To fail or not to fail? Assessing the microprudential efficiency of Basel III
Methodology details

December 3, 2025

1 Models, performance measures and interpretation

In this section, we briefly present the methodology this paper is based on. In particular, we provide insight on how the SMOTE procedure works, on the general idea behind each model, on the performance measures we rely on, and finally on the tools we use to draw economic interpretations out of the models. These all aim at estimating a function $f(.)$ defined as follows:

$$\Pr(y = 1|X = x) = f(X) + \epsilon,$$

where $\Pr(y = 1|X = x)$ is the probability that a specific bank belongs to class 1 (default) knowing its specific characteristics $X = x$ (where x is the realization of X), and ϵ the error term. Since data are unbalanced,[1] we find ourselves in a cross-sectional analysis: when $t \neq t'$, a bank j observed at date t is considered as different from the same bank as observed at date t'. To take time dynamics into account and allow for causal relationship to be established, a one-period time lag between X and y is introduced. The function $f(.)$ estimated for bank j at time t can thus be rewritten as follows:

$$\Pr(y_{j,t} = 1|X_{j,t-1} = x_{j,t-1}) = f(X_{j,t-1}) + \epsilon_{j,t}.$$

The following procedure is applied throughout the paper to ensure models' interpretability: 1) k-cross-validation on out-of-sample macro recall is used to determine the hyperparameters of the models, 2) Shapley values are then computed to provide an idea of the significance and of the nature of the impact of the considered variables on the probability of default, 3) partial dependence plots are provided to assess the marginal impact of a given feature on the predicted probability of default.

1.1 SMOTE

Introduced by Chawla et al. (2002), the Synthetic Minority Over-sampling Technique (SMOTE) is inspired by Ha and Bunke (1997). It is built in such a way that it replicates the initial data distribution. More precisely, SMOTE uses the k-nearest neighbors of all the instances found in the minority class (failed banks in our case) to synthesize new minority class instances: synthetic observations are created

[1] While "imbalanced" refers to the fact that the minority class (failed banks) is far less represented than the majority class (unfailed banks), "unbalanced" here refers to the fact that the total number of observations is inferior to the total number of banks times the number of periods.

on the line between existing ones. Using nearest neighbors ensures that the distribution of the balanced sample is the same as that of the original imbalanced sample.

1.2 Models

We very briefly detail here the main idea behind each of the seven models that are used in the paper. In addition, we highlight their main *hyperparameters* which constrain the optimization process and are chosen according to performance scores (detailed in section 1.3). Models are implemented in Python thanks to Scikit-learn (Pedregosa et al., 2011).

Logistic regression (LR) (or **Logit**) is a linear model that estimates the probability with which each observation enters either of the classes of the outcome variable. In addition to the standard statistical model, one can also include some penalties into the cost function (Lasso, Ridge or a combination of the two: Elastic Net) which shrink the estimated coefficients (Hastie et al., 2009). The intensity of this regularization is itself a hyperparameter.

Random forest (RF) (Breiman, 2001) is an ensemble method based on the aggregation of decision trees (bagging). More precisely, trees are run in parallel so that the prediction of the model consists in the average of their outcomes. In order to avoid overfitting, trees can be constrained by limiting their maximum depth or the maximum number of variables (or features) considered when splitting observations, as well as increasing the minimum number of samples a leaf should have. The size of the forest is an important hyperparameter of this model: the number of aggregated trees (or estimators) affects the ability of the model to generalize to unobserved data as opposed to single decision trees.

Gradient boosting classifier (GBC) (Friedman, 2001) is based on the aggregation of simple models (here decision trees). While RF simply aggregates several trees with bagging, gradient boosting draws a prediction out of a simple model and improves it by implementing another simple model (boosting). This procedure is repeated several times, focusing on residual errors of the previous model. As compared to RF, an additional hyperparameter is the learning rate that affects how fast the algorithm improves the decision trees, with a lower value usually limiting overfitting. We also consider **Histogram-based Gradient Boosting classifiers (HGBC)** which implement optimized versions of GB, namely XGBoost (Chen and Guestrin, 2016) and LightGBM (Ke et al., 2017).

Support vector classifier (SVC) (Boser et al., 1992) (or Support vector machine) consists in finding the hyperplane in a N-dimensional space (N being the number of features) that best classifies the data points. One of the main hyperparameters is the regularization parameter (denoted C) that impacts the margins of error during the training: a larger C restricts the number of misclassifications but also reduces out-of-sample performances. In addition, one might allow for nonlinear separations of the dataset with the risk of overfitting as the polynomial degree increases.

Multi-layer perceptron (MLP) is a class of artificial neural network (McCulloch and Pitts, 1943; Hastie et al., 2009). We only consider here this feedforward type of network that may be composed of one or several layers of neurons, activated according to some usual functions, namely logistic, hyperbolic tangent (tanh) or Rectified Linear Unit (ReLU). The flexibility of MLP comes at the cost of an important risk of overfitting that one should control for with a combination of hyperparameters: a shrinkage of parameters, early stopping of the training and/or a maximum number of iterations in the backpropagation process.

K-nearest neighbors (KNN) is a non-parametric supervised learning classifier which classifies points based on their proximity. It is computed such that an observation is assigned to the class that is the most frequent within its neighbors. In addition to the considered number of neighbors K, one may also choose the metric as well as attribute uniform or higher weights to closer points.

1.3 Performance measures

Each of the models presented above generates predictions that allow to classify banks in either of the two considered categories (i.e. failed or unfailed). The variety of predictors implies the need to efficiently compare their performance. For a classification task, the usual criteria rely on the well-known confusion matrix (Hand, 2012) that consists in: the number of true positives (TP, failed banks identified as failed banks), the number of true negatives (TN, unfailed banks correctly identified), the number of false positives (FP, unfailed banks identified as failed ones), and the number of false negatives (FN, failed banks identified as unfailed ones). From these four categories, immediate performance scores can be computed: *accuracy* that is defined as the proportion of accurate predictions among all predictions, *recall* that is defined as the proportion of a given class that is properly identified, *precision* that is defined as the proportion of predictions for a given class that actually belongs to this class, and $F1$-score that is computed as the harmonic mean of recall and precision.[2]

There generally is a tradeoff between recall and precision, especially when the considered model is highly flexible and that the convergence criterion leads to focus on one of these scores. As an extreme example, a model may very well classify all instances in a given class and therefore reach perfect recall. In that case, such model would display a very low precision. The $F1$-score tackles this problem by assessing models with the average of these two scores. However, with severely imbalanced datasets come two additional challenges. First, the accuracy score overestimates the performance of the model if the overrepresented class is well predicted. Second, precision and recall scores imply another tradeoff regarding the class to predict. Indeed, even in a binary classification problem, the performance measure of the model is affected by the class defined as a *positive* instance. In our case, we do observe that models exhibit very high $F1$-scores when considering the prediction of unfailed banks as they easily predict a large number of TP as compared to both FN and FP. On the contrary, considering very low $F1$-scores in the prediction of failed banks actually underestimates the performance of models. As a result, a relevant performance measure is the average between the proportion of 1 (failed banks) that is well predicted and the proportion of 0 (unfailed banks) that is well predicted, which is the very definition of the average between the recall scores of the two classes, also known as *macro recall.* It is worth noticing that, when the classification problem is a binary problem, the area under the ROC curve can be computed the same way as the macro recall (Fawcett, 2006; Sokolova and Lapalme, 2009; Muschelli, 2020). In that case, these two metrics are thus equal.

1.4 k-cross validation

A complementary point should be made when comparing the performance of the models: they may not generalize identically on unobserved data. This is where cross-validation is useful (Ojala and Garriga, 2010). k-cross validation is a resampling method that uses different portions of the dataset to train and evaluate a given model on different iterations. More precisely, it consists in splitting observations into k folds and training k times the same model leaving each time a different kth part of the data to perform score measures. Cross-validation helps in the determination of the relevant classification model in two ways. First, by averaging each score reached on the k validation sets any score becomes a more robust

[2]These scores are given by the following formula: Accuracy $= \frac{TP+TN}{TP+TN+FP+FN}$, Recall $= \frac{TP}{TP+FN}$, Precision $= \frac{TP}{TP+FP}$ and $F1$-score $= 2\frac{\text{Recall}\times\text{Precision}}{\text{Recall}+\text{Precision}}$.

performance measure. Second, it helps to identify hyperparameters that limit the gap between the train and validation scores, that is to limit overfitting.

1.5 Interpretation

As already mentioned, we resort to two tools to draw economic interpretation out of the models: Shapley value and partial dependence plots.

Shapley value. For all models, we resort to the Shapley value (Shapley, 1953; Strumbelj and Kononenko, 2013). The rationale behind the computation of this latter is grounded in game theory: all features are assumed to be players engaged in a game where the payout is the prediction. In this context, the Shapley value indicates how this payout is distributed among the features given their contribution. See Molnar (2020) for more details on computation. In addition, as a robustness check, we rely on permutation feature importance (Molnar, 2020).

Partial dependence plots (PDPs). Economic interpretations of our results mostly rely on PDPs (Friedman, 2000; Hastie et al., 2009). PDPs average the Individual Conditional Expectation (ICE) of all individuals. Considering the i-th individual and the variable X_j and fixing all the other variables to their level taken for the i-th individual, the ICE corresponds to the predictions of the model when X_j varies from its minimum to its maximum value with step k. ALEs are used to take into account the fact that PDPs can be biased when features are strongly correlated.

References

B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory - COLT '92*, pages 144–152, Pittsburgh, Pennsylvania, United States, 1992. ACM Press. ISBN 978-0-89791-497-0. doi: 10.1145/130385.130401. URL `http://portal.acm.org/citation.cfm?doid=130385.130401`.

L. Breiman. Random Forests. *Machine Learning*, 45(1):5–32, 2001. ISSN 08856125. doi: 10.1023/A:1010933404324. URL `http://link.springer.com/10.1023/A:1010933404324`.

N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. SMOTE: Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence Research*, 16:321–357, 2002.

T. Chen and C. Guestrin. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, aug 2016. doi: 10.1145/2939672.2939785. URL `https://doi.org/10.1145%2F2939672.2939785`.

T. Fawcett. An introduction to ROC analysis. *Pattern Recognition Letters*, 27(8):861–874, June 2006. ISSN 01678655. doi: 10.1016/j.patrec.2005.10.010. URL `https://linkinghub.elsevier.com/retrieve/pii/S016786550500303X`.

J. Friedman. Greedy Function Approximation: A Gradient Boosting Machine. *The Annals of Statistics*, 29, 11 2000.

J. H. Friedman. Greedy Function Approximation: A Gradient Boosting Machine. *The Annals of Statistics*, 29(5):1189–1232, 2001. ISSN 00905364. URL `http://www.jstor.org/stable/2699986`. Publisher: Institute of Mathematical Statistics.

T. M. Ha and H. Bunke. Off-line, handwritten numeral recognition by perturbation method. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(5):535–539, 1997.

D. J. Hand. Assessing the performance of classification methods. *International Statistical Review*, 80, 12 2012.

T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* Springer series in statistics. Springer, 2009. ISBN 9780387848846.

G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu. Lightgbm: A highly efficient gradient boosting decision tree. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL `https://proceedings.neurips.cc/paper/2017/file/6449f44a102fde848669bdd9eb6b76fa-Paper.pdf`.

W. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:115–133, 1943.

C. Molnar. *Interpretable machine learning: a guide for making Black Box Models interpretable.* 2020. ISBN 978-0-244-76852-2. URL `https://christophm.github.io/interpretable-ml-book/`. OCLC: 1105728953.

J. Muschelli. ROC and AUC with a Binary Predictor: a Potentially Misleading Metric. *Journal of Classification*, 37(3):696–708, Oct. 2020. ISSN 0176-4268, 1432-1343. doi: 10.1007/s00357-019-09345-1. URL `http://link.springer.com/10.1007/s00357-019-09345-1`.

M. Ojala and G. C. Garriga. Permutation tests for studying classifier performance. *J. Mach. Learn. Res.*, 11:1833–1863, aug 2010.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12: 2825–2830, 2011.

L. S. Shapley. A Value for n-Person Games. *Princeton University Press*, pages 307–317, 1953.

M. Sokolova and G. Lapalme. A systematic analysis of performance measures for classification tasks. *Information Processing & Management*, 45(4):427–437, July 2009. ISSN 03064573. doi: 10.1016/j.ipm.2009.03.002. URL `https://linkinghub.elsevier.com/retrieve/pii/S0306457309000259`.

E. Strumbelj and I. Kononenko. Explaining prediction models and individual predictions with feature contributions. *Knowledge and Information Systems*, 41:647–665, 2013. doi: 10.1007/s10115-013-0679-x.