %>% select(Y, `cogs_/_ta`:`trec_/_ta`,
    -`ta_/_ta`))

Coefficients:
                       Estimate Std. Error z value Pr(>|z|)
(Intercept)          -4.355e+00  1.040e-01 -41.873  < 2e-16 ***
`cogs_/_ta`           4.738e+03  7.662e+04   0.062    0.951
`current_asset_/_ta` -1.479e+00  2.576e-01  -5.742 9.35e-09 ***
`da_/_ta`             4.822e+03  7.629e+04   0.063    0.950
`ebit_/_ta`           4.822e+03  7.629e+04   0.063    0.950
`gp_/_ta`             4.738e+03  7.662e+04   0.062    0.951
`invt_/_ta`           2.415e+00  3.568e-01   6.768 1.31e-11 ***
`mv_/_ta`            -1.286e-01  2.167e-02  -5.935 2.94e-09 ***
`ni_/_ta`            -1.995e-02  1.465e-02  -1.362    0.173
`nsale_/_ta`         -9.560e+03  1.527e+05  -0.063    0.950
`opex_/_ta`           4.822e+03  7.629e+04   0.063    0.950
`re_/_ta`            -6.454e-03  1.274e-03  -5.064 4.10e-07 ***
`tcl_/_ta`            2.362e-02  1.447e-02   1.632    0.103
`tl_/_ta`            -2.738e-02  1.856e-02  -1.476    0.140
```

```
`tltd_/_ta`          -3.072e-02  5.032e-02  -0.611    0.542
`trec_/_ta`           4.037e-01  4.682e-01   0.862    0.389
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 5654.9  on 59765  degrees of freedom
Residual deviance: 5474.5  on 59750  degrees of freedom
AIC: 5506.5

Number of Fisher Scoring iterations: 13
```

- using scaled (`ta`) and winsorized financial variables:

```
model_full <- glm(Y~., data=default_exam %>%
                    filter(!company_id %in% outliers_ebit) %>%
                    select(Y, `cogs_/_ta_ws`:`trec_/_ta_ws`, -`ta_/_ta_ws`),
                  family="binomial")
summary(model_full)
```

```
Call:
glm(formula = Y ~ ., family = "binomial", data = default_exam %>%
    filter(!company_id %in% outliers_ebit) %>% select(Y, `cogs_/_ta_ws`:`trec_/_ta_ws`,
    -`ta_/_ta_ws`))

Coefficients:
                       Estimate Std. Error z value Pr(>|z|)
(Intercept)            -5.52337    0.18182 -30.379  < 2e-16 ***
`cogs_/_ta_ws`          0.09517    0.17651   0.539 0.589757
`current_asset_/_ta_ws` -0.61344    0.27686  -2.216 0.026712 *
`da_/_ta_ws`            0.21248    1.44201   0.147 0.882855
`ebit_/_ta_ws`         -0.29101    0.31938  -0.911 0.362201
`gp_/_ta_ws`            0.10555    0.28778   0.367 0.713794
`invt_/_ta_ws`          1.34941    0.43616   3.094 0.001976 **
`mv_/_ta_ws`           -0.57389    0.04917 -11.672  < 2e-16 ***
`ni_/_ta_ws`           -0.61617    0.20706  -2.976 0.002922 **
`nsale_/_ta_ws`        -0.39699    0.19045  -2.085 0.037111 *
`opex_/_ta_ws`          0.39587    0.15017   2.636 0.008386 **
`re_/_ta_ws`            0.03856    0.01834   2.102 0.035563 *
`tcl_/_ta_ws`           1.18070    0.30152   3.916 9.01e-05 ***
`tl_/_ta_ws`            1.47822    0.21763   6.792 1.10e-11 ***
```

```
`tltd_/_ta_ws`          -0.02260    0.28617  -0.079 0.937056
`trec_/_ta_ws`          -1.77709    0.52941  -3.357 0.000789 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 5654.9  on 59765  degrees of freedom
Residual deviance: 4808.8  on 59750  degrees of freedom
AIC: 4840.8

Number of Fisher Scoring iterations: 9
```

```r
model_full_alt <- glm(Y_alt~., data=default_exam %>%
                      filter(!company_id %in% outliers_ebit) %>%
                      select(Y_alt, `cogs_/_ta_ws`:`trec_/_ta_ws`, -`ta_/_ta_ws`),
                      family="binomial")
summary(model_full_alt)
```

```
Call:
glm(formula = Y_alt ~ ., family = "binomial", data = default_exam %>%
    filter(!company_id %in% outliers_ebit) %>% select(Y_alt,
    `cogs_/_ta_ws`:`trec_/_ta_ws`, -`ta_/_ta_ws`))

Coefficients:
                        Estimate Std. Error z value Pr(>|z|)
(Intercept)            -2.769722   0.061466 -45.061  < 2e-16 ***
`cogs_/_ta_ws`         -0.070232   0.076887  -0.913 0.361010
`current_asset_/_ta_ws` -0.327037   0.090486  -3.614 0.000301 ***
`da_/_ta_ws`            3.171041   0.539271   5.880 4.10e-09 ***
`ebit_/_ta_ws`         -0.864918   0.146262  -5.913 3.35e-09 ***
`gp_/_ta_ws`           -0.465351   0.118132  -3.939 8.17e-05 ***
`invt_/_ta_ws`          0.806648   0.159406   5.060 4.19e-07 ***
`mv_/_ta_ws`           -0.115249   0.011562  -9.968  < 2e-16 ***
`ni_/_ta_ws`           -0.024189   0.099757  -0.242 0.808408
`nsale_/_ta_ws`         0.230260   0.083945   2.743 0.006088 **
`opex_/_ta_ws`          0.030587   0.066363   0.461 0.644864
`re_/_ta_ws`            0.004962   0.007449   0.666 0.505307
`tcl_/_ta_ws`           0.129001   0.142750   0.904 0.366165
`tl_/_ta_ws`            0.333587   0.100396   3.323 0.000891 ***
`tltd_/_ta_ws`          1.148081   0.132095   8.691  < 2e-16 ***
```

```
`trec_/_ta_ws`          -1.764874   0.192291  -9.178  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 32195  on 59765  degrees of freedom
Residual deviance: 30718  on 59750  degrees of freedom
AIC: 30750

Number of Fisher Scoring iterations: 5
```

```
car::Anova(model_full, type=3, test.statistic = "Wald")
```

```
Analysis of Deviance Table (Type III tests)

Response: Y
                       Df    Chisq Pr(>Chisq)
(Intercept)             1 922.8581  < 2.2e-16 ***
`cogs_/_ta_ws`          1   0.2907  0.5897569
`current_asset_/_ta_ws` 1   4.9093  0.0267121 *
`da_/_ta_ws`            1   0.0217  0.8828553
`ebit_/_ta_ws`          1   0.8302  0.3622008
`gp_/_ta_ws`            1   0.1345  0.7137942
`invt_/_ta_ws`          1   9.5720  0.0019757 **
`mv_/_ta_ws`            1 136.2371  < 2.2e-16 ***
`ni_/_ta_ws`            1   8.8554  0.0029223 **
`nsale_/_ta_ws`         1   4.3453  0.0371112 *
`opex_/_ta_ws`          1   6.9492  0.0083859 **
`re_/_ta_ws`            1   4.4180  0.0355627 *
`tcl_/_ta_ws`           1  15.3336  9.010e-05 ***
`tl_/_ta_ws`            1  46.1345  1.104e-11 ***
`tltd_/_ta_ws`          1   0.0062  0.9370557
`trec_/_ta_ws`          1  11.2676  0.0007887 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
car::Anova(model_full, type=3, test.statistic = "LR")
```

```
Analysis of Deviance Table (Type III tests)
```

```
Response: Y
                       LR Chisq Df Pr(>Chisq)
`cogs_/_ta_ws`            0.290  1  0.5903294
`current_asset_/_ta_ws`   5.074  1  0.0242892 *
`da_/_ta_ws`              0.022  1  0.8828848
`ebit_/_ta_ws`            0.829  1  0.3625239
`gp_/_ta_ws`              0.135  1  0.7136342
`invt_/_ta_ws`            9.612  1  0.0019332 **
`mv_/_ta_ws`            227.707  1  < 2.2e-16 ***
`ni_/_ta_ws`              8.798  1  0.0030157 **
`nsale_/_ta_ws`           4.264  1  0.0389328 *
`opex_/_ta_ws`            6.621  1  0.0100762 *
`re_/_ta_ws`              4.532  1  0.0332606 *
`tcl_/_ta_ws`            15.984  1  6.390e-05 ***
`tl_/_ta_ws`             40.562  1  1.905e-10 ***
`tltd_/_ta_ws`            0.006  1  0.9370648
`trec_/_ta_ws`           11.181  1  0.0008266 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
# manual_selection <- "Y~ `current_asset_/_ta_ws` + `invt_/_ta_ws` +
#                         `mv_/_ta_ws` + `ni_/_ta_ws` + `nsale_/_ta_ws` +
#                         `opex_/_ta_ws` + `re_/_ta_ws`  + `tcl_/_ta_ws` +
#                         `tl_/_ta_ws`  + `trec_/_ta_ws`"

manual_selection <- "Y~ `current_asset_/_ta_ws` + `invt_/_ta_ws` + `mv_/_ta_ws` + `ni_/_ta

model_manual <- glm(as.formula(manual_selection), data=default_exam %>%
                    filter(!company_id %in% outliers_ebit) %>%
                    select(Y, `cogs_/_ta_ws`:`trec_/_ta_ws`, -`ta_/_ta_ws`), family="binom
summary(model_manual)
```

```
Call:
glm(formula = as.formula(manual_selection), family = "binomial",
    data = default_exam %>% filter(!company_id %in% outliers_ebit) %>%
        select(Y, `cogs_/_ta_ws`:`trec_/_ta_ws`, -`ta_/_ta_ws`))

Coefficients:
                       Estimate Std. Error z value Pr(>|z|)
(Intercept)            -5.50416    0.15663 -35.142  < 2e-16 ***
```

```
`current_asset_/_ta_ws` -0.59540    0.26382  -2.257 0.024018 *
`invt_/_ta_ws`           1.31565    0.43299   3.039 0.002377 **
`mv_/_ta_ws`            -0.57473    0.04883 -11.771  < 2e-16 ***
`ni_/_ta_ws`            -0.73999    0.15702  -4.713 2.44e-06 ***
`nsale_/_ta_ws`         -0.39231    0.12379  -3.169 0.001529 **
`opex_/_ta_ws`           0.48239    0.11792   4.091 4.30e-05 ***
`re_/_ta_ws`             0.03534    0.01748   2.021 0.043260 *
`tcl_/_ta_ws`            1.20682    0.24267   4.973 6.59e-07 ***
`tl_/_ta_ws`             1.45107    0.17062   8.505  < 2e-16 ***
`trec_/_ta_ws`          -1.81961    0.52618  -3.458 0.000544 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 5654.9  on 59765  degrees of freedom
Residual deviance: 4809.9  on 59755  degrees of freedom
AIC: 4831.9

Number of Fisher Scoring iterations: 9
```

We perform stepwise variable selection:

```r
model_full <- glm(Y~., data=default_exam %>%
                    filter(!company_id %in% outliers_ebit) %>%
                    select(Y, `cogs_/_ta_ws`:`trec_/_ta_ws`, -`ta_/_ta_ws`), family="binom

intercept_only <- glm(Y~1, data=default_exam %>%
                    filter(!company_id %in% outliers_ebit) %>%
                    select(Y, `cogs_/_ta_ws`:`trec_/_ta_ws`, -`ta_/_ta_ws`), family="binom
```

```r
forward_aic <- step(intercept_only,
                    direction='forward',
                    test = 'LRT',
                    scope=formula(model_full), k=2,
                    trace = FALSE)
summary(forward_aic)
```

```
Call:
glm(formula = Y ~ `tl_/_ta_ws` + `mv_/_ta_ws` + `ni_/_ta_ws` +
    `tcl_/_ta_ws` + `trec_/_ta_ws` + `invt_/_ta_ws` + `current_asset_/_ta_ws` +
```

```
    `opex_/_ta_ws` + `nsale_/_ta_ws` + `re_/_ta_ws`, family = "binomial",
    data = default_exam %>% filter(!company_id %in% outliers_ebit) %>%
        select(Y, `cogs_/_ta_ws`:`trec_/_ta_ws`, -`ta_/_ta_ws`))

Coefficients:
                       Estimate Std. Error z value Pr(>|z|)
(Intercept)            -5.50416    0.15663 -35.142  < 2e-16 ***
`tl_/_ta_ws`            1.45107    0.17062   8.505  < 2e-16 ***
`mv_/_ta_ws`           -0.57473    0.04883 -11.771  < 2e-16 ***
`ni_/_ta_ws`           -0.73999    0.15702  -4.713 2.44e-06 ***
`tcl_/_ta_ws`           1.20682    0.24267   4.973 6.59e-07 ***
`trec_/_ta_ws`         -1.81961    0.52618  -3.458 0.000544 ***
`invt_/_ta_ws`          1.31565    0.43299   3.039 0.002377 **
`current_asset_/_ta_ws` -0.59540    0.26382  -2.257 0.024018 *
`opex_/_ta_ws`          0.48239    0.11792   4.091 4.30e-05 ***
`nsale_/_ta_ws`        -0.39231    0.12379  -3.169 0.001529 **
`re_/_ta_ws`            0.03534    0.01748   2.021 0.043260 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 5654.9  on 59765  degrees of freedom
Residual deviance: 4809.9  on 59755  degrees of freedom
AIC: 4831.9

Number of Fisher Scoring iterations: 9
```

```
forward_bic <- step(intercept_only,
                    direction='forward',
                    test = 'LRT',
                    scope=formula(model_full),
                    k=log(nrow(default_exam %>%
                      filter(!company_id %in% outliers_ebit))),
                    trace = FALSE)
summary(forward_bic)
```

```
Call:
glm(formula = Y ~ `tl_/_ta_ws` + `mv_/_ta_ws` + `ni_/_ta_ws` +
    `tcl_/_ta_ws` + `trec_/_ta_ws`, family = "binomial", data = default_exam %>%
    filter(!company_id %in% outliers_ebit) %>% select(Y, `cogs_/_ta_ws`:`trec_/_ta_ws`,
```

```
    -`ta_/_ta_ws`))

Coefficients:
               Estimate Std. Error z value Pr(>|z|)
(Intercept)    -5.51516    0.12614 -43.722  < 2e-16 ***
`tl_/_ta_ws`    1.39110    0.15689   8.867  < 2e-16 ***
`mv_/_ta_ws`   -0.56653    0.04657 -12.165  < 2e-16 ***
`ni_/_ta_ws`   -0.84524    0.12022  -7.031 2.06e-12 ***
`tcl_/_ta_ws`   1.31383    0.22393   5.867 4.43e-09 ***
`trec_/_ta_ws` -2.12094    0.42905  -4.943 7.68e-07 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 5654.9  on 59765  degrees of freedom
Residual deviance: 4843.8  on 59760  degrees of freedom
AIC: 4855.8

Number of Fisher Scoring iterations: 9
```

```r
backward_bic <- step(model_full,
                     direction='backward',
                     test = 'LRT',
                     scope=formula(model_full),
                     k=log(nrow(default_exam %>%
                       filter(!company_id %in% outliers_ebit))),
                     trace = FALSE)
summary(backward_bic)
```

```
Call:
glm(formula = Y ~ `mv_/_ta_ws` + `ni_/_ta_ws` + `tcl_/_ta_ws` +
    `tl_/_ta_ws` + `trec_/_ta_ws`, family = "binomial", data = default_exam %>%
    filter(!company_id %in% outliers_ebit) %>% select(Y, `cogs_/_ta_ws`:`trec_/_ta_ws`,
    -`ta_/_ta_ws`))

Coefficients:
               Estimate Std. Error z value Pr(>|z|)
(Intercept)    -5.51516    0.12614 -43.722  < 2e-16 ***
`mv_/_ta_ws`   -0.56653    0.04657 -12.165  < 2e-16 ***
`ni_/_ta_ws`   -0.84524    0.12022  -7.031 2.06e-12 ***
```

```
`tcl_/_ta_ws`   1.31383    0.22393   5.867 4.43e-09 ***
`tl_/_ta_ws`    1.39110    0.15689   8.867  < 2e-16 ***
`trec_/_ta_ws` -2.12094    0.42905  -4.943 7.68e-07 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 5654.9  on 59765  degrees of freedom
Residual deviance: 4843.8  on 59760  degrees of freedom
AIC: 4855.8

Number of Fisher Scoring iterations: 9
```

Forward and Backward stepwise selection using the BIC criterion using the full dataset agree on the choice of variables.

We perform penalized Logistic Regression (Lasso):

```
default_lasso <- glmnetUtils::cv.glmnet(Y ~ .,
                     data=default_exam %>%
                       filter(!company_id %in% outliers_ebit) %>%
                       select(Y, `cogs_/_ta_ws`:`trec_/_ta_ws`, -`ta_/_ta_ws`),
                     family="binomial",
                     alpha=1,
                     standardize = TRUE,
                     type.measure = "auc")
```

```
lasso_coeffs_1se <- coef(default_lasso, s = "lambda.1se")
tibble(name = lasso_coeffs_1se@Dimnames[[1]][lasso_coeffs_1se@i + 1],
       coefficient = lasso_coeffs_1se@x)
```

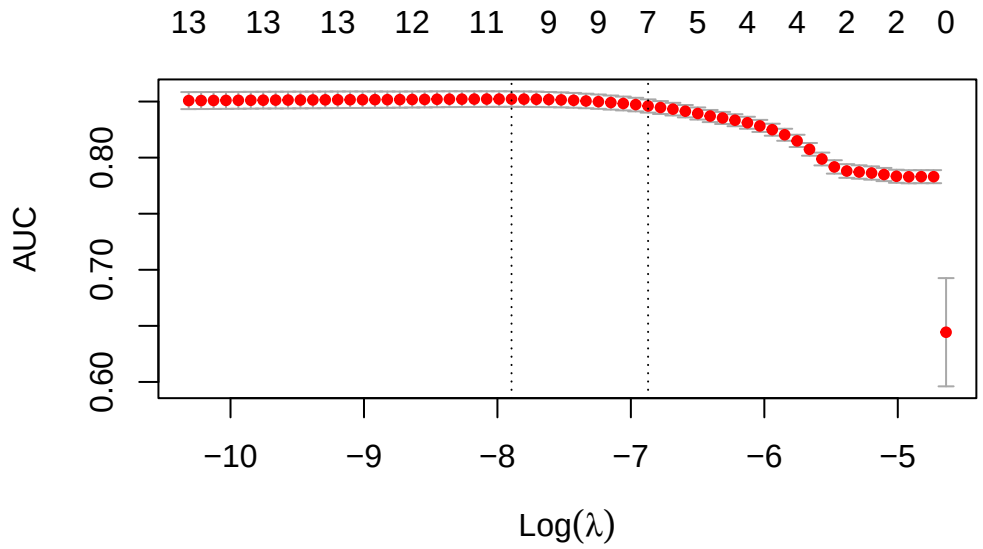
```
# A tibble: 8 x 2
  name         coefficient
  <chr>              <dbl>
1 (Intercept)      -5.72
2 ebit_/_ta_ws     -0.0871
3 invt_/_ta_ws      0.157
4 mv_/_ta_ws       -0.368
5 ni_/_ta_ws       -0.628
6 tcl_/_ta_ws       0.981
7 tl_/_ta_ws        1.42
```

```
8 trec_/_ta_ws     -0.732
```

```
lasso_coeffs_min <- coef(default_lasso, s = "lambda.min")
tibble(name = lasso_coeffs_min@Dimnames[[1]][lasso_coeffs_min@i + 1],
       coefficient = lasso_coeffs_min@x)
```

```
# A tibble: 12 x 2
   name                   coefficient
   <chr>                        <dbl>
 1 (Intercept)                -5.58
 2 current_asset_/_ta_ws      -0.361
 3 ebit_/_ta_ws               -0.298
 4 gp_/_ta_ws                 -0.0346
 5 invt_/_ta_ws                0.927
 6 mv_/_ta_ws                 -0.484
 7 ni_/_ta_ws                 -0.609
 8 opex_/_ta_ws                0.0829
 9 re_/_ta_ws                  0.00114
10 tcl_/_ta_ws                 1.08
11 tl_/_ta_ws                  1.40
12 trec_/_ta_ws               -1.53
```

```
plot(default_lasso)
```



We briefly show how to interpret model coefficients (odds ratio, tests, confidence intervals):

We choose the variables `ni_/_ta_ws` and `ti_/_ta_ws` in the manual model

In terms of odds ratio:

A 1 unit increase in `ni_/_ta_ws` decreases the odds of default by a factor of 0.48.

A 1 unit increase in `tl_/_ta_ws` increases the odds of default by a factor of 4.27.

```
exp(cbind(odds_ratio = coef(model_manual), confint.default(model_manual)))
```

```
                        odds_ratio       2.5 %      97.5 %
(Intercept)             0.00406982 0.002994025 0.005532163
`current_asset_/_ta_ws` 0.55134039 0.328741216 0.924667221
`invt_/_ta_ws`          3.72718889 1.595213911 8.708510461
`mv_/_ta_ws`            0.56285827 0.511490519 0.619384760
`ni_/_ta_ws`            0.47711711 0.350726984 0.649053960
`nsale_/_ta_ws`         0.67549616 0.529970234 0.860982446
`opex_/_ta_ws`          1.61994670 1.285667541 2.041139886
`re_/_ta_ws`            1.03596904 1.001071023 1.072083624
`tcl_/_ta_ws`           3.34283262 2.077542013 5.378726322
`tl_/_ta_ws`            4.26765850 3.054630976 5.962392587
`trec_/_ta_ws`          0.16208839 0.057792366 0.454604121
```

We show the Wald confidence interval:

```
confint.default(model_manual)
```

```
                              2.5 %      97.5 %
(Intercept)             -5.811136548 -5.19717644
`current_asset_/_ta_ws` -1.112484415 -0.07832137
`invt_/_ta_ws`           0.467007841  2.16430076
`mv_/_ta_ws`            -0.670426229 -0.47902862
`ni_/_ta_ws`            -1.047747180 -0.43223942
`nsale_/_ta_ws`         -0.634934436 -0.14968116
`opex_/_ta_ws`           0.251278071  0.71350842
`re_/_ta_ws`             0.001070449  0.06960407
`tcl_/_ta_ws`            0.731185470  1.68245160
`tl_/_ta_ws`             1.116658792  1.78547184
`trec_/_ta_ws`          -2.850898589 -0.78832830
```

And the profiled confidence interval:

```
confint(model_manual)
```

```
                              2.5 %      97.5 %
(Intercept)             -5.813477852 -5.19940203
`current_asset_/_ta_ws` -1.122170860 -0.08724336
`invt_/_ta_ws`           0.468235691  2.16664422
`mv_/_ta_ws`            -0.674573425 -0.48286486
`ni_/_ta_ws`            -1.046148344 -0.43071839
`nsale_/_ta_ws`         -0.634133871 -0.14830586
`opex_/_ta_ws`           0.248520330  0.71127938
`re_/_ta_ws`             0.001528179  0.07008587
`tcl_/_ta_ws`            0.732062528  1.68360435
`tl_/_ta_ws`             1.113442046  1.78241715
`trec_/_ta_ws`          -2.849883037 -0.78630104
```

We perform likelihood ratio test for **ni_/_ta_ws** and **tl_/_ta_ws** (vs manual_model) and conclude that the two variables are statistically significant within the manual model:

```
# manual_selection_wo_ni <- "Y~ `current_asset_/_ta_ws` + `invt_/_ta_ws` +
                              # `mv_/_ta_ws`  + `nsale_/_ta_ws` + `opex_/_ta_ws` +
                              # `re_/_ta_ws`  + `tcl_/_ta_ws` + `tl_/_ta_ws`  + `trec_/_
```

```
# manual_selection_wo_tl <- "Y~ `current_asset_/_ta_ws` + `invt_/_ta_ws` +
                                # `mv_/_ta_ws` + `ni_/_ta_ws` + `nsale_/_ta_ws` +
                                # `opex_/_ta_ws` + `re_/_ta_ws`  + `tcl_/_ta_ws`  + `trec


manual_selection_wo_ni <- "Y~ `current_asset_/_ta_ws` + `invt_/_ta_ws` + `mv_/_ta_ws`  + `

manual_selection_wo_tl <- "Y~ `current_asset_/_ta_ws` + `invt_/_ta_ws` + `mv_/_ta_ws` + `n

model_manual_wo_ni  <- glm(as.formula(manual_selection_wo_ni), data=default_exam %>%
                    filter(!company_id %in% outliers_ebit) %>%
                    select(Y, `cogs_/_ta_ws`:`trec_/_ta_ws`, -`ta_/_ta_ws`), family="binom

model_manual_wo_tl  <- glm(as.formula(manual_selection_wo_tl), data=default_exam %>%
                    filter(!company_id %in% outliers_ebit) %>%
                    select(Y, `cogs_/_ta_ws`:`trec_/_ta_ws`, -`ta_/_ta_ws`), family="binom
```

Testing variable `ni_/_ta_ws`:

```
anova(model_manual_wo_ni, model_manual, test ="LRT")
```

```
Analysis of Deviance Table

Model 1: Y ~ `current_asset_/_ta_ws` + `invt_/_ta_ws` + `mv_/_ta_ws` +
    `nsale_/_ta_ws` + `opex_/_ta_ws` + `re_/_ta_ws` + `tcl_/_ta_ws` +
    `tl_/_ta_ws` + `trec_/_ta_ws`
Model 2: Y ~ `current_asset_/_ta_ws` + `invt_/_ta_ws` + `mv_/_ta_ws` +
    `ni_/_ta_ws` + `nsale_/_ta_ws` + `opex_/_ta_ws` + `re_/_ta_ws` +
    `tcl_/_ta_ws` + `tl_/_ta_ws` + `trec_/_ta_ws`
  Resid. Df Resid. Dev Df Deviance  Pr(>Chi)
1     59756     4831.6
2     59755     4809.9  1   21.706 3.179e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Testing variable `tl_/_ta_ws`:

```
anova(model_manual_wo_tl, model_manual, test ="LRT")
```

```
Analysis of Deviance Table
```

```
Model 1: Y ~ `current_asset_/_ta_ws` + `invt_/_ta_ws` + `mv_/_ta_ws` +
    `ni_/_ta_ws` + `nsale_/_ta_ws` + `opex_/_ta_ws` + `re_/_ta_ws` +
    `tcl_/_ta_ws` + `trec_/_ta_ws`
Model 2: Y ~ `current_asset_/_ta_ws` + `invt_/_ta_ws` + `mv_/_ta_ws` +
    `ni_/_ta_ws` + `nsale_/_ta_ws` + `opex_/_ta_ws` + `re_/_ta_ws` +
    `tcl_/_ta_ws` + `tl_/_ta_ws` + `trec_/_ta_ws`
  Resid. Df Resid. Dev Df Deviance  Pr(>Chi)
1     59756     4876.5
2     59755     4809.9  1   66.558 3.397e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Inference for lasso is not straightforward see for example these references here, here and here for a R package

We compare using the likelihood ratio test a full model (**full_model** using all the variables you retained in the Data Exploration part)

```
anova(model_manual, model_full, test = "LRT")
```

```
Analysis of Deviance Table

Model 1: Y ~ `current_asset_/_ta_ws` + `invt_/_ta_ws` + `mv_/_ta_ws` +
    `ni_/_ta_ws` + `nsale_/_ta_ws` + `opex_/_ta_ws` + `re_/_ta_ws` +
    `tcl_/_ta_ws` + `tl_/_ta_ws` + `trec_/_ta_ws`
Model 2: Y ~ `cogs_/_ta_ws` + `current_asset_/_ta_ws` + `da_/_ta_ws` +
    `ebit_/_ta_ws` + `gp_/_ta_ws` + `invt_/_ta_ws` + `mv_/_ta_ws` +
    `ni_/_ta_ws` + `nsale_/_ta_ws` + `opex_/_ta_ws` + `re_/_ta_ws` +
    `tcl_/_ta_ws` + `tl_/_ta_ws` + `tltd_/_ta_ws` + `trec_/_ta_ws`
  Resid. Df Resid. Dev Df Deviance Pr(>Chi)
1     59755     4809.9
2     59750     4808.8  5   1.1165   0.9526
```

which is equivalent to the following command:

```
car::Anova(model_full, type=3, test.statistic = "LR")
```

```
Analysis of Deviance Table (Type III tests)

Response: Y
```

```
                       LR Chisq Df Pr(>Chisq)
`cogs_/_ta_ws`            0.290  1  0.5903294
`current_asset_/_ta_ws`   5.074  1  0.0242892 *
`da_/_ta_ws`              0.022  1  0.8828848
`ebit_/_ta_ws`            0.829  1  0.3625239
`gp_/_ta_ws`              0.135  1  0.7136342
`invt_/_ta_ws`            9.612  1  0.0019332 **
`mv_/_ta_ws`            227.707  1  < 2.2e-16 ***
`ni_/_ta_ws`              8.798  1  0.0030157 **
`nsale_/_ta_ws`           4.264  1  0.0389328 *
`opex_/_ta_ws`            6.621  1  0.0100762 *
`re_/_ta_ws`              4.532  1  0.0332606 *
`tcl_/_ta_ws`            15.984  1  6.390e-05 ***
`tl_/_ta_ws`             40.562  1  1.905e-10 ***
`tltd_/_ta_ws`            0.006  1  0.9370648
`trec_/_ta_ws`           11.181  1  0.0008266 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Keeping all the variables (full model vs simpler manual model removing 5 variables/df) do not make the model significantly more accurate using the likelihood ratio test:

```r
full_LL <- logLik(model_full)
manual_LL <- logLik(model_manual)

(LRT <- -2 * (as.numeric(manual_LL)-as.numeric(full_LL)))
```

```
[1] 1.11648
```

```r
(p_val <- pchisq(LRT, df = 5, lower.tail = FALSE))
```

```
[1] 0.9526346
```

Although it is generally not recommended by practitioners and theoreticians, the Hosmer & Lemeshow test (see here or here) allows to quickly asses the "goodness of fit" of a Logistic Regression. But more than the test in itself, the underlying motivation is interesting: the Logistic regression model provides an estimate of the probability of an outcome (success/failure, here the default is success or 1). It is desirable that the estimated probability of this outcome is close to the true probability. A first step is to order the predicted probabilities of the outcome and divide it into Q groups (usually using deciles, Q=10).

Then the average predicted probability for each group is computed and compared to the observed probability.

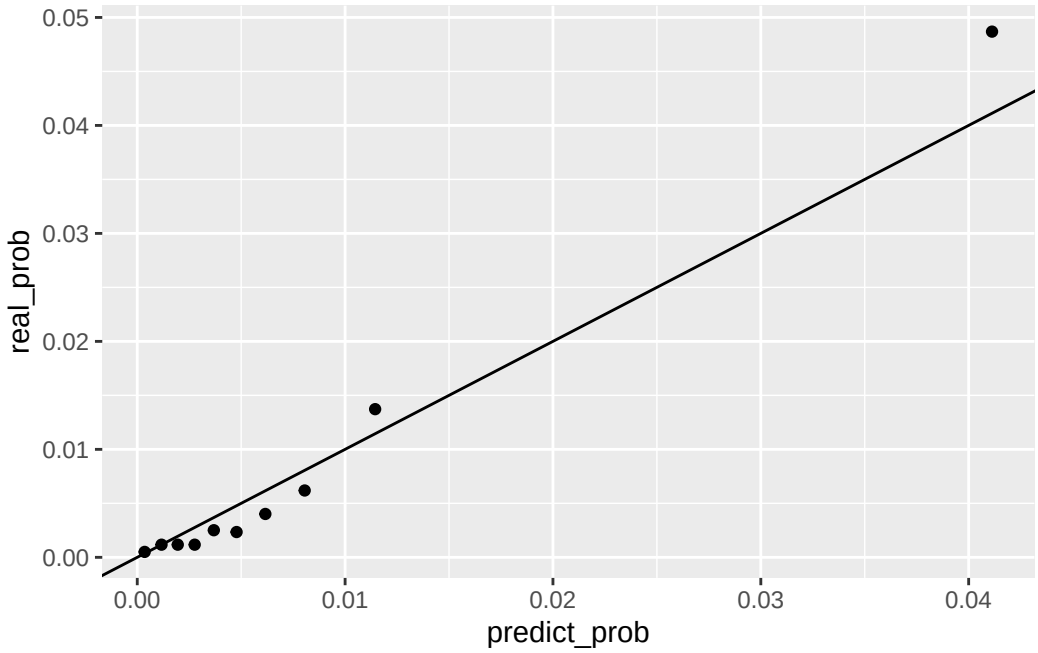
The Hosmer & Lemeshow test uses the predicted vs observed per group to compute a statistic and compare it to a χ^2_{Q-2} distribution. The test is dependent on the choice of Q and the binning performed on probabilities and is sometimes considered unreliable.

As an alternative, it is usual to assess or diagnose the good calibration of the model using Calibration Plots or Probability Calibration Curves (see here for a recent R package from the Tidyverse ecosystem and here for a scikit-learn version): they are used to visualize if predictions are consistent with the observed event rates (be it on the training set or a testing set, which is better). For example considering all the training data, and the manually selected model we have:

```
check_default_prob <- as_tibble(cbind(fitted=model_manual$fitted.values,
                                      Y= default_exam %>%
                                        filter(!company_id %in% outliers_ebit) %>% pull(Y)
(calibration_data <- check_default_prob %>%
  mutate(bins_prob = cut(fitted, breaks = quantile(fitted,seq(0,1,0.10)), include.lowest =
  group_by(bins_prob) %>%
  summarize(n = n(),
            def = sum(Y),
            no_def = n - def,
            predict_prob = mean(fitted),
            real_prob = def/n,
            forecast_acc = def / sum(check_default_prob$Y)))
```

```
# A tibble: 10 x 7
   bins_prob               n   def no_def predict_prob real_prob forecast_acc
   <fct>               <int> <dbl>  <dbl>        <dbl>     <dbl>        <dbl>
 1 [1.19e-05,0.000758]  5977     3   5974     0.000355  0.000502      0.00616
 2 (0.000758,0.00156]   5977     7   5970     0.00117   0.00117       0.0144
 3 (0.00156,0.00234]    5976     7   5969     0.00195   0.00117       0.0144
 4 (0.00234,0.00319]    5977     7   5970     0.00276   0.00117       0.0144
 5 (0.00319,0.00421]    5976    15   5961     0.00368   0.00251       0.0308
 6 (0.00421,0.0054]     5977    14   5963     0.00478   0.00234       0.0287
 7 (0.0054,0.00699]     5976    24   5952     0.00616   0.00402       0.0493
 8 (0.00699,0.00929]    5977    37   5940     0.00806   0.00619       0.0760
 9 (0.00929,0.0147]     5976    82   5894     0.0114    0.0137        0.168
10 (0.0147,0.508]       5977   291   5686     0.0411    0.0487        0.598
```

```r
ggplot(calibration_data, aes(x = predict_prob, y = real_prob)) +
  geom_point() +
  geom_abline()
```



The manually selected model tends to slightly overestimate the default rate for the 8 first deciles, then underestimate the last two deciles.

4.2 Second step (10% total points)

4.2.1 TO DO

In a **second step**, you will need to split the data set using the hold-out approach into a training and testing set.

Explain carefully how you will split the data set given its specific nature (temporal/time series aspect).

Do not use stratified sampling.

Once the split is done, assess using the testing set the two models predictions (**your_model** vs **full_model**) using a ROC curve / AUC approach as seen in the course.

As a fallback (if you cannot plot the ROC or compute the AUC) you can provide a confusion matrix as defined in course and use any metric of your choice to compare models (Accuracy, F1-score,... more details at this Wikipedia page).

Note that usually the first and second step are performed together and the "valid" model is chosen as the result of the second step.

If for some reason, you encounter issues (to slow, memory errors) to run your analysis using the full dataset, you may restrict to the first 1000 companies (company_id 1 to 1000) for the in-class analysis. Then at home expand to either all companies or the maximum you can run on your computer.

4.2.2 Proposed solution

Due to the temporal aspect of the data set, we cannot split randomly the data set into training and testing set. Usually when dealing with time series a chronological or time-series split is preferred (see for example here).

We choose here to train on dates up to 2011 (included) and test on dates after 2011

```
train <- default_exam %>%
          filter(!company_id %in% outliers_ebit, fiscal_year<=2011)

test <- default_exam %>%
          filter(!company_id %in% outliers_ebit, fiscal_year>2011)
```

In complete rigor, we should re-winsorize data using only the training set, then apply (or not) winsorization to testing data using quantiles from training set. Otherwise we are learning some of the testing set when winsorizing (data leak).

This is not what we do in the following, assuming without verifying a comparable distribution between training and testing set.

Fitting full model on training set:

```
model_full_holdout <- glm(Y~.,
                          data = train %>%
                            select(Y, `cogs_/_ta_ws`:`trec_/_ta_ws`, -`ta_/_ta_ws`),
                          family="binomial")

summary(model_full_holdout)
```

```
Call:
glm(formula = Y ~ ., family = "binomial", data = train %>% select(Y,
    `cogs_/_ta_ws`:`trec_/_ta_ws`, -`ta_/_ta_ws`))

Coefficients:
                       Estimate Std. Error z value Pr(>|z|)
(Intercept)            -5.57213    0.20123 -27.690  < 2e-16 ***
`cogs_/_ta_ws`          0.07209    0.19174   0.376  0.70692
`current_asset_/_ta_ws` -0.51779    0.30226  -1.713  0.08670 .
`da_/_ta_ws`           -1.98627    1.62247  -1.224  0.22087
`ebit_/_ta_ws`         -0.05232    0.35309  -0.148  0.88221
`gp_/_ta_ws`            0.03376    0.31026   0.109  0.91336
`invt_/_ta_ws`          1.31341    0.47085   2.789  0.00528 **
`mv_/_ta_ws`           -0.54378    0.05368 -10.130  < 2e-16 ***
`ni_/_ta_ws`           -0.63930    0.22498  -2.842  0.00449 **
`nsale_/_ta_ws`        -0.47671    0.20615  -2.312  0.02075 *
`opex_/_ta_ws`          0.55666    0.16947   3.285  0.00102 **
`re_/_ta_ws`            0.02433    0.02026   1.201  0.22971
`tcl_/_ta_ws`           1.02532    0.33145   3.093  0.00198 **
`tl_/_ta_ws`            1.49649    0.23799   6.288 3.21e-10 ***
`tltd_/_ta_ws`          0.08618    0.31285   0.275  0.78297
`trec_/_ta_ws`         -1.57252    0.56951  -2.761  0.00576 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 4652.8  on 49589  degrees of freedom
Residual deviance: 3994.6  on 49574  degrees of freedom
AIC: 4026.6

Number of Fisher Scoring iterations: 9
```

Fitting manually selected model on training set:

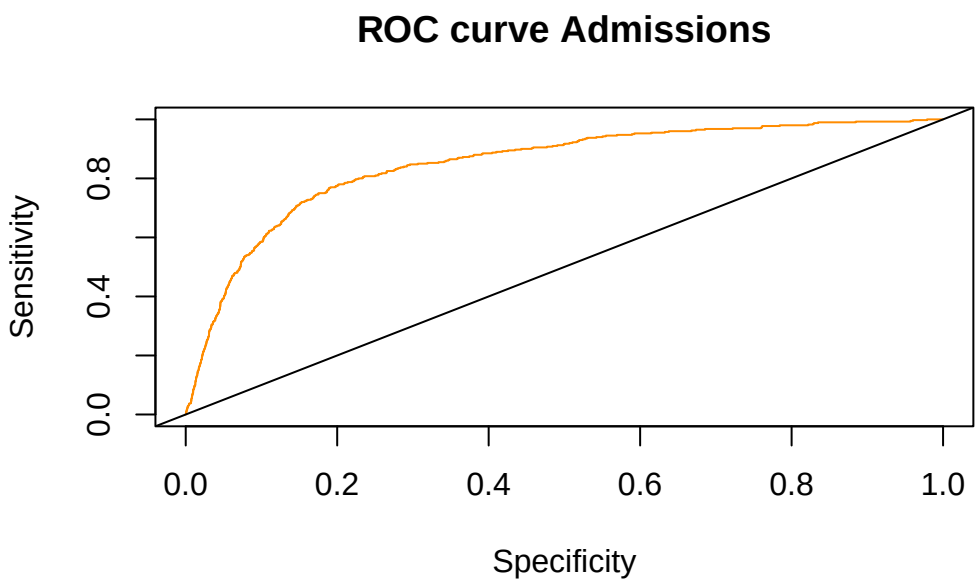
```
# manual_selection <- "Y~ `current_asset_/_ta_ws` + `invt_/_ta_ws` +
                         # `mv_/_ta_ws` + `ni_/_ta_ws` + `nsale_/_ta_ws` +
                         # `opex_/_ta_ws` + `re_/_ta_ws`  + `tcl_/_ta_ws` +
                         # `tl_/_ta_ws`  + `trec_/_ta_ws`"

manual_selection <- "Y~ `current_asset_/_ta_ws` + `invt_/_ta_ws` + `mv_/_ta_ws` + `ni_/_ta
```

```r
model_manual_holdout <- glm(as.formula(manual_selection),
                            data = train %>%
                              select(Y, `cogs_/_ta_ws`:`trec_/_ta_ws`, -`ta_/_ta_ws`),
                            family="binomial")
summary(model_manual_holdout)
```

```
Call:
glm(formula = as.formula(manual_selection), family = "binomial",
    data = train %>% select(Y, `cogs_/_ta_ws`:`trec_/_ta_ws`,
        -`ta_/_ta_ws`))

Coefficients:
                       Estimate Std. Error z value Pr(>|z|)
(Intercept)            -5.67204    0.17357 -32.678  < 2e-16 ***
`current_asset_/_ta_ws` -0.44409    0.28848  -1.539 0.123708
`invt_/_ta_ws`          1.35311    0.46836   2.889 0.003864 **
`mv_/_ta_ws`           -0.54241    0.05333 -10.171  < 2e-16 ***
`ni_/_ta_ws`           -0.58330    0.17235  -3.384 0.000713 ***
`nsale_/_ta_ws`        -0.48248    0.13884  -3.475 0.000511 ***
`opex_/_ta_ws`          0.60013    0.13128   4.571 4.84e-06 ***
`re_/_ta_ws`            0.02626    0.01927   1.363 0.172865
`tcl_/_ta_ws`           0.96874    0.27007   3.587 0.000334 ***
`tl_/_ta_ws`            1.54360    0.18504   8.342  < 2e-16 ***
`trec_/_ta_ws`         -1.53371    0.56660  -2.707 0.006792 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 4652.8  on 49589  degrees of freedom
Residual deviance: 3996.5  on 49579  degrees of freedom
AIC: 4018.5

Number of Fisher Scoring iterations: 9
```

```r
# plot ROC / compute AUC for the training set
pred_full_train <- ROCR::prediction(model_full_holdout$fitted.values,
                                    train$Y)

perf_full_train <- ROCR::performance(pred_full_train, measure = "tpr", x.measure = "fpr")
plot(perf_full_train, main="ROC curve Admissions", xlab="Specificity",
```

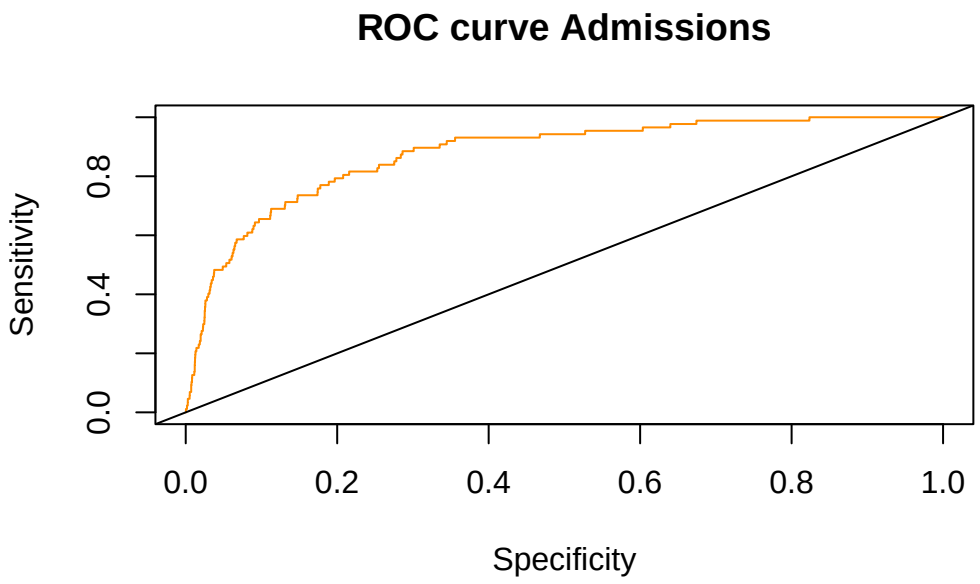
```
     ylab="Sensitivity", col = "darkorange")
abline(0, 1) #add a 45 degree line
```




```
auc_full_train <- ROCR::performance(pred_full_train, measure = "auc")
auc_full_train <- auc_full_train@y.values[[1]]
auc_full_train
```

```
[1] 0.8487214
```

```
# plot ROC / compute AUC for the testing set

predicted_prob_full_test <- predict(model_full_holdout, newdata=test, type="response")

pred_full_test <- ROCR::prediction(predicted_prob_full_test,
                                   test$Y)

perf_full_test <- ROCR::performance(pred_full_test, measure = "tpr", x.measure = "fpr")
plot(perf_full_test, main="ROC curve Admissions", xlab="Specificity",
     ylab="Sensitivity", col = "darkorange")
abline(0, 1) #add a 45 degree line
```

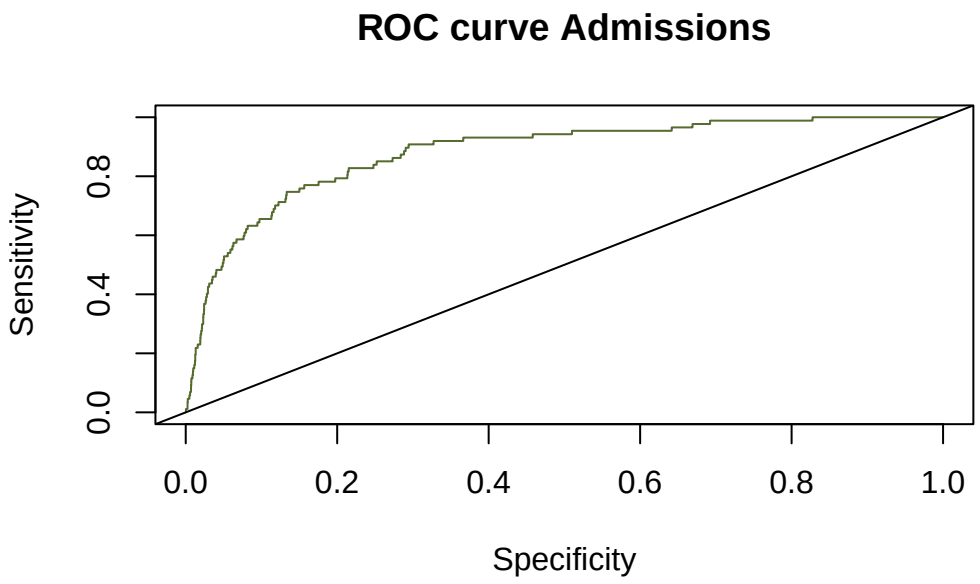



```
auc_full_test <- ROCR::performance(pred_full_test, measure = "auc")
auc_full_test <- auc_full_test@y.values[[1]]
auc_full_test
```

```
[1] 0.8761608
```

The AUC stands at ca. 87.65% on the testing set for the full model.

```
# plot ROC / compute AUC for the testing set

predicted_prob_manual_test <- predict(model_manual_holdout, newdata=test, type="response")

pred_manual_test <- ROCR::prediction(predicted_prob_manual_test,
                                     test$Y)

perf_manual_test <- ROCR::performance(pred_manual_test, measure = "tpr", x.measure = "fpr"
plot(perf_manual_test, main="ROC curve Admissions", xlab="Specificity",
     ylab="Sensitivity", col = "darkolivegreen")
abline(0, 1) #add a 45 degree line
```




```
auc_manual_test <- ROCR::performance(pred_manual_test, measure = "auc")
auc_manual_test <- auc_manual_test@y.values[[1]]
auc_manual_test
```

```
[1] 0.8794602
```

The AUC stands at ca. 87.94% on the testing set for the manually selected model.

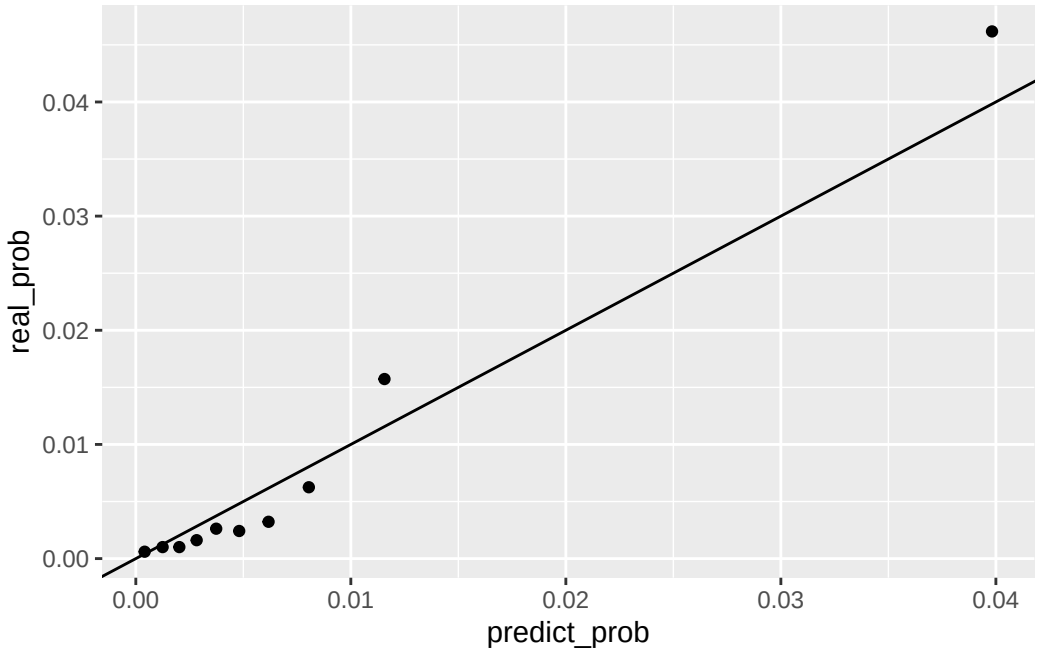
We also provide calibration plots to quickly asses the goodness of fit of the model:

```
check_default_prob_manual_train <- as_tibble(cbind(fitted=model_manual_holdout$fitted.valu
                                      Y= train$Y))
(calibration_data_manual_train <- check_default_prob_manual_train %>%
  mutate(bins_prob = cut(fitted,
                         breaks = quantile(fitted,seq(0,1,0.10)),
                         include.lowest = TRUE)) %>%
  group_by(bins_prob) %>%
  summarize(n = n(),
            def = sum(Y),
            no_def = n - def,
            predict_prob = mean(fitted),
            real_prob = def/n,
```

```
            forecast_acc = def / sum(check_default_prob_manual_train$Y)))
```

```
# A tibble: 10 x 7
   bins_prob               n   def no_def predict_prob real_prob forecast_acc
   <fct>               <int> <dbl>  <dbl>        <dbl>     <dbl>        <dbl>
 1 [1.26e-05,0.000834]  4959     3   4956     0.000411  0.000605       0.0075
 2 (0.000834,0.00164]   4959     5   4954     0.00125   0.00101        0.0125
 3 (0.00164,0.00241]    4959     5   4954     0.00202   0.00101        0.0125
 4 (0.00241,0.00326]    4959     8   4951     0.00283   0.00161        0.02
 5 (0.00326,0.00424]    4959    13   4946     0.00374   0.00262        0.0325
 6 (0.00424,0.00543]    4959    12   4947     0.00481   0.00242        0.03
 7 (0.00543,0.00698]    4959    16   4943     0.00617   0.00323        0.04
 8 (0.00698,0.00929]    4959    31   4928     0.00805   0.00625        0.0775
 9 (0.00929,0.0149]     4959    78   4881     0.0116    0.0157         0.195
10 (0.0149,0.532]       4959   229   4730     0.0398    0.0462         0.572
```

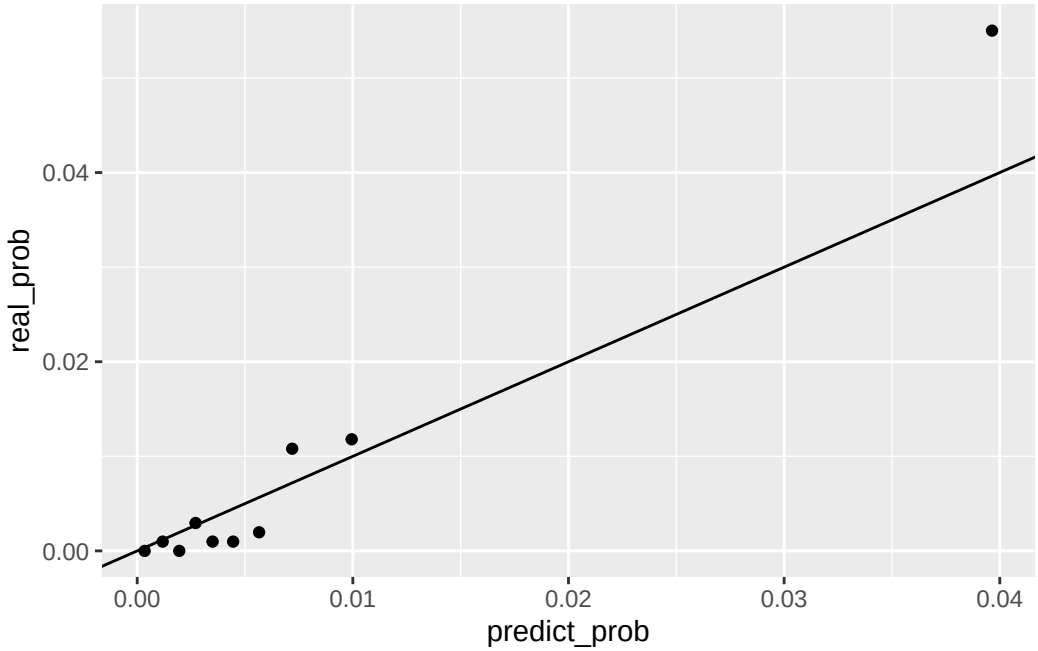
```
ggplot(calibration_data_manual_train, aes(x = predict_prob, y = real_prob)) +
  geom_point() +
  geom_abline()
```



```
check_default_prob_manual_test <- as_tibble(cbind(fitted=predicted_prob_manual_test,
                                        Y= test$Y))
(calibration_data_manual_test <- check_default_prob_manual_test %>%
  mutate(bins_prob = cut(fitted,
                         breaks = quantile(fitted,seq(0,1,0.10)),
                         include.lowest = TRUE)) %>%
  group_by(bins_prob) %>%
  summarize(n = n(),
            def = sum(Y),
            no_def = n - def,
            predict_prob = mean(fitted),
            real_prob = def/n,
            forecast_acc = def / sum(check_default_prob_manual_test$Y)))
```

```
# A tibble: 10 x 7
   bins_prob              n   def no_def predict_prob real_prob forecast_acc
   <fct>              <int> <dbl>  <dbl>        <dbl>     <dbl>        <dbl>
 1 [2.5e-05,0.000769]  1018     0   1018     0.000345  0              0
 2 (0.000769,0.00158]  1018     1   1017     0.00118   0.000982       0.0115
 3 (0.00158,0.00232]   1017     0   1017     0.00195   0              0
 4 (0.00232,0.00309]   1018     3   1015     0.00271   0.00295        0.0345
 5 (0.00309,0.00395]   1017     1   1016     0.00350   0.000983       0.0115
 6 (0.00395,0.00499]   1018     1   1017     0.00445   0.000982       0.0115
 7 (0.00499,0.00637]   1017     2   1015     0.00565   0.00197        0.0230
 8 (0.00637,0.00818]   1018    11   1007     0.00719   0.0108         0.126
 9 (0.00818,0.0128]    1017    12   1005     0.00995   0.0118         0.138
10 (0.0128,0.474]      1018    56    962     0.0396    0.0550         0.644
```

```
ggplot(calibration_data_manual_test, aes(x = predict_prob, y = real_prob)) +
  geom_point() +
  geom_abline()
```



4.3 Third step (5% total points)

4.3.1 TO DO at-home (~30 minutes)

Complete the **first step** variable selection using two unused approaches of your choice. See the effect on **second step**.

When finishing this step you will have 3 models at hand (**your_models**), to keep for take-home part of the exam.

4.3.2 Proposed solution

4.3.2.1 Backward BIC

Fitting stepwise backward model using BIC criterion on training set:

```
model_full_train_holdout <- glm(Y~., data=train %>%
                    select(Y, `cogs_/_ta_ws`:`trec_/_ta_ws`, -`ta_/_ta_ws`),
                    family="binomial")

intercept_only_train_holdout <- glm(Y~1, data=train %>%
                    select(Y, `cogs_/_ta_ws`:`trec_/_ta_ws`, -`ta_/_ta_ws`),
```

```
                    family="binomial")
```

```
forward_aic_holdout <- step(intercept_only_train_holdout,
                    direction='forward',
                    test = 'LRT',
                    scope=formula(model_full_train_holdout),
                    k=2,
                    trace = FALSE)
summary(forward_aic_holdout)
```

```
Call:
glm(formula = Y ~ `tl_/_ta_ws` + `mv_/_ta_ws` + `ni_/_ta_ws` +
    `tcl_/_ta_ws` + `trec_/_ta_ws` + `opex_/_ta_ws` + `nsale_/_ta_ws` +
    `invt_/_ta_ws` + `current_asset_/_ta_ws`, family = "binomial",
    data = train %>% select(Y, `cogs_/_ta_ws`:`trec_/_ta_ws`,
        -`ta_/_ta_ws`))

Coefficients:
                      Estimate Std. Error z value Pr(>|z|)
(Intercept)           -5.61326    0.16736 -33.540  < 2e-16 ***
`tl_/_ta_ws`           1.47765    0.17871   8.268  < 2e-16 ***
`mv_/_ta_ws`          -0.55043    0.05329 -10.330  < 2e-16 ***
`ni_/_ta_ws`          -0.51860    0.16527  -3.138 0.001701 **
`tcl_/_ta_ws`          0.92078    0.26926   3.420 0.000627 ***
`trec_/_ta_ws`        -1.47652    0.56526  -2.612 0.008999 **
`opex_/_ta_ws`         0.54784    0.12567   4.359  1.3e-05 ***
`nsale_/_ta_ws`       -0.43052    0.13307  -3.235 0.001215 **
`invt_/_ta_ws`         1.44976    0.46268   3.133 0.001728 **
`current_asset_/_ta_ws` -0.51049    0.28477  -1.793 0.073026 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 4652.8  on 49589  degrees of freedom
Residual deviance: 3998.4  on 49580  degrees of freedom
AIC: 4018.4

Number of Fisher Scoring iterations: 9
```

```r
forward_bic_holdout <- step(intercept_only_train_holdout,
                    direction='forward',
                    test = 'LRT',
                    scope=formula(model_full_train_holdout),
                    k=log(nrow(train)),
                    trace = FALSE)
summary(forward_bic_holdout)
```

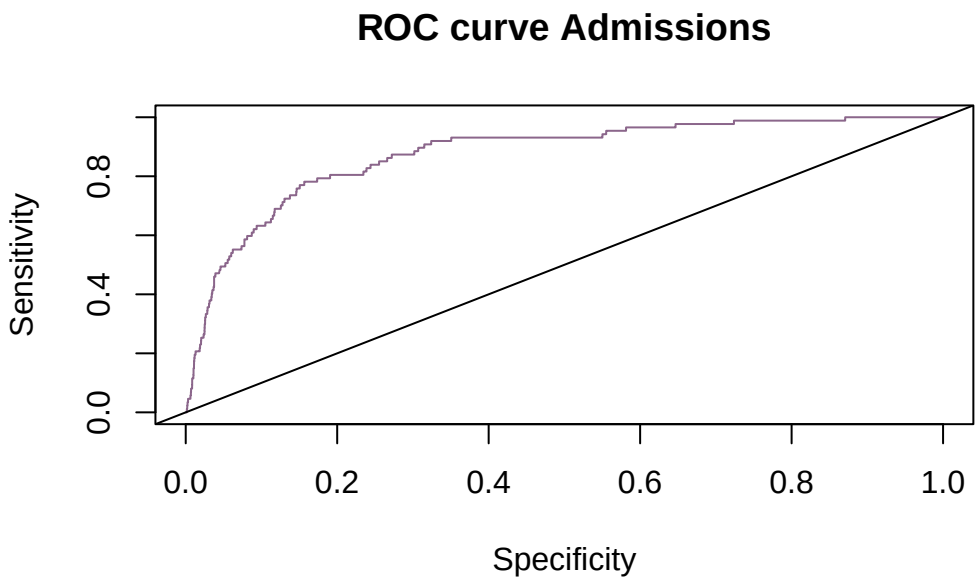
```
Call:
glm(formula = Y ~ `tl_/_ta_ws` + `mv_/_ta_ws` + `ni_/_ta_ws` +
    `tcl_/_ta_ws` + `trec_/_ta_ws` + `opex_/_ta_ws`, family = "binomial",
    data = train %>% select(Y, `cogs_/_ta_ws`:`trec_/_ta_ws`,
        -`ta_/_ta_ws`))

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept)    -5.74230    0.14161 -40.549  < 2e-16 ***
`tl_/_ta_ws`    1.47018    0.17031   8.632  < 2e-16 ***
`mv_/_ta_ws`   -0.52887    0.05037 -10.500  < 2e-16 ***
`ni_/_ta_ws`   -0.70425    0.13319  -5.288 1.24e-07 ***
`tcl_/_ta_ws`   0.92291    0.26254   3.515 0.000439 ***
`trec_/_ta_ws` -2.32225    0.49101  -4.730 2.25e-06 ***
`opex_/_ta_ws`  0.24809    0.06450   3.847 0.000120 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 4652.8  on 49589  degrees of freedom
Residual deviance: 4017.4  on 49583  degrees of freedom
AIC: 4031.4

Number of Fisher Scoring iterations: 9
```

```r
backward_bic_holdout <- step(model_full_train_holdout,
                    direction='backward',
                    test = 'LRT',
                    scope=formula(model_full_train_holdout),
                    k=log(nrow(train)),
                    trace = FALSE)
summary(backward_bic_holdout)
```

```
Call:
glm(formula = Y ~ `mv_/_ta_ws` + `nsale_/_ta_ws` + `opex_/_ta_ws` +
    `tcl_/_ta_ws` + `tl_/_ta_ws` + `trec_/_ta_ws`, family = "binomial",
    data = train %>% select(Y, `cogs_/_ta_ws`:`trec_/_ta_ws`,
        -`ta_/_ta_ws`))

Coefficients:
               Estimate Std. Error z value Pr(>|z|)
(Intercept)    -5.67210    0.14727 -38.514  < 2e-16 ***
`mv_/_ta_ws`   -0.57407    0.05358 -10.714  < 2e-16 ***
`nsale_/_ta_ws` -0.59919    0.10950  -5.472 4.45e-08 ***
`opex_/_ta_ws`  0.75706    0.10253   7.384 1.54e-13 ***
`tcl_/_ta_ws`   1.02534    0.25970   3.948 7.87e-05 ***
`tl_/_ta_ws`    1.58170    0.16948   9.333  < 2e-16 ***
`trec_/_ta_ws` -1.97461    0.51053  -3.868  0.00011 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 4652.8  on 49589  degrees of freedom
Residual deviance: 4015.9  on 49583  degrees of freedom
AIC: 4029.9

Number of Fisher Scoring iterations: 9
```

Forward and Backward stepwise selection using the BIC criterion using the full data set agree on the choice of variables.

```
# plot ROC / compute AUC for the testing set

predicted_prob_back_bic_test <- predict(backward_bic_holdout, newdata=test, type="response

pred_back_bic_test <- ROCR::prediction(predicted_prob_back_bic_test,
                                       test$Y)

perf_back_bic_test <- ROCR::performance(pred_back_bic_test, measure = "tpr", x.measure = "
plot(perf_back_bic_test, main="ROC curve Admissions", xlab="Specificity",
     ylab="Sensitivity", col = "plum4")
abline(0, 1) #add a 45 degree line
```




```
auc_back_bic_test <- ROCR::performance(pred_back_bic_test, measure = "auc")
auc_back_bic_test <- auc_back_bic_test@y.values[[1]]
auc_back_bic_test
```

```
[1] 0.875222
```

The AUC stands at ca. 87.52% on the testing set for the stagewise selected (backward BIC) model.

4.3.2.2 Lasso

```
set.seed(1)
model_lasso_holdout <- glmnetUtils::cv.glmnet(Y ~ .,
                            data=train %>%
                              select(Y, `cogs_/_ta_ws`:`trec_/_ta_ws`, -`ta_/_ta_ws`),
                            family="binomial",
                            alpha=1,
                            standardize = TRUE,
                            type.measure = "auc")
```

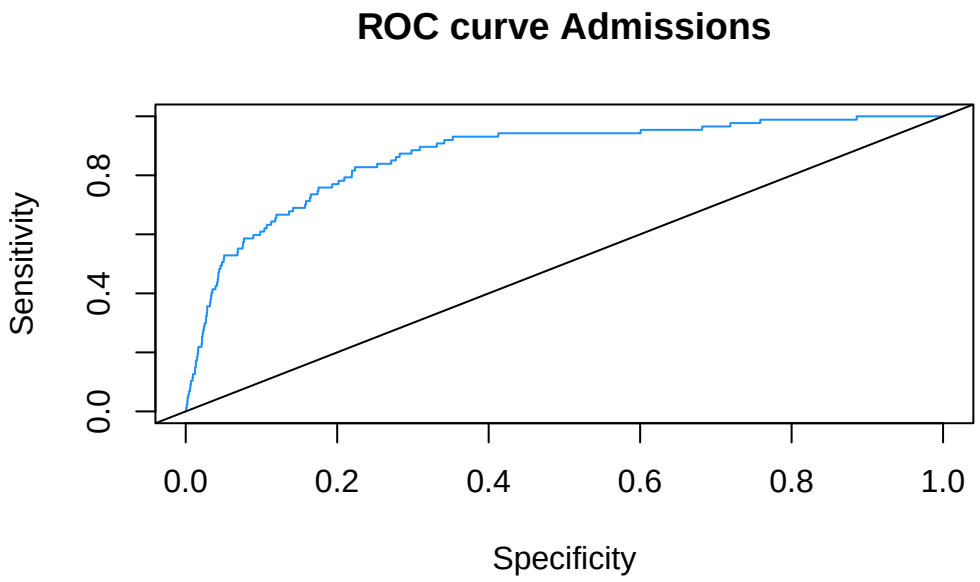
```r
lasso_holdout_coeffs_1se <- coef(model_lasso_holdout, s = "lambda.1se")
tibble(name = lasso_holdout_coeffs_1se@Dimnames[[1]][lasso_holdout_coeffs_1se@i + 1],
       coefficient = lasso_holdout_coeffs_1se@x)
```

```
# A tibble: 9 x 2
  name         coefficient
  <chr>              <dbl>
1 (Intercept)      -5.86
2 ebit_/_ta_ws     -0.0822
3 invt_/_ta_ws      0.422
4 mv_/_ta_ws       -0.355
5 ni_/_ta_ws       -0.599
6 opex_/_ta_ws      0.0645
7 tcl_/_ta_ws       0.822
8 tl_/_ta_ws        1.49
9 trec_/_ta_ws     -0.726
```

```r
lasso_holdout_coeffs_min <- coef(model_lasso_holdout, s = "lambda.min")
tibble(name = lasso_holdout_coeffs_min@Dimnames[[1]][lasso_holdout_coeffs_min@i + 1],
       coefficient = lasso_holdout_coeffs_min@x)
```

```
# A tibble: 15 x 2
   name                  coefficient
   <chr>                       <dbl>
 1 (Intercept)              -5.63
 2 current_asset_/_ta_ws    -0.405
 3 da_/_ta_ws               -1.25
 4 ebit_/_ta_ws             -0.0943
 5 gp_/_ta_ws               -0.128
 6 invt_/_ta_ws              1.19
 7 mv_/_ta_ws               -0.504
 8 ni_/_ta_ws               -0.620
 9 nsale_/_ta_ws            -0.226
10 opex_/_ta_ws              0.394
11 re_/_ta_ws                0.00957
12 tcl_/_ta_ws               0.952
13 tl_/_ta_ws                1.49
14 tltd_/_ta_ws              0.0288
15 trec_/_ta_ws             -1.54
```

```r
# plot ROC / compute AUC for the testing set

predicted_prob_lasso_test <- as.vector(predict(model_lasso_holdout, newdata=test, type="re

pred_lasso_test <- ROCR::prediction(predicted_prob_lasso_test,
                                    test$Y)

perf_lasso_test <- ROCR::performance(pred_lasso_test, measure = "tpr", x.measure = "fpr")
plot(perf_lasso_test, main="ROC curve Admissions", xlab="Specificity",
     ylab="Sensitivity", col = "dodgerblue")
abline(0, 1) #add a 45 degree line
```




```r
auc_lasso_test <- ROCR::performance(pred_lasso_test, measure = "auc")
auc_lasso_test <- auc_lasso_test@y.values[[1]]
auc_lasso_test
```

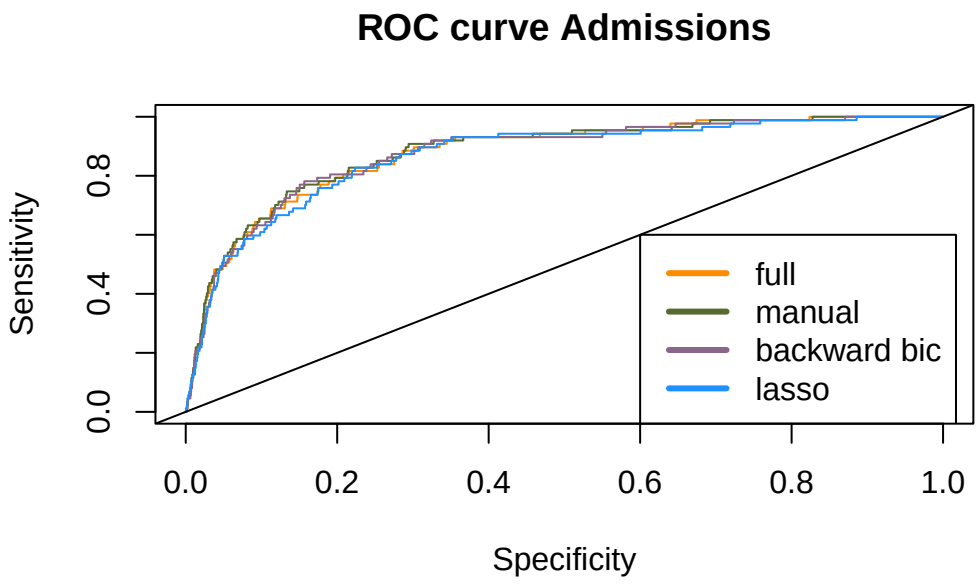
```
[1] 0.8677312
```

The AUC stands at ca. 86.77% on the testing set for the lasso model (using penalization parameter lambda.1se).

4.3.2.3 ROC curves

We display the four models ROC curves:

```
plot(perf_full_test, main="ROC curve Admissions", xlab="Specificity",
     ylab="Sensitivity", col = "darkorange")
plot(perf_manual_test, add = TRUE, main="ROC curve Admissions", xlab="Specificity",
     ylab="Sensitivity", col = "darkolivegreen")
plot(perf_back_bic_test, add = TRUE, main="ROC curve Admissions", xlab="Specificity",
     ylab="Sensitivity", col = "plum4")
plot(perf_lasso_test, add = TRUE, main="ROC curve Admissions", xlab="Specificity",
     ylab="Sensitivity", col = "dodgerblue")
abline(0, 1) #add a 45 degree line
legend(0.6,0.6,
       c('full', 'manual', 'backward bic', 'lasso'),
       col=c("darkorange", "darkolivegreen", "plum4", "dodgerblue"),lwd=3)
```



5 Model improvement and prediction using a validation set

TO DO at-home (~3-4 hours)

5.1 Feature engineering

5.1.1 TO DO

In this part, you will try to improve the model devised in last part (**your_improved_model**):

- it may involve improving variable selection, creating or transforming variables,
- you may, if relevant, consider potential non-linearities or interactions in features (for example using: polynomial expansion or `splines` to transform variables or by binning some variables into qualitative variables, as seen in class for the mixture data set),
- given the time series / temporal aspect you might also enrich the data set with any relevant macroeconomic indicator of your choice, remembering that the companies are public companies listed in the US. For this you can use for example the FRED website providing financial and macroeconomic time series. Be cautious not to use time series data that would be more current than the observations, use appropriate lag to avoid this pitfall,
- you can also try to deal with response variable imbalance (there are more healthy companies than defaulting in the data set) by sampling differently your training set (for example undersampling the healthy cases).

In this part you need to implement three different improvements to get full score (i.e. three models). You may combine these improvements in a fourth model.

Explain or motivate briefly your choices and assumptions.

Note that you might find that **your_improved_models** do not improve predictions using your metric. It happens in real life, what is important is the approach you follow.

5.1.2 Proposed solution

5.1.2.1 Altman/Ohlson

We use here predictors used by two classical bankruptcy prediction models first Altman's Z-score (Altman 1968) then Ohlson's O-score (Ohlson 1980).

```
default_exam_classic <- default_exam %>%
                        mutate(wc=current_asset-tcl,
                               wcta = wc / ta, # Altman's 5 ratios
                               reta = re / ta,
                               ebta = ebit / ta,
                               mvtl = mv / tl,
                               salta = nsale / ta,
```

```
                                tlta = tl / ta, # Completing with Ohlson's
                                clca = tcl / current_asset,
                                oeneg = as.factor(if_else(tcl > ta, 1, 0)),
                                nita = ni / ta,
                                futl = ebit / tl)

default_exam_classic <- default_exam_classic %>%
                          group_by(company_id) %>%
                          arrange(company_id, fiscal_year) %>%
                          mutate(ni_1 = lag(ni),
                                 chin = (ni - ni_1)/(abs(ni) + abs(ni_1))) %>%
                          ungroup()
```

ent ratio-profiles. The final discriminant function is as follows:

$$\text{(I)} \quad Z = .012X_1 + .014X_2 + .033X_3 + .006X_4 + .999X_5$$

where $X_1 =$ Working capital/Total assets
$X_2 =$ Retained Earnings/Total assets
$X_3 =$ Earnings before interest and taxes/Total assets
$X_4 =$ Market value equity/Book value of total debt
$X_5 =$ Sales/Total assets
$Z =$ Overall Index

```
altman_var <- "Y ~ wcta + reta + ebta + mvtl + salta"

model_altman <- glm(as.formula(altman_var),
                          data=default_exam_classic %>%
                           filter(!company_id %in% outliers_ebit),
                          family="binomial")
summary(model_altman)
```

```
Call:
glm(formula = as.formula(altman_var), family = "binomial", data = default_exam_classic %>%
    filter(!company_id %in% outliers_ebit))

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -4.023999   0.067655 -59.478  < 2e-16 ***
wcta         0.012411   0.004331   2.865  0.00416 **
reta        -0.003327   0.000837  -3.975 7.03e-05 ***
ebta        -0.106797   0.023089  -4.625 3.74e-06 ***
```

```
mvtl        -0.331315   0.029712 -11.151  < 2e-16 ***
salta       -0.014116   0.020595  -0.685  0.49308
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 5654.9  on 59765  degrees of freedom
Residual deviance: 5317.2  on 59760  degrees of freedom
AIC: 5329.2

Number of Fisher Scoring iterations: 12
```

5. *Ratios and Basic Results*

For purposes of the present report, no attempt was made to develop any "new or exotic" ratios. The criterion for choosing among different predictors was simplicity. The first three models estimated, Models 1–3, were composed of an intercept and the following nine independent variables:[10]

1. *SIZE* = log(total assets/GNP price-level index). The index assumes a base value of 100 for 1968. Total assets are as reported in dollars. The index year is as of the year prior to the year of the balance sheet date. The procedure assures a real-time implementation of the model. The log transform has an important implication. Suppose two firms, A and B, have a balance sheet date in the same year, then the sign of $P_A - P_B$ is independent of the price-level index. (This will not follow unless the log transform is applied.) The latter is, of course, a desirable property.
2. *TLTA* = Total liabilities divided by total assets.
3. *WCTA* = Working capital divided by total assets.
4. *CLCA* = Current liabilities divided by current assets.
5. *OENEG* = One if total liabilities exceeds total assets, zero otherwise.
6. *NITA* = Net income divided by total assets.
7. *FUTL* = Funds provided by operations divided by total liabilities.

[9] See McFadden [1973] for a comprehensive analysis of the logit model.

[10] No attempt was made to select predictors on the basis of rigorous theory. To put it mildly, the state of the art seems to preclude such an approach. (The first six predictors were partially selected simply because they appear to be the ones most frequently mentioned in the literature.)

8. *INTWO* = One if net income was negative for the last two years, zero otherwise.
9. $CHIN = (NI_t - NI_{t-1})/(|NI_t| + |NI_{t-1}|)$, where NI_t is net income for the most recent period. The denominator acts as a level indicator. The variable is thus intended to measure change in net income. (The measure appears to be due to McKibben [1972]).

Previous studies, "common sense," and perhaps even theory, would suggest that the sign of the coefficients of the different ratios should be as follows:

Positive	Negative	Indeterminate
TLTA	*SIZE*	*OENEG*
CLCA	*WCTA*	
INTWO	*NITA*	
	FUTL	
	CHIN	

```
ohlson_var <- "Y ~ log(ta) + tlta + wcta + clca + oeneg + nita + futl + chin"

model_ohlson <- glm(as.formula(ohlson_var),
                        data=default_exam_classic %>%
                         filter(!company_id %in% outliers_ebit),
                        family="binomial")
summary(model_ohlson)
```

```
Call:
glm(formula = as.formula(ohlson_var), family = "binomial", data = default_exam_classic %>%
    filter(!company_id %in% outliers_ebit))

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -4.8841110  0.1298407 -37.616  < 2e-16 ***
log(ta)     -0.0039803  0.0211015  -0.189    0.850
tlta        -0.0186393  0.0280929  -0.663    0.507
wcta        -0.0247161  0.0357460  -0.691    0.489
clca        -0.0001842  0.0006912  -0.266    0.790
```

```
oeneg1       1.9644069  0.1517277  12.947  < 2e-16 ***
nita        -0.0050698  0.0099572  -0.509    0.611
futl        -0.0054175  0.0063394  -0.855    0.393
chin        -0.5806459  0.0829174  -7.003 2.51e-12 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 5405.8  on 51959  degrees of freedom
Residual deviance: 5154.6  on 51951  degrees of freedom
  (7806 observations deleted due to missingness)
AIC: 5172.6

Number of Fisher Scoring iterations: 8
```

```r
altman_ohlson_var <- "Y ~ log(ta) + wcta + reta + ebta + mvtl + salta + tlta + clca + oene

model_altman_ohlson <- glm(as.formula(altman_ohlson_var),
                           data=default_exam_classic %>%
                            filter(!company_id %in% outliers_ebit),
                           family="binomial")
summary(model_altman_ohlson)
```

```
Call:
glm(formula = as.formula(altman_ohlson_var), family = "binomial",
    data = default_exam_classic %>% filter(!company_id %in% outliers_ebit))

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -3.8156406  0.1611241 -23.681  < 2e-16 ***
log(ta)     -0.0493043  0.0220187  -2.239   0.0251 *
wcta        -0.0353317  0.0447694  -0.789   0.4300
reta        -0.0009983  0.0012747  -0.783   0.4335
ebta        -0.0322933  0.0327397  -0.986   0.3240
mvtl        -0.3231861  0.0301270 -10.727  < 2e-16 ***
salta       -0.0304602  0.0310506  -0.981   0.3266
tlta        -0.0321786  0.0395162  -0.814   0.4155
clca        -0.0005469  0.0010449  -0.523   0.6007
oeneg1       1.3330978  0.1565987   8.513  < 2e-16 ***
nita         0.0044129  0.0228619   0.193   0.8469
```

```
futl        -0.2439928  0.0319349  -7.640 2.17e-14 ***
chin        -0.4533910  0.0796290  -5.694 1.24e-08 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 5405.8  on 51959  degrees of freedom
Residual deviance: 4873.4  on 51947  degrees of freedom
  (7806 observations deleted due to missingness)
AIC: 4899.4

Number of Fisher Scoring iterations: 12
```

```r
model_altman_ws <- glm(as.formula(altman_var),
                       data=default_exam_classic %>%
                         filter(!company_id %in% outliers_ebit) %>%
                         mutate(across(.cols = wc:clca,
                                       .fns = ~ DescTools::Winsorize(., probs = c(0.05, 0.95)
                         mutate(across(.cols = nita:futl,
                                       .fns = ~ DescTools::Winsorize(., probs = c(0.05, 0.95)
                      family="binomial")
summary(model_altman_ws)
```

```
Call:
glm(formula = as.formula(altman_var), family = "binomial", data = default_exam_classic %>%
    filter(!company_id %in% outliers_ebit) %>% mutate(across(.cols = wc:clca,
    .fns = ~DescTools::Winsorize(., probs = c(0.05, 0.95)))) %>%
    mutate(across(.cols = nita:futl, .fns = ~DescTools::Winsorize(.,
        probs = c(0.05, 0.95)))))

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -4.24351    0.09639 -44.023  < 2e-16 ***
wcta        -1.69787    0.22149  -7.666 1.78e-14 ***
reta         0.01348    0.02213   0.609   0.5425
ebta        -1.79454    0.23173  -7.744 9.63e-15 ***
mvtl        -0.28346    0.03103  -9.135  < 2e-16 ***
salta        0.12033    0.05363   2.244   0.0249 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 5654.9  on 59765  degrees of freedom
Residual deviance: 5018.6  on 59760  degrees of freedom
AIC: 5030.6

Number of Fisher Scoring iterations: 10
```

```r
model_ohlson_ws <- glm(as.formula(ohlson_var),
                       data=default_exam_classic %>%
                         filter(!company_id %in% outliers_ebit) %>%
                         mutate(across(.cols = wc:clca,
                                       .fns = ~ DescTools::Winsorize(., probs = c(0.05, 0.95)
                         mutate(across(.cols = nita:futl,
                                       .fns = ~ DescTools::Winsorize(., probs = c(0.05, 0.95)
                      family="binomial")
summary(model_ohlson_ws)
```

```
Call:
glm(formula = as.formula(ohlson_var), family = "binomial", data = default_exam_classic %>%
    filter(!company_id %in% outliers_ebit) %>% mutate(across(.cols = wc:clca,
    .fns = ~DescTools::Winsorize(., probs = c(0.05, 0.95)))) %>%
    mutate(across(.cols = nita:futl, .fns = ~DescTools::Winsorize(.,
        probs = c(0.05, 0.95)))))

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -7.69956    0.30068 -25.607  < 2e-16 ***
log(ta)      0.09118    0.02290   3.981 6.85e-05 ***
tlta         2.51451    0.19157  13.126  < 2e-16 ***
wcta         0.11391    0.41240   0.276 0.782382
clca         0.66570    0.15517   4.290 1.79e-05 ***
oeneg1      -0.59322    0.17276  -3.434 0.000595 ***
nita        -0.41436    0.21140  -1.960 0.049986 *
futl        -0.50200    0.11715  -4.285 1.83e-05 ***
chin        -0.45500    0.08812  -5.163 2.43e-07 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)
```

```
    Null deviance: 5405.8  on 51959  degrees of freedom
Residual deviance: 4704.8  on 51951  degrees of freedom
  (7806 observations deleted due to missingness)
AIC: 4722.8

Number of Fisher Scoring iterations: 8
```

```r
altman_ohlson_var <- "Y ~ log(ta) + wcta + reta + ebta + mvtl + salta + tlta + clca + oene

model_altman_ohlson_ws <- glm(as.formula(altman_ohlson_var),
                     data=default_exam_classic %>%
                       filter(!company_id %in% outliers_ebit) %>%
                       mutate(across(.cols = wc:clca,
                                     .fns = ~ DescTools::Winsorize(., probs = c(0.05, 0.95)
                       mutate(across(.cols = nita:futl,
                                     .fns = ~ DescTools::Winsorize(., probs = c(0.05, 0.95)
                    family="binomial")
summary(model_altman_ohlson_ws)
```

```
Call:
glm(formula = as.formula(altman_ohlson_var), family = "binomial",
    data = default_exam_classic %>% filter(!company_id %in% outliers_ebit) %>%
        mutate(across(.cols = wc:clca, .fns = ~DescTools::Winsorize(.,
            probs = c(0.05, 0.95)))) %>% mutate(across(.cols = nita:futl,
        .fns = ~DescTools::Winsorize(., probs = c(0.05, 0.95)))))

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -7.04584    0.35021 -20.119  < 2e-16 ***
log(ta)      0.04222    0.02687   1.571 0.116162
wcta         0.71280    0.42676   1.670 0.094871 .
reta         0.07168    0.02605   2.752 0.005928 **
ebta         0.63064    0.38826   1.624 0.104318
mvtl        -0.22561    0.03102  -7.273 3.51e-13 ***
salta        0.15763    0.05904   2.670 0.007582 **
tlta         2.11107    0.21479   9.829  < 2e-16 ***
clca         0.76844    0.15924   4.826 1.39e-06 ***
oeneg1      -0.32753    0.18278  -1.792 0.073143 .
nita        -0.84990    0.26399  -3.219 0.001285 **
futl        -1.12288    0.15124  -7.424 1.13e-13 ***
```

```
chin        -0.31256    0.08915  -3.506 0.000455 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 5405.8  on 51959  degrees of freedom
Residual deviance: 4570.2  on 51947  degrees of freedom
  (7806 observations deleted due to missingness)
AIC: 4596.2

Number of Fisher Scoring iterations: 10
```

```r
altman_ohlson_manual_var <- "Y ~ reta + mvtl + salta + tlta + clca + oeneg + nita + futl +

model_altman_ohlson_manual_ws <- glm(as.formula(altman_ohlson_manual_var),
                data=default_exam_classic %>%
                  filter(!company_id %in% outliers_ebit) %>%
                  mutate(across(.cols = wc:clca,
                                .fns = ~ DescTools::Winsorize(., probs = c(0.05, 0.95)))) 
                  mutate(across(.cols = nita:futl,
                                .fns = ~ DescTools::Winsorize(., probs = c(0.05, 0.95)))),
               family="binomial")
summary(model_altman_ohlson_manual_ws)
```

```
Call:
glm(formula = as.formula(altman_ohlson_manual_var), family = "binomial",
    data = default_exam_classic %>% filter(!company_id %in% outliers_ebit) %>%
        mutate(across(.cols = wc:clca, .fns = ~DescTools::Winsorize(.,
            probs = c(0.05, 0.95)))) %>% mutate(across(.cols = nita:futl,
        .fns = ~DescTools::Winsorize(., probs = c(0.05, 0.95)))))

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -6.39258    0.19358 -33.022  < 2e-16 ***
reta         0.09619    0.02372   4.056 5.00e-05 ***
mvtl        -0.22458    0.03076  -7.300 2.87e-13 ***
salta        0.13144    0.05538   2.373 0.017634 *
tlta         2.06948    0.21088   9.813  < 2e-16 ***
clca         0.52201    0.08842   5.904 3.55e-09 ***
oeneg1      -0.38329    0.18032  -2.126 0.033532 *
```

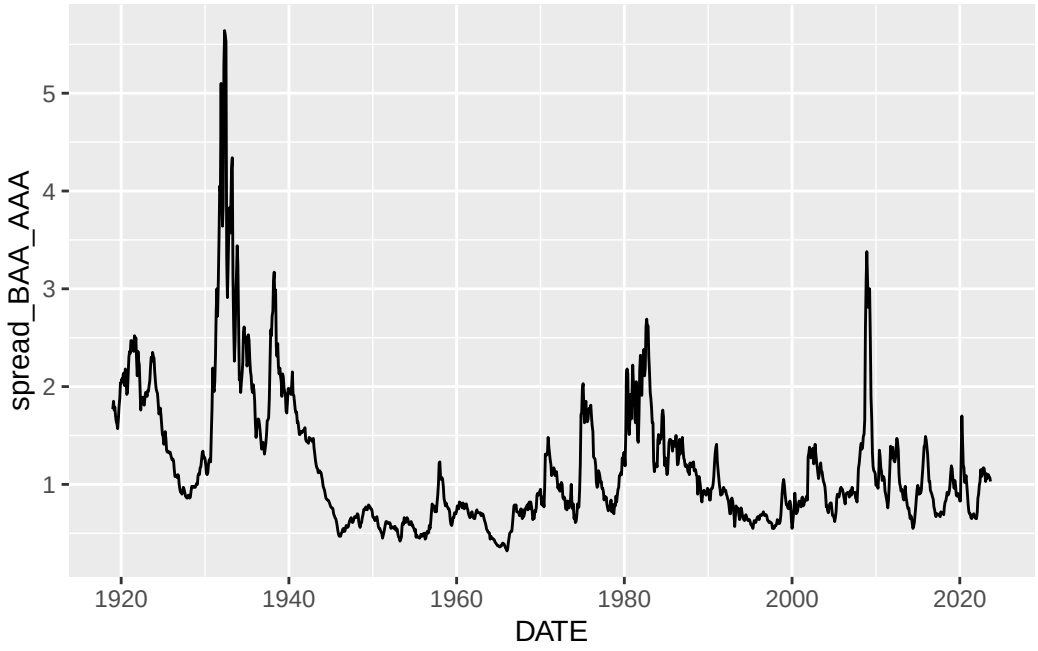
```
nita        -0.57376    0.22265  -2.577 0.009967 **
futl        -0.97897    0.13110  -7.468 8.17e-14 ***
chin        -0.32211    0.08839  -3.644 0.000268 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 5405.8  on 51959  degrees of freedom
Residual deviance: 4578.5  on 51950  degrees of freedom
  (7806 observations deleted due to missingness)
AIC: 4598.5

Number of Fisher Scoring iterations: 10
```

5.1.2.2 FRED

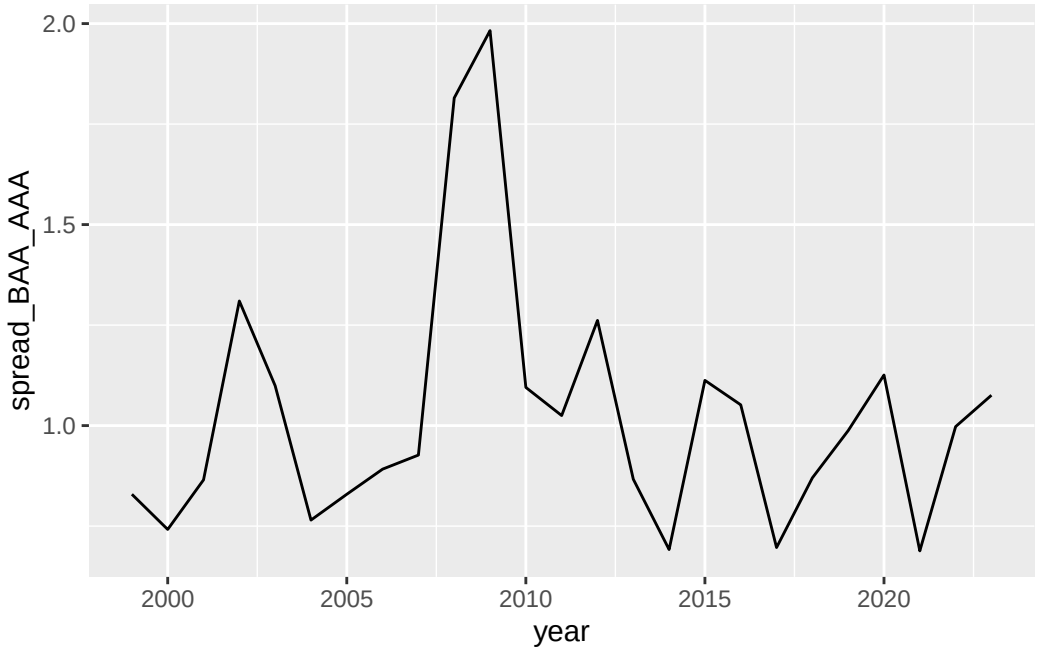
We take inspiration from this FRED blog post (https://fredblog.stlouisfed.org/2014/05/dating-the-financial-crisis-using-fixed-income-markets-yields-spreads/) using in particular the spread between Moody's seasoned Aaa corporate bond yield and Moody's seasoned Baa corporate bond yield to "date" the Great Financial Crisis. Significant breaks in the dynamics of yield spreads from U.S. fixed income markets are supposed to be indicator of a crisis and can be used to predict future rise in defaults.

```
url_AAA <- "https://fred.stlouisfed.org/series/AAA/downloaddata/AAA.csv"
series_AAA <- read_csv(url_AAA) %>%
                rename(AAA = VALUE)

url_BAA <- "https://fred.stlouisfed.org/series/BAA/downloaddata/BAA.csv"
series_BAA <- read_csv(url_BAA) %>%
                rename(BAA = VALUE)
```

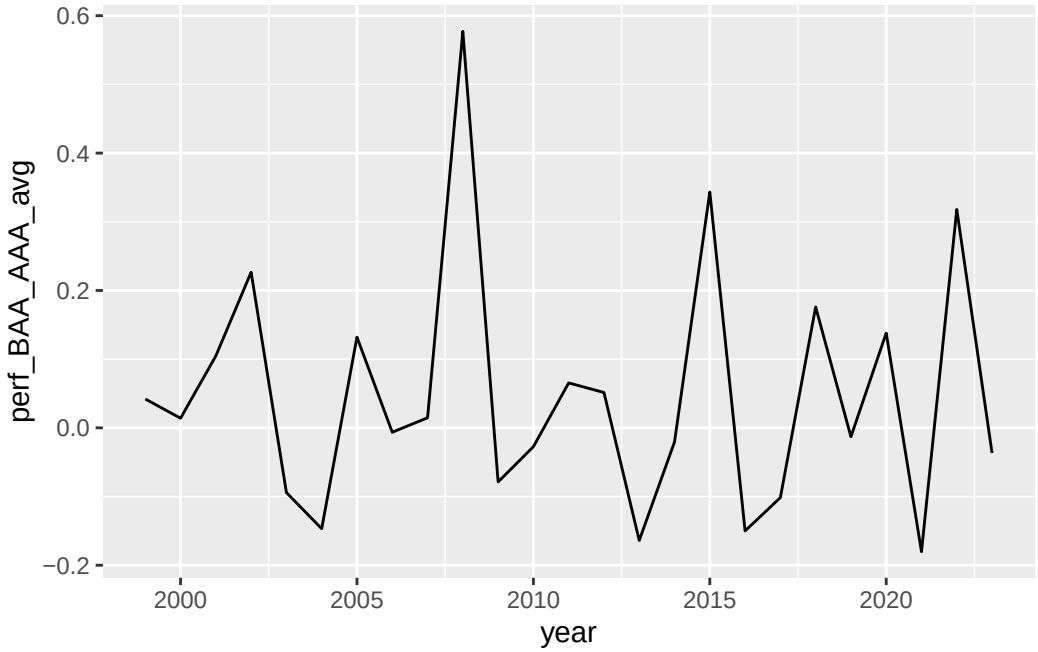
```
spread_corporate <- series_AAA %>%
                      left_join(series_BAA, by = c("DATE")) %>%
                      mutate(spread_BAA_AAA = BAA - AAA,
                             spread_BAA_AAA_min1 = lag(spread_BAA_AAA,n=6),
                             perf_BAA_AAA = spread_BAA_AAA / spread_BAA_AAA_min1-1,
                             year = lubridate::year(DATE))

ggplot(spread_corporate, aes(x=DATE, y=spread_BAA_AAA)) + geom_line()
```




```
spread_corporate_yearly <- spread_corporate %>%
                                group_by(year) %>%
                                summarise(spread_BAA_AAA=mean(spread_BAA_AAA),
                                          perf_BAA_AAA_avg=mean(perf_BAA_AAA)) %>%
                                mutate(year_1 = year + 1,
                                       year_2 = year + 2)

ggplot(spread_corporate_yearly %>%
         filter(year>=1999), aes(x=year, y=spread_BAA_AAA)) + geom_line()
```




```
ggplot(spread_corporate_yearly %>%
         filter(year>=1999), aes(x=year, y=perf_BAA_AAA_avg)) + geom_line()
```



```r
# manual_selection_FRED <- "Y~ `current_asset_/_ta_ws` + `invt_/_ta_ws` +
                            # `mv_/_ta_ws` + `ni_/_ta_ws` + `nsale_/_ta_ws` +
                            # `opex_/_ta_ws` + `re_/_ta_ws`  + `tcl_/_ta_ws` +
                            # `tl_/_ta_ws`  + `trec_/_ta_ws` + spread_BAA_AAA"

manual_selection_FRED <- "Y~ `current_asset_/_ta_ws` + `invt_/_ta_ws` + `mv_/_ta_ws` + `ni

model_manual_FRED <- glm(as.formula(manual_selection_FRED),
                  data=default_exam %>%
                     left_join(spread_corporate_yearly, by = c("fiscal_year"="year_2")) %>%
                     filter(!company_id %in% outliers_ebit) %>%
                     select(Y, `cogs_/_ta_ws`:`trec_/_ta_ws`, spread_BAA_AAA, -`ta_/_ta_ws`
                  family="binomial")

summary(model_manual_FRED)
```

```
Call:
glm(formula = as.formula(manual_selection_FRED), family = "binomial",
    data = default_exam %>% left_join(spread_corporate_yearly,
        by = c(fiscal_year = "year_2")) %>% filter(!company_id %in%
        outliers_ebit) %>% select(Y, `cogs_/_ta_ws`:`trec_/_ta_ws`,
        spread_BAA_AAA, -`ta_/_ta_ws`))

Coefficients:
                       Estimate Std. Error z value Pr(>|z|)
(Intercept)            -6.01774    0.20296 -29.651  < 2e-16 ***
`current_asset_/_ta_ws` -0.60396    0.26514  -2.278 0.022733 *
`invt_/_ta_ws`          1.32448    0.43497   3.045 0.002327 **
`mv_/_ta_ws`           -0.58825    0.04939 -11.909  < 2e-16 ***
`ni_/_ta_ws`           -0.80414    0.15901  -5.057 4.25e-07 ***
`nsale_/_ta_ws`        -0.38714    0.12403  -3.121 0.001801 **
`opex_/_ta_ws`          0.47793    0.11835   4.038 5.39e-05 ***
`re_/_ta_ws`            0.03871    0.01750   2.211 0.027003 *
`tcl_/_ta_ws`           1.21697    0.24327   5.003 5.66e-07 ***
`tl_/_ta_ws`            1.43598    0.17138   8.379  < 2e-16 ***
`trec_/_ta_ws`         -1.76147    0.52737  -3.340 0.000838 ***
spread_BAA_AAA          0.49141    0.11922   4.122 3.76e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)
```

```
    Null deviance: 5654.9  on 59765  degrees of freedom
Residual deviance: 4794.1  on 59754  degrees of freedom
AIC: 4818.1

Number of Fisher Scoring iterations: 9
```

5.1.2.3 Downsampling

We now test down-sampling the healthy cases in the training set.

```
train <- default_exam %>%
          filter(!company_id %in% outliers_ebit, fiscal_year<=2011)

test <- default_exam %>%
          filter(!company_id %in% outliers_ebit, fiscal_year>2011)
```

For that we randomly select 25% of healthy companies in training set:

```
company_ids <- unique((train  %>% filter(default_will_occur == 0))$company_id)
set.seed(1987)
downsampled_default_exam_train <- train  %>%
  filter(company_id %in% sample(company_ids,floor(0.25*length(company_ids))))

downsampled_default_exam_train <- bind_rows(downsampled_default_exam_train,
                                            train %>% filter(default_will_occur == 1))
```

We artificially augment the default rate in the training set:

```
downsampled_default_exam_train %>%
  group_by(fiscal_year) %>%
  summarise(total_firm = n_distinct(company_id),
            default = sum(default),
            pct_default = default / total_firm)
```

```
# A tibble: 12 x 4
   fiscal_year total_firm default pct_default
         <dbl>      <int>   <dbl>       <dbl>
 1        2000       1611       7     0.00435
 2        2001       1527      10     0.00655
 3        2002       1469      17     0.0116
 4        2003       1404      29     0.0207
 5        2004       1367      46     0.0337
```

```
 6        2005       1319      40     0.0303
 7        2006       1275      51     0.04
 8        2007       1223      59     0.0482
 9        2008       1156      58     0.0502
10        2009       1091      23     0.0211
11        2010       1038      35     0.0337
12        2011       1002      25     0.0250
```

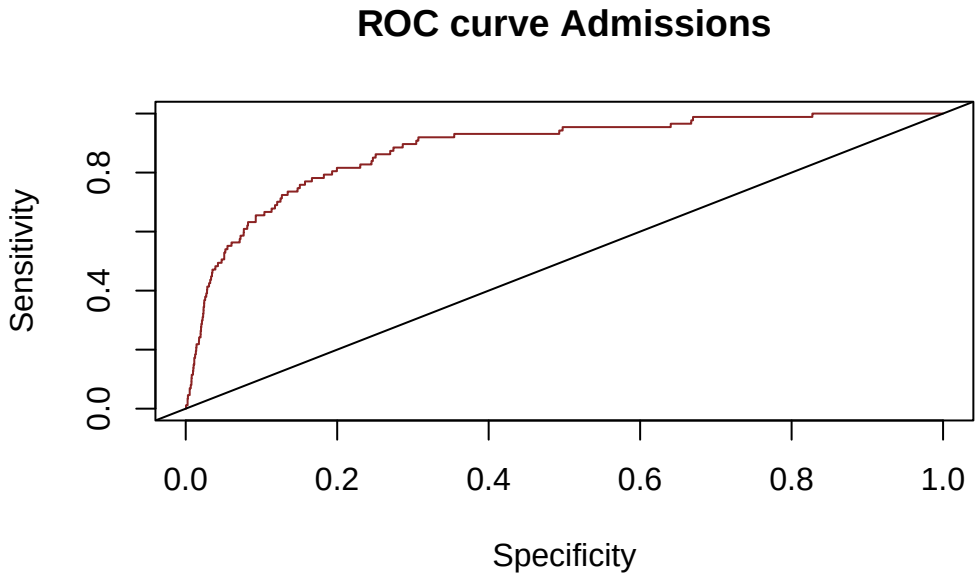
```
manual_selection <- "Y ~ `current_asset_/_ta_ws` + `invt_/_ta_ws` + `mv_/_ta_ws` + `ni_/_t

model_manual_DS <- glm(as.formula(manual_selection),
                       data=downsampled_default_exam_train %>%
                          filter(!company_id %in% outliers_ebit) %>%
                          select(Y, `cogs_/_ta_ws`:`trec_/_ta_ws`, -`ta_/_ta_ws`), family=

summary(model_manual_DS)
```

```
Call:
glm(formula = as.formula(manual_selection), family = "binomial",
    data = downsampled_default_exam_train %>% filter(!company_id %in%
        outliers_ebit) %>% select(Y, `cogs_/_ta_ws`:`trec_/_ta_ws`,
        -`ta_/_ta_ws`))

Coefficients:
                       Estimate Std. Error z value Pr(>|z|)
(Intercept)            -4.54248    0.17640 -25.751  < 2e-16 ***
`current_asset_/_ta_ws` -0.34093    0.30293  -1.125 0.260401
`invt_/_ta_ws`          1.36961    0.48973   2.797 0.005164 **
`mv_/_ta_ws`           -0.55194    0.05423 -10.179  < 2e-16 ***
`ni_/_ta_ws`           -0.64483    0.18659  -3.456 0.000548 ***
`nsale_/_ta_ws`        -0.51690    0.15336  -3.370 0.000750 ***
`opex_/_ta_ws`          0.56382    0.14767   3.818 0.000134 ***
`re_/_ta_ws`            0.02250    0.02074   1.085 0.277859
`tcl_/_ta_ws`           1.04908    0.27982   3.749 0.000177 ***
`tl_/_ta_ws`            1.48279    0.18961   7.820 5.27e-15 ***
`trec_/_ta_ws`         -1.30003    0.57064  -2.278 0.022715 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)
```

```
    Null deviance: 3714.4  on 15481  degrees of freedom
Residual deviance: 3129.1  on 15471  degrees of freedom
AIC: 3151.1

Number of Fisher Scoring iterations: 8
```

We do not modify the testing set (keeping the default distribution unchanged):

```
# plot ROC / compute AUC for the testing set
predicted_prob_manual_DS_test <- predict(model_manual_DS, newdata=test, type="response")

pred_manual_DS_test <- ROCR::prediction(predicted_prob_manual_DS_test,
                                        test$Y)

perf_manual_DS_test <- ROCR::performance(pred_manual_DS_test, measure = "tpr", x.measure =
plot(perf_manual_DS_test, main="ROC curve Admissions", xlab="Specificity",
     ylab="Sensitivity", col = "brown4")
abline(0, 1) #add a 45 degree line
```



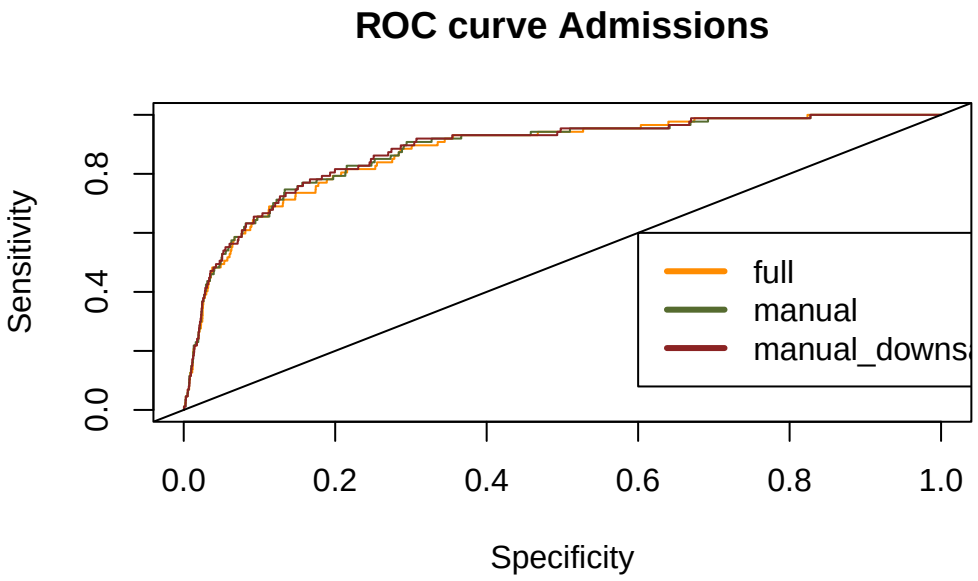

```
auc_manual_DS_test <- ROCR::performance(pred_manual_DS_test, measure = "auc")
auc_manual_DS_test <- auc_manual_DS_test@y.values[[1]]
```

```
auc_manual_DS_test
```

```
[1] 0.8809219
```

The AUC for the manually selected model is slightly better on the testing set when down-sampling.

```
plot(perf_full_test, main="ROC curve Admissions", xlab="Specificity",
     ylab="Sensitivity", col = "darkorange")
plot(perf_manual_test, add = TRUE, main="ROC curve Admissions", xlab="Specificity",
     ylab="Sensitivity", col = "darkolivegreen")
plot(perf_manual_DS_test, add = TRUE, main="ROC curve Admissions", xlab="Specificity",
     ylab="Sensitivity", col = "brown4")
abline(0, 1) #add a 45 degree line
legend(0.6,0.6,
       c('full', 'manual', 'manual_downsampled'),
       col=c("darkorange", "darkolivegreen", "brown4"),lwd=3)
```



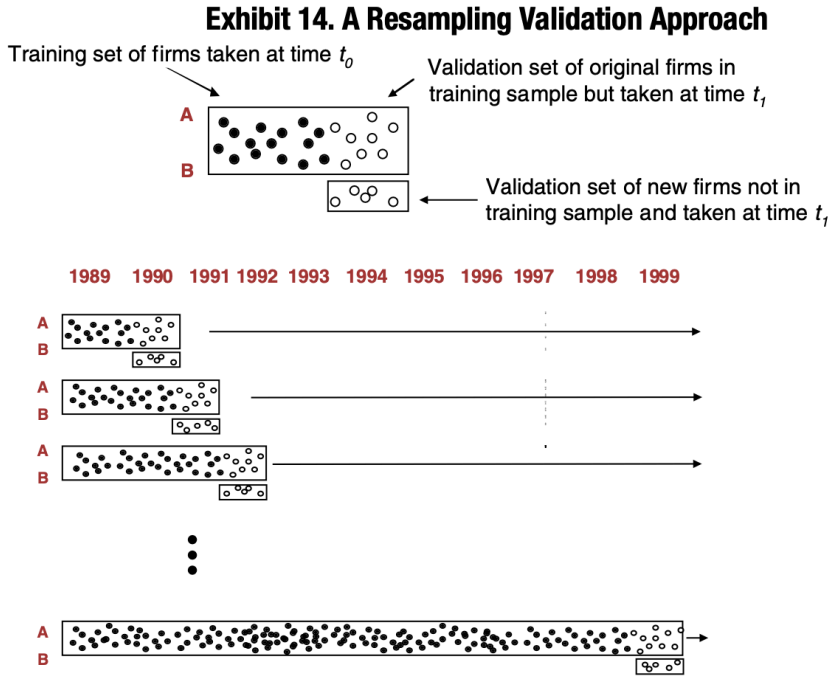
5.2 Resampling validation

5.2.1 TO DO

In this part you have to improve the hold-out approach used in the in-class part.

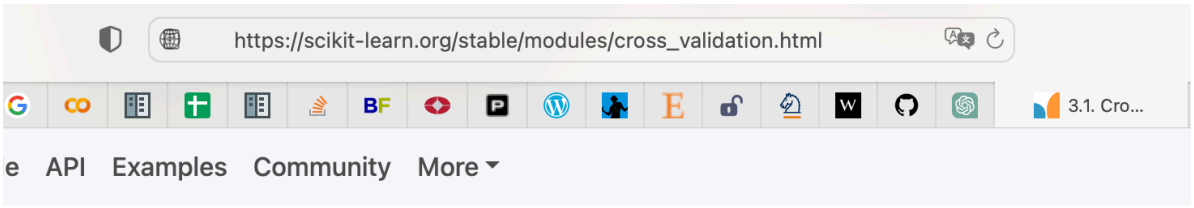
Explain why classic K-fold Cross-Validation might not be adapted to the data set.

A proposed approach is to implement something similar to the following scheme (also known as walk forward or time series split):

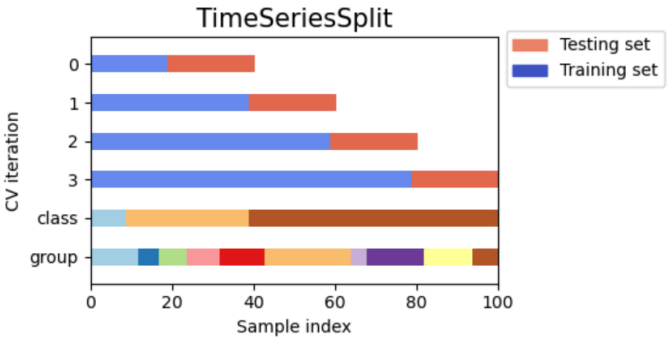


Moody's fits a model using a sample of historical data on firms and tests the model using both data on those firms one year later, and using data on new firms one year later (upper portion of exhibit). Dark circles represent in-sample data, used to estimate the model, and white circles represent testing data. We do "walk-forward testing" (bottom left) by fitting the parameters of a model using data through a particular year, and testing on data from the following year, and then inching the whole process forward one year.

which is also described here in a less specific way in the scikit-learn documentation, see below:



Here is a visualization of the cross-validation behavior.



You may adapt a little bit the length of the splits to your needs, and also your training set window may be either "fixed" (i.e. you keep all past data in training set) or "rolling" (i.e. with a fixed length or time period).

For each split, assess your models on the testing sets by computing the AUC.

Store the results in a data structure of your choice (dataframe, tibble, list) similarly to what has been shown in class (in the exercise at the very end of the second lesson 'Implement k-fold cross validation (using AUC as metric) manually'). Save the results to a `.rds` file especially if the code is long to run.

If you cannot implement a multiple split approach, use the simple hold-out approach for the next section. Nonetheless you can earn some points trying to implement something and detailing what you intended to achieve.

Once you have constructed your improved models, and your resampling method, you will need to make predictions on the validation set(s) and choose a best model within both **your_models** (**your_best_model**) and **your_improved_models** (**your_best_improved_model**).

5.2.2 Proposed solution

Similarly as before for the hold-out approach, due to the time series nature of the data, we cannot perform the usual K-Fold cross validation randomly sampling K folds/blocks.

We implement a walk forward split. Starting with a 5-year long training set (2000-2004) and using the next-year (2005) as a testing set. We increase sequentially the training set by 1-year and keep predicting the next year.

```
# Adding Altman/Ohlson variables
default_exam_split <- default_exam %>%
   mutate(wc=current_asset-tcl,
          wcta = wc / ta, # Altman's 5 ratios
          reta = re / ta,
          ebta = ebit / ta,
          mvtl = mv / tl,
          salta = nsale / ta,
          tlta = tl / ta, # Completing with Ohlson's
          clca = tcl / current_asset,
          oeneg = as.factor(if_else(tcl > ta, 1, 0)),
          nita = ni / ta,
          futl = ebit / tl)

default_exam_split  <- default_exam_split  %>%
          group_by(company_id) %>%
          arrange(company_id, fiscal_year) %>%
          mutate(ni_1 = lag(ni),
                 chin = (ni - ni_1)/(abs(ni) + abs(ni_1))) %>%
          ungroup() %>%
          mutate(across(.cols = wc:clca,
                        .fns = ~ DescTools::Winsorize(., probs = c(0.05, 0.95)))) %>%
          mutate(across(.cols = nita:futl,
                        .fns = ~ DescTools::Winsorize(., probs = c(0.05, 0.95))))

# Adding the FRED Baa - Aaa spread
default_exam_split <- default_exam_split %>%
  left_join(spread_corporate_yearly, by = c("fiscal_year"="year_2"))
```

```
    auc <- function(X,Y){
           pred <- ROCR::prediction(X, Y)
           auc <- ROCR::performance(pred, measure = "auc")
           auc <- auc@y.values[[1]]
    }
```

```
pred_list <- list()
auc_list <- list()
```

```r
i <- 1
for(split_year in 2005:2014) {
  print(split_year)
  train <- default_exam_split %>%
          filter(!company_id %in% outliers_ebit, fiscal_year < split_year)

  test <- default_exam_split %>%
          filter(!company_id %in% outliers_ebit, fiscal_year == split_year)

  test_altman <- test %>% filter(!is.na(chin))

  # fitting models on training set / predicting probabilities on testing set

  # full
  model_full_ts_split <- glm(Y~., data=train %>%
                                    select(Y, `cogs_/_ta_ws`:`trec_/_ta_ws`, -`ta_/_ta_ws`
                                  family="binomial")
  predicted_prob_full_ts_split <- predict(model_full_ts_split,
                                          newdata=test, type="response")
  auc_full_ts_split <- auc(predicted_prob_full_ts_split, test$Y)

  # summary(model_full_ts_split)

  # manual
    # manual_selection <- "Y~ `current_asset_/_ta_ws` + `invt_/_ta_ws` +
                              # `mv_/_ta_ws` + `ni_/_ta_ws` + `nsale_/_ta_ws` +
                              # `opex_/_ta_ws` + `re_/_ta_ws`  + `tcl_/_ta_ws` +
                              # `tl_/_ta_ws`  + `trec_/_ta_ws`"

  manual_selection <- "Y~ `current_asset_/_ta_ws` + `invt_/_ta_ws` + `mv_/_ta_ws` + `ni_/_

  model_manual_ts_split <- glm(as.formula(manual_selection),
                               data = train %>%
                                select(Y, `cogs_/_ta_ws`:`trec_/_ta_ws`, -`ta_/_ta_ws`),
                               family="binomial")
  predicted_prob_manual_ts_split <- predict(model_manual_ts_split,
                                            newdata=test, type="response")
  auc_manual_ts_split <- auc(predicted_prob_manual_ts_split, test$Y)

  # summary(model_manual_ts_split)
```

```r
# backward_bic
backward_bic_ts_split <- step(model_full_ts_split,
               direction='backward',
               test = 'LRT',
               scope=formula(model_full_ts_split),
               k=log(nrow(train)),
               trace = FALSE)
predicted_prob_back_bic_ts_split <- predict( backward_bic_ts_split,
                                             newdata=test, type="response")
auc_back_bic_ts_split <- auc(predicted_prob_back_bic_ts_split, test$Y)

# summary(backward_bic_ts_split)

# lasso
set.seed(1)
model_lasso_ts_split <- glmnetUtils::cv.glmnet(Y ~ .,
                          data=train %>%
                            select(Y, `cogs_/_ta_ws`:`trec_/_ta_ws`, -`ta_/_ta_ws`),
                          family="binomial",
                          alpha=1,
                          standardize = TRUE,
                          type.measure = "auc")
predicted_prob_lasso_ts_split <- as.vector(predict(model_lasso_ts_split,
                                                   newdata=test, type="response"))
auc_lasso_ts_split <- auc(predicted_prob_lasso_ts_split, test$Y)

# lasso_ts_split_coeffs_1se <- coef(model_lasso_ts_split, s = "lambda.1se")
# tibble(name = lasso_ts_split_coeffs_1se@Dimnames[[1]][lasso_ts_split_coeffs_1se@i + 1]
#        coefficient = lasso_ts_split_coeffs_1se@x)


# altman/ohlson
altman_ohlson_var <- "Y ~ reta + mvtl + salta + tlta + clca + oeneg + nita + futl + chin

model_altman_ohlson_ts_split <- glm(as.formula(altman_ohlson_var),
                            data=train,
                           family="binomial")

predicted_prob_altman_ohlson_ts_split <- predict( model_altman_ohlson_ts_split,
                                                  newdata= test_altman,
                                                  type="response")
```

```r
auc_altman_ohlson_ts_split <- auc(predicted_prob_altman_ohlson_ts_split, test_altman$Y)

# summary( model_altman_ohlson_ts_split)

# FRED
  # manual_selection_FRED <- "Y~ `current_asset_/_ta_ws` + `invt_/_ta_ws` +
                                # `mv_/_ta_ws` + `ni_/_ta_ws` + `nsale_/_ta_ws` +
                                # `opex_/_ta_ws` + `re_/_ta_ws`  + `tcl_/_ta_ws` +
                                # `tl_/_ta_ws`  + `trec_/_ta_ws` + spread_BAA_AAA"


manual_selection_FRED <- "Y~ `current_asset_/_ta_ws` + `invt_/_ta_ws` + `mv_/_ta_ws` + `

model_FRED_ts_split <- glm(as.formula(manual_selection_FRED), data=train, family="binomi

predicted_prob_FRED_ts_split <- predict( model_FRED_ts_split,
                                                 newdata= test,
                                                 type="response")
auc_FRED_ts_split <- auc(predicted_prob_FRED_ts_split, test$Y)

# summary(model_FRED_ts_split)

# downsample
company_ids <- unique((train %>% filter(default_will_occur == 0))$company_id)
set.seed(1987)
downsampled_default_exam_train <- train  %>%
  filter(company_id %in% sample(company_ids,floor(0.25*length(company_ids))))

downsampled_default_exam_train <- bind_rows(downsampled_default_exam_train,
                                            train %>% filter(default_will_occur == 1))

  # manual_selection <- "Y ~ `current_asset_/_ta_ws` + `invt_/_ta_ws` +
                              # `mv_/_ta_ws` + `ni_/_ta_ws` + `nsale_/_ta_ws` +
                              # `opex_/_ta_ws` + `re_/_ta_ws`  + `tcl_/_ta_ws` +
                              # `tl_/_ta_ws`  + `trec_/_ta_ws`"

manual_selection <- "Y ~ `current_asset_/_ta_ws` + `invt_/_ta_ws` + `mv_/_ta_ws` + `ni_/

 model_manual_DS_ts_split <- glm(as.formula(manual_selection),
                                 data=downsampled_default_exam_train %>%
                                   filter(!company_id %in% outliers_ebit) %>%
```

```
                                    select(Y, `cogs_/_ta_ws`:`trec_/_ta_ws`, -`ta_/_ta_ws
                                  family="binomial")

    predicted_prob_manual_DS_ts_split <- predict( model_manual_DS_ts_split,
                                                  newdata= test,
                                                  type="response")
    auc_manual_DS_ts_split <- auc(predicted_prob_manual_DS_ts_split, test$Y)
    # summary(model_manual_DS_ts_split)

    # storing results
    row_prob <- tibble_row(year = split_year,
                                       full = list(predicted_prob_full_ts_split),
                                       manual = list(predicted_prob_manual_ts_split),
                                       back_bic = list(predicted_prob_back_bic_ts_split)
                                       lasso = list(predicted_prob_lasso_ts_split),
                                       altman = list(predicted_prob_altman_ohlson_ts_spl
                                       fred = list(predicted_prob_FRED_ts_split),
                                       manual_DS = list(predicted_prob_manual_DS_ts_spli
                                       test_Y = list(test$Y),
                                       test_Y_altman = list(test_altman$Y))

   row_auc <- tibble_row(year = split_year,
                                      full = auc_full_ts_split,
                                      manual = auc_manual_ts_split,
                                      back_bic = auc_back_bic_ts_split,
                                      lasso = auc_lasso_ts_split,
                                      altman = auc_altman_ohlson_ts_split,
                                      fred = auc_FRED_ts_split,
                                      manual_DS = auc_manual_DS_ts_split)

  pred_list[[i]] <- row_prob
  auc_list[[i]] <- row_auc
  i <- i + 1

 }
```

```
[1] 2005
[1] 2006
[1] 2007
[1] 2008
[1] 2009
[1] 2010
```

```
[1] 2011
[1] 2012
[1] 2013
[1] 2014
```

```
pred_ts_split <- bind_rows(pred_list)
auc_ts_split <- bind_rows(auc_list) %>%
        mutate(year = as.character(year)) %>%
        add_row(year = "Total", summarise(., across(where(is.numeric), mean)))
```

```
# saveRDS(pred_ts_split,"pred_ts_split.rds")
# saveRDS(auc_ts_split,"auc_ts_split.rds")
```

```
pred_ts_split <- readRDS("pred_ts_split.rds")
auc_ts_split <- readRDS("auc_ts_split.rds")
```

```
auc_ts_split
```

```
# A tibble: 11 x 8
   year   full manual back_bic lasso altman  fred manual_DS
   <chr> <dbl>  <dbl>    <dbl> <dbl>  <dbl> <dbl>     <dbl>
 1 2005  0.813  0.814    0.781 0.777  0.800 0.823     0.818
 2 2006  0.846  0.850    0.840 0.833  0.845 0.855     0.851
 3 2007  0.827  0.814    0.832 0.814  0.822 0.823     0.819
 4 2008  0.893  0.901    0.899 0.894  0.895 0.904     0.903
 5 2009  0.909  0.907    0.898 0.897  0.891 0.913     0.917
 6 2010  0.869  0.869    0.838 0.861  0.858 0.871     0.871
 7 2011  0.915  0.921    0.912 0.917  0.934 0.920     0.920
 8 2012  0.948  0.950    0.944 0.944  0.943 0.951     0.951
 9 2013  0.832  0.832    0.837 0.810  0.823 0.832     0.834
10 2014  0.876  0.878    0.883 0.861  0.857 0.879     0.881
11 Total 0.873  0.874    0.867 0.861  0.867 0.877     0.877
```

```
auc_ts_split[11,7] %>% pull(fred)
```

```
[1] 0.8770205
```

```
auc_ts_split[11,8] %>% pull(manual_DS)
```

```
[1] 0.8765522
```

5.3 Prediction on validation test / final model selection

5.3.1 TO DO

At the end of the exam I have released a validation test (file `default_dataset_validation.csv`):

```
# # Uncomment to read the validation set,
# # ensuring the file default_dataset_validation.csv is in the data folder
default_dataset_validation <- read_csv('data/default_dataset_validation.csv')
```

Once you have selected your best models, you will conclude by making predictions on this validation set and choosing a final model (**final_model**) within (**full_model**, **your_best_model**, **your_best_improved_model**).

5.3.2 Proposed solution

```
# # Uncomment to read the validation set,
# # ensuring the file default_dataset_validation.csv is in the data folder
default_exam_validation <- default_dataset_validation %>%
  select(-ebitda,-tr,- fiscal_year_default) %>%
  mutate(Y = default,
         Y_alt = default_will_occur) %>%
    # scaling by `ta`
  mutate(across(.cols = cogs:trec,
                .fns = ~ . / ta,
                .names = "{.col}_/_ta")) %>%
  # scaling by `ta` and winsorizing
  mutate(across(.cols = cogs:trec,
                .fns = ~ DescTools::Winsorize(. / ta, probs = c(0.05, 0.95)),
                .names = "{.col}_/_ta_ws"))
```

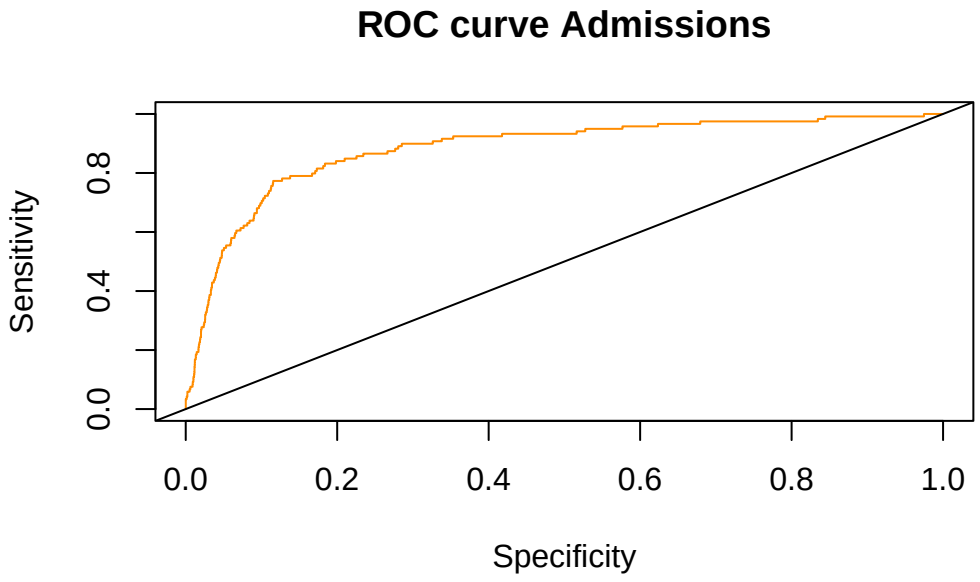
```
outliers_ebit_validation <- default_exam_validation %>%
  filter(`ebit_/_ta` < -10, default_will_occur == 0) %>%
  distinct(company_id) %>%
  pull(company_id)
```

```
# Adding Altman/Ohlson variables
default_exam_validation <- default_exam_validation %>%
   mutate(wc=current_asset-tcl,
          wcta = wc / ta, # Altman's 5 ratios
          reta = re / ta,
          ebta = ebit / ta,
          mvtl = mv / tl,
          salta = nsale / ta,
          tlta = tl / ta, # Completing with Ohlson's
          clca = tcl / current_asset,
          oeneg = as.factor(if_else(tcl > ta, 1, 0)),
          nita = ni / ta,
          futl = ebit / tl) %>%
  filter(!company_id %in% c(outliers_ebit_validation, outliers_ebit))


default_exam_validation  <- default_exam_validation  %>%
                  group_by(company_id) %>%
                  arrange(company_id, fiscal_year) %>%
                  mutate(ni_1 = lag(ni),
                         chin = (ni - ni_1)/(abs(ni) + abs(ni_1))) %>%
                  ungroup() %>%
                  mutate(across(.cols = wc:clca,
                                .fns = ~ DescTools::Winsorize(., probs = c(0.05, 0.95)
                  mutate(across(.cols = nita:futl,
                                .fns = ~ DescTools::Winsorize(., probs = c(0.05, 0.95)

# Adding the FRED Baa - Aaa spread
default_exam_validation <- default_exam_validation %>%
  left_join(spread_corporate_yearly, by = c("fiscal_year"="year_2"))
```

We retrain the full model, the manually selected model and the FRED model on 2000-2014 and assess on the validation set:

```
# full
model_full_final <- glm(Y~., data=default_exam_split%>%
                                  select(Y, `cogs_/_ta_ws`:`trec_/_ta_ws`, -`ta_/_ta_ws`),
                                family="binomial")
predicted_prob_full_final <- predict(model_full_final,
                                     newdata=default_exam_validation, type="response")
```

```
# auc_full_final <- auc(predicted_prob_full_final, default_exam_validation$Y)
# auc_full_final

pred_full_final <- ROCR::prediction(predicted_prob_full_final,
                                    default_exam_validation$Y)

perf_full_final <- ROCR::performance(pred_full_final, measure = "tpr", x.measure = "fpr")
plot(perf_full_final, main="ROC curve Admissions", xlab="Specificity",
     ylab="Sensitivity", col = "darkorange")
abline(0, 1) #add a 45 degree line
```

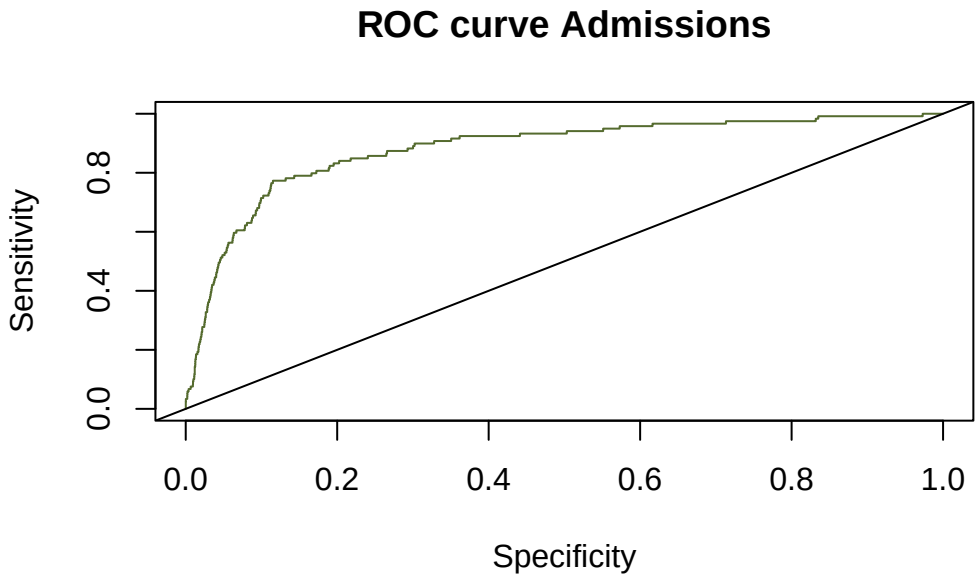



```
auc_full_final <- ROCR::performance(pred_full_final, measure = "auc")
auc_full_final <- auc_full_final@y.values[[1]]
auc_full_final
```

```
[1] 0.8822401
```

```
# manual
# manual_selection <- "Y~ `current_asset_/_ta_ws` + `invt_/_ta_ws` +
                          # `mv_/_ta_ws` + `ni_/_ta_ws` + `nsale_/_ta_ws` +
```

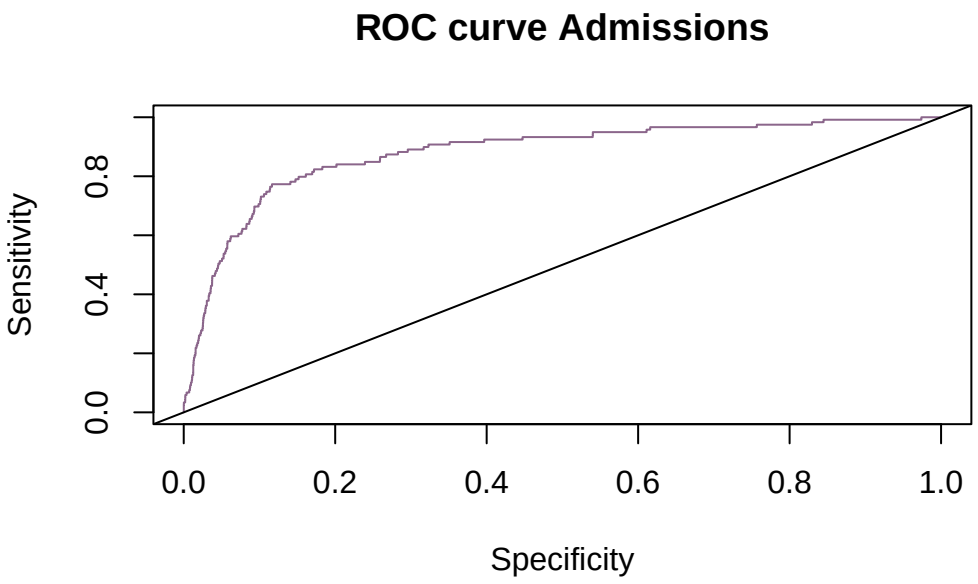
```
                          # `opex_/_ta_ws` + `re_/_ta_ws`  + `tcl_/_ta_ws` +
                          # `tl_/_ta_ws`  + `trec_/_ta_ws`"

manual_selection <- "Y~ `current_asset_/_ta_ws` + `invt_/_ta_ws` + `mv_/_ta_ws` + `ni_/_ta
model_manual_final <- glm(as.formula(manual_selection),
                          data=default_exam_split%>%
                                  select(Y, `cogs_/_ta_ws`:`trec_/_ta_ws`, -`ta_/_ta_ws`),
                                family="binomial")

predicted_prob_manual_final <- predict(model_manual_final,
                                       newdata=default_exam_validation,
                                       type="response")
# auc_manual_final <- auc(predicted_prob_manual_final, default_exam_validation$Y)
# auc_manual_final

pred_manual_final <- ROCR::prediction(predicted_prob_manual_final,
                                      default_exam_validation$Y)

perf_manual_final <- ROCR::performance(pred_manual_final, measure = "tpr", x.measure = "fp
plot(perf_manual_final, main="ROC curve Admissions", xlab="Specificity",
     ylab="Sensitivity", col = "darkolivegreen")
abline(0, 1) #add a 45 degree line
```



```
auc_manual_final <- ROCR::performance(pred_manual_final, measure = "auc")
auc_manual_final <- auc_manual_final@y.values[[1]]
auc_manual_final
```

```
[1] 0.8800922
```

```
# manual_selection_FRED <- "Y~ `current_asset_/_ta_ws` + `invt_/_ta_ws` +
                            # `mv_/_ta_ws` + `ni_/_ta_ws` + `nsale_/_ta_ws` +
                            # `opex_/_ta_ws` + `re_/_ta_ws`  + `tcl_/_ta_ws` +
                            # `tl_/_ta_ws`  + `trec_/_ta_ws` + spread_BAA_AAA"

manual_selection_FRED <- "Y~ `current_asset_/_ta_ws` + `invt_/_ta_ws` + `mv_/_ta_ws` + `ni

model_FRED_final <- glm(as.formula(manual_selection_FRED),
                        data=default_exam_split,
                        family="binomial")

predicted_prob_FRED_final <- predict(model_FRED_final,
                                     newdata=default_exam_validation,
                                     type="response")

# auc_FRED_final <- auc(predicted_prob_FRED_final, default_exam_validation$Y)
# auc_FRED_final

pred_FRED_final <- ROCR::prediction(predicted_prob_FRED_final,
                                    default_exam_validation$Y)

perf_FRED_final <- ROCR::performance(pred_FRED_final, measure = "tpr", x.measure = "fpr")
plot(perf_FRED_final, main="ROC curve Admissions", xlab="Specificity",
     ylab="Sensitivity", col = "plum4")
abline(0, 1) #add a 45 degree line
```

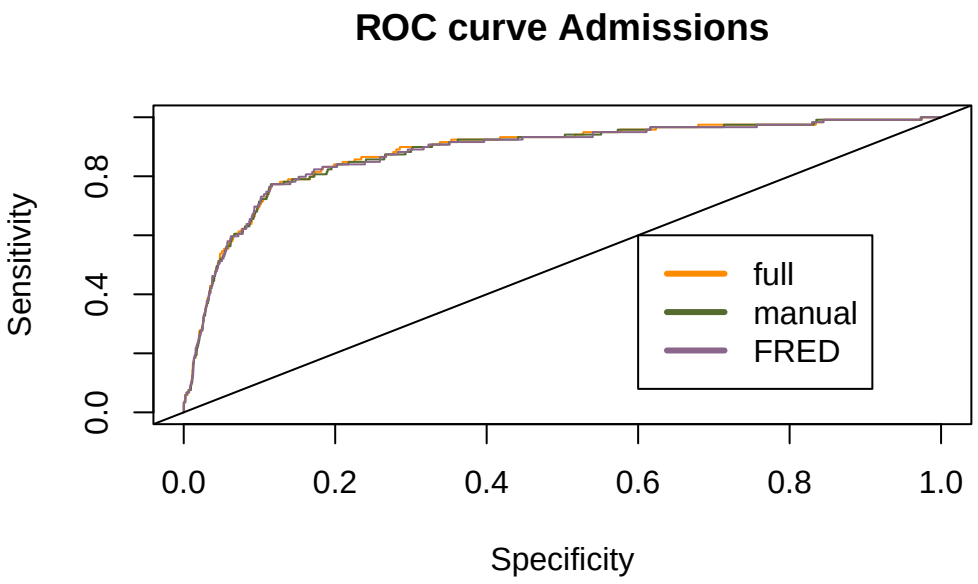



```
auc_FRED_final <- ROCR::performance(pred_FRED_final, measure = "auc")
auc_FRED_final <- auc_FRED_final@y.values[[1]]
auc_FRED_final
```

```
[1] 0.8794484
```

We display the three final models ROC curves:

```
plot(perf_full_final, main="ROC curve Admissions", xlab="Specificity",
     ylab="Sensitivity", col = "darkorange")
plot(perf_manual_final, add = TRUE, main="ROC curve Admissions", xlab="Specificity",
     ylab="Sensitivity", col = "darkolivegreen")
plot(perf_FRED_final, add = TRUE, main="ROC curve Admissions", xlab="Specificity",
     ylab="Sensitivity", col = "plum4")
abline(0, 1) #add a 45 degree line
legend(0.6,0.6,
       c('full', 'manual', 'FRED'),
       col=c("darkorange", "darkolivegreen", "plum4"),lwd=3)
```



Altman, Edward I. 1968. "FINANCIAL RATIOS, DISCRIMINANT ANALYSIS AND THE PREDICTION OF CORPORATE BANKRUPTCY." *The Journal of Finance* 23 (4): 589–609. https://doi.org/10.1111/j.1540-6261.1968.tb00843.x.

Ohlson, James A. 1980. "Financial Ratios and the Probabilistic Prediction of Bankruptcy." *Journal of Accounting Research* 18 (1): 109. https://doi.org/10.2307/2490395.