Exam - In class

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 at a given 1-year horizon, using annual financial statement data reported by companies and predictors derived from market data.

Financial statement data relating to these companies is available in a data set and has been scaled using one of the item (the Total Assets `at` from balance sheet) to ease the analysis.

The data set has been simulated.

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

Your goal is to predict corporate default with a 1-year horizon.

We briefly describe the columns of the data set (Rows: 10,000 Columns: 25):

- `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;

First a set of financial statements items mainly coming from companies balance sheets or income statements :

- `at`: Total Assets (see here).

Variables with `_at` suffix have been scaled/divided using `at`:

- `seq_at`, where `seq`: (Total Parent) Stockholders' Equity (see here).
- `lt_at`, where `lt`: Total Liabilities (see here); we have `lt` = `at` - `seq`.
- `lct_at`, where `lct`: Current Liabilities (see here).
- `che_at`, where `che`: Cash and Equivalents (see here).
- `rect_at`, where `rect`: (Accounts) Receivables (see here).
- `invt_at`, where `invt`: Inventories (see here).
- `act_at`, where `act`: Current Assets (see here).
- `ppent_at`, where `ppent`: Property, Plant, and Equipment (see here).
- `intan_at`, where `intan`: Intangible assets that lacks physical substance: patents, copyright, franchises, trademarks, … as well as any form of digital asset such as software and data. (see here)
- `sale_at`, where `sale`: Net Sales (see here).
- `cogs_at`, where `cogs`: Cost of Goods Sold (see here).
- `xsga_at`, where `xsga` : Selling, general, and administrative expenses (SG&A) are overhead expenses that keep a business running but are not directly tied to producing goods or services (see here).
- `ebitda_at`, where `ebitda`: Earnings Before Interest, Taxes, Depreciation and Amortization (see here); `ebitda` = `ebit` + `dp` = `sale` - `cogs` - `xsga`.
- `dp_at`, where `dp`: Depreciation and Amortization (see here).
- `ebit_at`, where `ebit`: Earnings Before Interest and Taxes (see here).
- `xint_at`, where `xint`: Interest Expense (see here).
- `ni_at`, where `ni`: Net Income (see here or here).
- `re_at`, where `re`: Retained Earnings (see here).
- `capx_at`, where `capx`: Capital Expenditures (see here).

- `dv_at`, where `dv` : Dividends are payments made by companies to their shareholders based on the number of shares they own. Dividends are usually paid when a company has excess cash that is not being reinvested into the company (see here).

Then a set of predictors derived from market data:

- `mktval_at`, where `mktval`: Market Value of Equity (see here).
- `sigma_E`: Historical Volatility of Equity (see here).
- `DtD`: Distance-to-default (DtD) is a measure of default risk derived from observed stock prices and firm capital structure using the structural credit risk model of Merton (1974).

You can also derive:

- `wcap`: Working Capital (see here) as `wcap` = `act` - `lct`.

You can load the data set and briefly inspect it (you might have to check `getwd()` and set up your working directory `setwd()` when using a .R script):

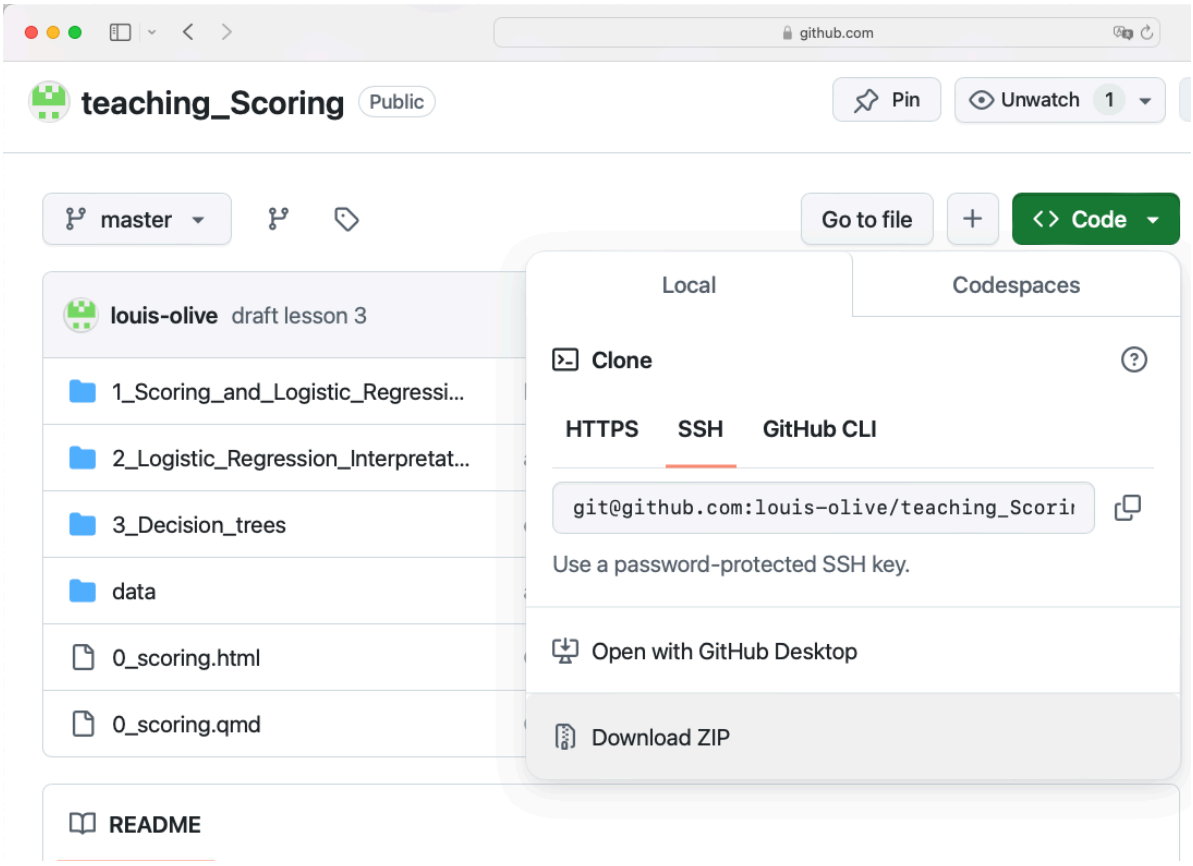
```
# loading data and inspecting it:
financial_data_scaled_exam <- readRDS('data/financial_data_scaled_exam.rds')
glimpse(financial_data_scaled_exam)
```

```
Rows: 10,000
Columns: 25
$ default   <int> 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, ~
$ at        <dbl> 670.07, 356.75, 278.03, 302.36, 616.59, 513.31, 795.32, 843.~
$ seq_at    <dbl> 0.14805127, 0.18285001, 0.12375207, 0.45742250, 0.19159025, ~
$ lt_at     <dbl> 0.8519487, 0.8171500, 0.8762479, 0.5425775, 0.8084098, 1.106~
$ lct_at    <dbl> 0.2466424, 0.2536894, 0.3748235, 0.1810911, 0.3049979, 0.354~
$ che_at    <dbl> 6.256543e-02, 1.792342e-02, 1.864240e-02, 2.691217e-02, 1.67~
$ rect_at   <dbl> 0.1751905, 0.2452370, 0.1296085, 0.1953389, 0.1198494, 0.187~
$ invt_at   <dbl> 0.18804674, 0.09854228, 0.07792034, 0.11191998, 0.14706389, ~
$ act_at    <dbl> 0.4730485, 0.4091822, 0.2752317, 0.3871283, 0.3139364, 0.372~
$ ppent_at  <dbl> 0.3650963, 0.3430971, 0.2345482, 0.1776721, 0.3000606, 0.232~
$ intan_at  <dbl> 0.04048263, 0.07779834, 0.16206834, 0.12871191, 0.02226753, ~
$ sale_at   <dbl> 1.4439853, 1.3237143, 0.4489586, 0.6927880, 1.2027942, 2.024~
$ cogs_at   <dbl> 0.7994676, 1.0014681, 0.2405407, 0.3640148, 0.8010651, 1.491~
$ xsga_at   <dbl> 0.16540193, 0.30587182, 0.05667495, 0.09020497, 0.15079478, ~
$ ebitda_at <dbl> 0.47911568, 0.01637433, 0.15174293, 0.23856832, 0.25093432, ~
$ dp_at     <dbl> 0.024923053, 0.014050164, 0.015513681, 0.005485946, 0.009309~
$ ebit_at   <dbl> 0.454192629, 0.002324168, 0.136229247, 0.233082378, 0.241625~
$ xint_at   <dbl> 0.005421302, 0.036152035, 0.032189441, 0.012881642, 0.021306~
```

```
$ ni_at     <dbl> 0.312598002, -0.083935874, 0.086779217, 0.166472225, 0.20814~
$ re_at     <dbl> 0.07428996, 0.09160539, 0.06208997, 0.22917223, 0.09612517, ~
$ capx_at   <dbl> 0.0178750554, 0.0113602274, 0.0140406356, 0.0080209685, 0.01~
$ dv_at     <dbl> 0.110157220, 0.000000000, 0.002293750, 0.019453022, 0.073742~
$ mktval_at <dbl> 0.28220932, 0.54875964, 0.72930979, 6.80453764, 1.35467653, ~
$ sigma_E   <dbl> 0.1021948, 0.1836031, 0.1805529, 0.3314238, 0.4640990, 0.222~
$ DtD       <dbl> 5.812598, 5.101043, 7.026926, 9.208506, 3.554835, 2.069808, ~
```

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, copy ChatGPT Q/A in appendix etc). You can reuse code and materials from the lessons hosted here: https://github.com/louis-olive/teaching_Scoring/, click on Code/Download ZIP for the last version:



- The first part of the exam is to be performed in-class (**TO DO in-class** in the exam document) on Monday, October 13. The in-class exam will last 90 minutes (08:00-09:30).
- You must use the R programming language, preferably with RStudio. Your "code for analysis" document (containing both R code and text/comments answering exam questions) 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, in that case comment directly in your code using #).
- Time permitting for the in-class exam, but it will ease my reading/correction: you can render your quarto/Rmarkdown to .html + folder or .pdf (it is up to you/optional for the in-class but it can be a way to check if your code works as intended). It will be required for the at-home part. You might check that rendering works for you before the exam if you intend to provide a rendered file.
- The "code for analysis" document (and optional rendered report) general readability especially for the take home 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, no GenAI for the comments please, I prefer your own simple/concise sentences, what is important is the content and insights, not a pompous style).
- The .qmd/.Rmd/.R "code for analysis" file should run without errors (if something is not working as you wish, comment out the code, and explain your intents). **Final documents should be posted before 09:30 on 13 October 2025 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 5-10 minutes before the end of the exam to check your .qmd/.Rmd/.R file is running and you have a readable "code for analysis" document (and optional rendered 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 Monday 27 October 2025 08:00. A document will be provided before next class.
- Regarding the grading:
  - 40% of the total points for the in-class part, 60% for the take-home part

- in-class data analysis (40%): 20% / 10% / 10% of the total points for each parts: "Scaled by Total Assets" ratios / "Altman" ratios / Models assessment using holdout
- in-class bonus exercise (10%): selecting lasso penalization parameter using `glmnet` built-in cross-validation procedure.

You have to perform the tasks between each **TO DO in-class - BEGIN** / **TO DO in-class - END** in your "code for analysis" document to get full credit:

3 "Scaled by Total Assets" ratios (20% total points)

TO DO in-class - BEGIN

- First explore the data set using variables `default` and `seq_at:DtD` (ie staring with `seq_at`, ending with `DtD`, using every variables in between), for example:
  - providing descriptive statistics,
  - showing correlated features,
  - showing/plotting individual features "interaction" with the response variable

  Visualizations are expected.

  After this step, you can remove observations (rows) from data if justified.

  You might also need to remove some variables/predictors is perfectly co-linear or any other justified reason.
- Secondly fit a "full" logistic regression model (**full_model**) on the data set using variables `default` and kept features from first exploration step among `seq_at:DtD`.
- Then use stepwise logistic regression (forward or backward, using the penalization/criterion of your choice, forward might be quicker to run) (**stepwise_model**) on the data set using variables `default` and kept features from exploration step among `seq_at:DtD`.
- Compare the **full_model** and **stepwise_model** using a Likelihood Ratio Test (LRT), ie test if **full_model** fits significantly better than the **stepwise_model** (if you cannot fit the stepwise model, select manually a subset of variables from the full model instead). [disclaimer: the comparison between the two models is only done to assess your skills in the exam, in general you should select models using resampling methods or look at post-selection inference methods in the literature that are beyond the scope of the course]
- Compare predicted probabilities to observed probabilities for the **stepwise_model** using a Calibration Plot.

TO DO in-class - END

4 "Altman" ratios (10% total points)

TO DO in-class - BEGIN

- First, using the data set at hand, create/select predictors closest as possible to Altman's Z-Score components (in case of doubt, go back to the introduction of the file, and get the closest variable, for example Book value of total debt might be close to Total Liabilities). In total you need to use 5 predictors X1 to X5 as defined below:

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

- Secondly, fit a logistic regression model **altman_model** using only these predictors, then:
  - give an interpretation for the coefficient `X1 = Working capital / Total Assets`
  - assess the "significance" of `X1 = Working capital / Total Assets` coefficient using Wald or Likelihood Ratio Test (LRT)
  - give a confidence interval for `X1 = Working capital / Total Assets`
- Assess an extended **altman_model_dtd** adding the variable `DtD` from data set to the model specification, compare the two nested models (ie with **altman_model**) using a LRT Test, decide to keep/remove the `DtD` variable given the result?
- Assess a diminished **altman_model_dtd_wo_x4** removing the variable `X4` from the **altman_model_dtd** model specification, compare the two nested models (ie with **altman_model_dtd**) using a LRT Test, decide to keep/remove the `X4` variable given the result?
- Depending on your answers, you might decide to carry on with an updated **altman_model** instead of the **altman_model** from first question.

TO DO in-class - END

5 Models assessment (10% total points)

TO DO in-class - BEGIN

- First use the hold-out technique to assess the preceding models **full_model** and **altman_model** (in particular you have to split the data set into training and testing set of your choice), time-permitting you can also re-run the **stepwise_model** on the training set.
- If you fail this first step you can proceed for the ROC curve part in the next question using the full data set.
- Then plot the ROC Curves assessed on the testing set and compare the AUC of **full_model**, **altman_model** (optionally **stepwise_model**)
- Conclude on the best model in terms of AUC given you hold-out split.
- What should you do to "robustify" your model assessment?

TO DO in-class - END

6 Bonus exercise / Lasso (10% total points)

TO DO in-class as a BONUS exercise or to finish at-home

- Starting with the same variables as in **full_model** and using function `glmnet::cv.glmnet` or more conveniently `glmnetUtils::cv.glmnet` as shown in lesson 3 select a "best value" for the lasso parameter "lambda" giving a penalized model **lasso_model**.
- Compare to the others models using a RCO cruve and the split/holdout from just before.

Functions `glmnet::cv.glmnet` / `glmnetUtils::cv.glmnet` fit the lasso path (ie multiple penalized models for multiple values of lambda) with the selection of the best lambda by K-Fold Cross-Validation (by default 10-fold) using a given criterion such as the AUC (`type.measure = "auc"`).

The functions computes two optimal values: `lambda.min` is the value of lambda that gives minimum mean Cross-validated criterion; `lambda.1se` is the value of lambda that gives the most regularized model (highest lambda) such that the Cross-validated criterion for this lambda is within one standard error of the minimum, it favours penalization/parsimony versus `lambda.min` (see here).

(example usage `model_cv_glmnet <- glmnetUtils::cv.glmnet(Y ~ ., data= YOUR_DATA, family="binomial", alpha=1, type.measure = "auc")`).

By default the `predict` function for a `cv.glmnet` model uses `lambda.1se`, but we can specify any value of lambda, in particular `lambda.min`, also note that the predict function outputs a `matrix` (`predict` functions in `R` outputs a vector in general) that might need to be converted to a vector (for latter use with `ROCR` package).

(example usage `predicted_proba <- as.vector(predict(model_cv_glmnet, newdata = YOUR_(NEW)_DATA, s = model_cv_glmnet$lambda.1se, type = "response")`)