TUM Data Innovation Lab
Munich Data Science Institute (MDSI)
Technical University of Munich

&

PwC Deutschland

Final report of project:

Explainable AI Applied to Dynamic Credit Risk Models

Authors	Abdullah Hesham, Ahmed Mokhtar, Gursel Naziroglu, Mingchen Wang, Papichaya Quengdaeng
Mentor(s)	Oliver Kobsik, Stephan Bautz, Sophie Mutze, Jan-Patrick Schulz
TUM Mentor	Prof. Dr. Massimo Fornasier (MDSI)
Project lead	Dr. Ricardo Acevedo Cabra (MDSI)
Supervisor	Prof. Dr. Massimo Fornasier (MDSI)

Feb 2024

Abstract

In the dynamic and complex domain of financial credit, accurately predicting loan defaults is paramount for risk management and decision-making processes. This project introduces a sophisticated predictive model that integrates Long Short-Term Memory (LSTM) networks with Reinforcement Learning (RL) and Genetic Algorithm (GA)s, creating a dynamic, adaptable, and highly accurate system for forecasting customer loan defaults. Unlike traditional static models, our approach continuously fine-tunes the LSTM hyperparameters through RL, ensuring the model's sensitivity and responsiveness to emerging data trends and economic shifts. The optimization of the RL model's hyperparameters is further refined using a GA, enhancing the model's efficiency and effectiveness in adapting to new patterns in financial credit sequential data.

A key feature of our project is the emphasis on model explainability and transparency, crucial in the financial sector where decisions such as loan application rejections require clear justification. We employ Shapley values to provide insights into the contribution of individual features to the prediction outcomes, alongside the Nearest Neighbour algorithm for comparative analysis, offering a comprehensive understanding of the factors influencing default predictions.

Our approach stands out by not only addressing the need for dynamic predictive capabilities in the face of emerging data evolutions, but also by fulfilling the financial industry's demand for transparent and Explainable AI (XAI) solutions.

Contents

1 Introduction

1.1 Credit Risk

Credit risk has become one of the key risk management challenges since the late 1990s. Globally, institutions are taking on an increasing amount of credit risk. There are generally 2 main types of credit risk: credit spread risk and credit default risk. Credit spread risk refers to the risk of financial loss due to changes caused by the volatility of credit spreads. Credit default risk is the risk that an obligor is unable to meet its financial obligations. Our project focuses on modeling and managing the credit default risk.

Expected losses have been considered a main topic in quantitative credit risk estimation. The calculation of expected losses consists of 3 risk components: Probability of Default (PD), Loss Given Default (LGD), and Exposure at Default (EAD).

$$ExpectedLosses = PD * LGD * EAD$$

Although the calculation is simple, determining the values for each input can be especially challenging for many financial institutions. Financial experts traditionally performed the estimation process, which was subject to various limitations. However, recent advancements in statistical techniques and data technologies have facilitated the development of several financial models tailored to address this challenge. Specifically, significant attention has been devoted to the prediction of PD [8].

Credit risk prediction brings forth many advantages to financial institutions, including profitable lending decisions, business steering, and predicting adequate capital reserves to manage crises. Efficient credit risk assessment offers crucial insights for decision-making processes, guiding choices such as capital investment in low-risk sectors and the maintenance of adequate capital reserves. In addition to the institution's operational stability, credit risk prediction also ensures regulatory compliance.

1.2 Problem Motivation

Time series data appears in many financial fields such as market analysis and quantitative risk management. In particular, we basically deal with multivariate time series with different lengths of sequences to predict individual PDs. This problem requires assessment in higher-dimensional data that a simple traditional linear model or logistic regression model can only partially capture its complexity [18]. Therefore, a more complicated and advanced model architecture is needed to fully tackle to problem.

Another problem with time series data is the sudden distribution shift due to unforeseen events. Market changes might occur from several unforeseeable events such as COVID-19 pandemic, wars, or bank collapses, impacting the performance of the models. To address such issue, dynamics models are designed to update the previous models more frequently whenever new data comes in. Additionally, we expect dynamic models using RL approach to respond confidently to a series of unexpected events, while making light changes to the model such that it maintains synergies to the previous one. We also take into account the regulation developments. While the original Basel framework originated in 1988, the latest iteration known as Basel III framework was developed in response to the global financial crisis in 2008. We believe that dynamic credit risk models are more flexible to align with rapid development of regulations.

Transparency and trustworthiness are also major concerns in applying complicated machine learning models in real-world decision making [6]. These models are often portrayed as a "black box" model, especially in deep neural network models which lack interpretability. In the context of financial institutions employing black-box models to determine e.g. loan approval or rejection, ensuring transparency in the rationale supporting such decisions is essential. The necessity goes beyond reduce the concerns in model acceptance; but also to comply with regulations such as the European General Data Protection Regulation (GDPR). This calls for an XAI framework to be applied to the credit risk model.

1.3 Proposed Methodology Overview

Our proposed approach aims to alleviate the shortcomings of traditional approaches by combining the powerful predictive ability of modern deep learning approaches, dynamic learning on new data with no human interaction, and the trustworthiness of explainable models. We do this by developing a deep learning model based on Bidirectional LSTM (Bi-LSTM)s. This model is fine-tuned on new data periodically to keep up with the rapid changes in the financial market.

As data shifts, the optimal hyperparameters for the model may shift as well. As a solution, we developed a RL approach to automatically re-tune the hyperparameters each time we adjust the model on new data. The idea of utilizing RL algorithms for hyperparameter tuning is based on the success of Jomaa et al. [11], although not as sophisticated. As RL agents are notoriously difficult to converge, we further employ the GA to find a suitable configuration for the RL algorithm. Using GA to assist learning is inspired by the work of Tran et al., 2016 [19] where they used a GA to extract patterns from data, although our usage of GA is different.

Finally, we combine two different approaches to explain the model. TimeSHAP [4], and a novel approach based on analyzing the nearest neighbor of each sample in space, see figure 3.

2 Background

2.1 Credit Risk Models

Financial institutions, including but not limited to banks, have applied credit risk models in more diverse portfolios with more complex products. Regulation changes also resulted in some of the recent developments in credit risk models, particularly the regulatory capital rules for credit risk based on the Basel III framework. In the European Union, the Basel III framework is implemented mainly through the Capital Requirements Regulation

(CRR) and Capital Requirements Directive (CRD). Even apart from regulatory considerations, financial institutions require increasing use of data-driven and quantitative models in their business.

There have been plenty of credit risk models developed in the financial industry. Banks have always been building frameworks for estimating credit risk, such as CreditMetrics developed by J.P. Morgan, and CREDITRISK+ introduced by Credit Suisse. [15] Some credit rating agencies, like S&P and Moody's, construct credit risk models to yield discrete ordinal groups that label firms by credit quality. Other institutions also spend a large amount of time building regulatory-compliant credit risk models.

In particular, PD models have gained popularity in recent years owing to their specific mentions in Basel II and III. Traditional PD models apply techniques ranging from logistic regression, probit/logit analysis, and hierarchical classification models. All of these methods can be shown to have some ability to distinguish high from low default likelihood firms.

2.2 LSTM for Time Series Prediction

Time series prediction with LSTM networks has emerged as a powerful and widely adopted approach in the field of machine learning, particularly for forecasting sequential data [9]. LSTMs, a type of Recurrent Neural Network (RNN), are designed to capture and remember long-term dependencies in time series, making them well-suited for tasks such as stock price prediction, weather forecasting, and energy consumption modeling.

LSTMs excel in handling the challenges posed by temporal data, where patterns and relationships can span across various time intervals. Unlike traditional feedforward neural networks, LSTMs leverage memory cells and gates to selectively remember or forget information over extended sequences, allowing them to capture both short-term fluctuations and intricate long-term patterns in time series data.

The architecture of LSTMs enables them to effectively model dynamic and nonlinear relationships present in time series. The recurrent nature of LSTMs allows the network to retain information over time steps, making them adept at learning complex temporal dependencies and making accurate predictions even when faced with irregularities or changing trends.

Training a time series prediction model with LSTMs involves feeding historical data into the network and adjusting its weights based on the prediction errors. The model then generalizes from the learned patterns to make predictions on unseen data. Hyperparameter tuning, such as the number of LSTM units, the sequence length, and the choice of activation functions, plays a crucial role in optimizing the model's performance.

The success of LSTM-based time series prediction lies in their ability to adapt to the inherent dynamics of sequential data, providing a valuable tool for researchers and practitioners across various domains. Whether applied to financial markets, meteorological phenomena, or industrial processes, LSTM networks have proven to be instrumental in capturing and forecasting intricate temporal patterns, contributing significantly to the advancement of predictive modeling in time series analysis.

2.3 Bidirectional LSTM

The idea behind Bidirectional sequnence models is to allow the flow of information in both directions. This can facilitate for the model to capture the whole context around a single point in time, from the future and the past. This concept is not new. In fact, it was introduced as far back as 1997 in the work of Schuster et al. [17]. Building on this idea is the concept of Bi-LSTMs [7]. This type of LSTMs proved pwerful in many applications such as machine translation or speech recognition.

In order to process the information from both directions, an Bi-LSTM layer uses twice as much LSTM units compared to a usual LSTM layer. The processing in both directions is inherently independent in the sense that the information from one direction does not directly influence the computations. This allows the model to reduce the impact of one direction or the other on the predictions. The forward and backward processing is done simaltniously, then combined in a certain way (usually concatination). Figure 1 illustrates this type of layer.

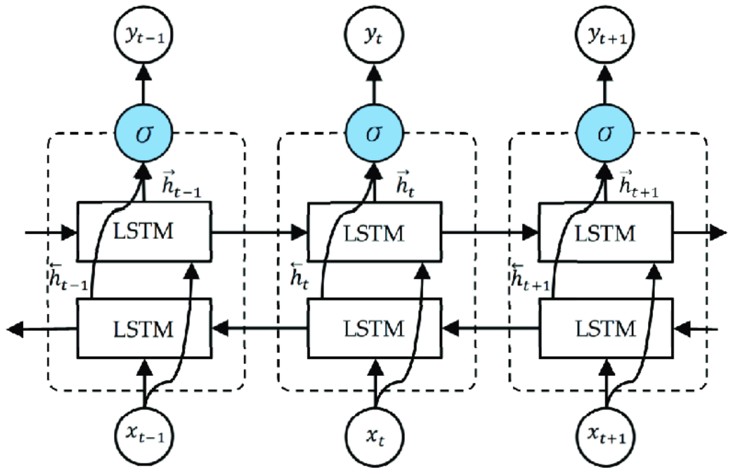


Figure 1: An unfolded 3-step Bi-LSTM layer [13].

2.4 Reinforcement Learning

2.4.1 Q-learning

Q-learning is a fundamental RL technique that has proven to be highly effective in training intelligent agents to make optimal decisions in dynamic environments [2]. Rooted in the field of machine learning, Q-learning operates by iteratively learning a quality function, represented as the Q-value, which measures the expected cumulative reward of taking a specific action in a given state. The agent explores the environment through trial and error, updating its Q-values based on the outcomes of its actions. This process enables the agent to learn a policy that guides its decision-making, maximizing cumulative rewards over time. Q-learning excels in scenarios where the agent interacts with an environment, navigating a complex state space to achieve predefined goals. Its versatility extends to applications in robotics, gaming, finance, and more. Despite its simplicity, Q-learning has demonstrated remarkable success in addressing a wide range of problems, making it a foundational tool in the realm of RL and artificial intelligence. Its adaptability and ability to learn from experience contribute to its prominence in developing intelligent systems capable of autonomous decision-making in diverse and dynamic settings.

2.4.2 SARSA

SARSA, short for State-Action-Reward-State-Action, is a RL algorithm that falls under the umbrella of temporal difference learning. Similar to Q-learning, SARSA is designed to enable agents to learn optimal policies by interacting with an environment. What distinguishes SARSA is its on-policy nature, meaning that it learns the Q-values for the policy it is currently following [2]. The algorithm is particularly well-suited for scenarios where the agent's actions influence the data it receives.

In SARSA, the agent starts in a particular state, takes an action based on its current policy, observes the next state and the corresponding reward, and then takes another action based on its policy in the new state. The Q-values are updated iteratively using the observed rewards and the Q-value of the next state-action pair. This allows the agent to refine its policy in a way that balances exploration and exploitation, ensuring a gradual convergence towards an optimal strategy.

SARSA has found applications in various domains, including robotics, gaming, and control systems. Its ability to adapt to changing environments and learn from its own experiences makes it a valuable tool for training agents to make sequential decisions in complex and dynamic scenarios. The simplicity and effectiveness of SARSA contribute to its popularity as a RL algorithm, and it serves as a foundational component in the development of intelligent systems capable of learning from interaction.

2.4.3 Expected SARSA

Expected SARSA, an extension of the SARSA algorithm, is a RL technique designed to improve the accuracy and stability of learning in dynamic environments. Like SARSA, Expected SARSA is an on-policy algorithm, meaning it learns the Q-values for the policy it is currently following.

The key distinction lies in how Expected SARSA updates its Q-values. In SARSA, the update is based on the Q-value of the next state-action pair, considering the action actually taken in the next state [2]. Expected SARSA, on the other hand, calculates the expected value over all possible actions in the next state, incorporating the probability of each action according to the current policy.

The update rule for Expected SARSA is given by:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[R + \gamma \sum_{a'} \pi(a'|s') Q(s',a') - Q(s,a) \right]$$

where:

$Q(s,a)$ is the Q-value for state-action pair (s,a),
α is the learning rate,
R is the immediate reward obtained after taking action a in state s,
γ is the discount factor determining the importance of future rewards, and
$\pi(a'|s')$ is the probability of taking action a' in the next state s'
according to the current policy.

The use of the expected value over actions provides a smoother and more stable update, reducing the impact of noise in the environment and leading to a more robust learning process. Expected SARSA has been employed in various applications, including game playing, control systems, and robotics, where reliable and stable learning is crucial for achieving optimal performance in dynamic environments.

2.5 Genetic Algorithm

GAs represent a powerful and innovative optimization technique inspired by the principles of natural selection and genetics [12]. Rooted in evolutionary biology, GAs harness the idea of survival of the fittest to iteratively search and refine solutions to complex problems. Comprising a population of potential solutions, each encoded as a set of parameters, GAs simulate the evolutionary process by applying genetic operators such as selection, crossover, and mutation. The pseudo-code (see appendix B) explains the flow of the operations.

Through successive generations, individuals with superior traits, as determined by a predefined fitness function, are more likely to pass on their genetic material, fostering the emergence of increasingly adept solutions. This iterative refinement mimics the evolutionary process, allowing GAs to explore vast solution spaces and adapt to diverse problem landscapes. Widely applicable across various domains, from optimization and machine learning to combinatorial problems, GAs offer a versatile and adaptive approach to problem-solving, capable of uncovering optimal or near-optimal solutions in scenarios where traditional methods may fall short. The inherent flexibility of GAs, combined with their capacity to explore diverse solution spaces, renders them a valuable tool in addressing complex, multidimensional challenges in fields ranging from engineering and finance to artificial intelligence and beyond.

2.6 Explainable AI

Feature Importances, particularly Permutation Feature Importance proposed by Breiman [5], is an explainability technique that was initially developed for Random Forest. Unlike other definitions of feature importance such as Gini Importance and Weight Importance limited to hierarchical models, Permutation Feature Importance is model-agnostic. It provides the relative importance of features for all input data on a global level, by estimating how much the model prediction variance changes due to the exclusion of individual features. However, it does not straightforwardly capture feature interactions.

Local Interpretable Model-agnostic Explanations (LIME), proposed by Ribeiro et al. [16], is a technique that explains the prediction of any classifier, by fitting a linear model around a prediction. LIME is fast and applicable to any type of model including text and image classifiers. However, LIME tends to provide lower accuracy as a trade-off for a simpler estimation model and its explanation is strictly local.

Shapley Additive Explanation (SHAP) Values approach was proposed by Lundberg and Lee [14] as a unified measure of feature importance using game theory. The SHAP value of a feature is calculated by taking a weighted average of the predicted output difference between when the feature is present and absent from all possible feature coalitions. The interpretation of the SHAP value is that it represents the marginal contribution of the future when it is present in the model, which can be positive or negative. A positive SHAP value means the presence of the feature value increases the prediction value, and a negative means the presence decreases the prediction value. The main disadvantage of SHAP is being unable to scale well with high-dimensional data as the complexity scales exponentially with the number of features. Implementations of SHAP are usually model-specific, for example, Linear SHAP for linear models and Tree SHAP for tree-based models, but it can be model-agnostic such as Kernel SHAP.

Combining LIME and SHAP, Lundberg and Lee also proposed Kernel SHAP [14], which estimates the local behavior of a complex model f with a linear model. Kernel SHAP is model-agnostic like LIME and maintains the local accuracy, missingness, and consistency properties of SHAP. Instead of calculating all possible feature coalitions, Kernel SHAP uses random samples of feature coalitions to estimate SHAP values, thus reducing complexity in computation. Detailed mathematical explanation of LIME, SHAP, and Kernel SHAP can be found in appendix F.

Adapting Kernel SHAP to a sequential setting, Bento et al.[4] proposed in their study TimeSHAP, extended kernel SHAP for RNNs. The main advantage of TimeSHAP is that it applies perturbations for feature attribution throughout the input sequence, considering the whole sequence instead of only one input recurrence. In addition to feature attribution, TimeSHAP is also able to calculate the contribution of an event or a single input vector in the sequence.

In addition to different feature attribution methods as XAI approach, Example-Based Explanation methods can also be deployed to provide further understanding of a model prediction [10]. An example of such methods is finding similar samples with different classification result using k-nearest neighbor model.

3 Methodology

3.1 Dataset

The synthetic dataset is generated and provided by PwC, and customer specific information is not included due to privacy reasons. Our dataset contains sequential data for 3975 customers. Each customer corresponds to a sequence with different lengths of timestamps. Compared with time series data with a single sequence, it is far more difficult to handle varying sequences with different lengths, since the sequence length is required to be equal within the same batch of the model.

3.1.1 Features

We have 32 features in total. In order to get a first insight of the dataset, we performed data exploration including but not limited to missing value analysis, correlation analysis, and outlier analysis.

3.1.2 Numerical Feature Observations

As for numerical features we have 22 company-level annual measures including financial performance and credit history, and 4 country-level annual measures for macro-economic development. Normal distribution and time series stationarity can be clearly observed in most of the numerical features.

3.1.3 Categorical Feature Observations

Given that the goal is to analyze the historical Default Flag, which indicates company defaults at a specific point in time. We investigated a class imbalance issue which actually makes sense, that only a few customers perform default behaviour in real-world conditions. Other than Default Flag, most of the other categorical features are also heavily imbalanced in our dataset.

3.1.4 Missing Value Analysis

We analyzed all the missing values in each feature in the dataset. Missing values not only cause loss of information, but also implies potential problems. Missing values identified are mapped to conditions in reality, since the realistic meaning behind is sometimes more important than the number itself.

3.1.5 Correlation Analysis

We plot the Pearson correlations between all the features in the dataset. The correlations between most features are quite weak, and the high correlations are reasonable and explainable by domain knowledge, which implicitly makes sense since the highly correlated features are metrics of the customers' financial statements and credit history. However, multicollinearity is not a problem for us, since our model is reliable to deal with highly correlated features.

3.1.6 Outlier Analysis

We applied Z-scores and quartiles respectively to detect outliers in our dataset. Z-score is primarily designed for normal distributed features and quartile performs much better with skewed features [1]. We further validate the rationality and consistency of the outliers to make decision for outlier removal.

3.2 Preprocessing steps

This section goes over all the steps taken to prepare the data for our LSTM model. Figure 2 shows an overview of the steps taken. They will be discussed further in this section.

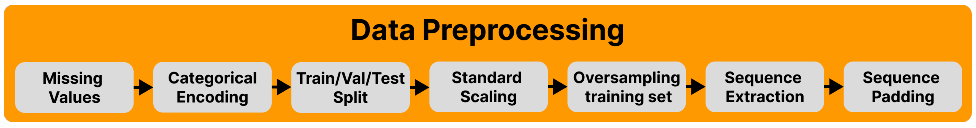


Figure 2: Preprocessing Pipeline Overview

3.2.1 Missing Value Imputation

Instead of removing the variables or omitting the entries with missing observations, which leads to loss of information, we implement some different methods to fill in the missing values. Firstly we explicitly impute missing values using an external data source of the United Nations Economic Commission for Europe (UNECE). We also use one straightforward method by simply replacing each missing observation with the mean of the variable. On this basis, we created some flag columns to indicate the missingness of certain features. In addition, we derived mathematical formulas to handle the missing values. See table 1 for a summary of imputations.

Feature	Missing	Handling
Unemployment country	Switzerland's unemployment rates from 2006 to 2010	Filled from a reliable source
Unemployment country trend (relative)	Relative unemployment rate changes in Switzerland from 2006 to 2011 and in Lithuania for the year 2006	Calculated according to a formula
Object Value Change	first year (2006) for each company	Calculated according to a formula
Object Value Change 3 Year	first year (2006) for each company	filled with Object Value Change
Tenant PD	Missing from commercial properties, given the value -9999.0	Replaced with mean (Property type serves as an indicator of misingness)
Vacancy rate	Missing from commercial properties, given the value -9999.0	Replaced with mean (Property type serves as an indicator of misingness)

Table 1: Missing values and how they were imputed

3.2.2 Categorical Encoding

We had two categorical features in the dataset. *CMS Country* and *Property Type*. The latter was label encoded, and the former was split into two features representing the average coordinate in that country. This helps capture the spatial relationships between different countries. Table 2 summarizes this step.

Feature	Old Value	Handling
CMS Country	ISO 3166-1 alpha-2 Country codes	Split into two features. CMS Country Longitude and CMS Country Latitude
Property Type	commercial or residential	Renamed to Commercial Property and values are 1 or 0 depending if commercial or residential

Table 2: Categorically encoded features

3.2.3 Train/Validation/Test Split

We split the data on the sequences, such that 30% of the customers/sequences are reserved for testing, and 70% are reserved for training. Out of those 70%, 20% are used for validation and hyperparameter tuning.

3.2.4 Standard Scaling

We fit a standard scaler on the training data, and use it to transform the training, validation, and test sets. This not only helps the model learn faster, but it is also crucial for the masking to work, this will be elaborated on further later.

3.2.5 Class Imbalance Handling

In order to balance the default and non-default sequence counts in the training set, we employ a custom random sampling technique which samples sequences instead of rows (a sequence can consist of multiple rows). In this method, we determine the number of sequences we need from the minority class in order to balance the data. We then sample from the dataset with replacement. To further augment the sampled data, those sequences are trimmed randomly to further make the model robust against varying sequence lengths. This results in a balanced dataset.

3.2.6 Sequence Extraction

In this step we group the rows of each sequence together. Each row represents a year for a certain customer. Therefore, a sequence is a single customer. After this step, the length of the dataset will be the number of customers we had. The sequcnes however, have varying number of years. There are long sequeces and short ones. We cannot divide the data into batches that way.

3.2.7 Sequence Padding

Training requires the batches fed into the model to be of the same length. To achieve this, we had to pad the shorter sequences to have the maximum sequence length. The padding value needs to be easily distinguishable from feature values. This is why scaling the inputs was crucial. The chosen padding value is -100 as it is near impossible to find a feature value 100 standard deviations away from its mean.

3.3 Approach Pipeline

Based on the literature research we did, we decided on the LSTM as our main neural network architecture. The reason behind this is the capability of LSTM to capture long-term dependencies which becomes very handy when we have time series data.

On top of our LSTM model, we applied RL to tune the hyperparameters of our deep learning model. The advantage of using RL with the deep learning model is to dynamically adapt our model to the new patterns that we might have as the new data arrives. Hyperparameters have an essential impact on the success of the model so tuning them has a significant importance.

Another important configuration of the overall architecture is the definition of the initial hyperparameters. Instead of manually defining the hyperparameters, we used the GA to make better RL hyperparameter combinations as suggested in the study [3]. The idea behind the GA is to combine the parts of the best combinations made before to get better ones. It also applies randomization to ensure we have a variety of combinations.

We also aimed to be able to explain how our model predicts our targets. Therefore, we used some explainability methods. It is very important to achieve that because deep learning is usually considered a black box and is complicated to explain what the model learns and how it concludes a specific target.

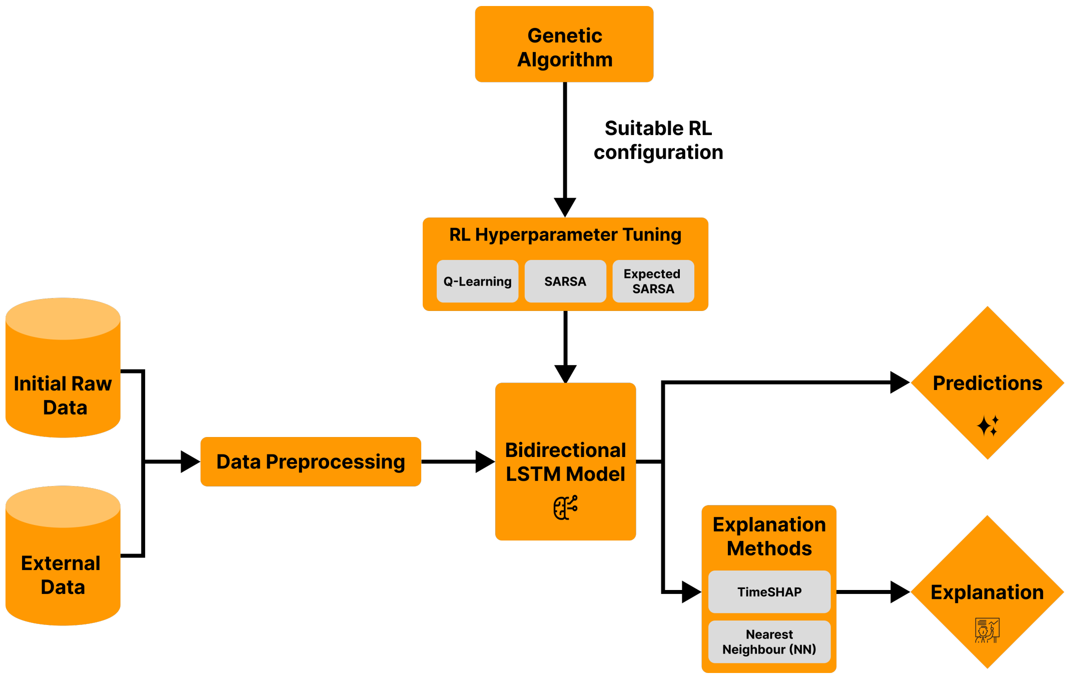


Figure 3: Model Pipeline Diagram

The figure (see figure 3) shows our architecture. The GA finds a suitable configuration for our RL model and the RL model itself tunes the hyperparameters of the LSTM model. After preprocessing the data and inputting the missing values with the help of some external resources, we apply feature engineering. Processed data goes through the LSTM model. Then, we get predictions of LSTM and apply explanation methods to interpret the behavior of the model. We can investigate these topics in more detail in the following sections to understand the reason behind using them.

3.4 Base Model

In the heart of our proposed approach is the deep learning algorithm responsible for the predictions. The patterns of the data were easily identified by sequence models. The main challenge was to minimize the bias of the network towards any of the classes in the presence of imbalance. Furthermore, the stability of training was difficult to achieve. Therefore, the Bi-LSTM model was proposed to address there issues. According to figure 4. the padded input sequence is given to the model. Any padded time-steps are quickly identified using a masking layer, hence the significance of scaling. The model input time-steps are then passed to a Bi-LSTM layer with 32 units for each direction. Each time-step returns an output of shape (batch_size, 64). This is followed by a ReLU activation. Next, a batch normalization layer is used to improve learning stability. The result is then given to a distributed dense layer of shape (batch_size, 64, 1). This means that the dense layer is applied for each time-step individually. Finally, the output of the dense layer goes through a Sigmoid activation to get the predicted probability.

Having a sequence of outputs helps with learning. The outputs of padded time-steps are ignored by the loss function (discarded). In evaluation time, only the last non-padded output is considered. Meaning that for a sequence of length 13, the model gives 13 outputs, the last output is used for evaluation purposes. Details on the training experiment can be found in table 3 and the loss curve in figure 6.

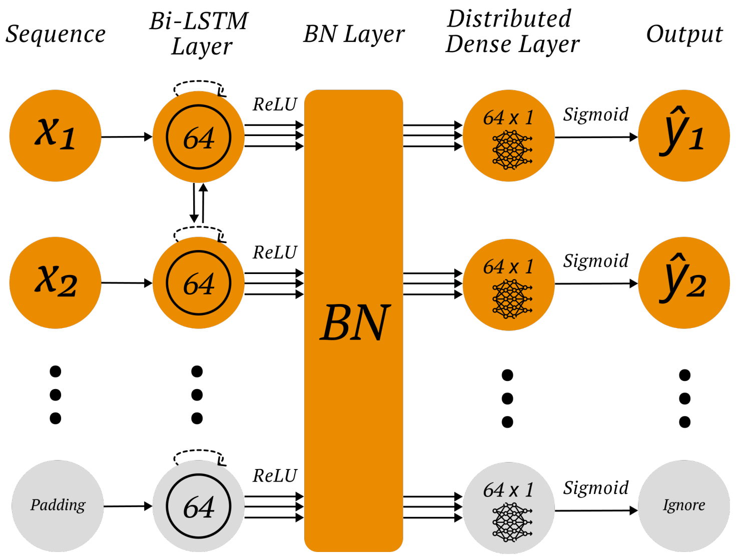


Figure 4: Proposed Bi-LSTM model

3.5 Reinforcment Learning

In our pipeline, different RL techniques were employed to optimize the parameters of the LSTM model. Specifically, we utilized Q-Learning, SARSA, and Expected SARSA algorithms for parameter tuning. The implementation of the Q-Table remained consistent across all approaches. The states within the Q-Table were discretized into a range of 0 to 100, representing the accuracy of the model. Each action within the RL framework corresponded to adjusting a specific parameter of the LSTM model. This structured approach facilitated the systematic exploration and exploitation of parameter configurations, ultimately enhancing the performance and robustness of the LSTM model.

The utilization of RL techniques, as an example of Q-Learning code, is within the context of parameter optimization for LSTM models. The presented algorithm intricately navigates the vast parameter space of LSTM architectures, systematically adjusting activation functions, LSTM unit numbers, output layer activation functions, optimizers, loss functions, epochs, batch sizes, and sequences to train. This process, encapsulated within the PreSetterModifier function, embodies the dynamic adaptation of model parameters in response to selected actions. Moreover, the PerformAction function exemplifies the seamless integration of RL with LSTM model evaluation, facilitating an efficient exploration of parameter configurations while leveraging historical performance data to expedite convergence toward optimal solutions.

Central to the Q-Learning framework is the iterative refinement of a Q-table, orchestrated by the QLearning function. This process embodies the essence of RL, as the algorithm progressively learns to associate actions with their respective outcomes, thereby guiding future decisions toward favorable parameter settings. Through an epsilon-greedy policy for action selection, the algorithm strikes a delicate balance between exploration and exploitation, ensuring thorough exploration of the parameter space while exploiting known high-performing configurations. Furthermore, the select_action and update_q_table functions play instrumental roles in action selection and Q-value updates, respectively, contributing to the iterative improvement of the Q-table and, consequently, the overall optimization process.

and the differences between the three algorithms are as follows:

1. **Q-Learning**:
   - **Nature**: Q-Learning is an off-policy learning algorithm.
   - **Update Rule**:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right]$$

   - **Policy**: Learns the optimal policy while following a behavior policy that explores the environment.
   - **Exploration vs. Exploitation**: Achieves exploration by choosing actions that maximize long-term rewards based on learned Q-values.
   - **Advantages**: Can converge to an optimal policy even with exploration strategies that might not be optimal.
   - **Disadvantages**: May take longer to converge compared to on-policy methods in environments with high variance.

2. **SARSA** (State-Action-Reward-State-Action):
   - **Nature**: SARSA is an on-policy learning algorithm.
   - **Update Rule**:

     $$Q(s,a) \leftarrow Q(s,a) + \alpha \left[r + \gamma Q(s',a') - Q(s,a)\right]$$

   - **Policy**: Learns the value of the policy that is being followed, making updates based on the policy's actions.
   - **Exploration vs. Exploitation**: Balances exploration and exploitation by incorporating the current policy's action selection strategy.
   - **Advantages**: More stable than Q-Learning in environments with high variance, as it directly updates Q-values based on the current policy.
   - **Disadvantages**: May converge to a suboptimal policy if the exploration strategy does not sufficiently explore the environment.

3. **Expected SARSA**:
   - **Nature**: Expected SARSA is an on-policy learning algorithm.
   - **Update Rule**:

     $$Q(s,a) \leftarrow Q(s,a) + \alpha \left[r + \gamma \sum_{a'} \pi(a'|s') Q(s',a') - Q(s,a)\right]$$

   - **Policy**: Learns the value of the policy being followed, similar to SARSA.
   - **Exploration vs. Exploitation**: Balances exploration and exploitation by considering the expected value of future actions.
   - **Advantages**: Tends to have smoother convergence and potentially greater stability compared to SARSA, particularly in environments with stochasticity.
   - **Disadvantages**: The computational complexity may be higher due to the need to compute expected values over all possible actions.

Our study exemplifies the effectiveness of RL techniques in optimizing LSTM model parameters. By systematically exploring and exploiting parameter configurations, our approach enhanced the performance and robustness of LSTM models. The judicious selection and integration of Q-Learning, SARSA, and Expected SARSA algorithms provided insights into optimal parameter settings, underscoring the versatility of RL in model optimization. This approach signifies a paradigm shift in automated model tuning, empowering practitioners with powerful tools to navigate the complexities of machine learning model development and deployment.

3.6 Genetic Algorithm

To further increase the adaptability of the model training, a GA is used as it selects the optimal hyperparameters for the RL model. The algorithm execution commences with initializing a population of potential solutions. Each solution is a dictionary representing a set of hyperparameters that includes a selected algorithm (Q-Learning, SARSA, Expected SARSA) and related parameters such as learning rate, discount factor, and exploration probability. It also includes an initial hyperparameter set for the LSTM model.

The fitness score of each solution is evaluated by employing a RL model that runs LSTM models to optimize its hyperparameters to reach optimal performance in terms of the evaluation metrics. If the solution has been processed before, it is skipped to save computation time; otherwise, the appropriate RL function is invoked based on the algorithm specified in the solution. Based on the fitness scores, the most fit individuals are selected to be parents for the next generation. Then the new children are generated by either crossover or mutation applied to the parents with the same probability.

The crossover combines pairs of parent solutions. In case the two solutions have the same RL algorithm, we crossover the lists of both solutions that are the initial hyperparameter setters of our LSTM model and average the other parameters to produce a new solution. Otherwise, we swap all parameters of the two solutions except the RL algorithm and get two new solutions considered children. The idea behind this approach is to apply the exact hyperparameter set on the other RL algorithm to observe its performance considering the high chance of leading to a good result with a parameter combination that already performed well on the other algorithm.

On the other hand, mutation introduces random changes to a solution to explore the search space further. This the probability of mutation rate, we select a random value among possible values for each parameter and hyperparameter set of the LSTM model to generate a new solution.

Over successive generations, the algorithm iteratively selects the best solutions, applies crossover or mutation, and evaluates the resulting population, aiming to maximize the fitness function. The best solution found during the process is returned as the optimal set of hyperparameters.

3.7 Explainability

Explainability helps build trust by providing insights into the model's decision-making process, making it more transparent and interpretable. Moreover, it can also help identify and mitigate biases in models by revealing how certain features impact predictions. SHAP approach can provide a local explanation from feature contribution values, and can further explain the model globally by aggregating for average contributions. Then an example-based explanation approach is employed using nearest neighbor algorithm to provide insights that are easy to understand.

3.7.1 SHAP

To apply SHAP to our credit risk model, we decided to use the TimeSHAP library by Bento et al. [4] which already extended Lundberg and Lee's KernelSHAP for RNN. TimeSHAP takes baseline data as uninformative values to use in perturbing the sequence and estimating the SHAP values. In this implementation, the average value of each feature (normalized and unpadded) is used as the baseline. Since Kernel SHAP is a local explanation method, it takes an input instance without padding, prediction function f, and baseline event to calculate SHAP values as well as other parameters such as random state and number of feature coalition samples.

With the predefined function from the TimeSHAP library, we can calculate event-level, feature-level, and cell-level local explanations for each input instance. Event-level SHAP value explains the contribution of each event of recurrence, which in this case is one year of client data in the sequence. Feature-level SHAP value explains the contribution of each feature across the sequence. Lastly, the Cell-level explanation provides further detail in the contribution of each recurrent unit, representing a feature at a certain time event.

To understand the driving factor of a model version at a global level, individual SHAP values from each input instance in the dataset can be collected and aggregated for mean SHAP values. This can be applied to event-level and feature-level explanations. Global feature attribution can be explained from the average SHAP values of each feature across all input instances, then we select features with top absolute SHAP values to investigate as the top driver of the prediction. The same applies to event-level explanation which can be aggregated to global event attribution to investigate how older years of data are no longer relevant or have a negative contribution to the prediction.

3.7.2 Nearest Neighbor

In addressing the challenge of explaining the outputs of an LSTM model, we first employed the SHAP technique. While SHAP provided insights into the model's decision-making process, it exhibited limitations in delivering easily understandable answers, often requiring expertise to interpret the model's decisions. To enhance interpretability, we developed an algorithm inspired by the principles of k-nearest neighbors, which we refer to as the "nearest neighbor for explainability" approach. This methodology involves comparing the non-default output with default answers already present in the dataset. By assessing the similarity between the current output and existing default answers, the algorithm quantifies the accuracy of the match and identifies the attributes influencing the decision. Subsequently, the algorithm generates natural language explanations that are comprehensible to non-technical individuals, facilitating an intuitive understanding of the model's reasoning process. This approach enables stakeholders to grasp the underlying factors driving the model's outputs and anticipate how changes in input attributes may impact the outcomes, ultimately promoting transparency and informed decision-making.

4 Results and Discussion

4.1 Base Model

For the results of the base model, we further evaluate the experiment mentioned in section 3.4 and table 3. Here, we evaluate the model on unseen sequences from the same period as the training data (2006-2018). Only the outputs of 2018 are considered in the metrics and figures.

After evaluating on the test set containing 1166 customers/sequences. We can see from the confusion matrix in figure 7 that even though though the false negatives are not many relative to the dataset size, it has a big impact on performance, as it's a sizable portion of the positive class. We can see that the false positives are not many as well and well. Looking further intro the metrics in table 4, we can see the difference between the performance in the default and non-default class. This is attributed to the fact that the default class is much rarer and harder to detect than the non-default class.

Lastly, we can take a look at the calibration curve in figure 8 to get insight on how realistic the probabilities outputted by the model are. We can see slight under-confidence in probabilities. Meaning that the average probabilities are slightly less than the observed portion of defaults. There are many ways to calibrate model probabilities, and should be considered in future works.

4.2 Entire Approach

In in the previous experiment, we evaluated the model individually on historic data. This experiment aims to simulate dynamic learning. This is done by splitting the data into 5 periods. Each period is a 5-year window except the first one. The first period is the initial 2006 to 2018 period. We run the GA-RL approach on a newly initialized model to get the best hyperparameter for this particular period. The hyperparameters optimized in the model are:

- The activation function after the Bi-LSTM layer: [Relu, Sigmoid, Tanh]
- Choice of optimizer: [Adam, SGD, RMSProp, AdaGrad]
- Number of epochs to tune the model on: [5, 10, 20, 50]
- The batch size: [32, 64, 128]

First results are seen in figure 9 and table 5. Naturally the highest performing set, is the one that covers the training period till 2018. Seeing the performance on future years, we can see that the model is effective in finding defaults, but it makes a sizable number of false positives in the process. This might be due to new observations that look like previous defaults, but are in fact not defaults. This suggests a shift happening in data starting in 2019.

The next period of training covers the years 2015 to 2019. In similar fashion to the previous period, the GA-RL approach runs to determine the most suitable hyperparameters for this period. However this is done on top of the model from the previous year. We do not reinitialize weights. Instead, we fine-tune them on the new data. It can be seen (figure 10, table 6) that the false negatives worsen, but the false positives get better. This might be due to the model training on the non-defaults that looked like defaults previously. In turn, the model got less confident in these defaults, classifying them wrongly as non-defaults. On the other hand, the number of false positives got cut by almost half in the datasets form 2019 to 2020.

The third period of training is from 2016 to 2020. From the results (figure 11, table 7), we are close to eliminating false positives entirely, in addition to fewer false negatives as well. By looking at the 2018 matrix, we can notice that the model is starting to forget the initial training period in favor of the present and future data. This is a desirable property, since markets change and evolve. We do not want the model to be stuck in the past.

The second to last period of this experiment is the period from 2017 to 2021. We can notice (figure 12, table 7) further forgetting in the model especially in the year 2019. The present dataset (2021) has also gotten slightly worse in terms of false positives. This might be due to the model forgetting important patterns that were present in the past data, along with the present testing set. However these patterns were not present in the current training set. In contrast, the future (2022) dataset is doing better.

Finally, we reach the last training period of this experiment. The period of 2018 to 2022. in figure 13 and table 9, we notice that the model now performs best for the current year (2022). The model also continues to forget past behaviors. We can see some past years improving as well. The reason for this might be due to the model picking up forgotten patterns that are shared between a past dataset and the present training set.

This concludes the experiment. The above explanations are merely educated guesses on how the model behaved in the experiment. Some other factors might have played a hand in the behavior. These factors will be discussed in the future works as areas of improvement.

4.3 Explainable AI

After the model is retrained into different versions, we apply XAI techniques to explain how the different model versions predict different results. The explainability dashboard was created using Streamlit to show the explanations of each model version on different datasets. The dashboard contains 2 pages: Model Overview and Local Explanation.

The "Model Overview" page (figure 14) provides a comparison between two model versions with global feature attribution and global event attribution plots, where the user can select model versions and dataset versions from the pre-generated files to see the plots. Global feature attribution plot (figure 15) shows SHAP values distribution for the top 15 features with the highest absolute mean SHAP of the selected model predictions on the selected dataset. Blue dots are individual values while red dots are the mean value of the feature. The top features are the main drivers of the predicted PD, contributing in a positive or negative direction. Global event attribution plot (figure 16) shows SHAP values distribution for each event (year of data) in the sequence of the selected model predictions on the selected dataset. Event index -1 means the latest year in sequence and the lower index means the older year. Green dots are individual values and red dots are the mean value of event.

An example use case of model overview comparison is shown in figure 15 and figure 16. The first model (bi_lstm[2006, 2018]) was trained as the initial base model using historical data from the year 2006 to 2018 and the second model (bi_lstm[2018, 2022]) was retrained using the dynamic model approach described in section 3.5 using latest data for 2022. The dataset used for calculating SHAP values for both models is the data with sequences from 2019 to 2022. We can see from figure 15a that the top risk driver of the first model is "Covenants Breach last 12 Month". While the same feature is still among the top drivers in the second model (see figure 15b), "Financial LTV" became the strongest driver. Moreover, even though the top 5 features remained the same, most of the next 5 top features had changed. This implies the possible change in underlying patterns of defaults.

The "Local Explanation" page (see figure 17) provides an in-depth explanation of a single input instance (one client data). The user can select a model version and data version to use for explanation, and then select a client ID in the selected dataset to generate explanation plots. After a client ID is selected, the data of that client is shown as a data frame to see what are the feature values and default flags of the client. The prediction value of the client is also shown on the dashboard, the value is the predicted PD of the last year in sequence from the dataset. Then the user can view event-level, feature-level, and cell-level explanations. Event-level explanation (figure 18a) is shown as a heatmap of the SHAP value of each event (year of data) in a sequence of the selected client ID. Feature-level explanation (figure 18b) is shown as a bar plot of the SHAP value of the top 15 features with the highest absolute SHAP value. And cell-level explanation (figure 18c) is represented with a heatmap of the contribution of the top 3 features at the top 3 events. Each cell represents the SHAP value of a feature at an event in the sequence.

If the predicted value of an input instance is lower than 0.5, which is considered predicted as non-default, the dashboard will calculate and output the explanation with the Nearest Neighbor method (see figure 19).

5 Conclusion

The strength of our project presents a sophisticated, multi-layered approach to predicting loan defaults, leveraging the strengths of LSTM networks, RL, and GAs to create a dynamic and adaptable model capable of adjusting to new financial trends. Our deep learning model, rooted in LSTM, is adept at processing and learning from sequential credit data, providing a robust framework for predicting customer defaults with high accuracy.

The model's dynamic nature is critical in the financial sector, where data evolves and reflects yearly market trends and economic shifts. Traditional static models are often inadequate for capturing such changes, leading to outdated predictions. Our approach counters this by using RL to continually fine-tune the LSTM hyperparameters, ensuring the model remains sensitive to fresh patterns and variances in the data.

Further enhancing the model's adaptability, a GA was employed to optimize the hyperparameters of the RL model itself. This meta-optimization process guarantees that not only is our primary model performing optimally at any given point, but also that the underlying RL model is operating with the most effective configuration for adjusting the LSTM. The resulting algorithm synergy creates a powerful predictive tool that self-adjusts in a principled and methodical manner.

Transparency and explainability were paramount in our methodology, addressing the financial industry's demand for clarity in decision-making processes, such as loan application outcomes. By incorporating Shapley values, we illuminate the "why" behind each prediction, offering insights into the contribution of each feature to the model's decision. Similarly, the Nearest Neighbour algorithm serves as a comparative tool, elucidating the characteristics that distinguish default from non-default predictions.

Comparison of different models from different years helps us understand the key factors changed in the model's decision-making process. The changes in the average Shapley values for each feature from each year lead us to have insight into the trend changes in the data. This in-depth analysis is visualized by our dashboard which also eases the interpretation of the reasoning behind each decision made for each customer.

There are certain areas that we wished to explore more in order to improve our approach. We believe that these recommendations will yield better results. Our RL approach is simple, and does not cover much of the hyperparameter search space. We believe that by employing more state-of-the-art techniques for hyperparameter tuning, we can get better and more consistent results. One other thing we were shy of doing, is to compare our approach to other available approaches. This would have given us better insights on where our model stands in the current market. Finally, including other XAI approaches in the explainability framework would have increased the trustworthiness of our model, and provided a more comprehensive assessment of the model.

In conclusion, our innovative combination of algorithms not only yields a model with superior predictive capabilities but also satisfies the crucial industry requirements of dynamism and explainability. The model's ability to adapt and learn from new data trends ensures its long-term applicability and accuracy in the ever-evolving financial landscape. Simultaneously, the integration of explainability techniques addresses the need for transparency, fostering trust and understanding in the automated decision-making that is increasingly pivotal in financial institutions. This project, thus, aligns with the ethical imperative of XAI, setting a new standard for responsible and dynamic financial modeling.

References

[1] Charu C Aggarwal and Charu C Aggarwal. *An introduction to outlier analysis.* Springer, 2017.

[2] Alex M Andrew. "REINFORCEMENT LEARNING: AN INTRODUCTION by Richard S. Sutton and Andrew G. Barto, Adaptive Computation and Machine Learning series, MIT Press (Bradford Book), Cambridge, Mass., 1998, xviii+ 322 pp, ISBN 0-262-19398-1,(hardback,£ 31.95)." In: *Robotica* 17.2 (1999), pp. 229–235.

[3] Uzair Aslam et al. "An empirical study on loan default prediction models". In: *Journal of Computational and Theoretical Nanoscience* 16.8 (2019), pp. 3483–3488.

[4] João Bento et al. "Timeshap: Explaining recurrent models through sequence perturbations". In: *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining.* 2021, pp. 2565–2573.

[5] Leo Breiman. "Random forests". In: *Machine learning* 45 (2001), pp. 5–32.

[6] Niklas Bussmann et al. "Explainable machine learning in credit risk management". In: *Computational Economics* 57 (2021), pp. 203–216.

[7] "Erratum". In: *Artificial Neural Networks: Formal Models and Their Applications â€" ICANN 2005.* Springer Berlin Heidelberg, 2005, E1â€"E1. ISBN: 9783540287568. DOI: 10.1007/11550907_163. URL: http://dx.doi.org/10.1007/11550907_163.

[8] Petr Gurnỳ, Martin Gurnỳ, et al. "Comparison of credit scoring models on probability of default estimation for us banks". In: *Prague economic papers* 22.2 (2013), pp. 163–181.

[9] Yuxiu Hua et al. "Deep learning with long short-term memory for time series prediction". In: *IEEE Communications Magazine* 57.6 (2019), pp. 114–119.

[10] Sheikh Rabiul Islam et al. "Explainable artificial intelligence approaches: A survey". In: *arXiv preprint arXiv:2101.09429* (2021).

[11] Hadi S. Jomaa, Josif Grabocka, and Lars Schmidt-Thieme. "Hyp-RL : Hyperparameter Optimization by Reinforcement Learning". In: *CoRR* abs/1906.11527 (2019). arXiv: 1906.11527. URL: http://arxiv.org/abs/1906.11527.

[12] Manoj Kumar et al. "Genetic algorithm: Review and application". In: *Available at SSRN 3529843* (2010).

[13] Yunghui Li et al. "Real-Time Cuffless Continuous Blood Pressure Estimation Using Deep Learning Model". In: *Sensors* 20 (Sept. 2020). DOI: 10.3390/s20195606.

[14] Scott M Lundberg and Su-In Lee. "A unified approach to interpreting model predictions". In: *Advances in neural information processing systems* 30 (2017).

[15] JP Morgan et al. "Creditmetrics-technical document". In: *JP Morgan, New York* 1 (1997), pp. 102–127.

[16] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "'Why should i trust you?' Explaining the predictions of any classifier". In: *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining.* 2016, pp. 1135–1144.

[17] M. Schuster and K.K. Paliwal. “Bidirectional recurrent neural networks”. In: *IEEE Transactions on Signal Processing* 45.11 (1997), pp. 2673–2681. DOI: 10.1109/78.650093.

[18] Si Shi et al. “Machine learning-driven credit risk: a systemic review”. In: *Neural Computing and Applications* 34.17 (2022), pp. 14327–14339.

[19] Khiem Tran, Thanh Duong, and Quyen Ho. “Credit scoring model: A combination of genetic programming and deep learning”. In: *2016 Future Technologies Conference (FTC)*. 2016, pp. 145–149. DOI: 10.1109/FTC.2016.7821603.

Appendix

A Acronymms

Bi-LSTM	Bidirectional LSTM
GA	Genetic Algorithm
LSTM	Long Short-Term Memory
PD	Probability of Default
RL	Reinforcement Learning
RNN	Recurrent Neural Network
XAI	Explainable AI

B Genetic Algorithm Pseudo-Code

1. Initialize Population:
    - Generate an initial population of individuals with random genes.
    - Each individual represents a potential solution to the problem.
2. Evaluate Fitness:
    - Calculate the fitness of each individual in the population.
    - The fitness function evaluates how well each individual solves the problem.
3. Repeat for a Fixed Number of Generations or Until Convergence:
    (a) Selection:
        - Select individuals from the population based on their fitness.
        - Individuals with higher fitness are more likely to be selected.
        - Common selection methods include roulette wheel selection or tournament selection.

    (b) Crossover (Recombination):
        - Pair selected individuals and perform crossover to create offspring.
        - Crossover combines genetic material from two parents to produce new individuals.
        - Common crossover methods include one-point or two-point crossover.

    (c) Mutation:
        - Apply mutation to some of the selected individuals.
        - Mutation introduces small random changes to individual genes.
        - Mutation helps explore new regions of the solution space.

    (d) Evaluate Fitness of Offspring:

    - Calculate the fitness of the new offspring individuals.

   (e) Replacement:

    - Replace old individuals in the population with new offspring having better fitness.
    - The new population is most likely to be a mixture of parents and offspring.

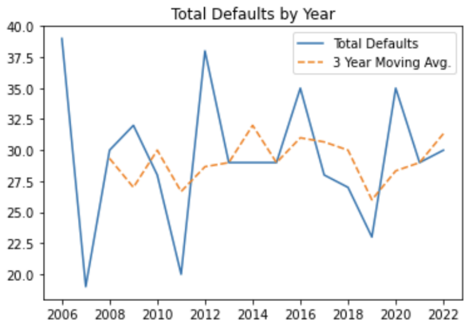
   (f) Termination Check:

    - Check if a termination condition is met (e.g., a maximum number of generations or a satisfactory fitness level).
    - If the termination condition is met, exit the loop.

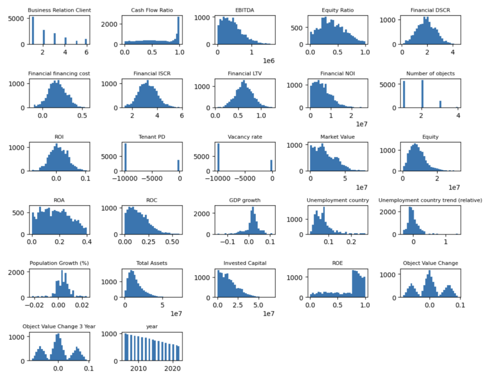
4. Output Result:

   - Output the best individual or the population's statistics as the final result.

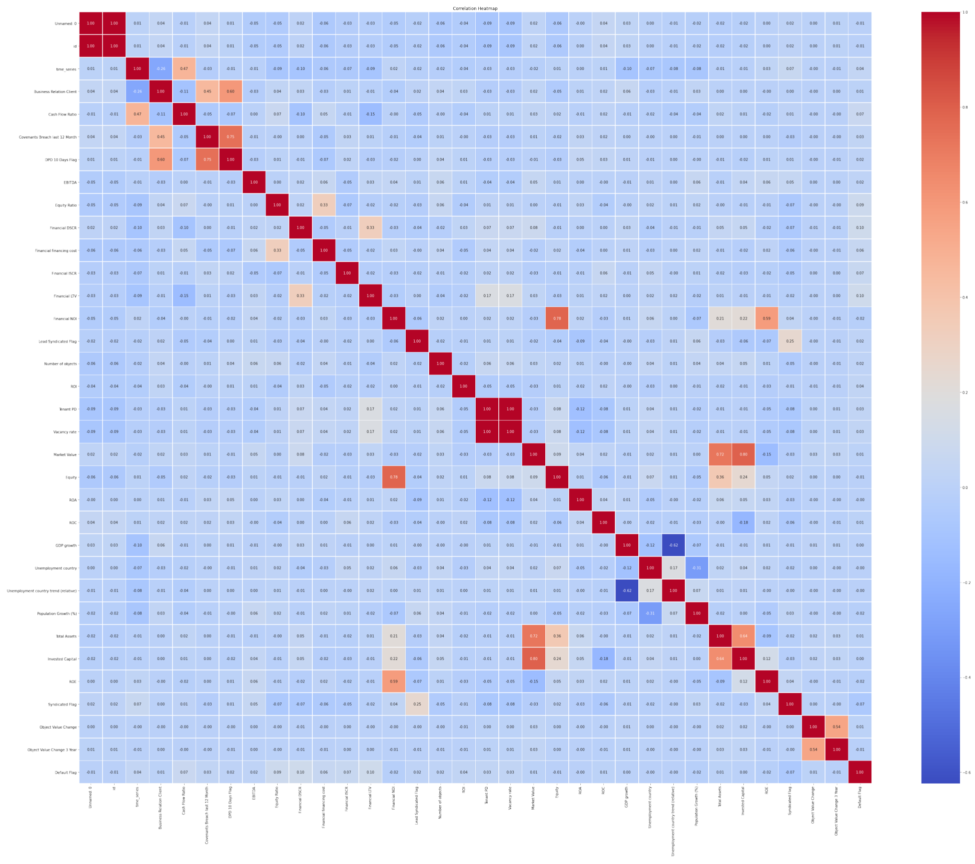
C Exploratory Data Analysis Results



(a) Defalut Moving Average



(b) Numerical Feature Distribution



(c) Correlation Matrix Visualization

Figure 5: Visualizations in data exploration

Configuration	Value
Dataset	data from 2006 to 2018
Optimizer	Adam
Sequence Length	13
Batch Size	32
Learning Rate	0.0001
Epochs 50	50

Table 3: Base model training experiment configuration

D Base model Training Experiment

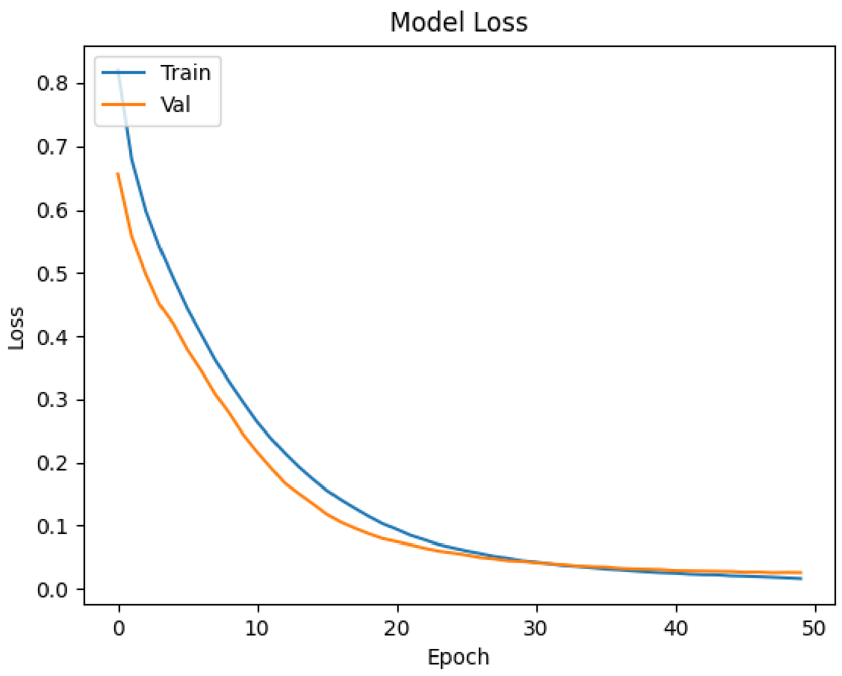


Figure 6: Base model training loss curve

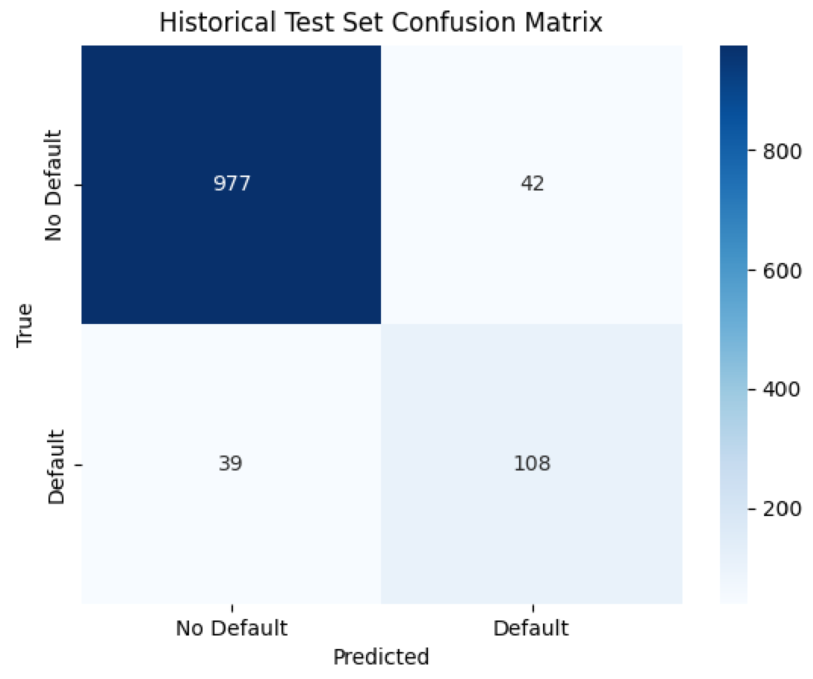


Figure 7: Base model confusion matrix on testing set. Both false positive and false negatives have a similar count. However false negatives have bigger impact due to the small number of positive samples.

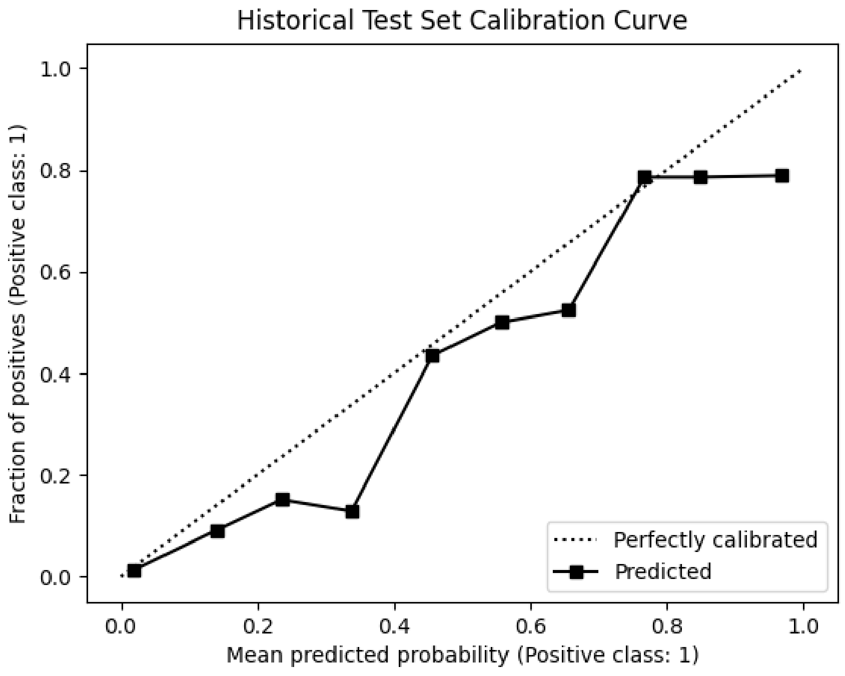


Figure 8: Base model calibration curve on the testing set. the model shows slight under-confidence in probabilities.

Metric	Default	Non-default	Macro-Average
Precision	0.7152	0.9962	0.8557
Recall	0.7347	0.9958	0.8652
F1-Score	0.7248	0.9960	0.8604

Table 4: Base model metrics on the testing set. The performance on the minority class is worse due to imbalance.

E Dynamic Model Experiment

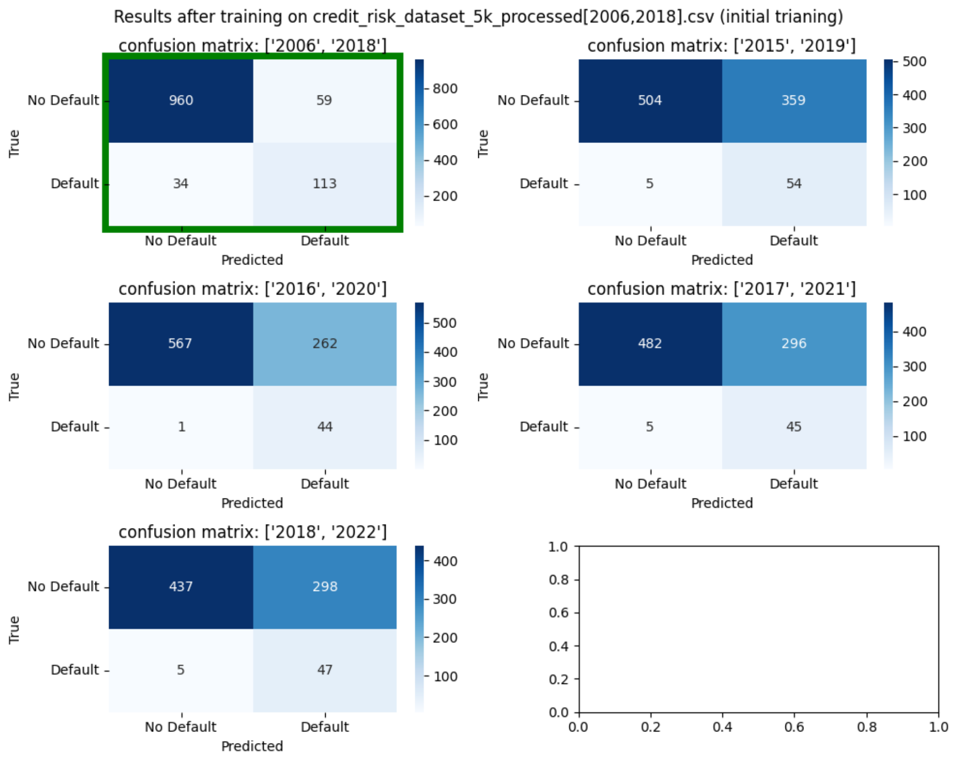


Figure 9: Confusion Matrices of all 5 period evaluated on the 2018 model

Training on period 2006 - 2018					
Metric	2018	2019	2020	2021	2022
Macro-Precision	**0.8241**	0.5640	0.5711	0.5643	0.5693
Macro-Recall	0.8848	0.9104	**0.9534**	0.9082	0.9269
Macro-F1	**0.8517**	0.5884	0.6058	0.5915	0.5982

Table 5: Metrics on all the 5 years when exposed to the years 2006 to 2018 only.

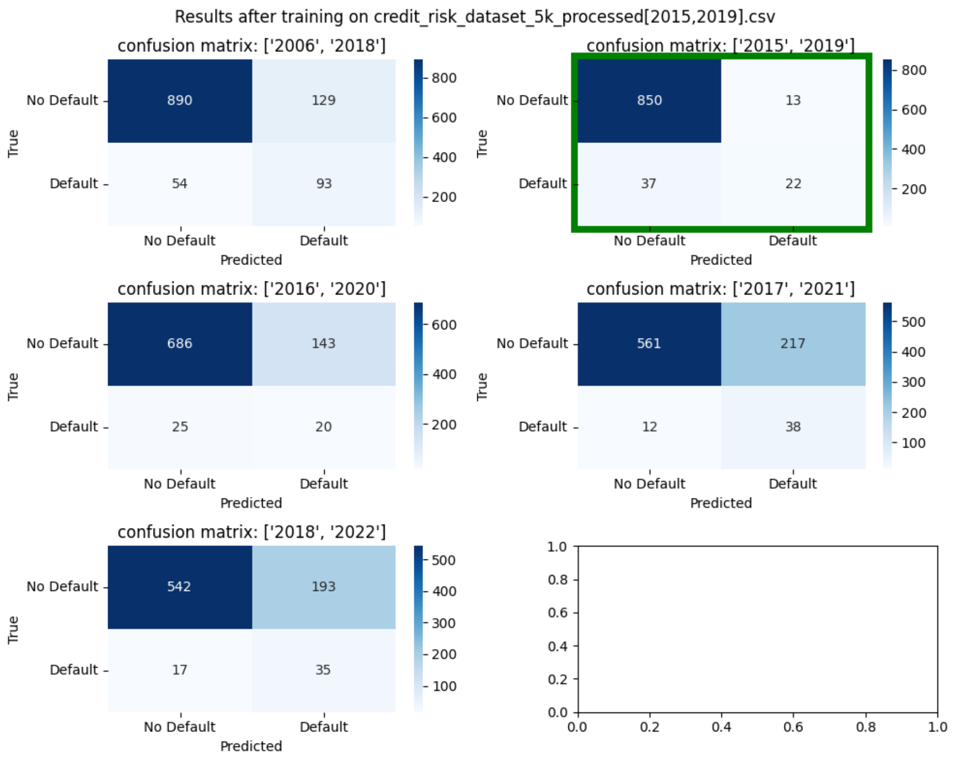


Figure 10: Confusion Matrices of all 5 period evaluated on the 2019 model

Training on period 2015 - 2019					
Metric	2018	2019	2020	2021	2022
Macro-Precision	0.5556	**0.7510**	0.5387	0.5227	0.5250
Macro-Recall	**0.7817**	0.6837	0.6931	0.7721	0.7499
Macro-F1	0.5794	**0.7119**	0.5543	0.4915	0.5029

Table 6: Metrics on all the 5 years when exposed to the years 2015 to 2019 only.

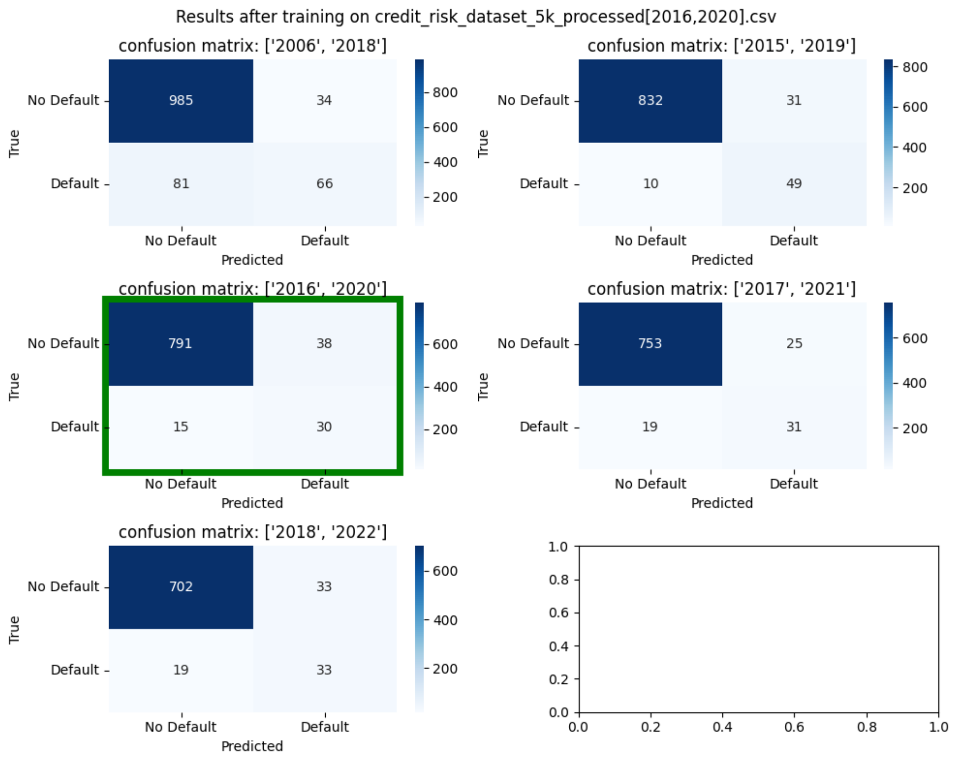


Figure 11: Confusion Matrices of all 5 period evaluated on the 2020 model

Training on period 2016 - 2020					
Metric	2018	2019	2020	2021	2022
Macro-Precision	**0.8196**	0.8075	0.7186	0.7646	0.7471
Macro-Recall	0.7227	**0.9197**	0.8282	0.8062	0.8030
Macro-F1	0.7622	**0.8545**	0.7619	0.7838	0.7720

Table 7: Metrics on all the 5 years when exposed to the years 2016 to 2020 only.

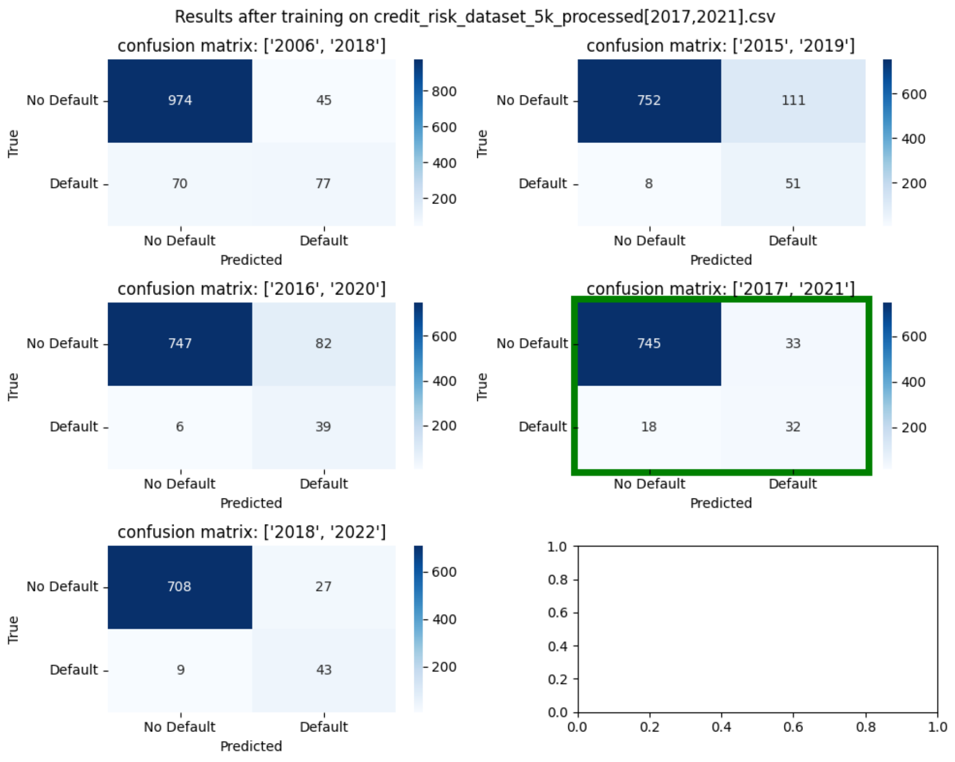


Figure 12: Confusion Matrices of all 5 period evaluated on the 2021 model

Training on period 2017 - 2021					
Metric	2018	2019	2020	2021	2022
Macro-Precision	0.7839	0.6554	0.6590	0.7436	**0.8058**
Macro-Recall	0.7591	0.9176	**0.9222**	0.8154	0.9095
Macro-F1	0.7709	**0.9693**	0.7275	0.7747	0.8498

Table 8: Metrics on all the 5 years when exposed to the years 2017 to 2021 only.

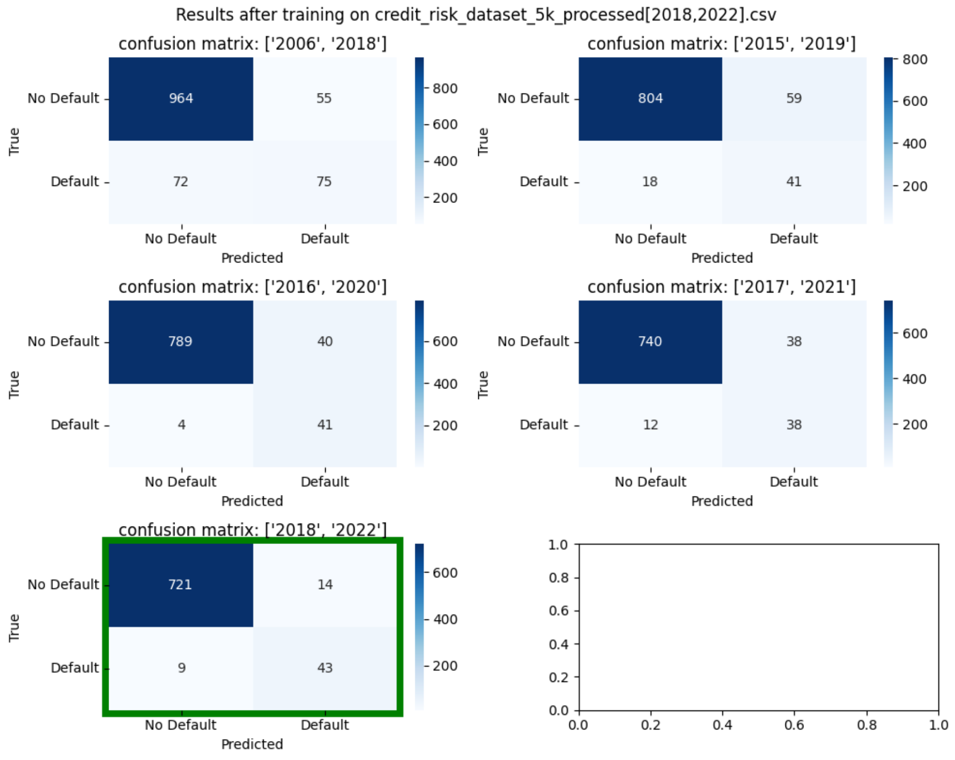


Figure 13: Confusion Matrices of all 5 period evaluated on the 2022 model

Training on period 2018 - 2022					
Metric	2018	2019	2020	2021	2022
Macro-Precision	0.7722	0.7026	0.7557	0.7483	**0.8694**
Macro-Recall	0.7521	0.8398	**0.9503**	0.8747	0.9113
Macro-F1	0.7617	0.7528	0.8251	0.7981	**0.8891**

Table 9: Metrics on all the 5 years when exposed to the years 2018 to 2022 only.

F Mathematics of XAI

The objective of LIME is to estimate the model function f with an explanation model $g \in G$, where g acts over the absence/presence of the interpretable components [16]. In explaining the classification of $f(x)$, a proximity measure $\pi_x(x')$ between a perturbed instance x' around x is used as a weight to define locality. Then, let $\mathcal{L}(f, g, \pi_{x'})$ be the fidelity function of the estimation and $\Omega(g)$ be a measure of complexity, the explanation by LIME can be achieved by minimizing the following objective function:

$$\xi(x) = \underset{g \in G}{\operatorname{argmin}}\, \mathcal{L}(f, g, \pi_{x'}) + \Omega(g) \tag{1}$$

The solution of equation 1 results in a linear regression that explains the local behavior of model function f around point x.

SHAP is a technique that utilizes the concept of cooperative game theory [14], aiming that the prediction value $f(x)$ can be distributed as contributions from each feature. The importance of a feature i is computed by comparing a model $f_{S\cup\{i\}}$ with the feature presenting and a model f_S without the feature presenting. Taking interaction effects between features into account, the difference must be calculated on all feature subsets $S \in F$, where F is the set of all features. The Shapley value of the feature i can be defined as:

$$\phi_i = \sum_{S\subseteq F\setminus\{i\}} \frac{|S|!(|F|-|S|-1)!}{|F|!} \left[f_{S\cup\{i\}(x_{S\cup\{i\}})-f_S(x_S)}\right] \tag{2}$$

Equation 2 above can be translated as the weighted average of differences between prediction value with and without the presence of feature i over all subsets of features excluding i.

Kernel SHAP combines LIME and SHAP by using a linear model to estimate the SHAP values of features [14]. Instead of computing all possible feature coalitions of the input, $z \in \{0,1\}^M$ where M is the number of features, Kernel SHAP only samples a certain amount of coalitions. The coalition z is then used to generate perturbed input using the perturbation function $h_x(z)$, which maps the coalition of features back into the original input space of x. When $z_i = 1$, this means the feature i maintains its original value as in x_i, and when $z_i = 0$, the feature i is represented as removed by being replaced with an uninformative background value [4]. New prediction values $f(h_x(z))$ can be computed from the perturbed values and used to estimate SHAP values ϕ_i by fitting the linear estimation function g.

$$g(z) = \phi_o + \sum_{i=1}^{M} \phi_i z_i \tag{3}$$

Lundberg and Lee also show that the optimal approximation of equation 3 can be achieved using a weighting kernel $\pi_x(z)$, loss metric $L(f, g, \pi_x)$, and regularization term Ω similarly as in equation 1 with:

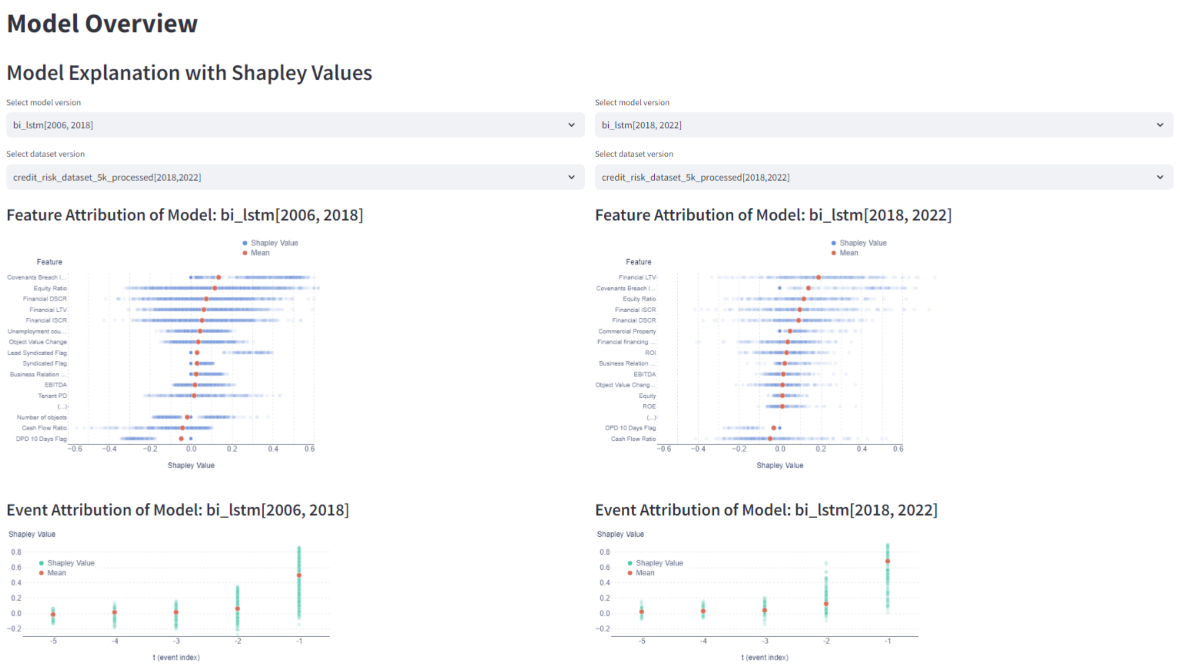
$$\Omega(g) = 0,$$

$$\pi_x(z) = \frac{(M-1)}{\binom{M}{|z|}|z|(M-|z|)},$$

$$L(f, g, \pi_x) = \sum_{z\in\{0,1\}^M} [f(h_x(z)) - g(z)]^2 \cdot \pi_x(z),$$

where $|z|$ is the number of non-zero elements of z.

G Explainability Dashboard User Interface



Model Overview

Model Explanation with Shapley Values

Select model version: bi_lstm[2006, 2018]

Select dataset version: credit_risk_dataset_5k_processed[2018,2022]

Select model version: bi_lstm[2018, 2022]

Select dataset version: credit_risk_dataset_5k_processed[2018,2022]

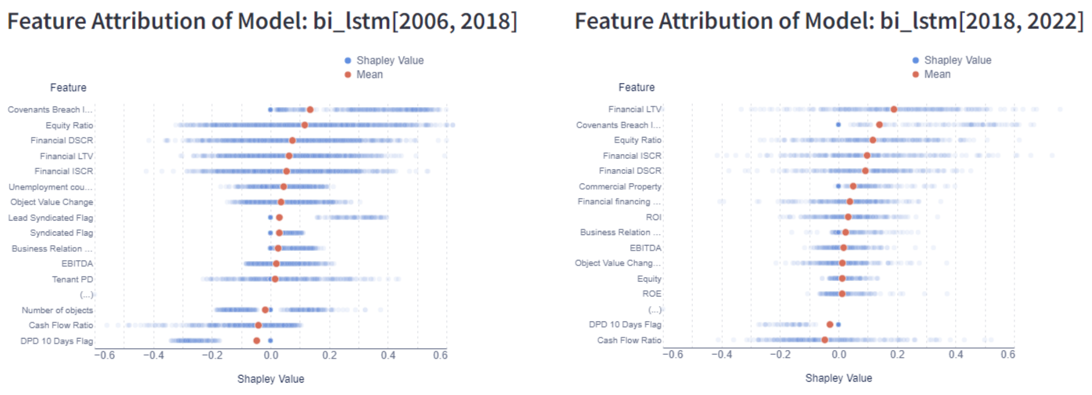
Feature Attribution of Model: bi_lstm[2006, 2018]

Feature Attribution of Model: bi_lstm[2018, 2022]

Event Attribution of Model: bi_lstm[2006, 2018]

Event Attribution of Model: bi_lstm[2018, 2022]

Figure 14: Model overview page of explainability dashboard



(a) (b)

Figure 15: Global feature attribution of (a) model trained with historical data and (b) model retrained for new data in 2022

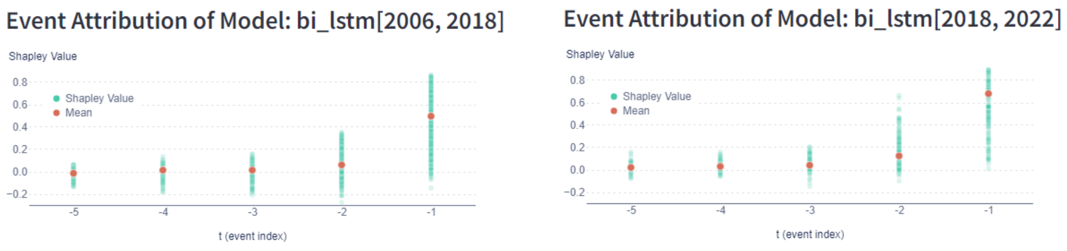
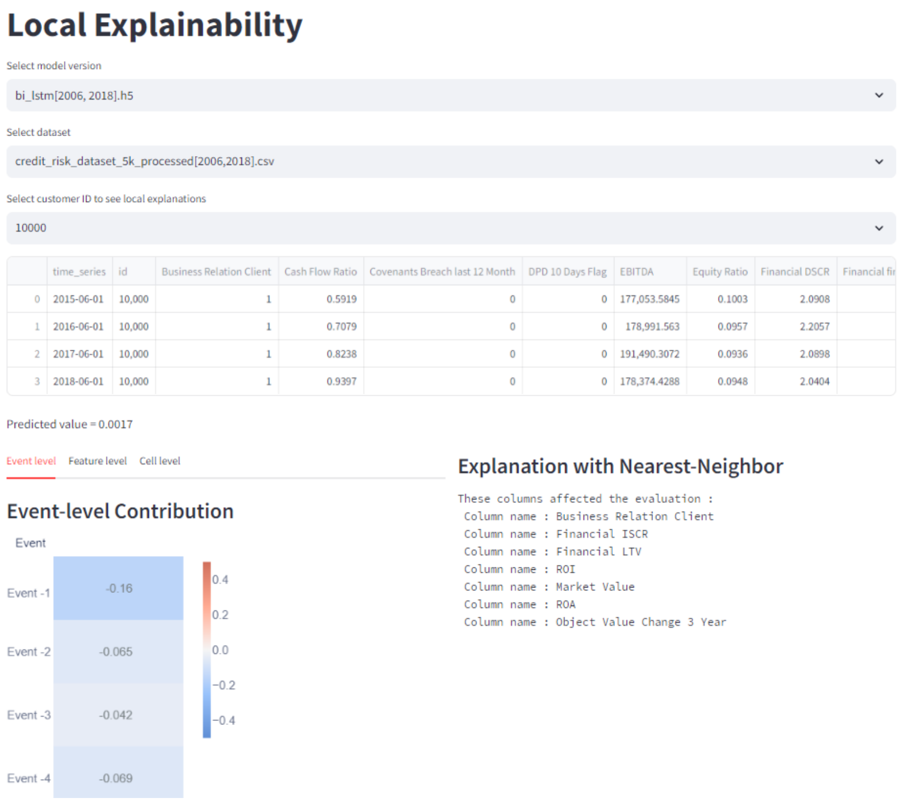


Figure 16: Global event attribution of (a) model trained with historical data and (b) model retrained for new data in 2022



Local Explainability

Select model version

bi_lstm[2006, 2018].h5

Select dataset

credit_risk_dataset_5k_processed[2006,2018].csv

Select customer ID to see local explanations

10000

	time_series	id	Business Relation Client	Cash Flow Ratio	Covenants Breach last 12 Month	DPD 10 Days Flag	EBITDA	Equity Ratio	Financial DSCR	Financial fi
0	2015-06-01	10,000	1	0.5919	0	0	177,053.5845	0.1003	2.0908	
1	2016-06-01	10,000	1	0.7079	0	0	178,991.563	0.0957	2.2057	
2	2017-06-01	10,000	1	0.8238	0	0	191,490.3072	0.0936	2.0898	
3	2018-06-01	10,000	1	0.9397	0	0	178,374.4288	0.0948	2.0404	

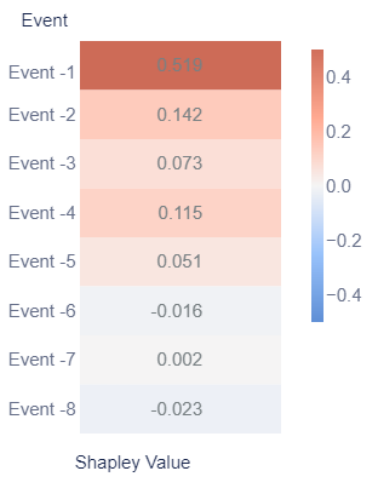
Predicted value = 0.0017

Event-level Contribution

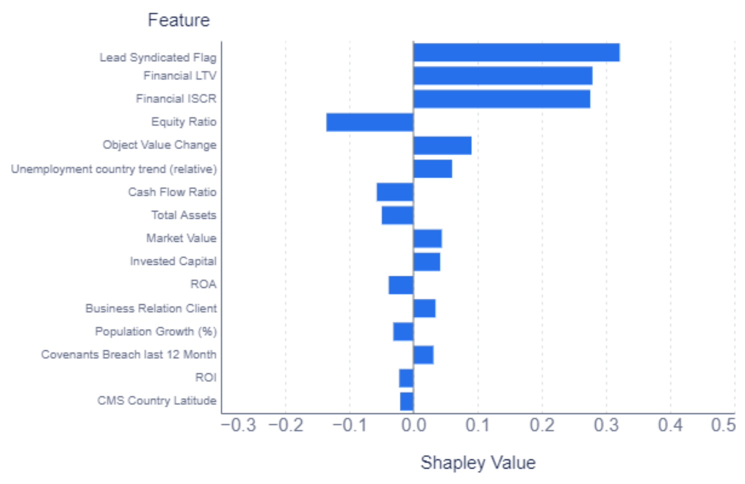
Explanation with Nearest-Neighbor

```
These columns affected the evaluation :
 Column name : Business Relation Client
 Column name : Financial ISCR
 Column name : Financial LTV
 Column name : ROI
 Column name : Market Value
 Column name : ROA
 Column name : Object Value Change 3 Year
```

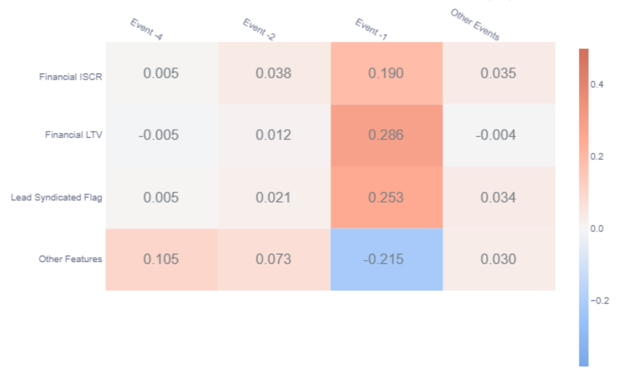
Figure 17: Local explanation page of explainability dashboard



(a)

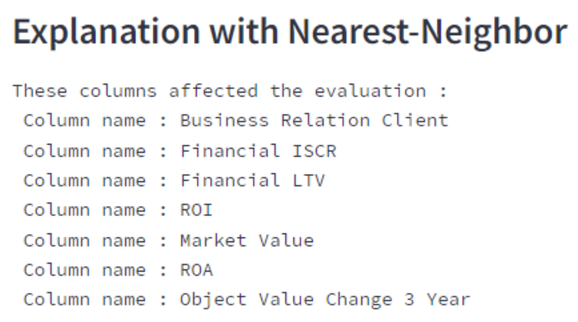


(b)



(c)

Figure 18: Local explanation using TimeSHAP of an input instance correctly predicted as default with high probability at (a) event-level, (b) feature-level, and (c) cell-level



Explanation with Nearest-Neighbor

```
These columns affected the evaluation :
 Column name : Business Relation Client
 Column name : Financial ISCR
 Column name : Financial LTV
 Column name : ROI
 Column name : Market Value
 Column name : ROA
 Column name : Object Value Change 3 Year
```

Figure 19: Local explanation with Nearest Neighbor method of an input instance correctly predicted as non-default with near-zero probability.