Co-Teaching Strategies for Credit Risk Assessment in the Presence of Label Noise

Alfonso Galván Fernández

Dissertation written under the supervision of

Susana Dias Brandão

Dissertation submitted in partial fulfillment of requirements for the MSc in Business Analytics, at the Universidade Católica Portuguesa, 2nd of January of 2024.

Abstract

This paper investigates the effect of co-teaching strategies on the handling of reject inference. The investigation explores the utility of co-teaching, a technique known for effectively handling noisy labels, specifically in computer vision, when applied to tabular data for the purpose of understanding its value in credit scoring models. This research provides a comprehensive analysis by comparing two different co-teaching strategies, fixed and incremental forget rates, to traditional the neural network approach in addressing reject inference issues. The findings suggest that although co-teaching has proven effective in other domains, the implementations that this study proposes for credit risk assessment do not provide a significant benefit in accurately predicting defaulting applications. The results indicate that traditional neural networks exhibit greater success in handling predictions for the minority class, which is crucial from a business standpoint. For future research, the study highlights the significance of improving co-teaching strategies, such as modifying the forget rate and investigating alternative network architectures. Future studies can further improve noise-robust learning techniques and their application in complex real-world scenarios by expanding current methodologies and exploring new adaptations. The research contributes to the existing literature by exploring the potential of co-teaching and providing a comprehensive analysis of its effectiveness in the presence of noisy labels. The study provides a basis for future investigations and practical implementations, with the goal of expanding the use and effectiveness of co-teaching strategies in the field of credit risk modeling.

Keywords: Co-teaching, reject inference, deep learning, credit risk, noisy labels, risk management

Este artigo investiga o efeito de estratégias de co-teaching no manejo da inferência de rejeição, focando na utilidade dessa técnica, conhecida por lidar eficazmente com rótulos ruidosos em visão computacional, em dados tabulares para modelos de pontuação de crédito. Realiza uma análise abrangente ao comparar duas estratégias de co-teaching no, taxas de esquecimento fixas e incrementais, com a abordagem neural tradicional, para resolver problemas de inferência de rejeição. Embora o co-teaching no seja eficaz em outros domínios, as implementações propostas para avaliação de risco de crédito não mostraram benefício significativo na previsão precisa de inadimplência. Os resultados apontam que redes neurais tradicionais têm mais sucesso em prever a classe minoritária, essencial para o negócio. O estudo encoraja pesquisas futuras a aprimorarem estratégias de co-teaching, ajustando a taxa de esquecimento e explorando novas arquiteturas de rede, para melhorar as técnicas de aprendizado robustas ao ruído em cenários reais. Contribui para a literatura existente ao explorar o potencial do co-teaching e sua eficácia frente a rótulos ruidosos, estabelecendo base para futuras investigações e implementações práticas em modelagem de risco de crédito.

Table of Contents

1. Introduction

Reject inference presents an important challenge in credit assessment, especially when it comes to predicting the probability of loan defaults. The issue arises because financial institutions have available data primarily for approved loan applications, resulting in a lack of information regarding the applicants who were rejected. This makes the sample less accurate because the model was only trained on data from people who were thought to be creditworthy and might not notice patterns related to default risk that exist in the rejected group.

A precise evaluation of credit applications is essential for ensuring the stability and effectiveness of financial institutions in the ever-evolving realm of financial risk management. However, label noise in the training data can occasionally weaken the robustness of these assessments, resulting in inaccurate risk predictions and related cost estimation errors. Traditional models often struggle to distinguish between erroneous and reliable data, underscoring the pressing need for innovative approaches that enhance the robustness and accuracy of the model.

This thesis looks at how co-teaching strategies deal with this problem. The goal is to mitigate the effects of label noise on credit scoring models. This study seeks to address the gap in current assessment methods by investigating the intersection of co-teaching strategies and credit scoring models.

The main purpose of this thesis is to test the idea that using co-teaching methods, such as using multiple models together to filter and learn from datasets with noise, can make credit risk predictions more accurate. The aim of this research is to explore the theoretical foundations of co-teaching methodologies in deep neural networks and offer empirical validation of their efficacy within the context of credit scoring models. The goal of this work is to establish a basis for financial institutions to enhance their assessment models through a comprehensive analysis that integrates quantitative procedures and modern deep learning techniques. The study's findings are expected to bring valuable insights to the field of credit scoring, specifically in terms of addressing costly errors in identifying and developing new approaches for assessing credit applications.

Research Question and Hypothesis

Does applying co-teaching strategies to neural networks improve their performance in credit risk assessment tasks in the presence of label noise?

H0: Co-teaching strategies applied to neural networks do not affect the handling of reject inference.

H1: Co-teaching strategies applied to neural networks improves the handling of reject inference.

The study follows the standard industry approach and set the significance level at 0.05 to reject the null hypothesis.

2. Literature Review

Reject Inference in Credit Assessment

The difficulty of incorporating rejected loan applications into prediction models limits the ability of credit scoring to accurately assess the likelihood of loan repayment. While this topic has been widely studied, many authors emphasize the absence of formalization and theoretical foundations in numerous approaches. This research will explore the effect of incorporating additional techniques to handle unreliable labels. The study will specifically look at how well these techniques work to improve the performance and reliability of deep neural networks by evaluating how well they can predict credit applications that will default.

Beben, S., (2021) introduced a theoretical framework that reconceptualizes rejecting inference as a missing data problem. The author assesses multiple approaches and provides an empirical evaluation of the strategies using both simulated and real data. The study looks into various approaches for inferring credit rejection in credit scoring. This study investigates the impact of excluding rejected loan applications from datasets on credit scoring models. Key tactics encompass disregarding clients without financial backing, employing fuzzy augmentation, reclassifying, augmenting, and utilizing twins.

The author concludes that none of the examined methods can be considered generally superior, emphasizing the difficulties in reliably estimating score functions due to inherent biases in the training samples of the models. The primary conclusion drawn from this research is the suggestion for credit modelers to mainly rely on data derived from financed loans. The author advise avoiding incorporating rejected applicants into models unless there is sufficient and trustworthy information accessible about these candidates or the acceptance process, particularly in cases involving non-ignorable missing data.

Machine Learning in Credit Assessment

Nevertheless, the continuing research on this subject has led to sophisticated strategies in the field of machine learning that can effectively tackle the challenges identified by Beben, S., (2021). To tackle the problem of unreliable labels, modern techniques have been developed that employ different approaches that demonstrate greater robustness in handling noisy labels. Considering this, it is probable that the conclusions drawn from previous studies may not be applicable to the upcoming generation of methods that utilize inferred labels for training predictive models.

Several studies have focused on automated scoring models. Li et al., (2017) introduces a Semi-Supervised Support Vector Machines (SSVM) technique within the framework of reject inference. The author utilizes a semi-supervised approach within the domain of online consumer loans, encompassing both approved and declined loan applications. The strategy considerably enhances conventional categorization techniques and effectively utilizes information from loan applicants who have been denied.

Additionally, Li et al., (2017) benchmarks the various machine learning methods used in credit scoring. The study stresses the importance of Clustered Support Vector Machines (CSVM) in supporting credit approval methods. However, the author also acknowledges the limitations of CSVM and highlights the need for further research to explore alternative techniques that can complement or improve credit approval methods. An alternate approach that could be considered is Random Forest, a widely used ensemble learning algorithm that mixes many decision trees to create predictions. The Random Forest algorithm has been extensively utilized across several fields and has demonstrated encouraging outcomes in credit scoring as well.

Furthermore, deep learning algorithms, such as neural networks, have garnered significant interest in recent years due to their capacity to acquire intricate patterns and generate precise forecasts (Li et al., 2017). The integration of these advanced techniques with CSVM has the potential to enhance credit approval systems, making them even more robust. This suggests that while CSVM is valuable, it may not be the best solution for credit scoring and there is room for exploring other machine learning algorithms or hybrid approaches to enhance the accuracy and efficiency of credit approval systems.

Limitations of State-of-the-Art Methods

Support Vector Machines (SVMs) and their related methods appear to be the most popular methodologies that researchers have embraced. However, SVMs are specifically developed to optimize the separation between distinct classes by maximizing the margin (Bootkrajang & Kabán, 2013). Noisy labeling can lead to a misrepresentation of the true margin. Incorrectly labeled data points might be used as support vectors, resulting in inaccurate decision bounds. Boosting algorithms, which prioritize misclassified samples, have the potential to overfit and diminish model performance as a result of noise. Both Support Vector Machines (SVM) and boosting models are highly prone to outliers, which can be attributed to the presence of label noise. The conventional loss functions employed in SVM and Boosting algorithms lack robustness in accurately labeling noisy data, resulting in significant penalties for

misclassifications. Noisy labels can hinder the modeling process in datasets with complicated structures and intricate class boundaries. In certain cases, SVMs and boosting algorithms may encounter difficulties in constructing precise models (Bootkrajang & Kabán, 2013). The author indicates that conventional boosting and SVM techniques exhibit inferior performance in the presence of label noise, resulting in higher error rates and diminished classification accuracy. In such cases, alternative methods or modifications to the existing algorithms may need to be explored.

Strategies for Mitigating Noisy Labels

Researchers have adopted several traditional methods to reduce and control noisy labeling. Like Natarajan, N., Dhillon, I. S., Ravikumar, P. K., & Tewari, A. (2013)., that addresses the issue of noisy labels in a binary classification setting. The author introduces the concept of surrogate loss function. Which is employed as an alternative to the primary loss function in cases where direct optimization is challenging due to properties such as non-differentiability, non-continuity, or computational infeasibility. It provides a proxy that, when minimized, leads to optimal properties in the original loss function.

The paper proposes two approaches for adapting the surrogate loss functions to handle such noise: the method of an unbiased estimator and the method of label-dependent costs.

First, the paper presents an unbiased estimate of any loss and gets performance limits for empirical risk minimization in the presence of noisy labels. The method leads to an effective algorithm for empirical minimization. The second approach suggests using a weighted surrogate loss to get strong empirical risk bounds. This is because it reduces the risk of risk minimization under noisy labels to classification with a weighted 0–1 loss.

Hence, the theoretical framework is implemented using practical techniques such as SVM and weighted logistic regression, demonstrating its ability to handle noisy labels well. The efficacy of the approaches is evaluated by subjecting them to testing using both actual and synthetic datasets. Additionally, several baseline models are included in the evaluation process to establish a standard for accuracy. The datasets used in the evaluation contain label corruption of up to 40%.

Deep learning models have effectively gained success in numerous domains by effectively handling noisy labels, similar to the previous approaches (Li et al., 2017). To mitigate the impact of lacking high-quality data in various real-life situations, it is essential to implement a

robust training procedure that keeps the overall performance of deep neural networks from being compromised. Song, (2020) provides a comprehensive list of 62 cutting-edge training methods. The authors divide the ways to deal with noisy labels in deep learning into five groups. They focus on making the supervised learning process more resistant to label noise. These areas include robust architecture, robust regularization, robust loss function, loss correction, and sample selection. As other research has shown, this study stresses how important it is to use a modified loss function and to make changes to the architecture of deep neural networks in order to improve generalization (Natarajan, N., Dhillon, I. S., Ravikumar, P. K., & Tewari, A., 2013).

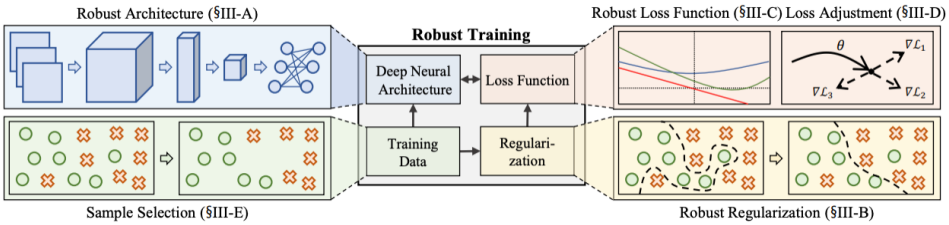

Figure 1: Deep learning methods to overcome noisy labels (Song, 2020)

In line with Song, (2020), the study explores network architectures that can mitigate noisy labels during the training of the deep neural network. Chen, Y, (2022). revisits dropout and nested dropout techniques as compression constraints. Furthermore, the models that have been trained via compression regularization are integrated with a new concept, co-teaching, to improve the model's performance.

The Role of Co-Teaching in Addressing Noisy Labels

The principle of co-teaching, as examined in recent research studies such as those published in Han, (2018), introduces an innovative strategy for addressing the difficulties caused by inaccurate labels in machine learning. Co-teaching in the context of noisy labels offers a significant benefit by effectively mitigating the influence of label noise during model training. Co-teaching is a method for learning under label noise where two deep networks train simultaneously, selecting their 100(1−λforget) percent small-loss instances. Based on the data subset that their peers have selected, the networks update themselves.

This technique is based on the idea that small-loss instances are more likely to be clean, making classifiers more resistant to noisy labels. To prevent constant bad selections, a scheduling approach is proposed, keeping all samples in the mini-batch, and gradually decreasing the sample size until the predefined number of epochs (Han, 2018).

The main breakthrough in this technique lies in the fact that one model acquires the ability to recognize and highlight distinct elements of the data, effectively instructing the other model on the subtleties of the dataset. The method of collaborative learning enables one model to leverage the insights and strengths of the other, resulting in a more resilient and precise learning output.

This selective learning process helps in filtering out the noise and focusing on the more reliable and accurate aspects of the data. Consequently, models trained using the co-teaching approach are less likely to overfit to noisy labels and are better at generalizing to new, unseen data (Han, 2018).

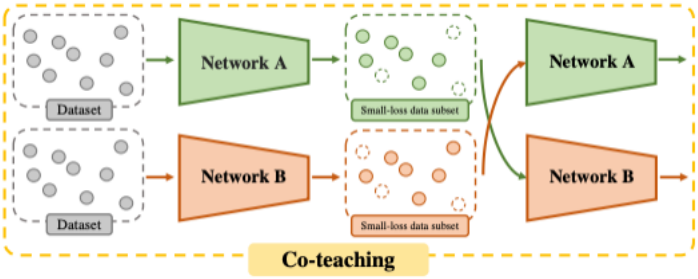

Figure 2: Co-teaching principle (Chen, Y, 2022)

This approach has gained popularity across various domains where it has been employed. Abdelgayed, T. S., Morsi, W. G., & Sidhu, T. S. (2018) presents a semi-supervised approach for fault classification of transmission and distribution systems, claiming that the classifier was able to perform better than other machine learning models in the presence of unlabeled data. Several other co-teaching variants like in Liu, (2023) were presented after the initial idea of co-teaching was introduced, with different logics that could help reduce the noise in datasets. Mainly, this technique has been incorporated into deep neural networks that are employed in the area of computer vision. Therefore, the main idea remains the same, but the input and general network architecture found in previous findings do not fit into the structured and tabular format in which financial institutions store the data for credit applications and their requestors.

Unfortunately, there is a lack of research available about the use of co-teaching in the context of credit scoring models. This thesis explores the impact of this novel approach and its potential to address the challenges associated with noisy labels. Specifically, it focuses on inferring the label for rejected applications, as financial institutions are unable to determine the outcome of such applications. This thesis aims to determine whether co-teaching strategies can enhance the performance of deep neural networks in the operational context of credit scoring models.

3. Data Pre-processing

Data Collection

The dataset offered by Home Credit is extensive and comprehensive, structured in a manner similar to a relational database. Home Credit, founded in 1997, is an experienced provider of consumer lending with a 24-year track record. Home loans, a company that operates in nine countries. The organization aims to improve financial inclusion for underserved customers, including those with limited or no credit histories, by enabling their access to credit through digital and traditional channels. Home Credit has initiated a Kaggle competition with the aim of motivating machine learning developers to create resilient predictive models.

The dataset has great granularity, containing a wide range of parameters such as loan kinds, demographic information, income levels, asset ownership details, and prior credit history.

A summary of the variables of the dataset includes:

Core Variables

- **SK_ID_CURR:** The unique identifier of the loan, crucial for merging with other data tables.
- **TARGET:** The dependent variable of interest indicating payment difficulties.
- **NAME_CONTRACT_TYPE, CODE_GENDER, FLAG_OWN_CAR, FLAG_OWN_REALTY:** Basic descriptors of the loan and applicant's personal status.

Financial and Social Context

- **AMT_INCOME_TOTAL, AMT_CREDIT, AMT_ANNUITY:** Key financial amounts describing the loan's size and the applicant's income level.
- **OBS_30_CNT_SOCIAL_CIRCLE, DEF_30_CNT_SOCIAL_CIRCLE**: Reflecting the applicant's social environment's financial reliability.

Credit History and Inquiry

- **AMT_REQ_CREDIT_BUREAU_HOUR, DAY, WEEK, MON, QRT, YEAR:** Indicating the frequency and recency of credit checks, important for understanding credit-seeking behavior.

Property and Housing

- **APARTMENTS_AVG, BASEMENTAREA_AVG, COMMONAREA_AVG:** Normalized metrics about the applicant's living conditions, which can be critical in understanding stability and lifestyle.

Employment and Stability Indicators:

- **DAYS_BIRTH, DAYS_EMPLOYED, OCCUPATION_TYPE:** Offering insights into the applicant's age, stability, and occupation, all of which are vital for risk assessment.

External Data Sources:

- **EXT_SOURCE_1, EXT_SOURCE_2, EXT_SOURCE_3:** Scores from external bureaus, often used as significant predictors in credit risk modeling.

Although all variables contribute to a complete understanding of the applicants, certain variables are interesting due to their possible impact on estimating an applicant's likelihood of default. Some key variables are:

- **TARGET:** The dependent variable indicating if the client has had payment difficulties.
- **AMT_INCOME_TOTAL** and **AMT_CREDIT**: These provide a clear indication of the applicant's financial situation and the magnitude of the loan.
- **DAYS_BIRTH** and **DAYS_EMPLOYED**: These give an idea of the applicant's age and employment stability, which are indirect indicators of financial stability and risk.
- **REGION_RATING_CLIENT** and **REGION_RATING_CLIENT_W_CITY**: These ratings provide a socioeconomic context of the applicant's environment, which can influence stability and ability to repay.
- **OBS_30_CNT_SOCIAL_CIRCLE** and **DEF_30_CNT_SOCIAL_CIRCLE:** Observations of the client's social circle with default history may reflect the applicant's financial behavior.
- **AMT_REQ_CREDIT_BUREAU_YEAR:** Frequent enquiries to credit bureaus can indicate distress or search for new credit lines, relevant for risk assessment.

- **EXT_SOURCE_1, EXT_SOURCE_2,** and **EXT_SOURCE_3** are numerical ratings obtained from external data sources. These ratings might serve as indicators of an applicant's general creditworthiness.

Exploratory Data Analysis (EDA)

The exploratory data analysis (EDA) has been carried out with “ydata-profiling”, an open-source Python library designed to facilitate EDA by generating profile reports from a pandas DataFrame. This library provides a detailed statistical summary, which includes information on variables, correlations, missing values, and duplicate data.

The library offers the option to limit certain computationally intensive analyses if the user chooses not to include them. Due to the size of the dataset, we have chosen to modify the default setting for the report. By setting the parameter "minimal=True", the tool will generate a thorough report that excludes complex analyses such as in-depth examinations of correlations and interactions.

The dataset contains 125 variables and 263491 observations in total. In the initial overview, 108 of those variables are numerical, and 17 are text. Surprisingly, there are approximately 23% of missing cells. This provides initial insights regarding the presence of missing values in the data that will need to be resolved.

The distribution of the target variable of this sample is significantly imbalanced due to the nature of credit applications. The class 0 with 92.3% of the observations indicating the applicant had no difficulties by paying it back. On the other hand, the class 1 with 7.7% implies that the requestor of the credit had defaulted.

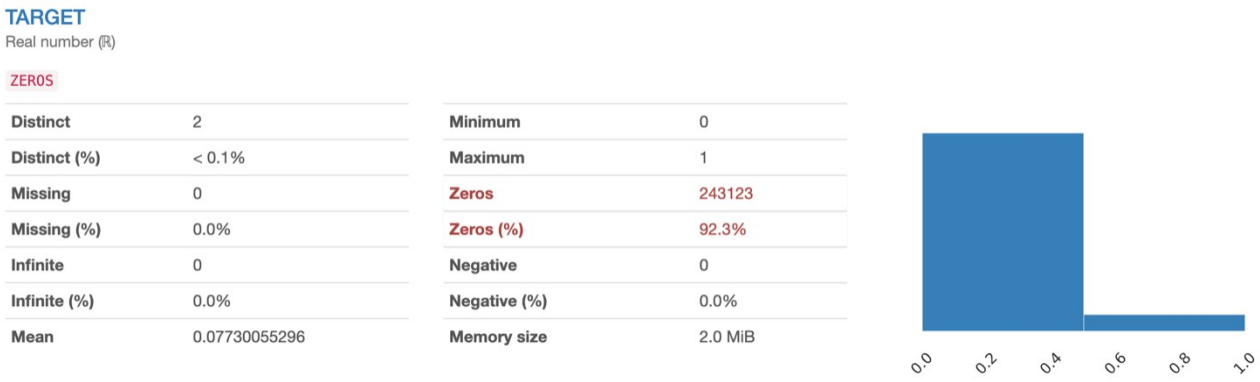

Figure 3: Distribution of target variable

It is crucial to thoroughly consider the suitable evaluation metrics and sample procedures to address this class imbalance. In our approach, we have decided to downsample the training

data, which involves reducing the number of instances in the majority class to match the number of instances in the minority class. This downsampling technique helps create a more balanced representation of the target variable, enabling the model to learn effectively from both classes.

However, this is just the representation of the starting state of the target variable. In a later stage, by leveraging the probability scores obtained from the prior model, we can employ a logic that infers the label for rejected applications. This approach allows us to make informed decisions based on the likelihood of an application being rejected, improving the overall performance of the models.

Regarding the missing values, we can figure out that they follow the Missing Completely at Random (MCAR) pattern after looking at the variables that have the most null values and understanding what those values mean in the dataset. Therefore, for variables with more than 60% of missing values, these were dropped. For the rest, imputing unobserved data with the median for numerical values and with a placeholder like 0 for the categorical ones could result in a lower risk of bias.

	Missing Values	% of Total Values
COMMONAREA_MODE	183219	69.5
COMMONAREA_AVG	183219	69.5
COMMONAREA_MEDI	183219	69.5
NONLIVINGAPARTMENTS_MEDI	182038	69.1
NONLIVINGAPARTMENTS_MODE	182038	69.1
NONLIVINGAPARTMENTS_AVG	182038	69.1
FONDKAPREMONT_MODE	179283	68.0
LIVINGAPARTMENTS_MODE	179191	68.0
LIVINGAPARTMENTS_MEDI	179191	68.0
LIVINGAPARTMENTS_AVG	179191	68.0
FLOORSMIN_AVG	177890	67.5
FLOORSMIN_MEDI	177890	67.5
FLOORSMIN_MODE	177890	67.5
YEARS_BUILD_MEDI	174239	66.1
YEARS_BUILD_MODE	174239	66.1
YEARS_BUILD_AVG	174239	66.1
OWN_CAR_AGE	171716	65.2
LANDAREA_AVG	155200	58.9
LANDAREA_MEDI	155200	58.9
LANDAREA_MODE	155200	58.9

Figure 4: Percentage of missing values

Then, it is necessary to encode the categorical variables to enable the machine learning model to handle them effectively. To address this problem, we employ a one-hot encoding technique, which involves generating a new column for every distinct category in a categorical data. Every observation is assigned a value of 1 in the column that corresponds to its category, and a value of 0 in the other newly created columns.

This brings us back to the data collection stage, during which certain variables were classified as being interesting, or at least that was our initial belief regarding the role of these variables in potential predictive models.

By looking at the variables of interests AMT_CREDIT and AMT_INCOME, it is evident that the variables display a left skew, suggesting the presence of outliers in the data. To mitigate the influence of skewness, a logarithmic transformation was performed before normalizing the data and feeding it into the models. This transformation was effective to reduce the skewness of the variables.

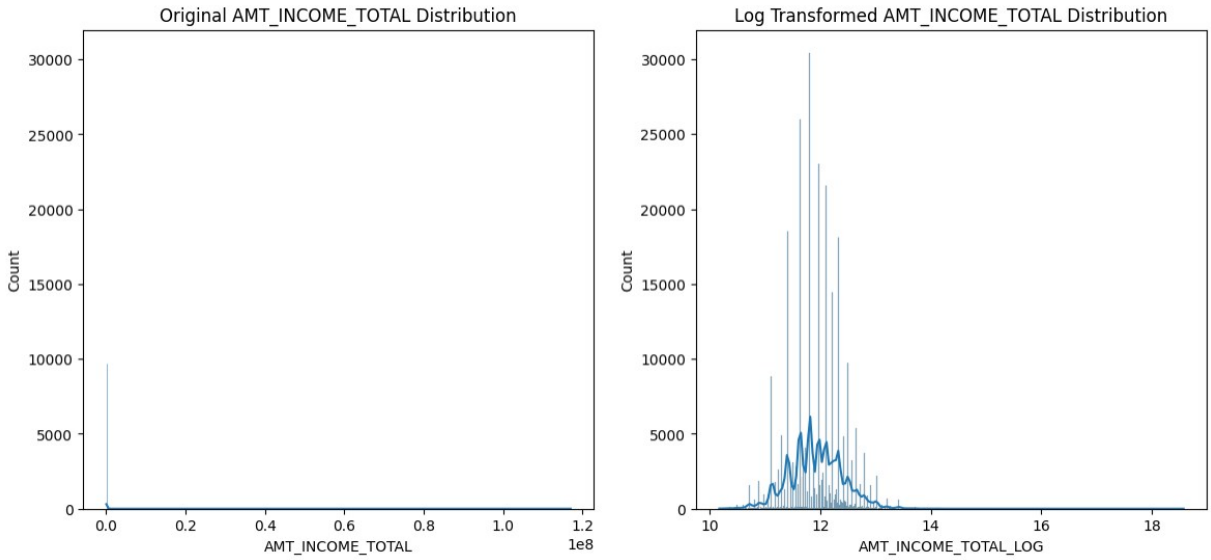

Figure 5: Distribution of AMT_INCOME_TOTAL

It looks like there is a long tail of higher incomes in the AMT_INCOME_TOTAL variable's initial distribution, which is highly skewed and has a positive skewness. This could relate to individuals who have high wages, which makes the mean an untrustworthy measure. To address this issue, a logarithmic transformation is utilized to condense the extensive range of values into a more compact scale. This leads to a more symmetrical and Gaussian income distribution, which is advantageous for statistical techniques and machine learning algorithms that rely on the assumption of a normal distribution.

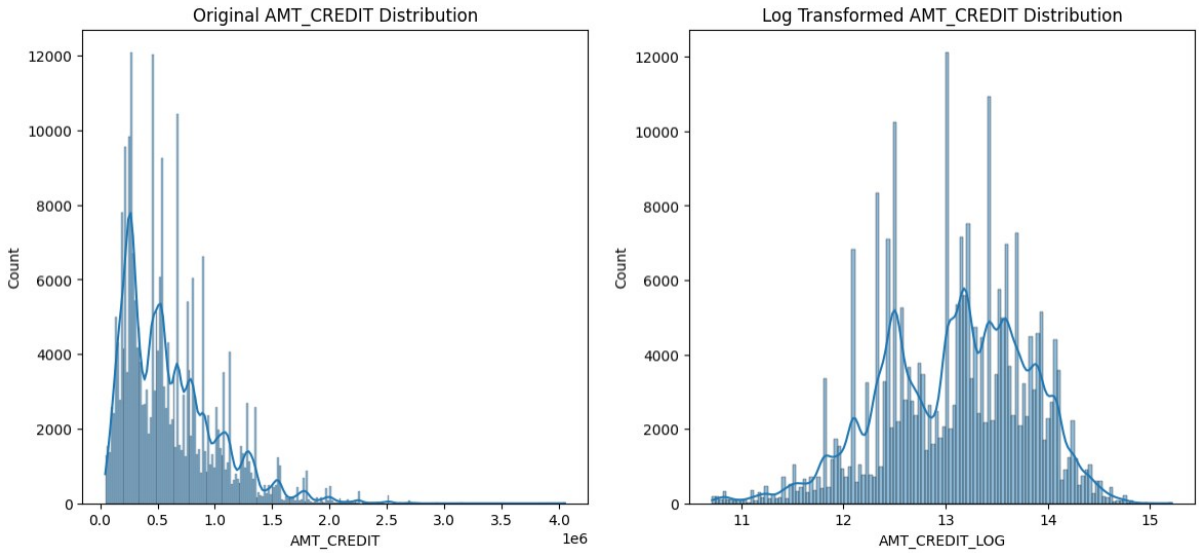

Figure 6: Distribution of AMT_CREDIT

The left side of the graph shows a high frequency of lower amounts, indicating the general pattern of the original distribution of AMT_CREDIT. This pattern suggests that most customers tend to borrow smaller loan amounts. On the other hand, the number of larger loans diminishes, as seen by the elongated distribution towards higher loan values. Log-transforming the data results in a reduction of skewness and an increase in symmetry in the distribution of AMT_CREDIT.

Furthermore, we are going to review the variable DAYS_BIRTH, which offers insight into the age of the credit applicants. Arguably, this is one of the most important elements that will be incorporated into the predictive models. To convert these values into years, divide by 365, resulting in an age range of around 27 to 69. The distribution has a very even pattern, with a little rise in occurrence within the middle age groups, indicating that a large proportion of the people in the dataset are in their adult years. This aligns with the demographic group that is most inclined to participate in the credit market, actively pursuing loans for purposes such as housing, education, or entrepreneurship.

DAYS_BIRTH

Real number (ℝ)

Distinct	17397	**Minimum**	-25201
Distinct (%)	6.6%	**Maximum**	-7489
Missing	0	**Zeros**	0
Missing (%)	0.0%	**Zeros (%)**	0.0%
Infinite	0	**Negative**	263491
Infinite (%)	0.0%	**Negative (%)**	100.0%
Mean	-16097.58174	**Memory size**	2.0 MiB

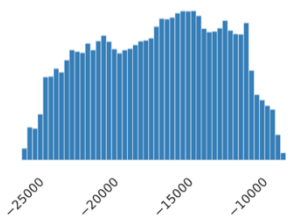

Figure 7: Distribution of DAYS_BIRTH

Lastly, the dataset contains EXT_SOURCE variables which represent external credit scores that are important in assessing the creditworthiness of prospective borrowers.

The distribution of scores in EXT_SOURCE_1 follows a regular pattern, but more than half of the data is absent. This indicates that although this score could offer useful information about the trustworthiness of borrowers, it may not be often used or accessible for making decisions. The commercial implication is that depending exclusively on EXT_SOURCE_1 may result in an inadequate evaluation of an individual's risk.

Figure 8: Distribution of EXT_SOURCE_1

On the contrary, EXT_SOURCE_2 possesses a far more comprehensive dataset with a minor bias towards the left, suggesting that a greater proportion of applicants have lower scores in this metric. This suggests that EXT_SOURCE_2 may be more rigorous or encompass a risk factor that EXT_SOURCE_1 does not include.

Figure 9: Distribution of EXT_SOURCE_2

Finally, EXT_SOURCE_3 exhibits a distribution that is biased towards the right.

EXT_SOURCE_3

Real number (ℝ)

MISSING

Distinct	812	Minimum	0.0005272652387
Distinct (%)	0.3%	Maximum	0.8960095495
Missing	18289	Zeros	0
Missing (%)	6.9%	Zeros (%)	0.0%
Infinite	0	Negative	0
Infinite (%)	0.0%	Negative (%)	0.0%
Mean	0.5097553509	Memory size	2.0 MiB

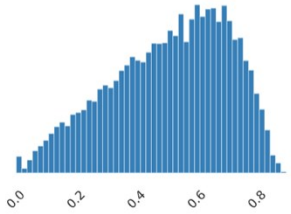

Figure 10: Distribution of EXT_SOURCE_3

The integration of all three sources, disregarding their different distributions and levels of completeness, would provide a more robust and comprehensive method for credit rating compared to relying on any individual source alone. It enables a wider angle of the candidate's profile, which can be critical for financial institutions.

This thesis focuses solely on examining the effects of co-teaching strategies in deep neural networks. As a result, no domain knowledge features, or feature engineering processes have been implemented. The deliberate choice to set boundaries is intended to prioritize the investigation of the potential of co-teaching strategies in improving both the learning process and reliability of deep learning models.

4. Methodology

This chapter introduces the methodology that serves as the foundation for our investigation into rejecting inferences for credit risk assessment. This topic entails the task of assessing the creditworthiness of individuals, which includes incomplete records resulting from declined loan applications. The presence of noisy labels in the training data, where the assigned labels may not accurately reflect the underlying classification of the data points, makes this challenge worse.

In the following sections, we provide a detailed description of our experimental design, the exact configuration details of our model, and the metrics chosen for evaluating its performance. This methodology is based on data science best practices to ensure a systematic and strong approach to our research question. The process is broken down into three parts: modeling, evaluation, validation, and testing.

Modelling

The modeling component of this thesis focuses on creating neural network models that are specifically suited to properly address the complexities of reject inference in credit scoring. By utilizing the probability scores provided by the prior model, we could provide a synthetic label for the reject set. However, this label is not true, therefore introducing noise into the dataset. To deal with these complexities, our approach consists of setting the labels of the rejected set to non-default. This study developed and evaluated three distinct neural network architectures. Every architecture presents a distinct approach to training models in scenarios characterized by label noise.

Baseline Neural Network: To ensure comparisons, we implement a standard neural network model as our reference starting point for future models. This initial model is created without utilizing specific methods to reduce noise, thereby providing a basis for evaluating the advantages of more advanced framework, like co-teaching.

T**wo Neural Networks implementing the Co-teaching principle with a constant forget rate:** The second model utilizes a co-teaching method that takes advantage of the combined learning dynamics of two neural networks to progress from the first stage. These networks are trained simultaneously, with one network utilizing the reliable forecasts generated by the other network. A predefined rate of 0.1 is employed to determine the proportion of information to be excluded from the training process based on their loss scores. This exclusion rule is established

on the assumption that data points with higher losses are more prone to misclassification. Therefore, by eliminating these data points, one can effectively diminish the impact of irrelevant information.

Two Neural Networks utilizing the Co-teaching principle with an incremental forget rate: The third model extends the co-teaching technique by incorporating an incremental forget rate starting at 0.1. Unlike the fixed-rate version, this model gradually modifies the threshold, increasing it by 0.01 for excluding input and becoming increasingly selective as the training progresses. This adaptive approach is designed to align more closely with the changing distribution of noise in the dataset.

The dataset was divided into subsets for the purpose of training, validating, and evaluating the predictive performance of a model. This was done using a stratified sampling strategy. 80% of the resampled dataset was assigned to a training and validation set, while the remaining 20% was designated for testing.

The training and validation sets were subsequently partitioned, allocating 75% for training and 25% for validation. The use of stratified sampling guarantees the integrity of evaluation measures across all models, irrespective of their complexity or co-teaching methodologies. The diligence in the design of the experiment guarantees a statistically valid and impartial basis for analyzing the performance of the model. This stratified division is critical for working with imbalanced datasets, minimizing biases, and contributing to a more balanced setting.

In the initial stage of the process, downsampling techniques were used to address class imbalances in the dataset in the training set. This step ensured that the models would not have a bias towards the dominant class and could learn more general patterns that are applicable to all classes.

Subsequently, the data normalization process was carried out using the MinMax Scaler, which effectively transformed the features to a limited range of values between 0 and 1. Normalization plays an important role in deep learning, as it enables faster convergence during training by ensuring a uniform scale for all input characteristics.

The hyperparameter tuning was performed using Keras Random Search with the objective of maximizing the AUC score on the valuation set. With focus on four parameters: L2 regularization, dropout rate, learning rate, and the number of units in the initial layer.

Each neural network follows a standardized architecture, consisting of an input layer designed to match the dimensionality of the feature space, a hidden layer with 60 neurons that use rectified linear unit (ReLU) activation, an extra layer of 20 neurons and an output layer that uses a sigmoid activation function suitable for binary classification. An Adam optimizer is also used due to its ability to adaptively adjust the learning rate, which has been demonstrated to be effective in different situations. The utilized loss function is binary cross-entropy, which is very appropriate for binary classification tasks.

The optimizer received a learning rate of 0.001. To mitigate overfitting, the model uses regularization and dropout techniques. More precisely, a hidden layer is subjected to L2 regularization with a coefficient of 0.00073. In addition, a dropout rate of 0.4 is used to randomly disable the neurons in the hidden layer during training.

The exploration is structured around three different models: a baseline model and two varieties of co-teaching—one with a consistent forget rate and the other with a gradually increasing rate of forgetting. A control baseline model was built in the beginning without the use of co-teaching principle to facilitate further comparative analysis. The baseline was used as a reference point to evaluate the additional value that the co-teaching strategies provided.

Subsequently, a model was implemented incorporating the co-teaching framework with a fixed rate of forgetting. The forget rate, which is set at 0.1, determines the fraction of predictions that are regarded as noisy and hence removed from the calculation of the loss function during training.

The chosen rate was based on preliminary trials that demonstrated its effectiveness in achieving a balance between preserving important learning signals and filtering out harmful noise, with the assumption that only 10% of the data contains noise.

Lastly, a model employing co-teaching and an incremental forget rate was proposed to enhance the complexity of the models. Every epoch after the initial setting of this model's forget rate at 0.1, it increased by 0.01. The model's ability to adapt and refine its learning process resulted in increased selectivity towards the data it acquired. The hypothesis was that as the model's weights approached an optimal state, its sensitivity to noise would diminish, necessitating a higher threshold for excluding input. By using these models, it became practical to explore the impact of co-teaching methodologies on the performance of deep neural networks for credit scoring models. Our study comparing these models to the baseline helped us figure out the best

conditions for co-teaching strategies to work and showed that the theory behind co-teaching in neural network training is sound.

The experimental work carried out in this thesis adds to the wider discussion on the effectiveness of co-teaching and suggests a new method for handling noise in large datasets when training deep learning models. Then, the derived models are evaluated according to their performance on a distinct test set, which is essential to confirm the practicality of the findings.

As an illustration of the concept behind the algorithm that was implemented, the following can be seen as an example of how the co-teaching principle was applied to the models that included such a framework.

```
| Algorithm 1: Co-teaching Neural Network with K-Fold CV
| Input: X_train, y_train, best_hps
| Output: loss_per_fold, accuracy_per_fold, auc_per_fold, precision_per_fold
| Initialize kfold (n_splits=5, shuffle=True, random_state=42)
| Set epochs=30, batch_size=32, forget_rate=0.1
| fold_no = 1
| Initialize metric lists
| Function TRAIN (X_train, y_train, best_hps)
| --- For each split in kfold:
| ------ Set seeds
| ------ Initialize models (best_hps)
| ------ Print fold_no
| ------ Split fold data
| ------ For epoch = 1 to epochs
| --------- Shuffle training data
| --------- For each batch in training data:
| ---------- Get predictions
| ---------- Calculate loss
| ---------- Get smallest loss indices
| ---------- Compute reduced losses
| ---------- Update models
| ------ Evaluate models
| ------ Record metrics
| ------ fold_no = fold_no + 1
| --- End For
| --- Return metrics
| End Function
```

Evaluation

Evaluating our neural network models is critical for understanding the influence of co-teaching strategies in mitigating label noise for credit scoring models.

Area Under the Receiver Operating Characteristic Curve (AUC-ROC)

The Area Under the Receiver Operating Characteristic Curve (AUC-ROC) is a highly helpful indicator when evaluating credit risk for loan applications. It quantifies the model's capacity to differentiate between clients who will default and those who will not. A higher AUC-ROC value signifies increased discriminatory capability, which is essential in financial decision-making due to the significant costs associated with false positives (approving credit for an undeserving applicant) and false negatives (rejecting credit for a deserving candidate). The AUC-ROC enables credit institutions to modify their risk thresholds based on their risk tolerance and regulatory obligations by offering a comprehensive metric that encompasses the trade-off between sensitivity and specificity at all thresholds.

F1 Score

The F1 score is especially applicable in imbalanced datasets commonly encountered in credit application procedures, where instances of defaults may be significantly less frequent than instances of non-defaults. In such instances, both accuracy (the capacity to correctly identify defaulters) and comprehensiveness (the capacity to collect as many genuine defaulters as feasible) are vital. The F1 score reconciles these two features, offering a singular metric that evaluates the equilibrium between precisely identifying defaulters and minimizing the number of defaulters that are overlooked. Accuracy is crucial in the credit sector, since failure to detect a defaulter can result in financial losses, while mistakenly classifying a reliable customer as a defaulter can result in missed opportunities and consumer discontent.

Area Under the Precision-Recall Curve (AUPRC).

The AUPRC is crucial in credit application reviews since credit datasets often exhibit an uneven character. The concepts of accuracy and recall are of utmost importance in this context. Accuracy refers to the expense incurred when granting a loan to an undeserving applicant (false positives), whereas recall pertains to the cost associated with rejecting a loan application from a deserving candidate (false negatives). Within the context of evaluating credit risk, the occurrence of false positives can lead to the accumulation of unpaid debts, while false negatives

might result in missed chances for the bank. The AUPRC offers a thorough assessment of a model's performance in these aspects, specifically emphasizing the capacity to identify all prospective defaulters (high recall) while retaining a high level of accuracy. This guarantees that financial organizations will generate profits while properly handling risk.

Through the application of a common approach, each model is examined using several performance indicators, including AUC-ROC, precision, recall, F1 score, and AUPRC. Together, these criteria offer a comprehensive view of the model's performance.

Additionally, to assess the business impact, some KPIs were determined from the available data to provide further insights on the impact of the predictive model when deployed to production. These metrics are default rate, average loss per default, expected loss, and application success rate.

Application Failure Rate: Proportion of loans that the model rejected.

Application Success Rate: Proportion of loans that the model approved.

Expected Profit: The expected profit is determined by calculating the gains minus the losses. To simplify, we will assume that on overall the lender gets 20% on top of their exposure. As for the losses, we will assume a fixed loss rate of 80% in the event of default. This represents the portion of the credit that the lender expects to lose if it defaults. The exposure is a constant of 0.6 million based on the mean of the variable AMT_CREDIT across the entire dataset.

The models follow a systematic training and validation approach, using consistent dataset splits. This guarantees consistency in the evaluation. The development of each model is closely observed, with callbacks implemented to record the model state at its highest validation performance. After the training process is over, the models proceed to the test set phase. This phase replicates the uncertainty of a real-life scenario and works as an important assessment of their ability to use in diverse settings.

Validation and Testing

During the validation and testing stages of our credit risk assessment study, we highlight the implementation of KFold cross-validation to guarantee the reliability and relevance of our neural network models. KFold cross-validation is a systematic technique that partitions the entire dataset into 5 separate subgroups. Within each fold, a distinct subset is set aside as the validation set, while the model is trained on the other 'k-1' subsets. The process is iterated 'k' times, with each subset being used once as a validation set. The results obtained from each of the 'k' folds are subsequently averaged to get a thorough assessment of the model's performance. This strategy reduces the risk of overfitting by ensuring that each data point is validated, providing a strong understanding of the model's predictive ability across different subsets of data.

In addition, the test set, which is a distinct collection of data, is completely omitted from the training procedure and set aside for the ultimate assessment. It remains unaltered throughout both the training and cross-validation stages to preserve its integrity as an impartial assessor. The test set is introduced only after the model has undergone training and validation using the KFold approach. During this phase, the model's predictions are evaluated by comparing them to the actual values of the test data. This comparison offers essential insights about the model's performance in real-world situations, acting as a substitute for unseen data. The purpose of the test set is to evaluate the model's capacity to apply its acquired knowledge to unfamiliar situations, thus demonstrating its potential usefulness in real-world credit risk assessment tasks.

Given the nature of co-teaching, the two neural networks that were used for training are then combined with a majority prediction method that compares the scores of both classes and comes up with a final set of predictions that are used to evaluate the performance.

The models adopt a conservative decline rate of 20% to categorize possible defaults, with the objective of mitigating the likelihood of issuing high-risk loans. Strategically, erroneously classifying a consumer as creditworthy might result in significant financial losses. The implementation of a 20% decline rate ensures that only applicants with a substantial likelihood of repayment are categorized as 'default'. The model's strict standards minimize the probability of incorrect positive results, effectively managing the trade-off between the desire for credit growth and attracting new customers.

5. Results

The models were evaluated using the previously described metrics. The training process involved storing the scores of the relevant metrics to track the performance of the models across different folds and overall. To illustrate the motivation to opt for the threshold, ROC-AUC curves were constructed for the three models, along with their decline rate (DR) at each threshold. However, due to the uneven distribution of data in the dataset, it is necessary to consider additional metrics to accurately evaluate the performance of the models and obtain a comprehensive understanding of the situation.

Baseline Neural Network

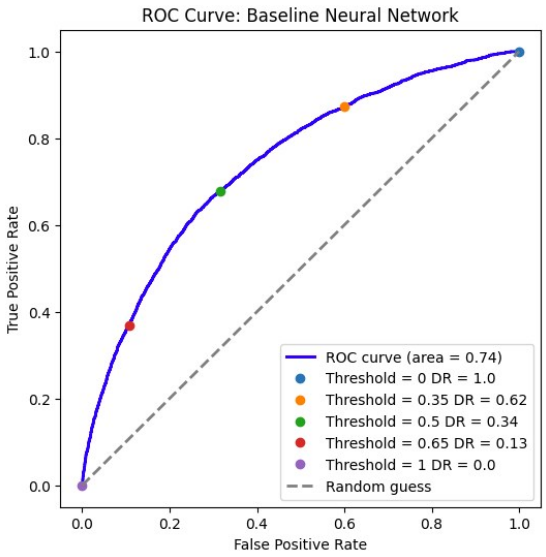

Figure 11: ROC Baseline

Co-teaching Fixed Forget Rate

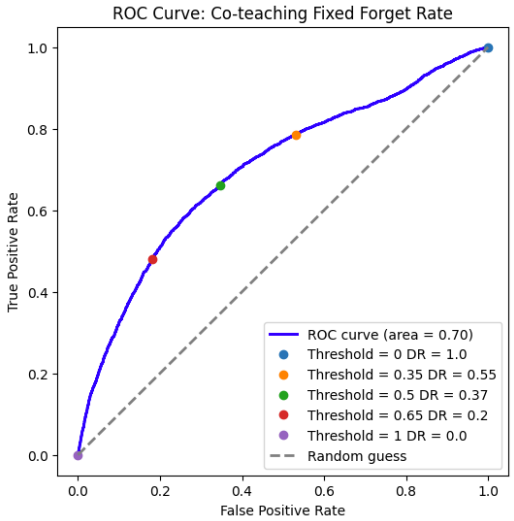

Figure 12: ROC Co-teaching Fixed Rate

Co-teaching Incremental Forget Rate

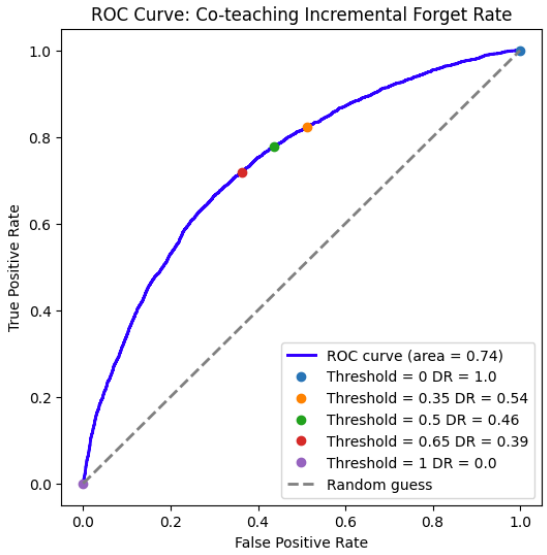

Figure 13: ROC Co-teaching Incremental Rate

The visual allows a quick comparison of the model's fluctuations and trends. The box plots provide a concise representation of the median, quartiles, and outliers for each model. The AUPRC scores stored during the training process are distributed as follows:

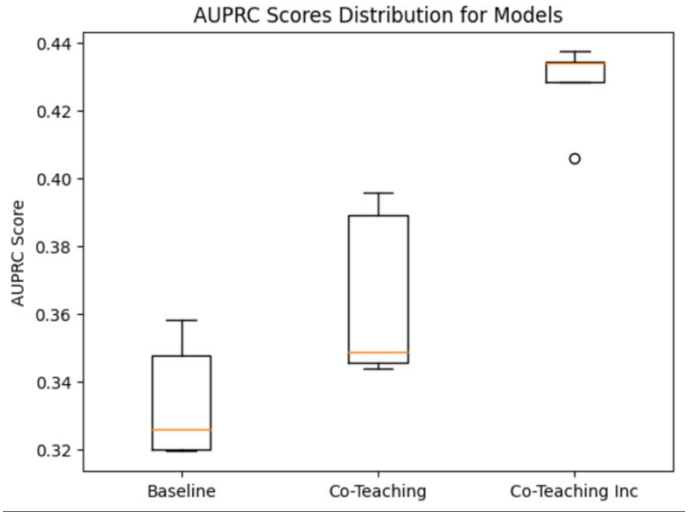

Figure 14: Distributions of AUPRC Scores

Following industry standards, the decline rate was established at 20% to achieve a balance between performance and the inclusion of the business perspective in the scores. With this rate, we simulate a real scenario in personal loans, where lenders might be more reluctant when

choosing the risk appetite. Given that defaults are costly, this percentage of rejected applications provide a high enough true positive rate with the expense of some false positives.

The required threshold was calculated for each of the models based on the decline rate. By establishing a consistent decline rate across the models, we can conduct a more accurate analysis that allows for improved comparisons, even if the thresholds vary. The table below presents the relevant metrics for evaluating the models:

Metric	**Baseline (Threshold = 0.59)**	**Co-teaching Fixed Rate (Threshold = 0.65)**	**Co-teaching Incremental Rate (Threshold = 0.90)**
Accuracy	0.80	0.80	0.80
Precision	0.19	0.18	0.19
Recall	0.50	0.47	0.49
Specifity	0.82	0.82	0.82
F1-Score	0.28	0.26	0.27
ROC-AUC	0.66	0.65	0.66
AUPRC	0.36	0.35	0.36

Furthermore, the business KPIs were computed based on the predictions made by the models, in addition to the previously mentioned metrics. By having access to the ground truth, we can accurately assess the financial consequences and gain valuable insights from a business standpoint for the models. The following table outlines the definition of a perfect score and the actual results compared to the estimates made by the models for various KPIs.

KPI	**Ground Truth**	**Baseline**	**Co-teaching Fixed Rate**	**Co-teaching Incremental Rate**
Application Default Rate	7.7%	3.8%	3.6%	3.7%
Expected Profit (in Millions)	4086.4	2688.96	2643.12	2673.72
Application Approval Rate	92.3%	96.2%	96.4%	96.3%

The baseline model with incremental rate would have identified the most defaults to materialize across the three models. The expected profit varies slightly among the models, with models exhibiting higher rates of application default generally experiencing lower losses. The KPIs help us to complete the view of the models, in addition to the evaluation metrics that are

processed during training, these values are more tangible. Particularly, when it comes to communicating with executives regarding the realization of the model's predictions in the real world.

To assess the effect in the handling of reject inference, the co-teaching models and baseline were compared to analyze its significance, the AUPRC scores of the three models were examined to test the hypothesis of this study.

First, a Kruskal-Wallis H test was employed to assess the statistical significance among the three groups. The analysis revealed a test statistic of approximately 10.5 and a p-value of about 0.005, indicating that there was a statistically significant difference among the three groups of model scores at the 0.05 significance level. This means there is enough evidence to reject the null hypothesis that there is no difference between the groups.

Then, a pair-wise test was performed to test the alternative hypothesis. The following scenarios were tested:

- **Baseline and co-teaching fixed rate**
- **Baseline and co-teaching incremental rate**

The Mann-Whitney U tests suggested that there was no statistically significant difference between the baseline and the co-teaching fixed rate models, as evidenced by a p-value of approximately 0.1507. However, the test indicates a statistically significant difference between the baseline and co-teaching incremental rate models, as shown by a p-value of 0.0079.

6. Discussion

The analysis results do not provide evidence that supports the hypothesis that co-teaching strategies are able to improve the performance of neural networks in the specific context of credit assessment. Specifically, the baseline model demonstrated better results compared to the standard co-teaching model and the co-teaching model with an incremental forget rate.

The Mann-Whitney U test indicates a statistically significant difference between the baseline and co-teaching incremental rate models. Still, while there is a significant difference, the direction of the effect is opposite to what was expected or desired.

The first impression is that the comparable accuracy and AUC-ROC scores among the models may not accurately reflect the models' performance. Accuracy might not be an appropriate metric in this situation because the ideal model should also be able to identify the minority class to some degree to prevent financial losses from defaulted credits. At the same time the low specifity and accuracy scores mean that the models could incorrectly predict a non-default as default, increasing the number of false alarms.

The baseline and the co-teaching with incremental forget rate models demonstrate superior performance compared to the co-teaching with fixed rate, with an AUPRC score of approximately 0.36. The AUPRC scores provide insight into the model's efficacy in an imbalanced scenario. In addition to having the highest recall score among the models, these two are crucial within the context of our study. The recall score offers valuable insights into the models' ability to accurately detect instances of the minority class, specifically the defaults.

In addition to that, the business KPIs serve to broaden the perspective on what is expected in relation to reality. This is especially significant because the financial effects of the models play a pivotal role in determining which model to implement in production. The ground truth is subsequently leveraged to assess the consequences of the models, using a benchmark that enables us to quantify the financial aspect.

The baseline model leads to a predicted default rate of 3.8%, which is somewhat comparable to the optimal default rate of 7.7%. In contrast, the co-teaching with incremental rate identified 3.7% and the co-teaching with fixed rate 3.6% of defaults. The small difference can be attributed to the respective model's profit. In practice, if we were deploying the baseline model to production, that would result in an expected profit of 2688.96 million, which is somewhat better than the co-teaching models.

Besides, within the context of credit assessment, costs incurred from false negatives outweigh those from false positives. Therefore, the baseline model seems to be more appropriate for this scenario at the determined decline rate of 20%.

That said, this study has certain constraints. The study lacks the real risk appetite of the lender, which could impact the performance of the models. This study omitted any domain characteristics in the pre-processing process. Allocating additional time to feature engineering could lead to better results compared to the evaluated models. The dataset includes numerous variables that influence the learning of the models, but it lacks specific information regarding the time aspect.

Also, the models are only trained using a single dataset. Therefore, the models might not have the ability to effectively apply their knowledge in a new setting if it significantly differs from their original training data. It would be ideal to augment the dataset with time series data, including its complete historical aspect. Because of this, the proprietary data utilized to train the models may not be applicable to other scenarios where the data has not been collected. It is important to emphasize that lenders should prioritize the utilization of datasets that accurately reflect the conditions in which they operate.

7. Conclusion and Recommendations

The research findings suggest that utilizing co-teaching strategies is ineffective in addressing label noise in neural network models used for credit assessment. The findings of this study provide evidence that co-teaching models, specifically those with a fixed forget rate, are less effective than traditional neural network models in dealing with the issues of reject inference.

This study contributes to the existing literature on the co-teaching principle and how noisy labels mitigate the effects of noise during the training process of the model. This research explores the potential use of co-teaching in tabular data, rethinking its utilization compared to previous studies, which predominantly was used in computer vision. It showcases the effectiveness of these strategies in mitigating the impact of noisy labels.

Applying the co-teaching principle to credit assessment has not proven to be beneficial in identifying applications with a high probability of defaulting. The baseline model predicts a higher percentage of observations from the minority class compared to the co-teaching models. From a business perspective, this is essential when considering the choice of the model, given the cost of defaults. This could be attributed to the high sensitivity of the primary parameter of the co-teaching approach, the forgetting rate. The selection of samples that the two neural networks use during training is influenced by the estimated forget rate. Besides that, the models solely rely on data from a single dataset, potentially compromising the reliability of the model when applied to a distinct environment.

While this study specifically examines the application of co-teaching in neural networks, there exist alternative approaches that utilize similar concepts to address the issues related to noisy labels (Yu, 2019). Hence, performing a comparative analysis of these techniques could be beneficial in evaluating which are appropriate for credit assessment.

In future studies, through deliberate adjustments to the forget rate, researchers could explore the potential variations in performance and determine which aspects of co-teaching are particularly influenced by the specific types of label noise found in credit risk datasets. Like De Vos et al., (2023), where the co-teaching principle is applied with unknown levels of noisy labels. In addition to that, we suggest to allocate additional time towards feature engineering and develop domain-specific features for future work.

Researchers should also investigate custom loss functions that are robust to label noise and credit data specificities. This approach could help optimize co-teaching for better performance

by focusing on minimizing mislabeled data impact and enhancing model sensitivity to subtle patterns. Custom loss functions could also incorporate domain-specific knowledge.

To conclude, the research contributes to the field by introducing and validating co-teaching strategies to tackle the ongoing issue of label noise. This study examines the applications of co-teaching in the domain of credit risk and establishes the basis for its broader adoption.

Bibliography

De Vos, B. D., Jansen, G., & Išgum, I. (2023, October 6). Stochastic co-teaching for training neural networks with unknown levels of label noise. Scientific Reports. https://doi.org/10.1038/s41598-023-43864-7

Liu, F. (2023, January 1). Asymmetric Co-teaching with Multi-view Consensus for Noisy Label Learning. arXiv.org. https://arxiv.org/abs/2301.01143

Chen, Y. (2022, June 27). Compressing Features for Learning with Noisy Labels. arXiv.org. https://arxiv.org/abs/2206.13140

Credit Card Approval Predictions Using Logistic Regression, Linear SVM and Naïve Bayes Classifier. (2022, February 1). IEEE Conference Publication | IEEE Xplore. https://ieeexplore.ieee.org/abstract/document/9763647Ehrhardt, A., Biernacki, C., Vandewalle, V., Heinrich, P., & Beben, S. (2021, May 20).

Ehrhardt, A., Biernacki, C., Vandewalle, V., Heinrich, P., & Beben, S. (2021, May 20). Reject inference methods in credit scoring. *Journal of Applied Statistics*, *48*(13–15), 2734–2754. https://doi.org/10.1080/02664763.2021.1929090

Beben, S. (2021). Reject inference methods in credit scoring. Journal of Applied Statistics, 48(13–15), 2734–2754. doi:10.1080/02664763.2021.1929090

Song, H. (2020, July 16). Learning from Noisy Labels with Deep Neural Networks: A Survey. arXiv.org. https://doi.org/10.48550/arXiv.2007.08199

Yu, X. (2019, May 24). How does Disagreement Help Generalization against Label Corruption? PMLR. https://proceedings.mlr.press/v97/yu19b.html

Han, B. (2018, April 18). Co-teaching: Robust Training of Deep Neural Networks with Extremely Noisy Labels. arXiv.org. https://arxiv.org/abs/1804.06872

Abdelgayed, T. S., Morsi, W. G., & Sidhu, T. S. (2018). Fault Detection and Classification Based on Co-training of Semisupervised Machine Learning. IEEE Transactions on Industrial Electronics, 65(2), 1595–1605. doi:10.1109/TIE.2017.2726961

Li, Z., Tian, Y., Li, K., Zhou, F., & Yang, W. (2017, May 1). Reject inference in credit scoring using Semi-supervised Support Vector Machines. Expert Systems With Applications. https://doi.org/10.1016/j.eswa.2017.01.011

Bootkrajang, J., & Kabán, A. (2013, September 26). Boosting in the presence of label noise. ResearchGate. https://www.researchgate.net/publication/257069474_Boosting_in_the_presence_of_label_noise

Natarajan, N., Dhillon, I. S., Ravikumar, P. K., & Tewari, A. (2013). Learning with Noisy Labels. In C. J. Burges, L. Bottou, M. Welling, Z. Ghahramani, & K. Q. Weinberger (Eds.), Advances in Neural Information Processing Systems (Vol. 26). Retrieved from https://proceedings.neurips.cc/paper_files/paper/2013/file/3871bd64012152bfb53fdf04b401193f-Paper.pdf

Appendix

Implementation of the models

```
# IMPORTS
import torch
import torch.nn as nn
import pandas as pd
from sklearn.model_selection import train_test_split
import numpy as np
import torch.optim as optim
import torch.nn.functional as F
import numpy  as np
from tensorflow import keras
import matplotlib.pyplot as plt
from ydata_profiling import ProfileReport
import keras_tuner as kt
from keras.models import Sequential
from keras.layers import Dense, Dropout
from keras.optimizers import Adam
from keras.regularizers import l2

# HELPER FUNCTIONS
def missing_values_table(df):
        # Total missing values
        mis_val = df.isnull().sum()

        # Percentage of missing values
        mis_val_percent = 100 * df.isnull().sum() / len(df)

        # Make a table with the results
        mis_val_table = pd.concat([mis_val, mis_val_percent], axis=1)

        # Rename the columns
        mis_val_table_ren_columns = mis_val_table.rename(
        columns = {0 : 'Missing Values', 1 : '% of Total Values'})

        # Sort the table by percentage of missing descending
        mis_val_table_ren_columns = mis_val_table_ren_columns[
            mis_val_table_ren_columns.iloc[:,1] != 0].sort_values(
        '% of Total Values', ascending=False).round(1)
        return mis_val_table_ren_columns

def transform_variables(df):
    df = df.fillna(0)
    categorical_columns = df.select_dtypes(include=['object']).columns
    df = pd.get_dummies(df, columns=categorical_columns)
    return df
```

```
def downsample(X, y):
    if y.ndim > 1:
      y = y.flatten()

    # Now y is guaranteed to be 1-dimensional
    y = pd.Series(y, name='defaulted')

    # Combine X and y into a single DataFrame
    X['defaulted'] = y.values
    data = X

    # Separate majority and minority classes
    majority_class = y.value_counts().idxmax()
    minority_class = 1 - majority_class

     df_majority = data[data['defaulted'] == majority_class]
     df_minority = data[data['defaulted'] == minority_class]

    # Downsample majority class
    df_majority_downsampled = resample(df_majority,
                                       replace=False,
                                       n_samples=len(df_minority),
                                       random_state=123)

    # Combine minority class with downsampled majority class
    downsampled_data = pd.concat([df_minority, df_majority_downsampled])

    # Separate features and labels
    X_downsampled = downsampled_data.iloc[:, :-1]  # Exclude the target
column
    y_downsampled = downsampled_data['defaulted'].values.reshape(-1, 1)
# Convert back to ndarray

    return X_downsampled, y_downsampled


def build_model(hp):
    model = Sequential()
    # Tune the number of units in the first Dense layer
    model.add(Dense(units=hp.Int('units1', min_value=10, max_value=100,
step=10),
                    activation='relu',
                    kernel_regularizer=l2(hp.Float('l2_1',
min_value=1e-5, max_value=1e-1, sampling='LOG')),
                    input_shape=(input_size,)))
    model.add(Dropout(hp.Float('dropout1', min_value=0, max_value=0.5,
step=0.1)))
```

```
    # Tune whether to add an extra Dense layer or not
    if hp.Boolean('extra_layer'):
        model.add(Dense(units=hp.Int('units2',             min_value=10,
max_value=100, step=10),
                        activation='relu',
                        kernel_regularizer=l2(hp.Float('l2_1',
min_value=1e-5, max_value=1e-1, sampling='LOG'))))
        model.add(Dropout(hp.Float('dropout1',              min_value=0,
max_value=0.5, step=0.1)))

    model.add(Dense(1, activation='sigmoid')) # Output layer

    # Tune the learning rate for the optimizer
    hp_learning_rate = hp.Choice('learning_rate', values=[1e-2, 1e-3,
1e-4])

    model.compile(optimizer=Adam(learning_rate=hp_learning_rate),
                  loss='binary_crossentropy',
                  metrics=['accuracy', 'AUC', keras.metrics.Precision(),
keras.metrics.Recall()])

    return model



# LOAD DATA
df                                                                    =
pd.read_csv('https://media.githubusercontent.com/media/alfonsogalvan10/
credit_inference_coteaching/master/application_train_all.csv')

# PROFILLING
profile = ProfileReport(df, title="Home Credit Dataset", html={"style":
{"full_width": True}}, sort=None, minimal=True)
#profile = ProfileReport(stratified_sample, interactions=None)
profile.to_notebook_iframe()

# HANDLING OF MISSING VALUES
threshold = 0.6
non_null_threshold = len(df) * (1 - threshold)
df = df.dropna(thresh=non_null_threshold, axis=1)
for column in df.select_dtypes(include=[np.number]).columns:
    if df[column].isnull().any():
        median_value = df[column].median()
        df[column].fillna(median_value, inplace=True)
for column in df.select_dtypes(include=['object']).columns:
     if df[column].isnull().any():
         mode_value = df[column].mode()[0]
         df[column].fillna(mode_value, inplace=True)
```

```
# DATA PREPROCESSING
df['AMT_INCOME_TOTAL_LOG'] = np.log(df['AMT_INCOME_TOTAL'] + 1)

plt.figure(figsize=(14, 6))
plt.subplot(1, 2, 1)
sns.histplot(df['AMT_INCOME_TOTAL'], kde=True)
plt.title('Original AMT_INCOME_TOTAL Distribution')

plt.subplot(1, 2, 2)
sns.histplot(df['AMT_INCOME_TOTAL_LOG'], kde=True)
plt.title('Log Transformed AMT_INCOME_TOTAL Distribution')

plt.show()

df['AMT_CREDIT_LOG'] = np.log(df['AMT_CREDIT'] + 1)

plt.figure(figsize=(14, 6))
plt.subplot(1, 2, 1)
sns.histplot(df['AMT_CREDIT'], kde=True)
plt.title('Original AMT_CREDIT Distribution')

plt.subplot(1, 2, 2)
sns.histplot(df['AMT_CREDIT_LOG'], kde=True)
plt.title('Log Transformed AMT_CREDIT Distribution')

plt.show()

df = transform_variables(df)

# SAMPLE SELECTION - ASSUMPTION TARGET TO 0 FOR REJECTED
# Approved batch
df_approved = df[df['rejected'] == 0]
df_approved['defaulted'] = df[df['rejected'] == 0]['TARGET']
# Rejected batch
df_rejected = df[df['rejected'] == 1]
df_rejected['defaulted'] = 0

df = pd.concat([df_approved, df_rejected])

# TRAIN TEST VALIDATION SPLIT
X_resampled = df.drop(['AMT_CREDIT', 'AMT_INCOME_TOTAL', 'defaulted',
'rejected', 'score', 'SK_ID_CURR'], axis=1)
y_resampled = df['defaulted']

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X_resampled,
y_resampled,
    test_size=0.2, shuffle=True, stratify=y_resampled, random_state=42)
```

```
X_train, X_val, y_train, y_val = train_test_split(X_train, y_train,
    test_size=0.25, shuffle=True, stratify=y_train, random_state=42)

ground_truth = X_test['TARGET']
X_test = X_test.drop(['TARGET'], axis=1)
X_train = X_train.drop(['TARGET'], axis=1)
X_val = X_val.drop(['TARGET'], axis=1)

X_train_downsampled, y_train_downsampled = downsample(X_train, y_train)

from sklearn.preprocessing import MinMaxScaler
scaler  = MinMaxScaler()
X_train = scaler.fit_transform(X_train_downsampled)
X_val   = scaler.transform(X_val)
X_test    = scaler.transform(X_test)
y_train = y_train_downsampled
# Define the input size
input_size = X_train.shape[1]

# HYPERPARAMETER OPTIMIZATION
from keras.callbacks import EarlyStopping
tuner = kt.RandomSearch(
    build_model,
    objective=kt.Objective("val_auc", direction="max"),
    max_trials=10,
    executions_per_trial=1,
    directory='my_dir',
    project_name='keras_tuner_auc_fin'
)

# Begin the random search
tuner.search(X_train, y_train,
             epochs=30,
             validation_data=(X_val, y_val),
             callbacks=[EarlyStopping(monitor='val_loss',
patience=10)])

# Get the optimal hyperparameters
best_hps = tuner.get_best_hyperparameters(num_trials=1)[0]


# BASELINE
seed(3)

loss_per_fold = []
accuracy_per_fold = []
auc_per_fold = []
precision_per_fold = []
```

```
# K-Fold Cross-Validator
k = 5  # Number of folds
kfold = KFold(n_splits=k, shuffle=True, random_state=3)

# Loop over each fold
fold_no = 1
for train in kfold.split(X_train, y_train):
    # Print the fold number
    print(f'Training on fold {fold_no}')

    model = build_model(best_hps)

    # Train the model
    history = model.fit(X_train, y_train,
                        epochs=30,
                        batch_size=32,
                        validation_data=(X_val, y_val),
                        callbacks=[EarlyStopping(monitor='val_loss',
patience=10)])

    # Increase fold number
    fold_no += 1

    final_scores = model.evaluate(X_val, y_val, verbose=0)

    avg_loss = (final_scores[0])
    avg_accuracy = (final_scores[1])
    avg_auc = (final_scores[2])
    avg_precision = (final_scores[3])
    avg_recall = (final_scores[4])

    #avg_auprc = auc(avg_recall, avg_precision)

    predictions = (model.predict(X_test) > 0.65).astype(int)

    precision, recall, thresholds = precision_recall_curve(ground_truth,
predictions)

    # Calculate the AUC (Area Under the Curve)
    auprc = auc(recall, precision)

    print(f'Score for fold {fold_no}: AUPRC {auprc}')

    print(f'Score for fold {fold_no}: Loss of {avg_loss}; Accuracy of
{avg_accuracy * 100}%; AUC of {avg_auc}; Precision of {avg_precision}')
    print('------------------------------------------------------------
-----------')
```

```
    loss_per_fold.append(avg_loss)
    accuracy_per_fold.append(avg_accuracy)
    auc_per_fold.append(avg_auc)
    precision_per_fold.append(avg_precision)

# == Provide average scores ==
print('------------------------------------------------------------------------
-------')
print('Score per fold')
for i in range(0, len(accuracy_per_fold)):
    print('------------------------------------------------------------------------
-----------')
    print(f'> Fold {i+1} - Loss: {loss_per_fold[i]} - Accuracy:
{accuracy_per_fold[i]*100}% - AUC: {auc_per_fold[i]} - Precision:
{precision_per_fold[i]}')
print('------------------------------------------------------------------------
-------')
print('Average scores for all folds:')
print(f'> Accuracy: {np.mean(accuracy_per_fold) * 100}% (+-
{np.std(accuracy_per_fold)})')
print(f'> Loss: {np.mean(loss_per_fold)}')
print(f'> AUC: {np.mean(auc_per_fold)}')
print(f'> Precision: {np.mean(precision_per_fold)}')
print('------------------------------------------------------------------------
-------')


# CO-TEACHING

# Initialize 5-fold cross-validation
kfold = KFold(n_splits=5, shuffle=True, random_state=42)

# Early Stopping Parameters
patience = 10
best_val_loss = float('inf')
patience_counter = 0

# Parameters
epochs = 30
batch_size = 32
forget_rate = 0.1
fold_no = 1
loss_per_fold = []
accuracy_per_fold = []
auc_per_fold = []
precision_per_fold = []

for train in kfold.split(X_train, y_train):
```

```
    # Reset the seeds and reinitialize the models for each fold using the
best hyperparameters
    seed(123)
    model1 = build_model(best_hps)
    seed(345)
    model2 = build_model(best_hps)

    print(f'Training on fold {fold_no}')

    # Co-teaching training loop
    for epoch in range(epochs):
        print(f"Epoch {epoch + 1}/{epochs}")

        # Shuffle the training data at the beginning of each epoch
        indices = np.arange(len(X_train))
        np.random.shuffle(indices)
        X_train_shuffled = X_train[indices]
        y_train_shuffled = y_train[indices]

        for i in range(0, len(X_train_shuffled), batch_size):
            X_batch = X_train_shuffled[i:i + batch_size]
            y_batch = y_train_shuffled[i:i + batch_size]

            # Train each model on the batch and get losses
            with tf.GradientTape() as tape1, tf.GradientTape() as tape2:
                predictions1 = model1(X_batch, training=True)
                predictions2 = model2(X_batch, training=True)
                loss1   =   tf.keras.losses.binary_crossentropy(y_batch,
predictions1)
                loss2   =   tf.keras.losses.binary_crossentropy(y_batch,
predictions2)

                # Select a portion of data with the smallest loss for
each model
                num_remember = int((1 - forget_rate) * batch_size)
                loss1_sorted_indices = tf.argsort(loss1)[:num_remember]
                loss2_sorted_indices = tf.argsort(loss2)[:num_remember]

                # Reduced losses for gradient computation
                reduced_loss1      =      tf.reduce_mean(tf.gather(loss1,
loss2_sorted_indices))
                reduced_loss2      =      tf.reduce_mean(tf.gather(loss2,
loss1_sorted_indices))

            # Compute gradients and update model parameters
            grads1            =            tape1.gradient(reduced_loss1,
model1.trainable_variables)
```

```
            model1.optimizer.apply_gradients(zip(grads1,
model1.trainable_variables))

            grads2            =            tape2.gradient(reduced_loss2,
model2.trainable_variables)
            model2.optimizer.apply_gradients(zip(grads2,
model2.trainable_variables))

        # Early Stopping Check (based on performance after each epoch)
        scores1 = model1.evaluate(X_val, y_val, verbose=0)
        scores2 = model2.evaluate(X_val, y_val, verbose=0)
        avg_loss = (scores1[0] + scores2[0]) / 2

        if avg_loss < best_val_loss:
            best_val_loss = avg_loss
            patience_counter = 0   # reset the counter if performance
improved
        else:
            patience_counter  +=  1    #  increment  the  counter  if  no
improvement

        # Break the loop if patience limit is reached
        if patience_counter >= patience:
            print(f"Early stopping triggered at epoch {epoch + 1}")
            break  # Breaks out of the epoch loop

    # Evaluate the models on the validation data
    final_scores1 = model1.evaluate(X_val, y_val, verbose=0)
    final_scores2 = model2.evaluate(X_val, y_val, verbose=0)

    avg_loss = (final_scores1[0] + final_scores2[0]) / 2
    avg_accuracy = (final_scores1[1] + final_scores2[1]) / 2
    avg_auc = (final_scores1[2] + final_scores2[2]) / 2
    avg_precision = (final_scores1[3] + final_scores2[3]) / 2

    predictions1    =    (model1.predict(X_test)    >    0.65).astype(int)
    predictions2    =    (model2.predict(X_test)    >    0.65).astype(int)
    majority_predictions  =  np.round((predictions1  +  predictions2)  /
2).astype(int)

    precision, recall, thresholds = precision_recall_curve(ground_truth,
majority_predictions)

    # Calculate the AUC (Area Under the Curve)
    auprc = auc(recall, precision)

    print(f'Score for fold {fold_no}: AUPRC {auprc}')
```

```
    print(f'Score for fold {fold_no}: Loss of {avg_loss}; Accuracy of
{avg_accuracy * 100}%; AUC of {avg_auc}; Precision of {avg_precision}')
    loss_per_fold.append(avg_loss)
    accuracy_per_fold.append(avg_accuracy)
    auc_per_fold.append(avg_auc)
    precision_per_fold.append(avg_precision)

    fold_no += 1
    best_val_loss = float('inf')
    patience_counter = 0

# == Provide average scores ==
print('------------------------------------------------------------------------
-------')
print('Score per fold')
for i in range(0, len(accuracy_per_fold)):
    print('------------------------------------------------------------------------
-----------')
    print(f'> Fold {i+1} - Loss: {loss_per_fold[i]} - Accuracy:
{accuracy_per_fold[i]*100}% - AUC: {auc_per_fold[i]} - Precision:
{precision_per_fold[i]}')
print('------------------------------------------------------------------------
-------')
print('Average scores for all folds:')
print(f'> Accuracy: {np.mean(accuracy_per_fold) * 100}% (+-
{np.std(accuracy_per_fold)})')
print(f'> Loss: {np.mean(loss_per_fold)}')
print(f'> AUC: {np.mean(auc_per_fold)}')
print(f'> Precision: {np.mean(precision_per_fold)}')
print('------------------------------------------------------------------------
-------')

# CO-TEACHING INCREMENTAL

# Initialize 5-fold cross-validation
kfold = KFold(n_splits=5, shuffle=True, random_state=42)

# Early Stopping Parameters
patience = 10
best_val_loss = float('inf')
patience_counter = 0

# Parameters
epochs = 30
batch_size = 32
fold_no = 1
loss_per_fold = []
accuracy_per_fold = []
```

```
auc_per_fold = []
precision_per_fold = []
initial_forget_rate = 0.1  # Starting forget_rate
forget_rate_increment = 0.01   # Amount to increment forget_rate each
epoch

for train in kfold.split(X_train, y_train):

  # Define a simple neural network model for binary classification
  seed(567)
  model3 = build_model(best_hps)
  seed(891)
  model4 = build_model(best_hps)

  print(f'Training on fold {fold_no}')

  # Co-teaching training loop
  for epoch in range(epochs):
      # Incrementally increase the forget_rate
      forget_rate = initial_forget_rate + epoch * forget_rate_increment
      forget_rate = min(forget_rate, 0.9)  # Cap forget_rate

      print(f"Epoch {epoch + 1}/{epochs}, Forget Rate: {forget_rate}")
      # Shuffle the training data at the beginning of each epoch
      indices = np.arange(len(X_train))
      np.random.shuffle(indices)
      X_train_shuffled = X_train[indices]
      y_train_shuffled = y_train[indices]

      for i in range(0, len(X_train_shuffled), batch_size):
        X_batch = X_train_shuffled[i:i + batch_size]
        y_batch = y_train_shuffled[i:i + batch_size]

        # Train each model on the batch and get losses
        with tf.GradientTape() as tape1, tf.GradientTape() as tape2:
          predictions1 = model3(X_batch, training=True)  predictions2
          = model4(X_batch, training=True)
          loss1      =      tf.keras.losses.binary_crossentropy(y_batch,
predictions1)
          loss2      =      tf.keras.losses.binary_crossentropy(y_batch,
predictions2)

          # Select a portion of data with the smallest loss for each
model
          num_remember = int((1 - forget_rate) * batch_size)
          loss1_sorted_indices = tf.argsort(loss1)[:num_remember]
          loss2_sorted_indices = tf.argsort(loss2)[:num_remember]
```

```
          # Reduced losses for gradient computation
          reduced_loss1        =        tf.reduce_mean(tf.gather(loss1,
loss2_sorted_indices))
          reduced_loss2        =        tf.reduce_mean(tf.gather(loss2,
loss1_sorted_indices))

        # Compute gradients and update model parameters
        grads1             =             tape1.gradient(reduced_loss1,
model3.trainable_variables)
        model3.optimizer.apply_gradients(zip(grads1,
model3.trainable_variables))

        grads2             =             tape2.gradient(reduced_loss2,
model4.trainable_variables)
        model4.optimizer.apply_gradients(zip(grads2,
model4.trainable_variables))

      # Early Stopping Check (based on performance after each epoch)
      scores1 = model3.evaluate(X_val, y_val, verbose=0)
      scores2 = model4.evaluate(X_val, y_val, verbose=0)
      avg_loss = (scores1[0] + scores2[0]) / 2

      if avg_loss < best_val_loss:
        best_val_loss = avg_loss
        patience_counter = 0
      else:
        patience_counter += 1  # increment the counter if no improvement

      if patience_counter >= patience:
              print(f"Early stopping triggered at epoch {epoch + 1}")
              break  # Breaks out of the epoch loop

  # Evaluate the models on the validation data
  final_scores1 = model3.evaluate(X_val, y_val, verbose=0)
  final_scores2 = model4.evaluate(X_val, y_val, verbose=0)

  avg_loss = (final_scores1[0] + final_scores2[0]) / 2
  avg_accuracy = (final_scores1[1] + final_scores2[1]) / 2
  avg_auc = (final_scores1[2] + final_scores2[2]) / 2
  avg_precision = (final_scores1[3] + final_scores2[3]) / 2

  predictions1 = (model3.predict(X_test) > 0.65).astype(int) predictions2
  = (model4.predict(X_test) > 0.65).astype(int) majority_predictions  =
  np.round((predictions1  +  predictions2)  /
2).astype(int)

  precision, recall, thresholds = precision_recall_curve(ground_truth,
majority_predictions)
```

```
  # Calculate the AUC (Area Under the Curve)
  auprc = auc(recall, precision)

  print(f'Score for fold {fold_no}: AUPRC {auprc}')


  print(f'Score for fold {fold_no}: Loss of {avg_loss}; Accuracy of
{avg_accuracy * 100}%; AUC of {avg_auc}; Precision of {avg_precision}')
  loss_per_fold.append(avg_loss)
  accuracy_per_fold.append(avg_accuracy)
  auc_per_fold.append(avg_auc)
  precision_per_fold.append(avg_precision)

  fold_no += 1
  best_val_loss = float('inf')
  patience_counter = 0

# == Provide average scores ==
print('------------------------------------------------------------------------
-------')
print('Score per fold')
for i in range(0, len(accuracy_per_fold)):
    print('------------------------------------------------------------------------
-----------')
    print(f'> Fold {i+1} - Loss: {loss_per_fold[i]} - Accuracy:
{accuracy_per_fold[i]*100}% - AUC: {auc_per_fold[i]} - Precision:
{precision_per_fold[i]}')
print('------------------------------------------------------------------------
-------')
print('Average scores for all folds:')
print(f'> Accuracy: {np.mean(accuracy_per_fold) * 100}% (+-
{np.std(accuracy_per_fold)})')
print(f'> Loss: {np.mean(loss_per_fold)}')
print(f'> AUC: {np.mean(auc_per_fold)}')
print(f'> Precision: {np.mean(precision_per_fold)}')
print('------------------------------------------------------------------------
-------')


# EVALUATION - BASELINE EXAMPLE
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import roc_curve, roc_auc_score

# Assuming ground_truth is defined and predicted_probs is the array of
predicted probabilities from your ensemble model
# Let's say we have a target decline rate of 20%
predicted_probs = model.predict(X_test).ravel()
```

```
target_decline_rate = 0.2

# Sort the predicted probabilities in descending order
sorted_probs = np.sort(predicted_probs)[::-1]

# Find the threshold which corresponds to the desired decline rate
threshold_index = int(len(sorted_probs) * target_decline_rate) - 1
threshold = sorted_probs[threshold_index]

# Calculate the new predictions using this threshold
new_predictions = (predicted_probs >= threshold).astype(int)

# Calculate the new decline rate, it should be close to the target decline
rate
new_decline_rate = 1 - (np.sum(new_predictions) / len(new_predictions))

# Calculate the ROC curve and ROC AUC from predicted probabilities
fpr, tpr, thresholds_roc = roc_curve(ground_truth, predicted_probs)
roc_auc = roc_auc_score(ground_truth, predicted_probs)

# Find the closest threshold index in the ROC curve data to our calculated
threshold
closest_threshold_index = np.argmin(np.abs(thresholds_roc - threshold))
closest_fpr = fpr[closest_threshold_index]
closest_tpr = tpr[closest_threshold_index]

# Plot the ROC curve
plt.figure(figsize=(6, 6))
plt.plot(fpr, tpr, color='blue', lw=2, label=f'ROC curve (area =
{roc_auc:.2f})')
plt.plot(closest_fpr, closest_tpr, 'o', label=f'Target DR =
{target_decline_rate:.2f} (Threshold = {threshold:.2f})')

plt.plot([0, 1], [0, 1], color='grey', lw=2, linestyle='--',
label='Random guess')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve with Target Decline Rate')
plt.legend(loc='lower right')
plt.show()

# Print the threshold that corresponds to the target decline rate
print(f"The threshold for a target decline rate of
{target_decline_rate:.0%} is: {threshold:.2f}")
print(f"The actual decline rate at this threshold is:
{new_decline_rate:.2f}")
```

```
# Calculate various evaluation metrics

accuracy = accuracy_score(ground_truth, new_predictions)
precision = precision_score(ground_truth, new_predictions)
recall = recall_score(ground_truth, new_predictions)
f1 = f1_score(ground_truth, new_predictions)
roc_auc = roc_auc_score(ground_truth, new_predictions)

# Calculate specificity
tn, fp, fn, tp = confusion_matrix(ground_truth, new_predictions).ravel()
specificity = tn / (tn + fp)

# Print the metrics
print(f"Accuracy: {accuracy:.2f}")
print(f"Precision: {precision:.2f}")
print(f"Recall: {recall:.2f}")
print(f"Specificity: {specificity:.2f}")
print(f"F1 Score: {f1:.2f}")
print(f"ROC-AUC: {roc_auc:.2f}")

# Calculate precision-recall
precision, recall, thresholds = precision_recall_curve(ground_truth,
new_predictions)

# Calculate the AUC (Area Under the Curve)
auprc = auc(recall, precision)

# Plot the precision-recall curve
plt.figure(figsize=(8, 6))
plt.plot(recall, precision, label=f'AUPRC = {auprc:.2f}')

# Add labels and title
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.title('Precision-Recall curve: Baseline Neural Network')
plt.legend(loc='best')

# Show the plot
plt.show()
```