unibz
Fakultät für Informatik
Facoltà di Scienze e Tecnologie informatiche
Faculty of Computer Science

Advanced Topic in Machine Learning - Project report

Financial crisis prediction with Multi Layer Perceptron and Recurrent Neural Networks

by
Hannes Wiedenhofer (14411) - Claudio Contini (18727)

February 7th, 2021

Contents

Chapter 1

Introduction

1.1 Problem description

Financial crises have major implications on the lives of billions of people. Spotting their warning signs early can facilitate the timely activation of countermeasures to prevent them. Many people have tried to come up with models for this challenging task.
Reference paper: Bank of England Staff Working Paper No. 848

1.2 Challenges

- One challenge is that we have only a limited set of crises to base our models on. Throughout recent years there have not been many major financial crises. This makes modeling difficult.
- Another challenge consists in the fact that many crisis indicators indicate crises too late to intervene.
- Also, it can be difficult to translate complex early warning models to simple indicators that can help prevent financial crises.
- The amount of variables is very high, therefore making it difficult to determine the significant ones.
- The countries are different from each other, making it difficult to find a "one fits all" solution. This problem can be overcome by taking the global average of some indicators.
- Replicating the paper's accuracy (AUC, a trade off between true positive rate and false positive) and determining the most important indicators.

1.3 Steps followed

- Dataset analysis
- Data preprocessing
- Reproduction of MLP model
- Implementation of RNN
- Evaluation

Chapter 2

Dataset

The dataset comes from the Jorda-Schularick-Taylor Macrohistory Database. It contains 50 features and 17 countries in a time span from 1870 to 2017. The features consist of various financial indicators and measures such as GDP per capita, investment-to-GDP ratio, imports/exports, and many more.

These indicators are available for 17 countries. These countries represent the most important economies in the world, among them the USA, Japan, and many European countries such as Germany, France, Spain and Italy. One of the features represents whether in a certain year a certain country suffered a financial crisis. Overall, there are 2499 observations, 90 of them representing a state of financial crisis, 2409 of them representing no state of crisis. The crises are defined as "events during which a country's banking sector experiences bank runs, sharp increases in default rates accompanied by large losses of capital that result in public intervention, bankruptcy, or forced merger of financial institutions."

In order to be able to achieve a better prediction we (and the paper) decided to remove all observations between 1933 and 1939, the later years of the Great Depression. Also, we excluded the two world wars (1914-1918, 1939-1945) as they represent an anomaly. Additionally, all observations with missing values were excluded. After these exclusions, we obtain 1249 observations of the original 2499. Of these observations, 95 have a positive class, indicating the (approaching) state of crisis.

The following is a graphic representation of the dataset. It shows the excluded periods of time, the periods of pre-crisis, crisis, and post-crisis.

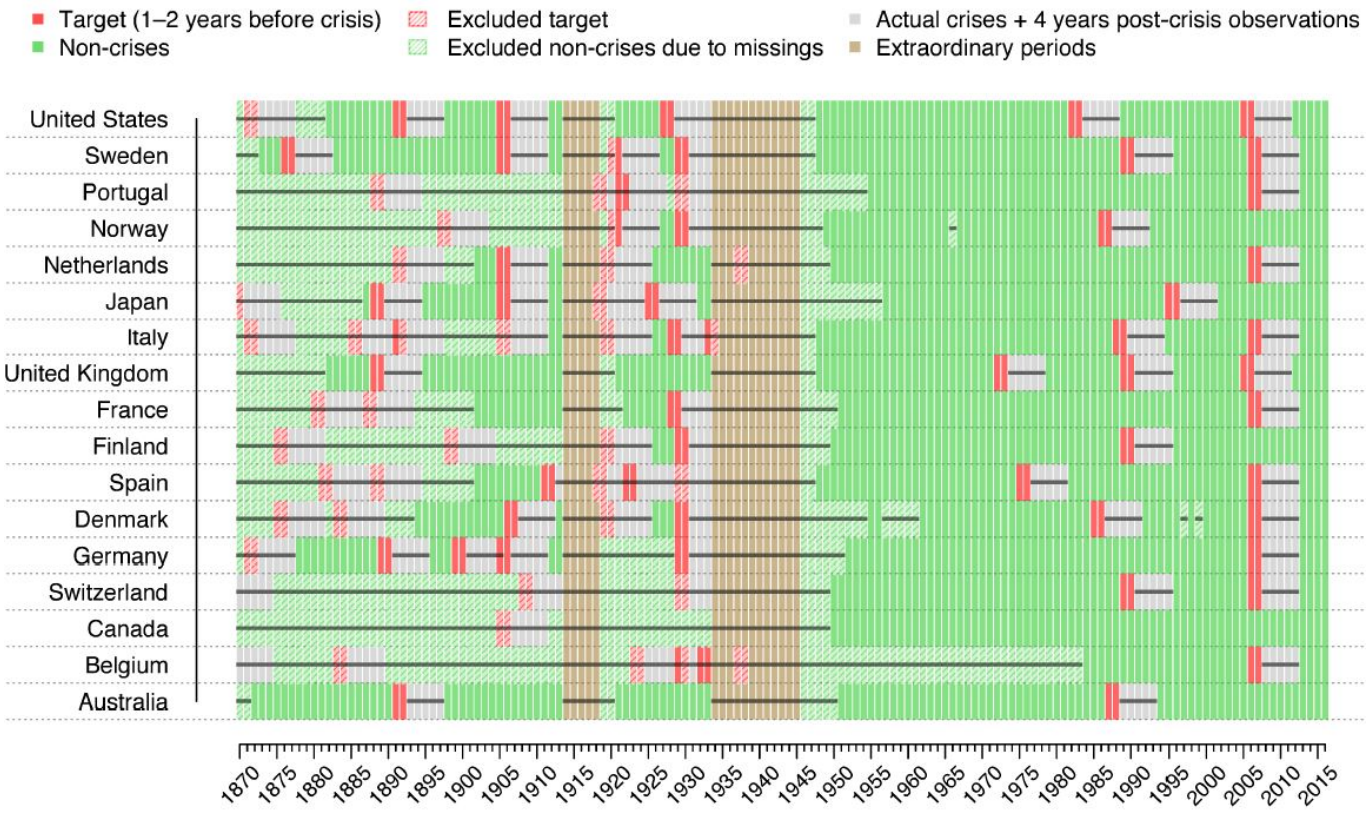

Figure 2.1: View of the Jorda-Schularick-Taylor Macrohistory dataset - target variable - author's work

Chapter 3

Data Preprocessing

3.1 Results

The feature selection performed by the authors is judgmental since just few features out of the 46 present in the dataset were considered. The steps followed are the following ones:

- exclusion of the years of crisis and the following four years to avoid post-crisis bias
- exclusion of particularly difficult years (Great Depression, WW1, WW2)

After these exclusions, 1249 observations remain from the original dataset (2500) and constitute our baseline dataset. Of these observations, 95 have a positive class value indicating the build-up phase to 49 distinct crises.

Further processing:

- creation of the variable "slope of the yield curve", one of the most important predictors for financial crises. This is calculated as the difference between long term rates and the short ones (typically 10 years vs 3 months). An upward sloping curve generally indicates that the financial markets expect higher future interest rates; a downward sloping curve indicates expectations of lower rates in the future. This is merely an econometric interpretation.
- creation of the variable "debt servicing" that represents the interests paid on the public debt (outstanding amount of loans to the non-financial private sector x long term interest rate)
- feature normalization: credit, money, public debt, debt servicing, investment and the current account are "normalized" by the national GDP. The result is ratio-feature.
- taking the 2-year-difference to be “immune for potential issues of comparability and stationarity”
- regression filter: the features are “detrended” using a regression method proposed by Hamilton (2008) that identifies the gap between the long-term trend of a variable and the observed change. The horizon used was h = 2, and the regression was performed on the four most recent values. [1](Hamilton Filter).
- calculation of two global variables, namely "global credit growth" and the "global slope of the yield curve". These were computed for a country-year pair by the mean credit to GDP growth (mean slope of the yield curve) in all countries except c in year y.

Subsequently followed the:

- identification of the target variable: since we wanted to predict crises ahead of time, we set our binary outcome variable to positive values for one and two years before the beginning of the crisis
- removal of rows with missing values
- feature scaling (standardization)

The following table summarises the features selection and transformation:

predictor	type	measure	transformation
slope of the yield curve (difference of short and long-term interest rates)	domestic	stir - ltrate	levels
stock prices	domestic	eq_tr	2-year growth rates
real consumption per capita	domestic	rconpc	2-year growth rates
CPI	domestic	cpi	2-year growth rates
credit (loans to the non-financial private sector)	domestic	tloans/gdp	2-year differences of GDP-ratios
debt service ratio (credit x long-term interest rate over GDP)	domestic	(tloans x ltrate)/gdp	2-year differences of GDP-ratios
investment	domestic	iy/gdp	2-year differences of GDP-ratios
current account	domestic	ca/gdp	2-year differences of GDP-ratios
public debt	domestic	debtgdp	2-year differences of GDP-ratios
broad money	domestic	money/gdp	2-year differences of GDP-ratios
credit growth	global	tloans/gdp growth	levels
global slope of the yield curve	global	stir - ltrate	levels

3.2 Most influential factors - SHAP values

After the dataset is processed it is possible to investigate the most influential factors driving the algorithms in the solution to our problem. To obtain these we used, as the authors did, the SHAP value.
SHAP (SHapley Additive exPlanation) is a method used to assign each feature an importance value for a particular prediction. In particular, SHAP values quantify the magnitude and direction (positive or negative) of a feature's effect on a prediction by temporarily removing it from the model.
The SHAP values can be calculated for any tree-based model, while other methods use linear regression or logistic regression models as the surrogate models. We used the library "shap" and as explainer the implemented eXtreme Gradient Boosting.
We obtained the following two graphs:

- one with the SHAP value
- one with the SHAP Feature Importance, the average of the absolute Shapley values per feature across the dataset. Features with large absolute Shapley values are "important" for model's output interpretation.

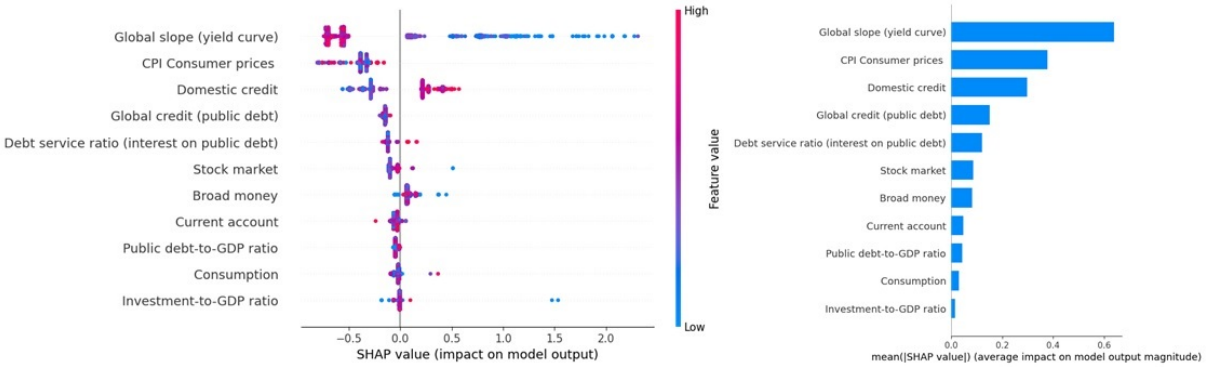

Figure 3.1: Shap value with the most influential factors

These graphs indicate which are the most influential features, in order of importance the top 5:

1. Global slope: the differential long - short interest rates (10ys vs 3mths)
2. Consumer Price Index (CPI): measure of the average change overtime in the prices paid by urban consumers for a market basket of consumer goods and services.
3. Domestic credit: the total amount of loans given to the non-financial private sector
4. Global credit growth
5. Debt service: represents the interests paid on the public debt (outstanding amount of loans to the non-financial private sector x long term interest rate)

Following the graph with the most influential factors individuated by the authors and partially overlapping with our results. The Global slope is the most importance factor (as for us), followed by Global credit growth, Domestic slope, Domestic credit, Consumer Price Index.

The authors commented: "All models consistently identify similar predictors for financial crises, although there are some variations across time reflecting changes in the nature of the global monetary and financial system over the past 150 years. These key early warning signs include:

1. prolonged high growth in domestic credit relative to GDP;
2. a flat or inverted yield curve especially when nominal yields are low, and
3. a shared global narrative in both of these dimensions as indicated by the importance of global variables.

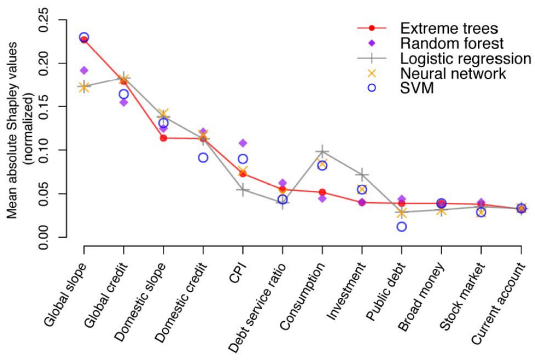

Figure 3.2: Shap value with the most influential factors - authors

While the crucial role of credit is an established result in the literature, the predictive power of the yield curve has obtained far less attention as an early warning indicator. Indeed, the global yield curve slope is a robust crisis indicator throughout the sample period. This contrasts with global credit, which proves a strong indicator only for the recent global financial crisis."

Chapter 4

Multi Layer Perceptron

The Multi Layer Perceptron is a class of feed forward artificial neural networks (ANN).

4.1 Implementation

The paper's authors used the Multi Layer Perceptron scikit-learn implementation.

4.2 Hyperparameter tuning

To obtain the set of optimal parameters for the MLP we performed a hyper parameter tuning with the following same combination used by the paper's authors:

- size of the hidden layer: following combination int(n/3), int(n/2), n, (n, int(n/2)), (n, n), (2n,n),(2n, 2n) where n is the number of the features (11)
- learning factor alpha: $10^{-3+6\times 0/9}$, $2^{-(3+6\times 2/9)}$, ..., $2^{-(3+6\times 9/9)}$
- activation function: tanH and ReLU

  tanh: it should converge quickly, but it tends to saturate (good values distribution around 0)

  ReLU: with slope is always 1 it should be fast to compute
- max iterations: 6.000
- cross validation: 10-fold
- solver: adam (default)

To evaluate the goodness of the model the AUC score was used.

4.2.1 Best structure and parameters for the MLP

The best performance of the model was obtained with the following configuration:

- input layer: size 1249 rows $\times$ 11 features
- 2 hidden layer: sizes 22, 11

- output layer: size 1249 rows $\times$ 1 target variable
- learning factor alpha: 0.0078
- activation function: ReLU
- accuracy obtained in terms of AUC: mean 0.85, standard deviation 0.13

Following the confusion matrix

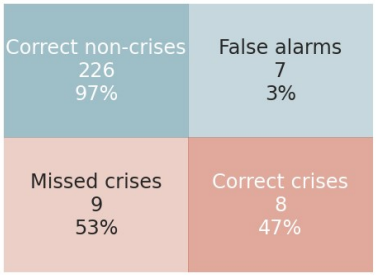

Figure 4.1: MLP confusion matrix

4.3 Predicted Crises

Following a convenient representation of the crises probability (red line) against the actual crisis (grey bar).

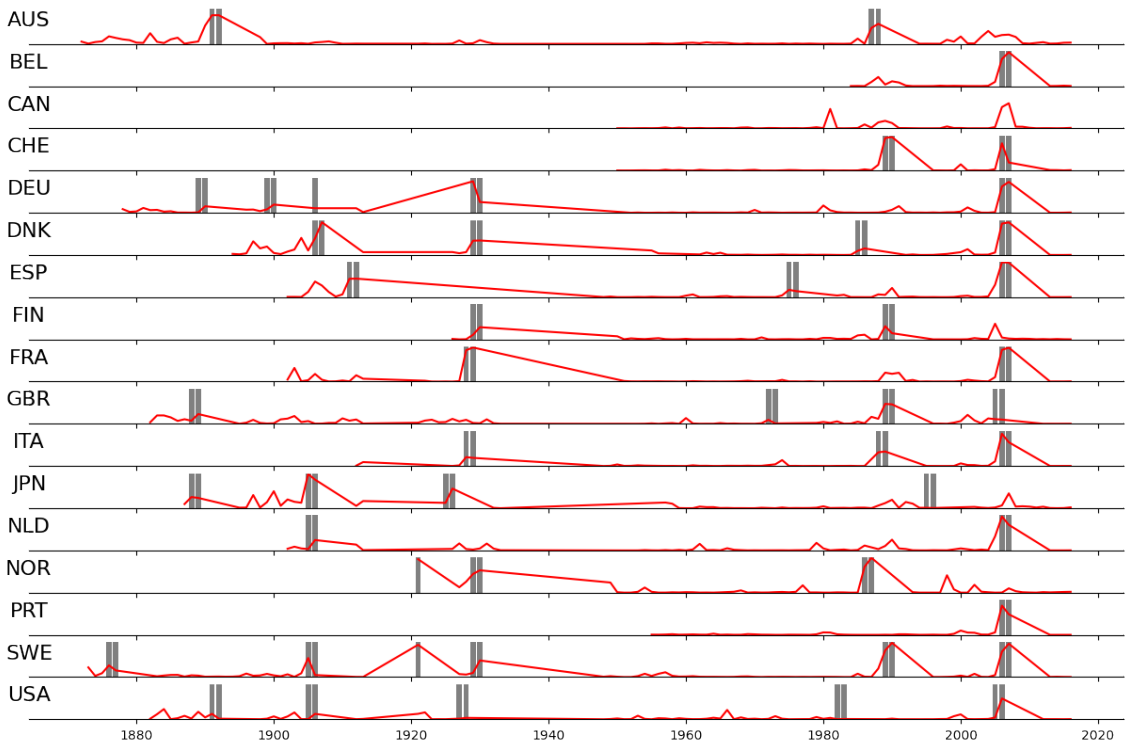

Figure 4.2: MLP predicted probabilities

Following another convenient representation of the predicted crises (green), missed crises (red), false alarm (orange) and correct not-crises (gray).

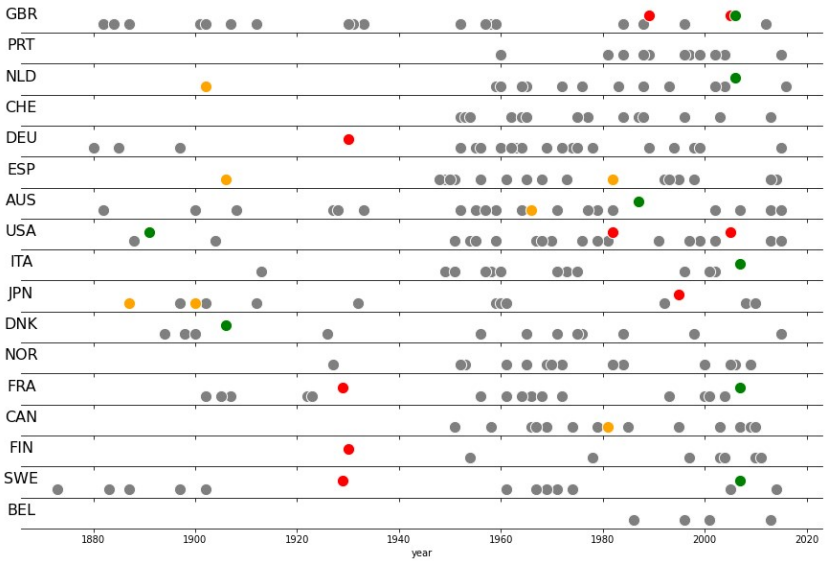

Figure 4.3: MLP timeline prediction

Just for the sake of image completeness, following the entire dataset (train + test) represented as above.

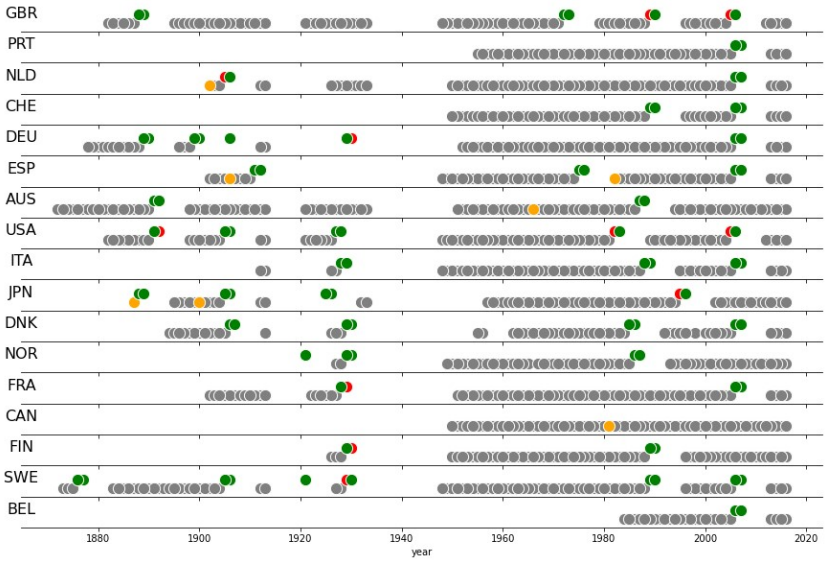

Figure 4.4: MLP timeline prediction both for the test and train dataset

4.4 Cross validation and comparison with other ML models

To validate the performance of the MLP we performed a 10-fold cross-validation. The results indicate a significant variability in the performance: the mean ROC is 0.76, with a standard deviation of 0.12; min AUC 0.6, max AUC 0.9.

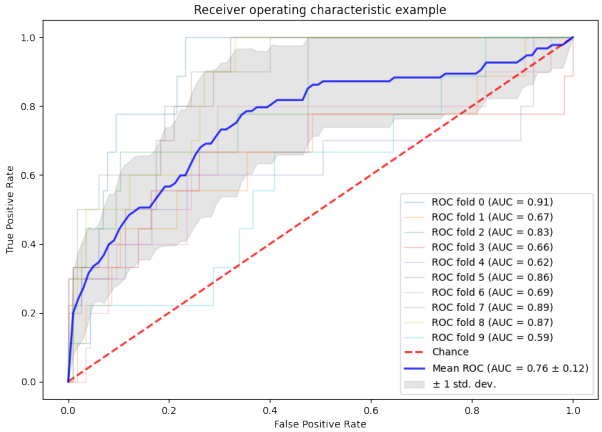

Figure 4.5: ROC curve - cross validation

The performance of the model was compared to other models (not fine-tuned because out-of-scope for the project). Again we performed a cross-validation (25-fold) to better compare the results:

- SVC
- RandomForestClassifier
- ExtraTreesClassifier
- SGDClassifier
- GradientBoostingClassifier
- KNeighborsClassifier

The performance of the optimized MLP was among the best ones but not the best in absolute. ExtraTrees and Random Forest seems the best suitable algorithms for this problem.

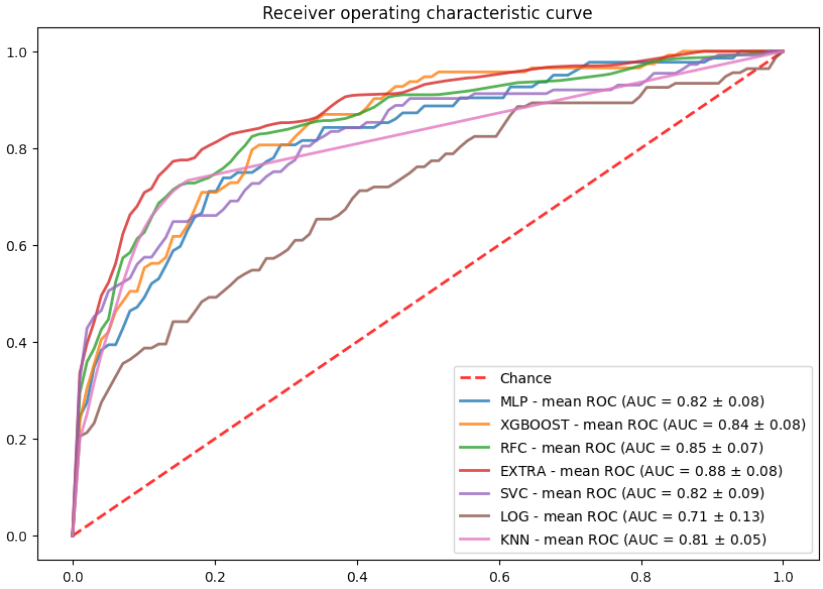

Figure 4.6: MLP vs other algorithms performance: ROC comparisons

Following the performance achieved by the papers' authors that confirm our results:

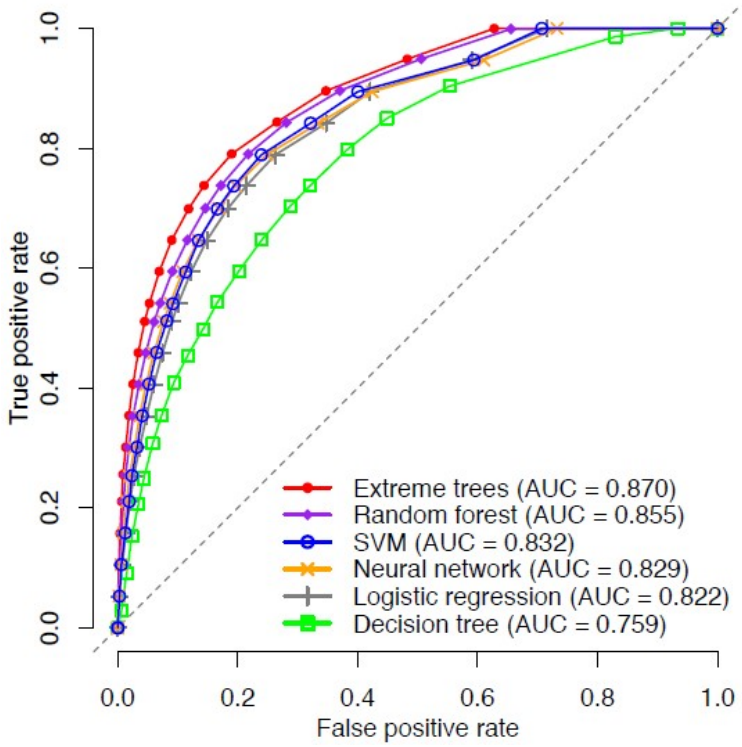

Figure 4.7: MLP vs other algorithms performance: ROC comparisons - paper

4.5 Conclusion of the reproducibility study

Although the paper doesn't mention a Github repository, we found it later in the project. This was useful for obtaining the same data preprocessing and comparing the results.
The code included was not 100 percent error-free. For example, in the MLP they use the following learning factor = $10^{np.linspace(-3.0,3.0,10)}$
The best parameters were not provided neither in the paper nor in the github repository.
However, based on our experiments, we can conlude that:

- the experiments are reproducible (we achieved the same results for the MLP) ✓
- the conclusions of the paper are largely validated by our findings ✓

Chapter 5

RNN

In an attempt to improve the results of the paper and its MLP approach, we decided to implement a Recurrent Neural Network. We thought RNNs would be very suitable for our cause given that they predict the future based on sequential input. Our goal was to forecast a time series. Each country represents a time series and based on the economic indicators we want to predict if we are approaching a financial crisis or not. This approach is more sophisticated than the one proposed in the paper by the Bank of England. Also, we did not only focus on the AUC measure, but we also tried to have reasonable results in the confusion matrix. There is no use in having a high AUC measure and a very high False Negative Rate at the same time.

5.1 Unbalanced Dataset

The dataset was not balanced. This causes problems: the model/algorithm never gets a sufficient look at the underlying class and splitting the data into train test sets is way more difficult. Therefore, we had to introduce a bias. The weights were calculated using this formula: $log(\frac{pos}{neg})$, where pos stands for the number of positive samples and neg is the number of negative samples. These weights were assigned to Keras' Constant Initializer, which was then applied on the final Dense Layer of the model.

5.2 Hyperparameter tuning

We tried to tune the hyperparameters using keras-tuner. Apparently though, keras-tuner does not work for the AUC measure we wanted to use. Therefore, we simply implemented our own rudimentary hyperparameter search using nested for loops. For creating our own hyperparameter search we researched similar models and tried to have a similar structure. In order to avoid too long running times, we decided to use 128 as number of units for the first LSTM layer and 32 units for the second LSTM layer. To obtain the set of optimal parameters for the LSTM we performed a hyper parameter tuning with a LSTM with the following characteristics:

- hiddenlayer 1, 2 sequential layer
- dropoutrate1 0.4, 0.5, 0.6
- dropoutrate2 0.4, 0.5, 0.6 (just for the 2 sequential layer)
- activation in ['sigmoid']
- learning rate 0.01, 0.005, 0.015

The result of the hyperparameter search, the structure of our final model:

Layer (type)	Output Shape	Param num	Dropout Rate	Activation function
LSTM	(None, 11, 128)	66560	-	-
Dropout	(None, 11, 128)	0	0.4	-
LSTM	(None, 32)	20608	-	-
Dropout	(None, 32)	0	0.5	-
Dense	(None, 1)	33	-	sigmoid

Total params: 87,201 Trainable params: 87,201 Non-trainable params: 0

The hypertuning was performed both for the fully preprocessed dataset and and the not fully preprocessed dataset. The total time for the elaboration was 113 minutes.

dataset	dropoutrate1	dropoutrate2	add layer	activation	learning rate	auc	time (secs)
fully prepr	0.4	0.5	TRUE	sigmoid	0.01	0.753	68
fully prepr	0.4	0.6	FALSE	sigmoid	0.015	0.742	55
fully prepr	0.6	0.4	TRUE	sigmoid	0.005	0.737	67
part prepr	0.5	0.4	TRUE	sigmoid	0.005	0.820	71
part prepr	0.4	0.5	FALSE	sigmoid	0.005	0.818	58
part prepr	0.4	0.4	FALSE	sigmoid	0.005	0.815	57

5.3 Regularization

A last exercise was performed to investigate the effect of the regularization. As the following test with/without a L2 regularization shows, during the training the regularization doesn't help. It actually prevents the model from learning. In keras we used the following parameters: kernel regularizer=l2(0.01), recurrent regularizer=l2(0.01), bias regularizer=l2(0.01).

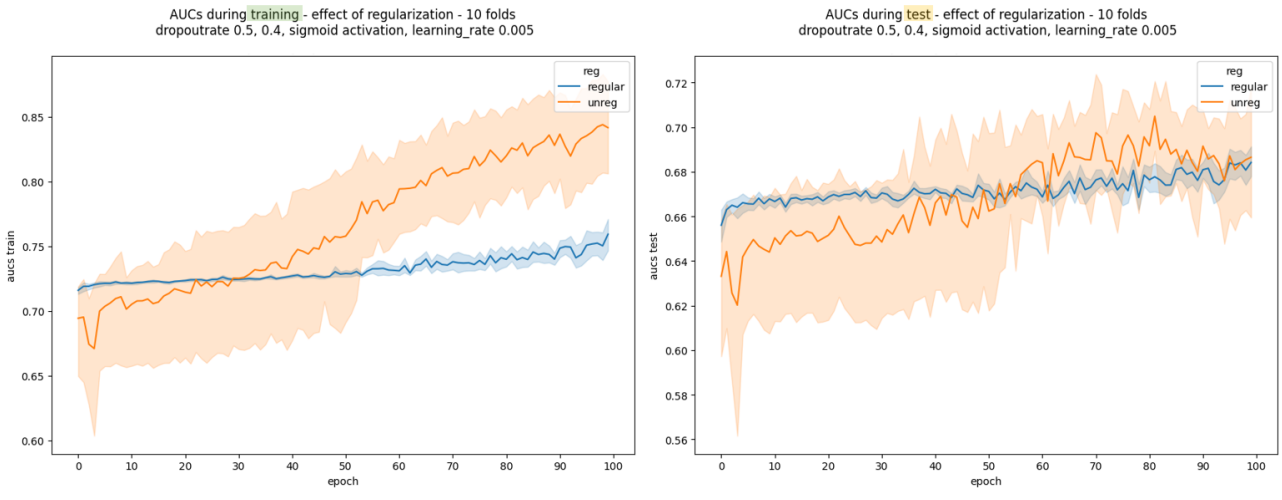

Figure 5.1: Effect of regularization

5.4 Dropout

One efficient way to deal with over-fitting, especially when the dataset is not particularly large, is to allow a random exclusion of a given amount of neurons from being activated. This dropout technique is widely accepted in the training phase. However, another approach could potentially boost performance of the model and reduce uncertainty of its results consists in letting the dropout rule being active also in inference (Monte- Carlo dropout)
The normal dropout is performed at training time only, not during test time, where all nodes/connections are present (the prediction is deterministic).
For Monte Carlo dropout, the dropout is applied at both training and test time. At test time, the prediction is no longer deterministic, but depending on which nodes/links you randomly choose to keep. Therefore, given a same data-point, your model could predict different values each time. So the primary goal of Monte Carlo dropout is to generate random predictions and interpret them as samples from a probabilistic distribution.
As displayed in figure 5.2, in our case the Monte Carlo dropout does not help in achieving a better result. The standard dropout yields better results than the Monte Carlo dropout.

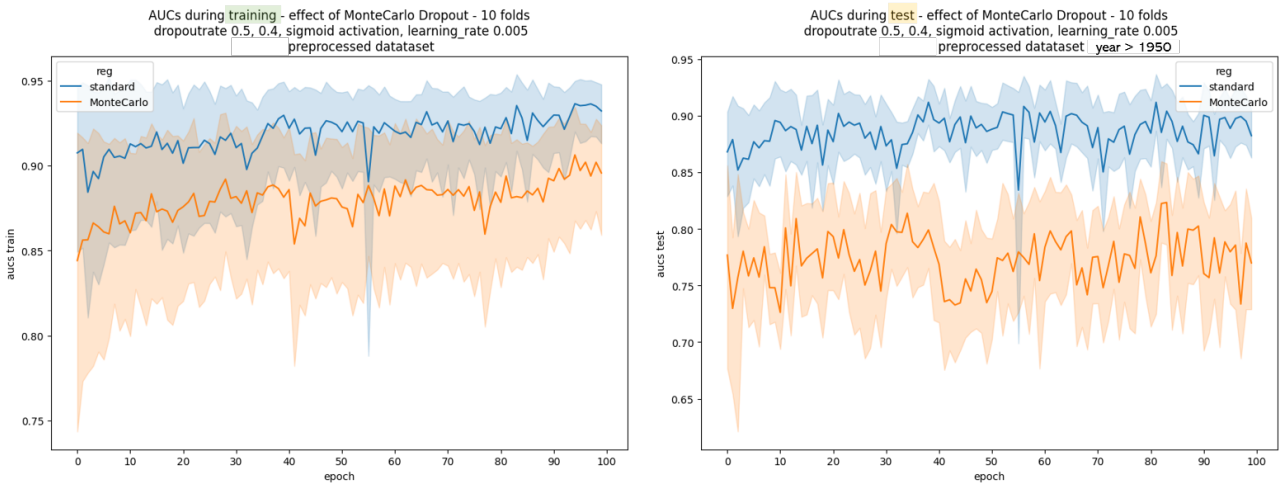

Figure 5.2: Effect of MC dropout

5.5 Year filtering

As for the MLP, we tested the effect of having a filtered dataset, after the WW2. As displayed in figure 5.3, the year filtering helps a lot.

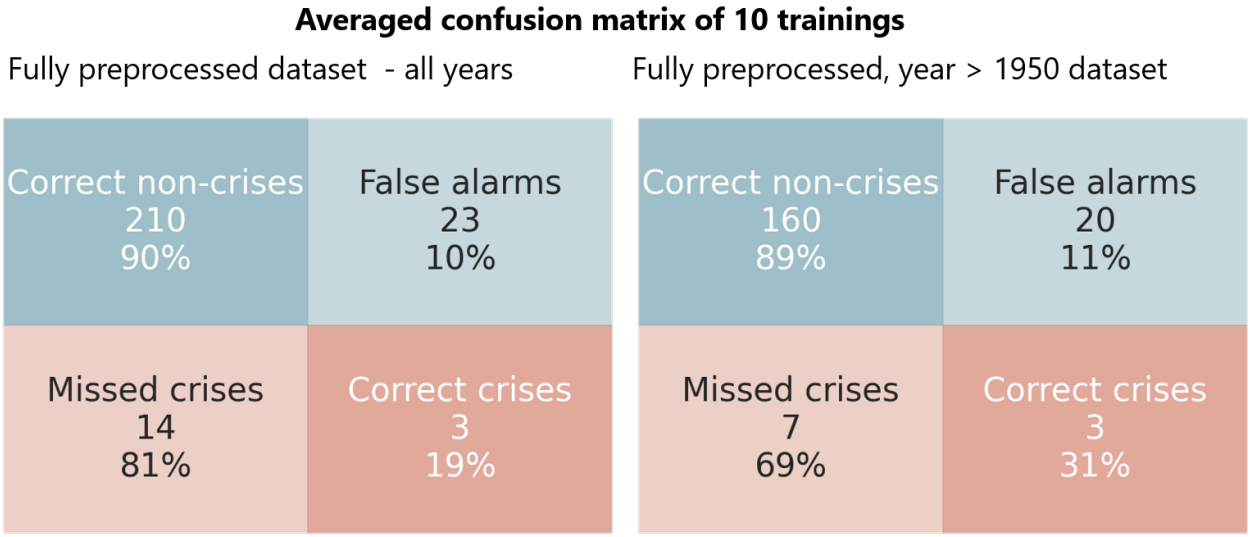

Figure 5.3: LSTM Effect of year filtering

5.6 Preprocessed Dataset

The dataset was preprocessed as described in chapter 3. The model misses many crises when using the preprocessed dataset, resulting in a poor confusion matrix.

Figure 5.4: AUC value by epoch for preprocessed data

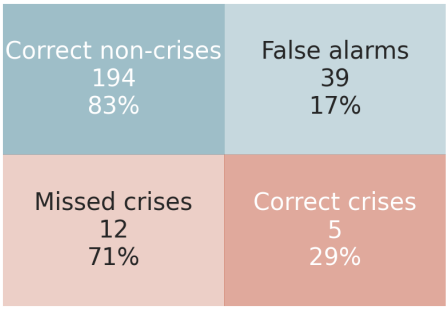

Figure 5.5: Confusion Matrix for preprocessed data

5.7 Not fully preprocessed Dataset

The results of the model are more convincing when using the dataset with less preprocessing, skipping the

- 2-year difference
- de-trending (Hamilton filtering)

This is understandable since the LSTM model is a Recurrent Neural Network where at each time step, every neuron receives both the input vector and the output vector from the previous time steps and is capable of learning order dependence.

"Recurrent neural networks contain cycles that feed the network activations from a previous time step as inputs to the network to influence predictions at the current time step. These activations are stored in the internal states of the network which can in principle hold long-term temporal contextual information. This mechanism allows RNNs to exploit a dynamically changing contextual window over the input sequence history." [2](Hassim 2014).

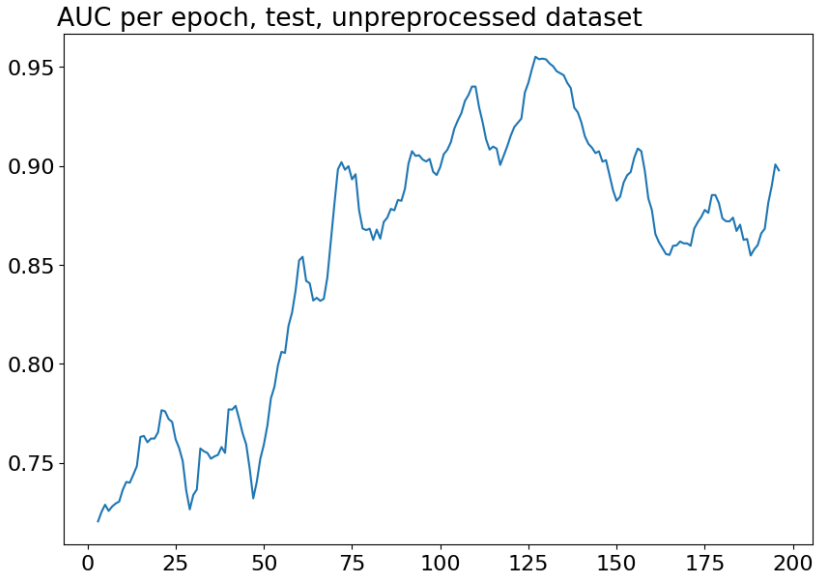

Figure 5.6: AUC value by epoch for unprocessed data

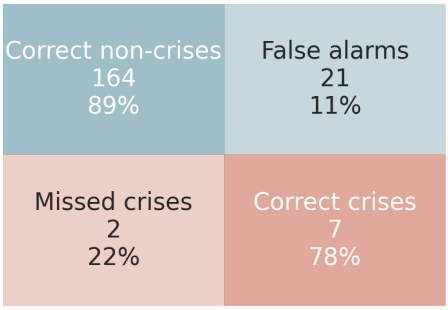

Figure 5.7: Confusion Matrix for unprocessed data

5.8 Threshold selection

The threshold is the specified cut off for an observation to be classified as either 0 (no crisis) or 1 (crisis). The standard setting is 0.5. By adjusting this threshold (on the training dataset) it is possible to achieve better model's prediction. By adjusting this threshold (on the training dataset) is possible to achieve better model's prediction. In the case of figure 5.8, the best choice would be to set the threshold to 0.05. For the train set it results in an AUC of 0.92, and for test set in AUC of 0.83. Another measure to take into consideration is the confusion matrix.

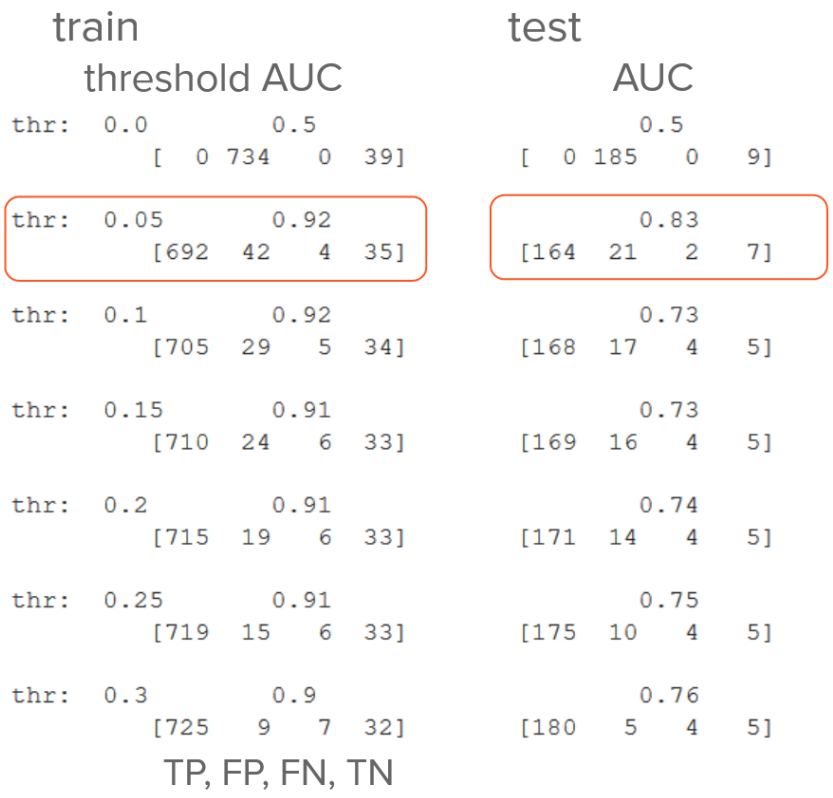

Figure 5.8: Threshold selection

Chapter 6

Conclusions and Lessons Learned

6.1 Reproducibility

- the experiments are reproducible (we achieved the same results for the MLP) ✓
- the conclusions of the paper are largely validated by our findings ✓

6.2 Discussion

The project consists of tasks of various difficulties. In the following sections we try to categorize them into easy and not so easy tasks.

6.2.1 What was easy

- replicating the results of the MLP using the sklearn libraries: the code was available on github, so we could see exactly how the data preprocessing was done. It was just a matter of creating the model and running it.

6.2.2 What was not easy

- understanding the different phases of the data preprocessing, why and how was quite unusual for people with no econometrical background: for example GDP normalization, the new variable introduced as 2-year difference in the ratio feature/GDP, Hamilton filtering)
- hyperparameter tuning: The automatic hyperparameter tuning of TF-Keras (keras-tuner) does not foresee the AUC as metric to be optimzed. The parameters are many and it is time-consuming.
- the Early Stopping condition implemented by Keras is based only on RMSE and not on AUC: this was not helpful in stopping the training as soon as the validation error reached a minimum.
- read and understand the author's python code: it is long, using too many small functions and classes. This is certainly an efficient way to run the code but not the best way to communicate with others.

6.3 Lessons learned

- Shallow networks are easier to train, but when tackling complex problems one needs to train deep neural network. A LSTM, with a dataset with not too much preprocessing achieves better performance than a Multi Layer Perceptron.
- Changing the types of dropout layers (Monte Carlo vs standard) does not significantly impact the performance
- Tensorflow Keras implementation of the NN is not so a stable/reliable library. In particular, we experienced problems with Tensorboard and keras-tuner (Hyperparameter tuning)
- The AUC measure is not a good indicator for general performance (confusion matrix)
- Choosing the right threshold is crucial for achieving the best results
- Depending on what type of model you are working with, data preprocessing may play a central role in a project
- Do not underestimate the importance of scaling
- Given the stochastic nature of the algorithm, the results may vary from one execution to the next → The cross validation is necessary to obtain robust and comparable results
- Choosing the best model often consists in making trade-offs between different measures
- Google Colab facilitates the run of small/large ML project, in particular if shared with other persons.
- Econometrics vs ML approach: the authors spent a lot of time in researching a rationale to include/ not include a feature (potential economic relevance)

List of Figures

Bibliography

[1] J. D. Hamilton, *Why you should never use the Hodrick-Prescott filter*. Review of Economics and Statistics, Vol. 100, No. 5, 2018.

[2] H. Sak, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling." `https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/43905.pdf,accessedonthe5-Feb-2021`, 2014.