Dimension Reduction Techniques for SME Loan Default Prediction

Ludvig Wärnberg Gerdin
ludvig.warnberggerdin@epfl.ch

***Abstract*—One of the most popular forms of small- and medium sized business financing is bank loans, and a crucial part of lending to these businesses is default prediction, i.e. predicting whether the borrower will be able to repay their loan. In this study, I examine the usefulness of noise-reduced financial statement information for SME default prediction. Three methods are applied - principal component analysis (PCA), isomap, and autoencoders - and compared to applying no noise reduction. The results show after applying PCA, the classifier performed the best. The classifier then yielded macro-averaged precision, recall, and F1-score of 0.51, 0.52, and 0.40, respectively.**

I. Introduction

Small- and medium sized enterprises (SME) account for 90% of businesses worldwide (The World Bank, n.d.). They are important to job creation, as 70% of jobs are provided by SME's, and are important for global economic development.

SME's are in particular need of financing to grow and stay competitive. One of of the most popular sources of financing for these businesses are bank loans and credit lines (European Union, 2019).

A crucial part of lending is to be able to asses whether the borrower will be able to repay their loan or not, i.e. if the customer will default on their loan. Traditionally, assessments of this kind has been done in a manual rather than digital way and were often subject to the risk of subjectivity. However, with the emergence and popularisation of tools and techniques to handle large amounts of data, these assessment processes have become predominantly digital and model-based.

A multitude of features can be used for default prediction. Zekic-Susac et al. (2004) explored the use of machine learning for small business default prediction in Croatia, where they used traditional credit scoring variables, such as activity of the firm, company age, and credit amount. Novel approaches include alternative data for default prediction. For example, Stevenson et al. (2020) used natural language processing and deep learning models applied to text data from 60000 records of manual credit assessments. They experiences model improvement when using text data in combination with traditional credit assessment variables.

Another approach is to use financial statement information. The challenge of using this type of information is that it is high-dimensional and noisy, which may harm the model's ability to determine the characteristics of a sample with high risk of default. An approach to alleviate these issues is to apply dimensionality and noise reduction techniques to the data. Thus, this project aims to explore the use of noise reduced financial statement data for prediction of SME loan default.

II. Methods

The `pandas` (McKinney, 2010), `scikit-learn` (Pedregosa et al., 2011), and `skorch` (Tietz et al., 2017) packages were used extensively for data preparation and modelling.

A. Data

The independent variables were financial statement information on loans that were disbursed during a two year period. Additionally, ratios derived from the financial statement information, pre-computed by a third-party provider of financial statement information, were included as features. The raw financial statement information ranged from net income to extraordinary events. An example of a pre-computed ratio is the equity ratio.

The dependent variable were default within 9 months. The rationale for choosing 9 months as a time period was to reduce the target class imbalance. The definition of default in this analysis is broad. It includes, but is not limited to, loans that have been sold to partnering payment collection companies and loans whose payments are above a pre-defined threshold of past due.

B. Models

Three candidate unsupervised machine learning approaches will be compared with the baseline approach of using no dimensionality reduction. The first unsupervised approach was chosen to be Principal Component Analysis (PCA), known as one of the most popular linear dimensionality reduction techniques (e.g. by Van Der Maaten et al. (2009)). The two other candidate approaches were chosen to be Isomap and autoencoders, two non-linear dimensionality reduction techniques.

1) PCA: PCA is a way to represent the D-dimensional space by linearly mapping to a K-dimensional space, $K \leq D$. From a probabilistic point of view, the PCA transformation decorrelates the input data (Jaggi et al., 2020). Formally, we are utilising the singular value decomposition of the feature set to find a $K \times N$ linear transform C and a $N \times K$ matrix R so that

$$||X - RCX||_F^2$$

is minimized, where $||A||_F$ represents the Frobenius norm of a matrix A.

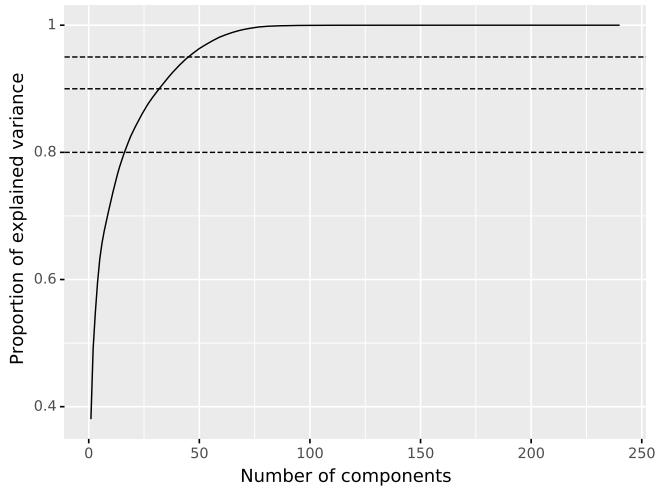

Fig. 1. PCA explained variance as function of K. The lines fro top to bottom mark 95%, 90%, and 80% of explained variance.

For PCA, I tweaked K in the hyper-parameter tuning. Figure 1 illustrates that more than 95% of the variance in the data is explained by less than 50 features. Thus, the configurations were chosen to be below 50 components, more specifically they 10, 15, 20, 25, and 30 components.

2) Isomap: PCA aim to preserve the pair-wise euclidean distances between samples in the data. However, the euclidean distance might not best represent the distance between the points when the geometry of the data is non-linear. Isomap instead measures the closeness of points by their geodesic distance rather than their euclidean distance, and therefore preserves the data geometry when projected to lower dimensions. Figure 2 explains this is a graphical sense, where the blue distance represent the euclidean distance and the red distance represent the geodesic distance. Formally, isomap seek to minimize

$$||\tau(D_G) - \tau(D_Y)||_F$$

where $\tau(D) = -\frac{1}{2}HSH$, D_G is a matrix of graph distances, D_Y is matrix of euclidean distances, S is a matrix of squared distances of D and H is the centering matrix (Tenenbaum et al., 2000).

The configurations that were tested for the isomap technique were 3, 5, and 10 neighbors and 10, 20, and 30 components.

3) Autoencoder: An autoencoder is a neural network that is constructed with an encoding component and a decoding component. In essence, autoencoders aims to learn the structure of the lower-dimensional manifold that represents the data. Is is the encoded, compressed sample that is used for subsequent prediction and classification tasks. In this study, the neural network learns to decompress the dataset in such as way that the cell-wise mean squared error is minimized between the original dataset and the compressed dataset.

The traditional autoencoder is a fully-connected neural network. One can extend this however, by introducing tweaks to the network that may improve its performance. For example, by introducing regularization to the loss function, we enforce that that the network learns a ”simple” representation where only parts of the network neurons are activated. This may be referred to as a sparse autoencoder, and will be explored in this study.

The number of layers, learning rate, regularization and batch size were given as hyper-parameters in the hyper-parameter tuning. The configurations that were tested is given in Table I.

4) Classifier: A classifier is applied on all features to obtain baseline performance estimates. The classifier is also applied to the denoised samples produced by the candidate unsupervised approaches. The classifier was chosen to be logistic regression with l_2 regularization.

C. Data preparation

The latest yearly financial statement information before the loan was disbursed were gathered for each loan. I subsetted the companies with legal form ”AB”, the Swedish equivalent to limited company, in order to be able to use several years of financial statement data as features. The yearly financial statement variables were converted into separate features. For example, net income became `net_income__latest`, `net_income__shift1` for the previous year, and so forth.

Two major candidate approaches of data preparation was explored. As the baseline, the untouched values of the variables were used as input. For the first candidate approach, the difference of subsequent years was used. For the second candidate approach, the sign of the differences of the values were used. The rationale for the latter two approaches was to reduce the complexity of the features. For the second and third approach, differences were not calculated for variables expressed as ratios. The data preparation approaches were tested in the cross-validation procedure.

Adding indicator variables were explored as an option to mark out missing values. An indicator variable were added for each feature that contained values of 0. The indicator variable were then given a value of 1 if the base column value was 0, and 0 otherwise. In the case when the second candidate data preparation approach was applied, the indicators represent either missing values or that the value remained constant between two years. Whether or not to use indicator variables were given as a separate hyper-parameter in the grid-search.

The sample was standardized by subtracting the mean and dividing by the standard deviation of each column. In addition, because there is a large class imbalance in the target variable, random under-sampling without replacement of the majority class were followed by applying the synthetic minority over-sampling technique (SMOTE). By random under-sampling, instances that have the majority target class are randomly deleted until class balance is reached. By SMOTE, synthetic samples are created along the line segments of the k nearest neighbors of the minority class samples. The original authors presents that under-sampling the majority class, followed by

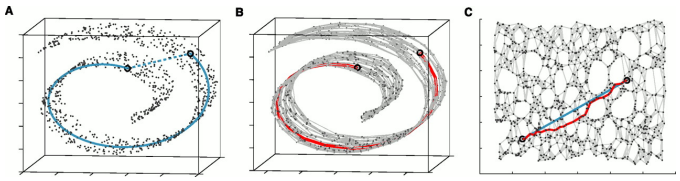

Fig. 2. "Swiss roll" illustration of the difference between geodesic and euclidean distances, as well as the resulting projection (Tenenbaum et al., 2000)

TABLE I
AUTOENCODER CONFIGURATIONS. THE "LAYERS" SHOWS THE NUMBER OF NODES IN EACH LAYER.

Layers	Learning rate	Activation function	Batch size	Epochs	Sparse
200, 100, 50; 100, 50	0.6 0.8	Leaky-ReLU ReLU	512	100	True False

over-sampling the minority class performed better than plain under-sampling (Chawla et al., 2002).

D. Model Evaluation

The models were evaluated using precision, recall, and the F1 score. The precision aims to answer "What proportion of classified defaults did default?", and is defined formally as

$$\text{Precision} = \frac{TP}{TP + FP}.$$

Recall answers "What proportion of actual defaults were classified as such?", and is defined formally as

$$\text{Recall} = \frac{TP}{TP + FN}$$

Other common measures for classification include the area under the receiver operating characteristics curve and accuracy, however both these measures can be deceptive when evaluated on a dataset with high class imbalance. Precision and recall have been described to be more informative in the setting of high target class imbalance (Saito & Rehmsmeier, 2015). The precision and recall were macro-averaged to weight penalize mis-classification of defaults higher.

The macro-averaged F1 score is a weighed average of the precision and recall, defined as

$$\text{F1} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

E. Model selection

The sample was first split into two partitions, the first denoted the training-validation partition and the second denoted the testing partition. To select the data preparation approach and hyper-parameters for the dimension reduction techniques, I ran 5-fold stratified cross validation on the training-validation partition. Stratification based on the target variable ensures that the same proportion of target classes are allocated to the training and validation folds in each iteration of the cross-validation. The rationale for choosing 5 folds rather than a higher number of folds was that the number of defaults is small - choosing a higher number of folds would leave a small number of defaults in each validation fold. Using the hyper-parameters found in the cross-validation, the model was run on the full training-validation set and evaluated on the training set.

III. RESULTS

For PCA, the optimal number of components were 35. For isomap, the optimal number of components and neighbors were 20 and 5, respectively. For both methods, using the sample prepared as differences with the sign attached were the optimal data preparation. Neither the first nor the second utilised missing indicators.

In order to diagnose whether or not the autoencoder were underfitting or overfitting to the data, I split the training-validation partition into a training and validation set. The validation set contained a random subset of 20% of the data, stratified by the target variable.

The grid search determined that the sparse option should be used. Figure 3 presents the training and validation loss when using the sparse option. Using this option, the precision and recall on the testing partition were both 0. Further, the training loss is above the validation loss, indicating that the model is behaving in an unstable manner. Therefore, I picked the second best option from the grid-search, without the sparse option.

Figure 4 presents the training and validation loss using the second best configuration. Because the model seem to be underfitting, I further validated the autoencoder using 1000 epochs, which is presented in Figure 5. The validation loss seem to stagnate after approximately 875 epochs, which was used for the final autoencoder configuration. The final configuration of the autoencoder is presented in Table II.

Confusion matrices on the testing partition for the baseline, PCA, isomap, and the autoencoder are presented in Figure 6, 7, 8, and 9, respectively. In all matrices, there is a tendency for high false negatives, i.e. predicting default when actually non-default. Since this is a pattern for all approaches, it is feasible that this is a result of the data re-sampling approach rather than specific to the models.

The precision, recall, and F1 score computed on the testing partition is presented in Table III. Using PCA, the classifier had the highest out-of-sample precision and F1-score for both non-defaults and defaults. Using all features, the classifier had

TABLE II
OPTIMAL AUTOENCODER HYPERPARAMETER CONFIGURATION. THE "LAYERS" SHOWS THE NUMBER OF NODES IN EACH LAYER.

Sample	MI	Layers	Learning rate	Activation function	Batch size	Max Epochs	Sparse
Difference and sign	False	100, 50	0.8	Leaky-ReLU	512	875	False

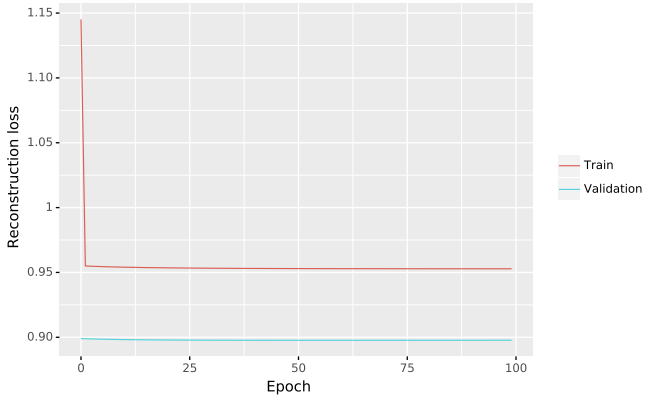

Fig. 3. Training and validation loss with sparse option.

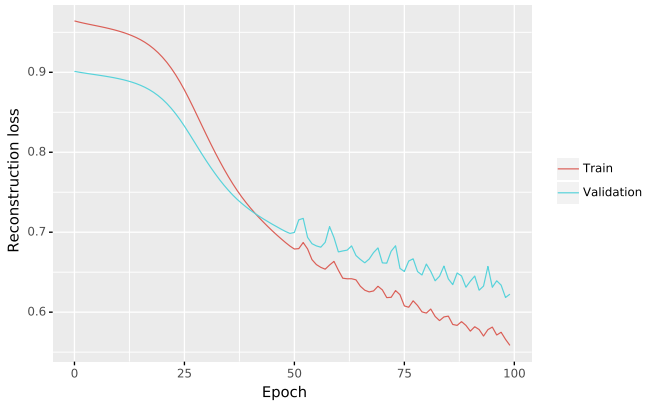

Fig. 4. Training and validation loss without sparse option, 100 epochs.

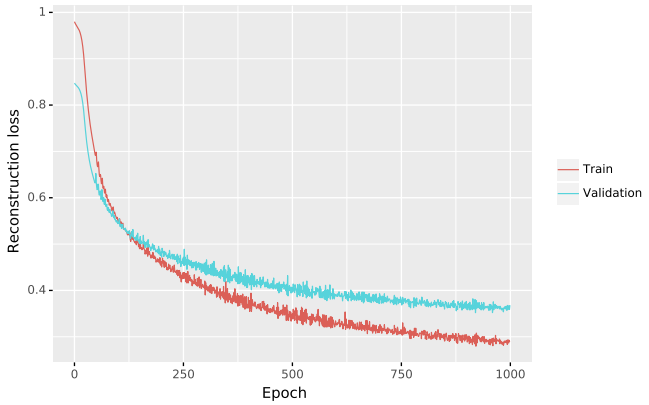

Fig. 5. Training and validation loss without sparse option, 875 epochs.

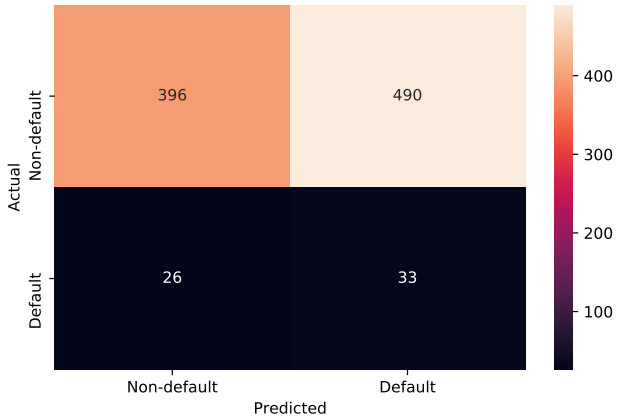

Fig. 6. Classification matrix on test partition using all features.

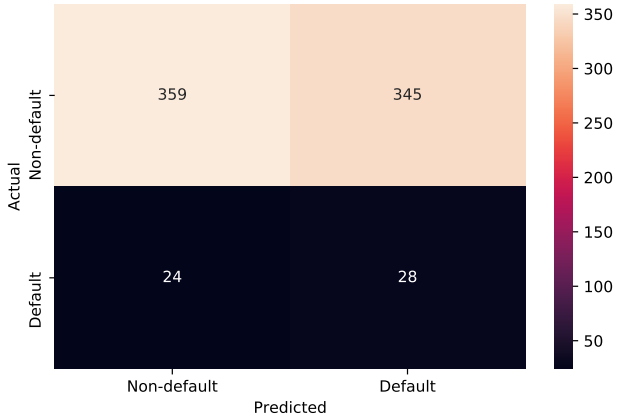

Fig. 7. Classification matrix on test partition using PCA for dimension reduction.

a higher out-of-sample recall than PCA, and tied with PCA on out-of-sample precision on non-defaults. Overall, i.e. with respect to macro-averaged precision, recall, and F1, using PCA yielded the highest performance on all metrics. Predictions after denoising with isomap and the autoencoder were less accurate than using all features and PCA on all categories of metrics.

TABLE III
IN-SAMPLE AND OUT-OF-SAMPLE ESTIMATES ON CLASS-LEVEL FOR ALL MODELS. THE TOP VALUE FOR EACH MEASURE, PARTITION, AND CLASS IS IN BOLD.

		All features			PCA			Isomap			Autoencoder		
	Class	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
Train	Non-default	0.62	0.59	0.61	**0.63**	0.61	0.62	0.61	**0.67**	**0.64**	**0.63**	0.61	0.62
	Default	0.61	0.63	0.62	0.62	**0.64**	**0.63**	**0.63**	0.57	0.60	0.62	**0.64**	**0.63**
	Macro-average	0.61	0.61	0.61	**0.63**	**0.63**	**0.63**	0.62	0.62	0.62	**0.63**	**0.63**	**0.63**
Test	Non-default	**0.94**	0.45	0.61	**0.94**	**0.51**	**0.66**	0.93	0.50	0.65	0.92	0.49	0.64
	Default	0.06	**0.56**	0.11	**0.08**	0.54	**0.13**	0.06	0.44	0.10	0.07	0.46	0.12
	Macro-average	0.50	0.50	0.36	**0.51**	**0.52**	**0.40**	0.49	0.47	0.38	0.49	0.47	0.38

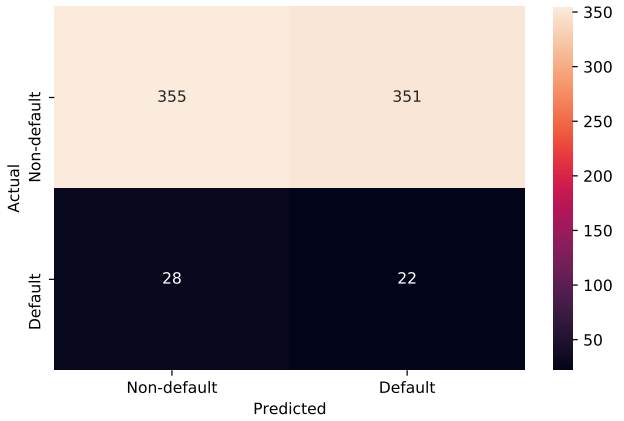

Fig. 8. Classification matrix on test partition using isomap for dimension reduction.

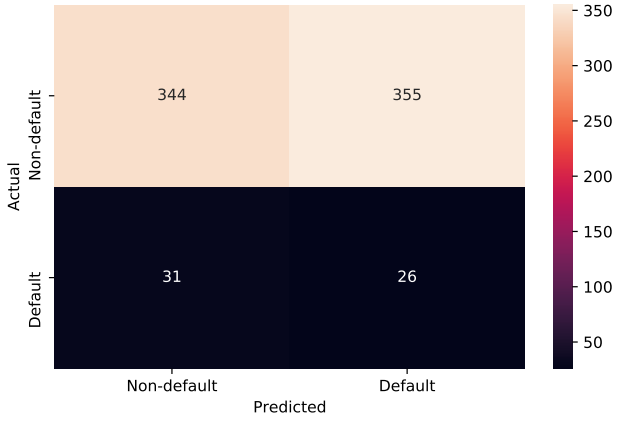

Fig. 9. Classification matrix on test partition using autoencoder for dimension reduction.

IV. DISCUSSION

In this study, I show that using PCA yielded the best out-of-sample performance. That being said, the performance of the different approaches are overall similar to each other, and - unfortunately - relatively discouraging. Previous studies have shown significantly better performance in default prediction using partly financial statement information (See for example Provenzano et al. (2020)). The following discussion outlines reasons as to why we observe low performance, and proposes improvements to the study design for future research.

A. Methodological Considerations and Future Research

Other than the financial statement information and ratios derived from it, no additional predictors were added. The rationale was to solely evaluate the use-fullness of these factors to determine whether customers will default or not. That being said, it is likely that adding additional features could have a positive impact on the model performance. For example, useful features could be the date when the loan was disbursed (as some businesses are more sensitive to seasonality), the type of business, and whether or not the borrower has any previous loan or not. Further, since only yearly financial statement information was used, one could suspect that it loses predictive value as time goes by. In the most extreme case, this means that the latest financial statement data that were given for loans disbursed at the end of the year were approximately one year old.

Apart from using raw statement values as input, two major approaches of feature engineering were explored - that is, taking differences between the statement information from different years and applying the sign to those differences. The feature engineering were applied in a general fashion, i.e. the same engineering were applied to all features (excluding the ratio features). Ideally, these transformations should be explored on a more granular basis, where they are considered on separate features. Further, several ways of feature engineering could be explored to increase the end performance of the logistic regression model. For instance, indicator variables for extreme values and rolling summary statistics such as rolling averages could be added. Then again, the focus of the analysis in this report should lie on the performance of the dimensionality reduction algorithms, which is why general approaches of feature engineering were applied.

The only model fitted to the reduced samples was a logistic regression model. The reason was to focus the analysis on the performance of the dimensionality reduction techniques rather than exploring the usefulness of more complex models. There are an abundance of models that could be explored as alternatives to the logistic regression model. For example, Provenzano et al. (2020) utilised a stacked model including a "light" gradient boosting machine and a random forest algorithm. Further, there are many different tweaks that can be explored for the autoencoder architecture. Moreover, one could explore utilising variational autoencoders as we've seen in class. Tweaks to the model architecture should be included in an expanded hyper-parameter grid search or randomized search. Moreover, the handling of the class-imbalance can be handled throgh model methods rather than using re-sampling methods. For exaample, one could specify weights for the target classes that are inversely proportional to the number of instances of each class.

As can be seen by Figure 5, in order to get the validation loss to stagnate, the autoencoder had to be fitted with a large number of epochs. The number of epochs in the grid-search was significantly lower than that number. It is possible that the grid search would have chosen a different model should the number of epochs been higher. In future research the number of epochs should be set higher in the cross-validation procedure. Another approach would be to utilise techniques to speed up training. Ioffe and Szegedy (2015) showed that utilising normalized batches can significantly speed up training of neural networks. Moreover, the grid search results had to be adjusted since refitting the selected model on the training-validation partition and testing on the testing partition gave unreliable results. This could perhaps be avoided by choosing a more sensible measure for the model selection.

Interpretability of features in the reduced samples were not taken into regard when choosing dimension reduction techniques for this project. This is, however, an important aspect of model development in the lending context. With current credit regulations, a borrower that is declined a loan have the right to know why that ended up being the case (Demajo et al., 2020). Another approach would be to utilis feature importances to conduct feature selection in a supervised manner rather unsupervised. To extract feature importance one could for example utilise novel methods such as Lime or Shapley Additive Explanations, or more traditional measures such as mean-decrease accuracy. The analyst would then have to decide on how many of the most importanct features to keep for the final model. Another approach would be to first conduct dimensionality and noise reduction on the data, and then determine the feature importance on the reduced dataset. However, the latter approach require that we know what the features represent after dimension reduction / noise reduction. All candidate approaches in this study map the features to a lower dimensional space, from which it is not possible to extract the original features.

V. Conclusion

This study evaluates the usefulness of three dimensionality and noise reduction techniques - PCA, isomap, and an autoencoder - for 9-month SME default prediction using financial statement data. The best performing approach with respect to the macro-averaged precision, recall, and F1-score was PCA. Neither isomap nor the autoencoder proved to be the most useful on any out-of-sample metric. All candidate approaches had similar performance to the baseline approach.

References

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic Minority Oversampling Technique. *Journal of Artificial Intelligence Research*, *16*, 321–357.

Demajo, L. M., Vella, V., & Dingli, A. (2020). Explainable AI for Interpretable Credit Scoring. *Computer Science & Information Technology (CS & IT)*, 185–203. https://doi.org/10.5121/csit.2020.101516

European Union. (2019). *SME access to finance conditions 2019 SAFE results - European Union* (tech. rep.).

Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *32nd International Conference on Machine Learning, ICML 2015*, *1*, 448–456.

Jaggi, M., Rüdiger Urbanke, & Khan, M. E. (2020). Motivation. *Machine learning course - cs-433* (pp. 13–16).

McKinney, W. (2010). Data Structures for Statistical Computing in Python. In S. van der Walt & J. Millman (Eds.), *Proceedings of the 9th Python in Science Conference* (pp. 56–61). https://doi.org/10.25080/Majora-92bf1922-00a

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

Provenzano, A. R., Trifirò, D., Datteo, A., Giada, L., Jean, N., Riciputi, A., Pera, G. L., Spadaccino, M., Massaron, L., & Nordio, C. (2020). Machine learning approach for credit scoring. *arXiv*, 1–28.

Saito, T., & Rehmsmeier, M. (2015). The precision-recall plot is more informative than the roc plot when evaluating binary classifiers on imbalanced datasets. *PLoS ONE*, *10*.

Stevenson, M., Mues, C., & Bravo, C. (2020). The value of text for small business default prediction: A deep learning approach. *arXiv*, 1–25.

Tenenbaum, J. B., de Silva, V., & Langford, J. C. (2000). A Global Geometric Framework for Nonlinear Dimensionality Reduction. *SCIENCE*, *290*(December), 151–180. https://doi.org/10.1007/978-3-642-27497-8_8

The World Bank. (n.d.). Small and Medium Enterprises (SMEs) Finance. Retrieved January 9, 2020, from https://www.worldbank.org/en/topic/smefinance

Tietz, M., Fan, T. J., Nouri, D., Bossan, B., & skorch Developers. (2017). *Skorch: A scikit-learn compatible neural network library that wraps pytorch*. https://skorch.readthedocs.io/en/stable/

Van Der Maaten, L. J. P., Postma, E. O., & Van Den Herik, H. J. (2009). Dimensionality Reduction: A Comparative Review. *Journal of Machine Learning Research*, *10*, 1–41. https://doi.org/10.1080/13506280444000102

Zekic-Susac, M., Sarlija, N., & Bensic, M. (2004). Small business credit scoring: A comparison of logistic regression, neural network, and decision tree models. *Proceedings of the International Conference on Information Technology Interfaces, ITI*, (May 2014), 265–270. https://doi.org/10.1109/ITI.2004.241696