Corporate default prediction

Statistics for Data science - AEM University of Brescia

Mateusz Dadej

June 26, 2022

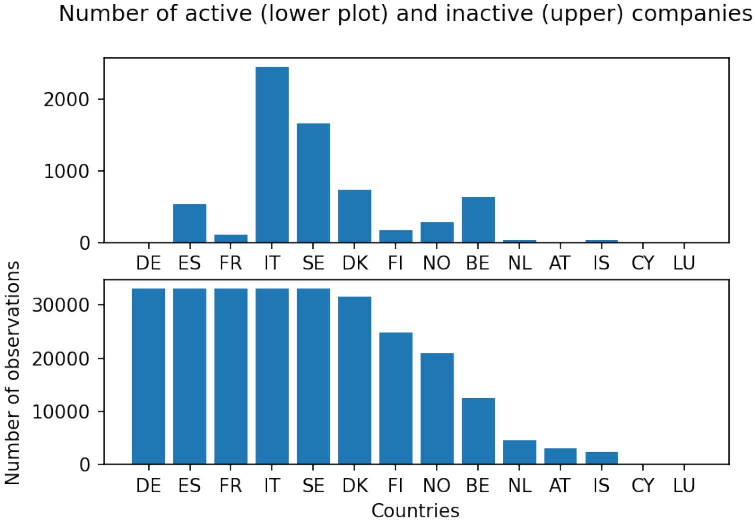
Research questions

- 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 - Private company database
- Geographical location of companies are developed countries in Europe
- in period 2018-2020
- observations were filtered with following criteria:
  - Active, bancrupt 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



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 proft, a relative change from previous year was introduced
- Most of the original variables were a standard positions from finacial 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 adress 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 compere them.

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 regularizaton
- C paraemters regulating preference to regularization

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

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

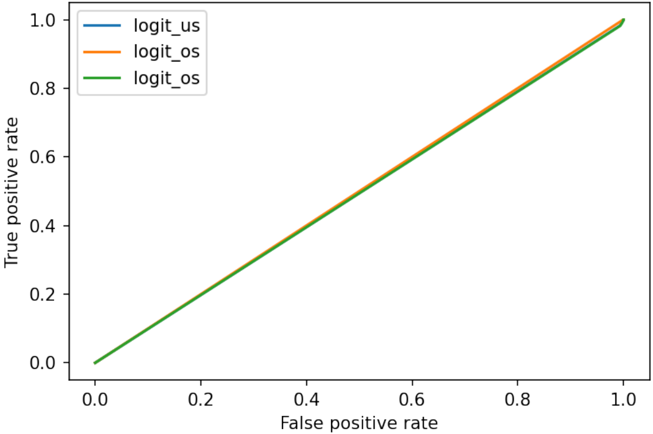
- A workhorse ML model, based on an ensemble of decision trees [T. Chen, C. Guestrin, 2016].
- Trained with a gradient boosting algorithm based on Friedman et al. 2000
- The model is trained in a sequential way, each training round consists of function estimated from previous round and a newly trained 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

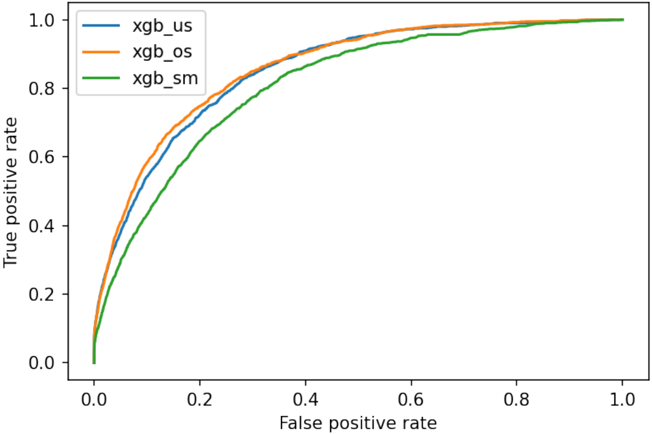
- Basic "vanilla" model of artificial neural networks
- Estimates weights of "neurons" iteratively with backpropagation
- Just like the previous model, can generalize non-linear patterns and has a regularization in a objective function

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

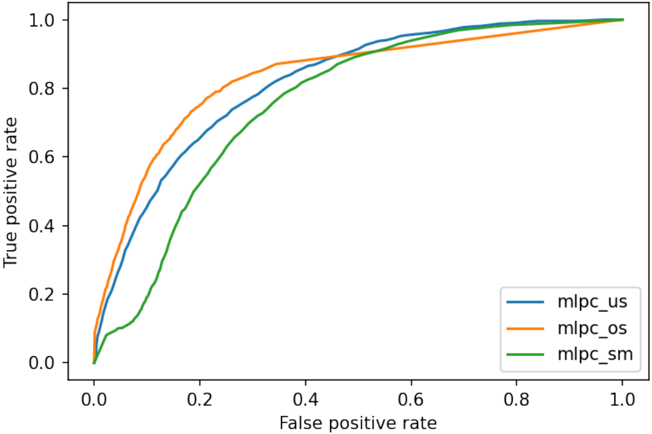
model fitness comparison - ROC



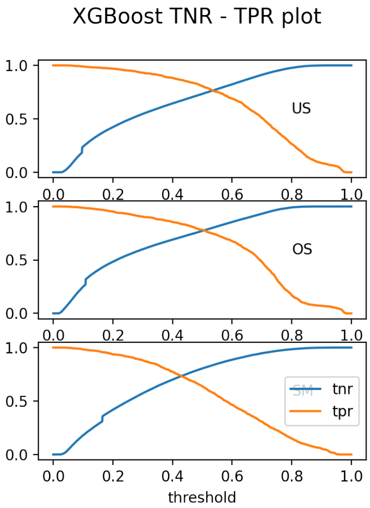
model fitness comparison - ROC

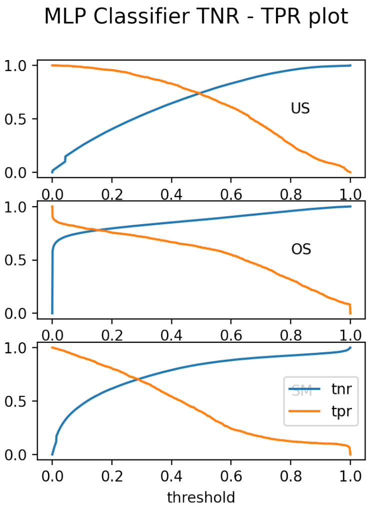


model fitness comparison - ROC



TPR, TNR vs. threshold





Metrics with optimal thresholds

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

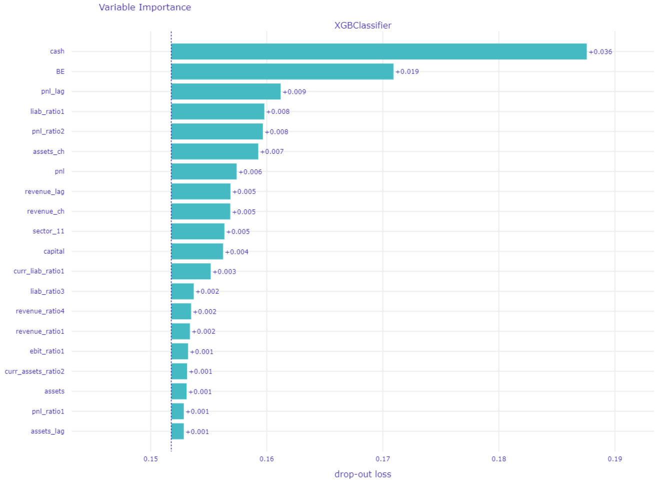
	logit_os	logit_us	logit_sm	xgb_us	xgb_os	xgb_sm	mlpc_us	mlpc_os	mlpc_sm
thresholds	49.9 %	49.0 %	48.9 %	48.7 %	45.6 %	35.4 %	42.2 %	13.8 %	22.0 %
tpr	100.0 %	100.0 %	100.0 %	82.3 %	82.1 %	83.3 %	81.2 %	78.8 %	78.8 %
tnr	0.1 %	0.1 %	0.1 %	72.2 %	73.6 %	64.6 %	66.8 %	77.3 %	63.9 %
balanced_acc	50.0 %	50.0 %	50.0 %	77.3 %	77.8 %	74.0 %	74.0 %	78.0 %	71.3 %

Base case metrics comparison

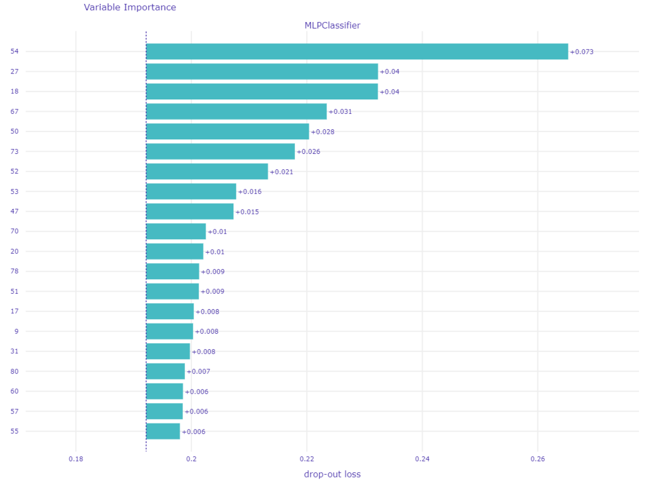
- Metrics calculated with 50% threshold (base case)

	logit_os	logit_us	logit_sm	xgb_us	xgb_os	xgb_sm	mlpc_us	mlpc_os	mlpc_sm
balanced accuracy	59.0 %	52.7 %	56.4 %	77.0 %	77.7 %	72.1 %	73.7 %	74.7 %	62.4 %
TPR	31.4 %	8.1 %	21.2 %	80.7 %	78.1 %	63.8 %	72.9 %	61.8 %	40.2 %
TNR	86.6 %	97.4 %	91.5 %	73.3 %	77.2 %	80.5 %	74.5 %	87.6 %	84.6 %
FPR	13.4 %	2.6 %	8.5 %	26.7 %	22.8 %	19.5 %	25.5 %	12.4 %	15.4 %
FNR	68.6 %	91.9 %	78.8 %	19.3 %	21.9 %	36.2 %	27.1 %	38.2 %	59.8 %

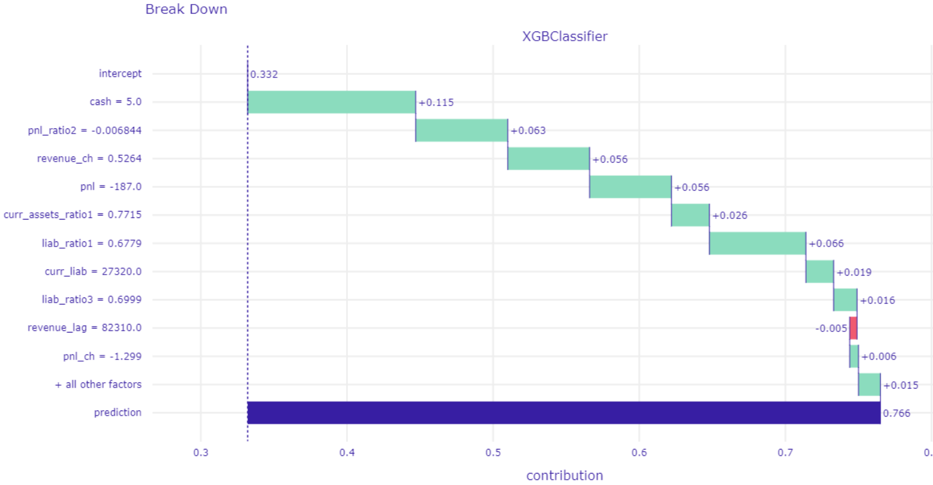
Base case metrics comparison



Base case metrics comparison



Base case metrics comparison



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