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.

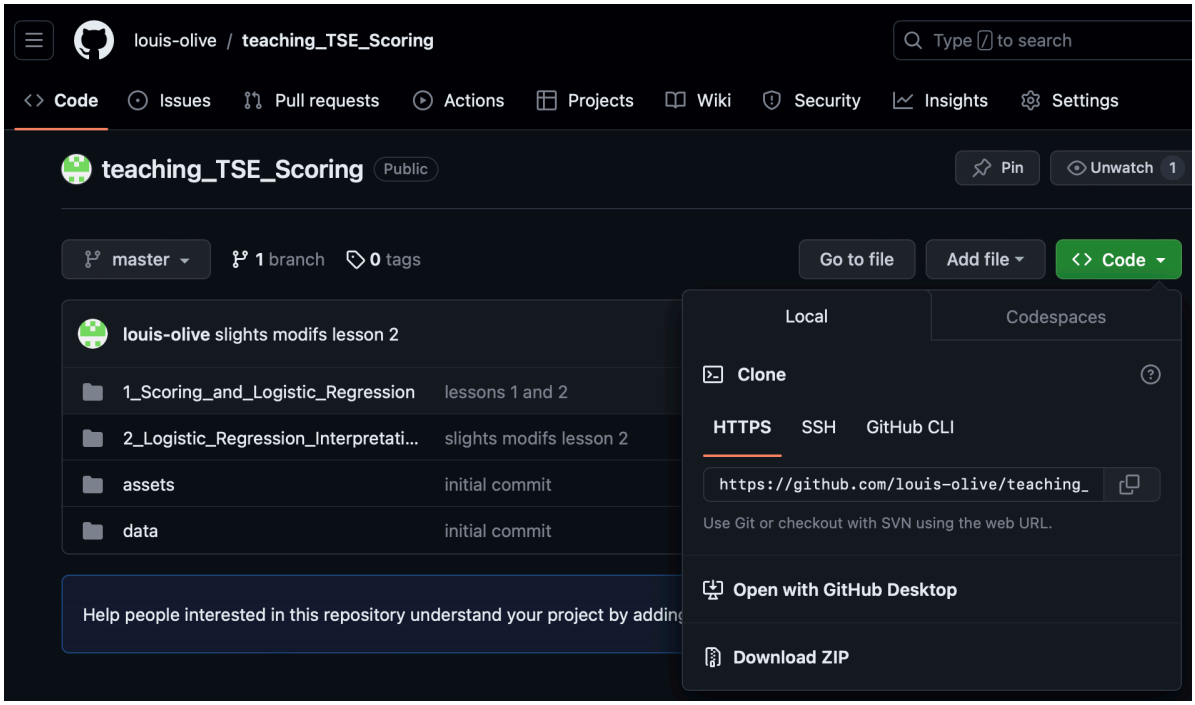
Below we load the data set and display its structure:

```
# 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, 2, 2,~
$ default             <dbl> 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,~
$ 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 4 October 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 (as well as completing the in class analysis) the deadline is 10 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)

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.

4 Model building and assessment

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

4.1 First step (10% total points)

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).

4.2 Second step (10% total points)

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.

4.3 Third step (5% total points)

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.

5 Model improvement and prediction using a validation set

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

5.1 Feature engineering

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 (for example creating the ratio of some variables), for example a natural idea is to scale variables by the total asset as in Altman Z-score, you can also winsorize variables if needed.
- 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 two different improvements to get full score (i.e. two models). You may combine these improvements into a third 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.

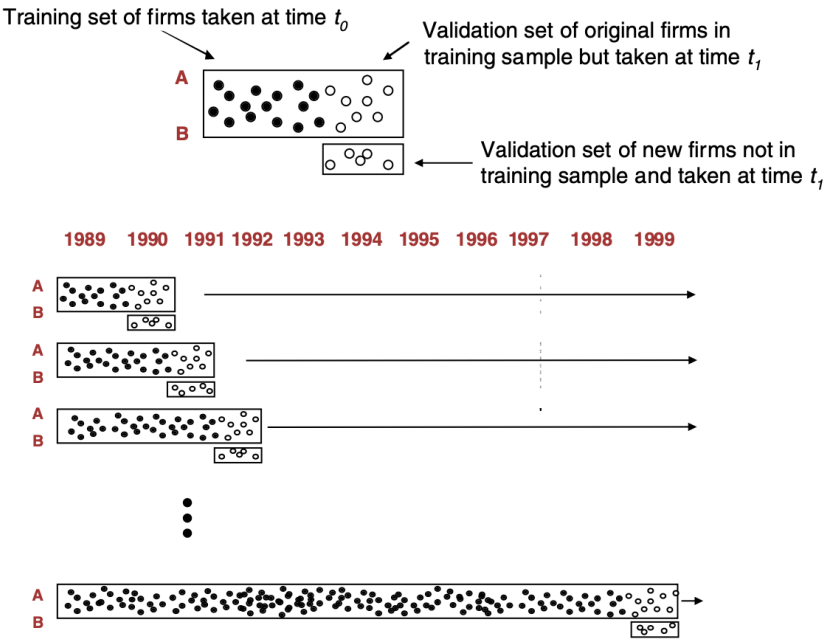
5.2 Resampling validation

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.

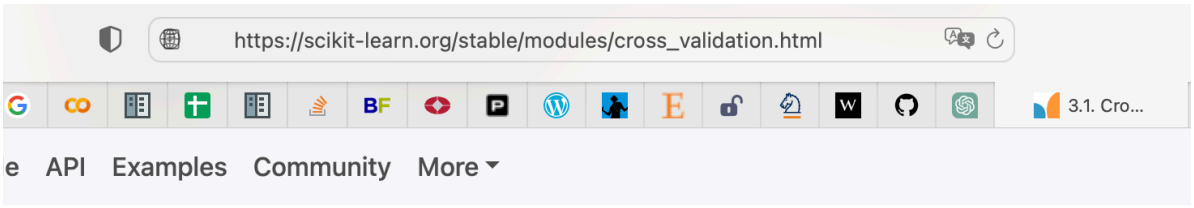
A proposed approach is to implement something similar to the following scheme (also known as walk forward or time series split):

Exhibit 14. A Resampling Validation Approach

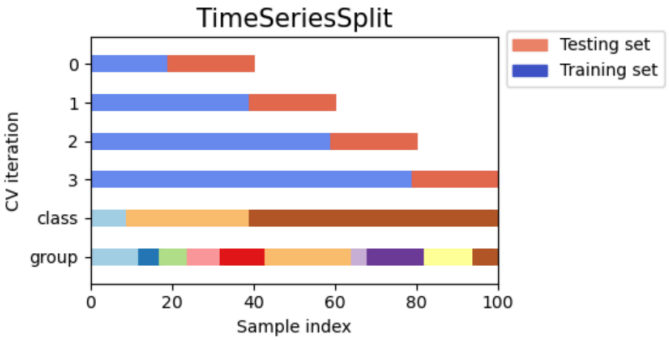


Moody's fits a model using a sample of historical data on firms and tests the model using both data on those firms one year later, and using data on new firms one year later (upper portion of exhibit). Dark circles represent in-sample data, used to estimate the model, and white circles represent testing 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

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**).

5.3 Prediction on validation test / final model selection

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')
# glimpse(default_dataset_validation)
```

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**).