

Corporate default prediction

Statistics for Data science - AEM University of Brescia

Mateusz Dadej

June 28, 2022

Research questions

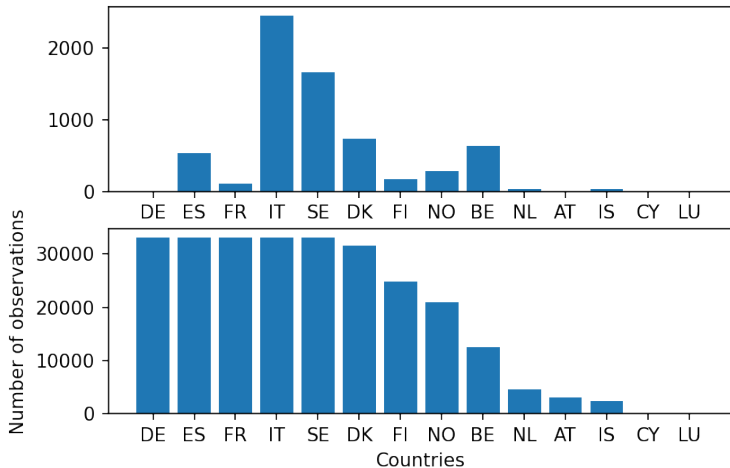
- What are the consequences of applying different resampling methods?
Is the more complex the better?
- What's the most efficient way to make a model of a corporate probability of default?

Overview of data: Basic information

- Data is sourced from Orbis - Database with private companies
- Geographical location of companies are developed countries in Europe
- observations were filtered with following criteria:
 - Active, bankrupt or dissolved at time t
 - With known value of some financial ratios at time $t - 1$
 - Standardised legal form: Private and public limited company
 - Entity type: corporate
 - Higher than 5 number of employees at time $t - 1$

Overview of data

Number of active (lower plot) and inactive (upper) companies



Data preprocessing

- Variables with more than 25% not available values were dropped
- Categorical variables were one-hot-encoded
- NAs were imputed with the median of a given variable
- Preprocessed dataset was split into training (80%) and testing dataset

For feature engineering, additional 20 new variables were introduced from initial 30, briefly:

- For some financial data like revenue, net profit, a relative change from previous year was introduced
- Most of the original variables were a standard positions from financial statement, thus financial ratios were introduced with some basic arithmetic operations.

Resampling methods

Corporate default data is known to have a huge class imbalance problem.

In order to address the problem and investigate their modeling consequences, I further apply 3 resampling methods to the training dataset:

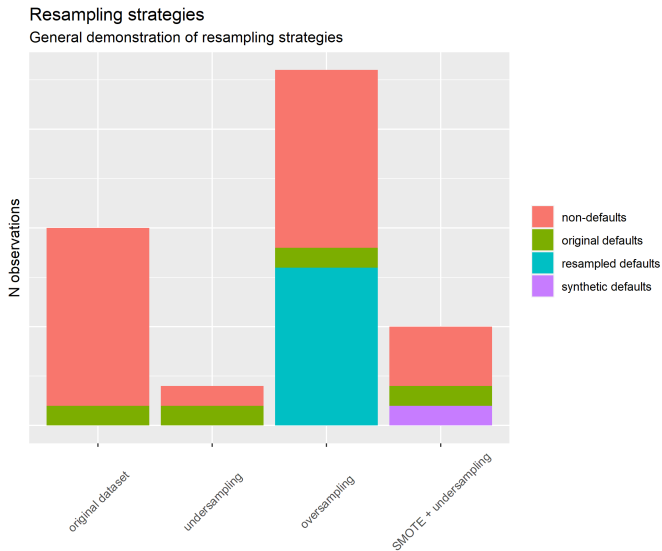
- Oversampling
- Undersampling
- Synthetic Minority Over-sampling Technique (SMOTE) [N. Chawla, et. al, 2002] + Undersampling

Further modeling workflow is done for datasets resampled with each of the methods above, in order to compare them.

Resampling methods - SMOTE

- Model-based oversampling technique
- synthetic observations - sampled from the feature space around N-nearest neighbours (5 neighbours in this project).
- The only method used that introduces some randomness
- combined with undersampling, as the original paper suggests [N. Chawla, et. al, 2002]

Resampling methods



Resampling methods - stratified resampling

- Problem: Imbalance not only on the dataset level but also between countries (see: slide 4)
- Consequence: ML favor countries with imbalance e.g:
 - IT with highest number of defaults → ML gives higher PD to Italian corporates
 - Is it economically sensible? (no, mostly sampling error)
 - Opposite effect for France
- Solution: weighted (undersampling) oversampling with weights (negatively) proportional to the share of defaults in the original dataset
- Effect: The training datasets have effectively balanced classes between countries

Applied models (1/3): Penalized logistic regression

$$\min_{w,c} \frac{1-\rho}{2} \beta^T \beta + \rho \|\beta\|_1 + C \sum_{i=1}^n \ln e^{-y_i(X_i^T \beta + c)} + 1$$

Where:

- ρ parameter regulating preference for l-2 or l-1 regularization
- C parameters regulating preference to regularization

Both of the parameters above were tuned with a grid search over cross validated results.

Applied models (2/3): XGBoost - Gradient boosted decision trees

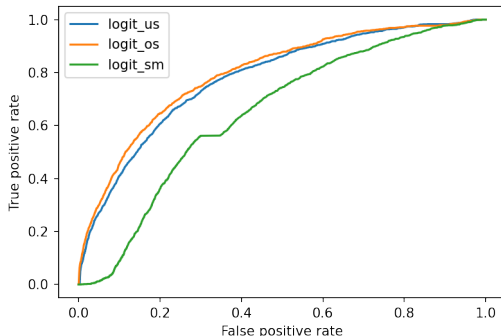
- A workhorse ML model, based on an ensemble of decision trees [T. Chen, C. Guestrin, 2016].
- Trained with a gradient boosting algorithm based on Friedman et al. 2000
- The model is trained in a sequential way, each training round consists of function estimated from previous round and a newly trained one.
- The objective function both tries to minimize log-loss and a regularization term, which penalizes number of leaves and a l-2 norm of leaf score
- all in all, a complex model....
- In herein project, the hyperparameters were tuned with random search.

Applied models (3/3): Multilayer perceptron

- Basic "vanilla" model of artificial neural networks
- Estimates weights of "neurons" iteratively with backpropagation
- Just like the previous model, can generalize non-linear patterns and has a regularization in a objective function

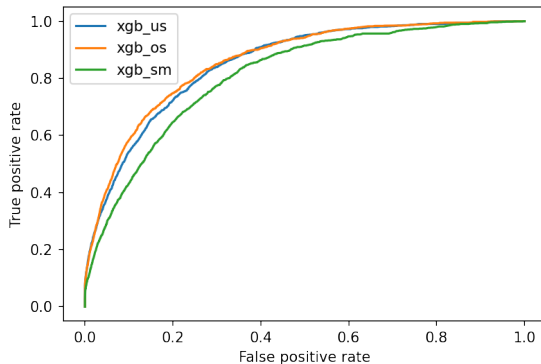
For this project, the hyperparameters were tuned with a random search.

model fitness comparison - ROC



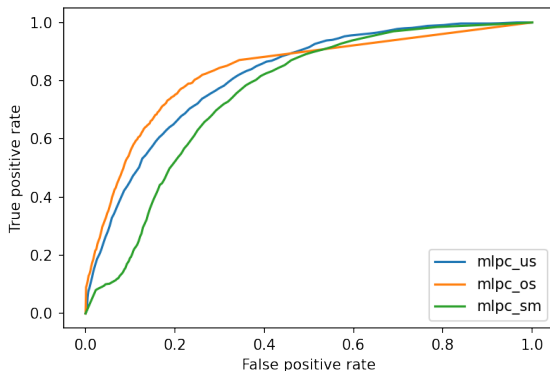
- Fairly well fit to the data, a weaker one with SMOTE
- SMOTE introduced some jumps with respect to the change of the threshold

model fitness comparison - ROC



- Similar shape
- Surprisingly, SMOTE lags behind other methods

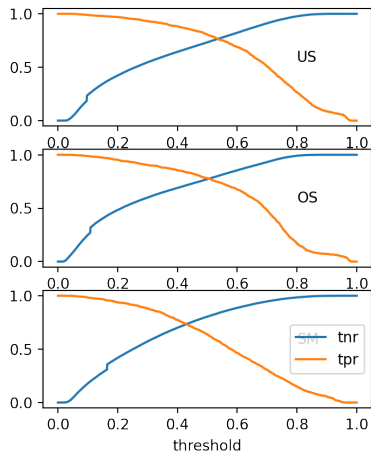
model fitness comparison - ROC



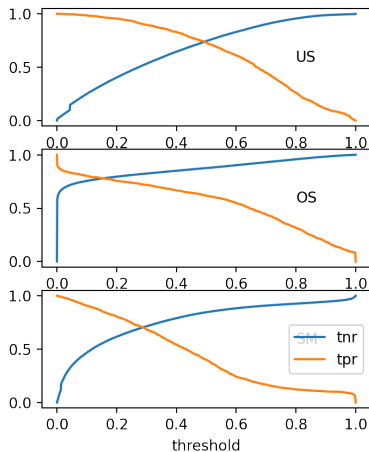
- Different shape for every method
- Once again, the SMOTE method is worse than the rest

TPR, TNR vs. threshold

XGBoost TNR - TPR plot



MLP Classifier TNR - TPR plot



Metrics with optimal thresholds

- Optimal threshold w.r.t balanced accuracy
- note, it's in-sample optimal (best case scenario)

	logit_os	logit_us	logit_sm	xgb_os	xgb_us	xgb_sm	mlpc_os	mlpc_us	mlpc_sm
thresholds	52.5 %	49.5 %	36.3 %	48.7 %	45.6 %	35.4 %	42.2 %	13.8 %	22.0 %
tpr	71.5 %	74.8 %	56.1 %	82.3 %	82.1 %	83.3 %	81.2 %	78.8 %	78.8 %
tnr	74.4 %	68.5 %	70.1 %	72.2 %	73.6 %	64.6 %	66.8 %	77.3 %	63.9 %
balanced_acc	73.0 %	71.7 %	63.1 %	77.3 %	77.8 %	74.0 %	74.0 %	78.0 %	71.3 %

Base case metrics comparison

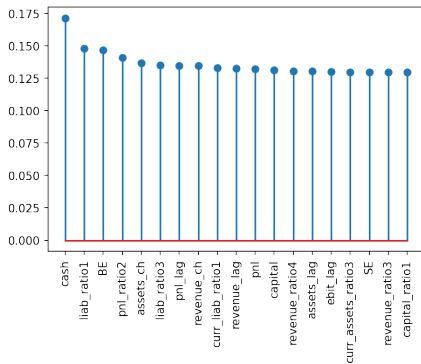
- Metrics calculated with 50% threshold (base case)

	logit_os	logit_us	logit_sm	xgb_os	xgb_us	xgb_sm	mlpc_os	mlpc_us	mlpc_sm
balanced accuracy	72.8 %	71.6 %	57.5 %	77.0 %	77.7 %	72.1 %	73.7 %	74.7 %	62.4 %
F1 score	11.4 %	10.6 %	7.6 %	13.1 %	14.5 %	13.6 %	12.3 %	18.9 %	10.7 %
TPR	74.1 %	74.1 %	34.3 %	80.7 %	78.1 %	63.8 %	72.9 %	61.8 %	40.2 %
TNR	71.5 %	69.1 %	80.7 %	73.3 %	77.2 %	80.5 %	74.5 %	87.6 %	84.6 %
FPR	28.5 %	30.9 %	19.3 %	26.7 %	22.8 %	19.5 %	25.5 %	12.4 %	15.4 %
FNR	25.9 %	25.9 %	65.7 %	19.3 %	21.9 %	36.2 %	27.1 %	38.2 %	59.8 %

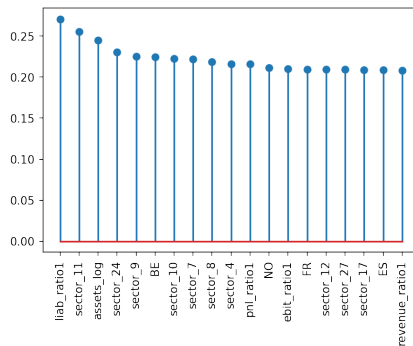
Feature importance

- Surprisingly high importance of variables in absolute values
- Smaller country effect than before stratified resampling
- Different pattern recognition between ML models (MLPC - sectors, XGB - ratios)

XGBoost - feature importance

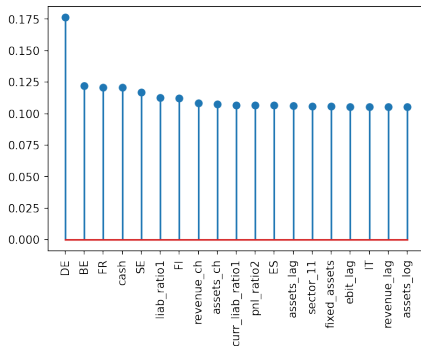


MLPC - feature importance



Feature importance without stratified resampling

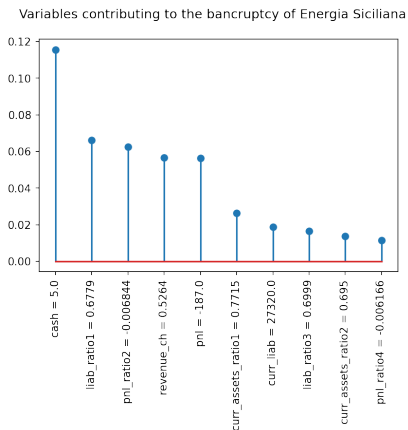
XGBoost - feature importance



	logit_os	logit_us	logit_sm	xgb_us	xgb_os	xgb_sm	mlpc_us	mlpc_os	mlpc_sm
balanced accuracy	59.0 %	52.7 %	56.4 %	80.3 %	80.8 %	79.2 %	79.3 %	75.5 %	75.2 %
TPR	31.4 %	8.1 %	21.2 %	83.3 %	82.1 %	74.5 %	81.6 %	65.6 %	61.8 %
TNR	86.6 %	97.4 %	91.5 %	77.3 %	79.6 %	83.8 %	76.9 %	85.4 %	88.6 %
FPR	13.4 %	2.6 %	8.5 %	22.7 %	20.4 %	16.2 %	23.1 %	14.6 %	11.4 %
FNR	68.6 %	91.9 %	78.8 %	16.7 %	17.9 %	25.5 %	18.4 %	34.4 %	38.2 %

Case study: Energia Siciliana S.R.L

- Big contribution to the variables in absolute levels
- correctly predicted bankruptcy 1 year before (76,6%)



further research

- What's the value-added of having single model for each country vs. a single model to rule them all?
- Choose final model:
 - best combination of resampling technique and algorithm
 - perhaps average few models
 - spend more time optimizing single model
 - What are the best metrics we can get?
- simulating a credit portfolio for a given empirical parameters (RR, ROI, default rate) Is the model profitable?

References

- N. V. Chawla, et. al, SMOTE: Synthetic Minority Over-sampling Technique, Journal of Artificial Intelligence Research 16, 2002
- T. Chen, C. Guestrin, XGBoost: A Scalable Tree Boosting System, 2016
- J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting. Annals of Statistics