CLASSIFICATION OF IMBALANCED CREDIT DATA SETS WITH BORROWER-SPECIFIC COST-SENSITIVE ALGORITHMS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF APPLIED MATHEMATICS
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

YASEMİN YAMAN KANMAZ

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF DOCTOR OF PHILOSOPHY
IN
FINANCIAL MATHEMATICS

JUNE 2023

Approval of the thesis:

CLASSIFICATION OF IMBALANCED CREDIT DATA SETS WITH BORROWER-SPECIFIC COST-SENSITIVE ALGORITHMS

submitted by **YASEMİN YAMAN KANMAZ** in partial fulfillment of the requirements for the degree of **Doctor of Philosophy in Financial Mathematics Department, Middle East Technical University** by,

Prof. Dr. Ayşe Sevtap Selçuk-Kestel
Dean, Graduate School of **Applied Mathematics** _______________

Prof. Dr. Ayşe Sevtap Selçuk-Kestel
Head of Department, **Financial Mathematics** _______________

Prof. Dr. Ayşe Sevtap Selçuk-Kestel
Supervisor, **Actuarial Sciences, IAM, METU** _______________

Prof. Dr. Şahap Kasırga Yıldırak
Co-supervisor, **Actuarial Sciences, Hacettepe Uni.** _______________

Examining Committee Members:

Prof. Dr. Ceylan Talu Yozgatlıgil
Department of Statistics, METU _______________

Prof. Dr. Ayşe Sevtap Selçuk-Kestel
Actuarial Sciences, IAM, METU _______________

Prof. Dr. Ömür Uğur
Scientific Computing, IAM, METU _______________

Prof. Dr. Çağdaş Hakan Aladağ
Department of Statistics, Hacettepe Uni. _______________

Assoc. Prof. Dr. Furkan Başer
Actuarial Sciences, Ankara University _______________

Date: _______________

Name, Last Name:	YASEMİN YAMAN KANMAZ
Signature :	

ABSTRACT

CLASSIFICATION OF IMBALANCED CREDIT DATA SETS WITH BORROWER-SPECIFIC COST-SENSITIVE ALGORITHMS

Kanmaz, Yasemin Yaman
Ph.D., Department of Financial Mathematics
Supervisor : Prof. Dr. Ayşe Sevtap Selçuk-Kestel
Co-Supervisor : Prof. Dr. Şahap Kasırga Yıldırak

June 2023, 163 pages

The unequal class distributions result in two types of prediction errors that incur different costs in imbalanced credit data sets. These are monetary losses for the misclassified defaults and opportunity cost of interest income for the misclassified non-defaults. Addressing these issues, this study proposes a novel approach to cost-sensitive learning and imbalanced data classification in credit data sets, using new borrower (instance)-specific cost/risk parameters to solve these two types of asymmetries.

The main objective of this study is to create a weight-signaling risk level for each instance by revealing instance-embedded information to strengthen ordinary algorithms with the generated weight and breaking the dominance of the majority class in the loss functions. The default probabilities of credit applicants provide valuable information about their risk level, and thus new instance-specific cost/risk parameters based on their default risk levels are proposed instead of class-specific ratios. Default probabilities are estimated with sampled sub-datasets, and before this step, analyses for the optimal class ratio of sub-datasets are conducted with the Simulated Annealing stochastic process. To estimate the default probabilities, non-linear complex models like logistic regressions, deep learning-based Graph Neural Networks, and Graph Attention Networks are employed. Three cost/risk parameters are generated with the target of equalizing the class losses based on their class-based default risk level aggregations.

AdaBoost, XGBoost, and ANN algorithms are then modified to incorporate these new parameters and the empirical analyses are conducted using eight credit data sets. The success of the proposed algorithms is particularly evident in the classification of data sets where the class ratios increase. The comparison analyses indicate that given Specificity values, the decrease in the monetary loss by new cost-sensitive algorithms can reach 33.7 % in the data set with the highest class imbalance.

Keywords: Instance-Specific, Default Probability, Logistic Regression, Graph Neural Networks, Graph Attention Networks, Articificial Neural Networks, XGBoost, AdaBoost

ÖZ

DENGESİZ KREDİ VERİ SETLERİNİN BORÇLUYA ÖZGÜ MALİYETE DUYARLI ALGORİTMALARLA SINIFLANDIRILMASI

Kanmaz, Yasemin Yaman
Doktora, Finansal Matematik Bölümü
Tez Yöneticisi : Prof. Dr. Ayşe Sevtap Selçuk-Kestel
Ortak Tez Yöneticisi : Prof. Dr. Şahap Kasırga Yıldırak

Haziran 2023, 163 sayfa

Dengesiz kredi veri setlerinde eşit olmayan sınıf dağılımları farklı maliyetlere yol açan iki tür yanlış tahmin hatası ile sonuçlanmaktadır. Bunlar, yanlış sınıflandırılan batık krediler için parasal kayıp ve yanlış sınıflandırılan kredisini ödeyecekler için kaçırılan fırsat maliyeti olarak faiz geliridir. Bu çalışma, belirtilen sorunları ele alarak kredi veri setlerinde maliyete duyarlı öğrenme ve dengesiz veri sınıflandırması asitmetrilerine yönelik borçluya özgü maliyet/risk parametrelerini çözebilen yeni bir yaklaşım önermektedir.

Çalışmanın temel amacı, her kredi başvurucusunun verilerinde saklı olan bilgileri ortaya çıkararak risk seviyesini gösteren bir ağırlık oluşturmak ve kayıp fonksiyonlarını bu ağırlık ile güçlendirmek ve çoğunluk sınıfının baskınlığını kırmaktır. Kredi almak için başvuranların temerrüt olasılıkları, risk seviyeleri hakkında değerli bilgiler sağlar. Bu çalışmayla, sınıfların veri büyüklük oranları yerine temerrüt risk seviyelerine dayalı her bir borçluya özgü maliyet /risk parametreleri önerilmektedir. Kredi ödememe olasılıkları, örneklenmiş alt veri kümeleriyle tahmin edilmekte ve bu aşamadan önce, Simüle Edilerek Kuvvetlendirilmiş (Annealing) stokastik süreciyle örneklenmiş alt veri kümelerinin en uygun sınıf oranı belirlenmektedir. Krediye başvuranların borçlarını ödememe olasılıklarını tahmin etmek için, lojistik regresyonlar ve derin öğrenmeye dayalı Grafik Sinir Ağları ve Grafik Dikkat Ağları gibi doğrusal olmayan karmaşık modeller kullanılmaktadır. Üç adet maliyet/risk parametresi, sınıf bazındaki kayıpları yine sınıf bazında toplam risk seviyelerine dayalı eşitleme hedefiyle oluşturulmaktadır.

AdaBoost, XGBoost ve ANN algoritmaları daha sonra bu yeni parametreleri içerecek şekilde değiştirilerek sekiz kredi veri seti üzerinde ampirik analiz gerçekleştirilmiştir. Bu algoritmaların başarısı, özellikle sınıf oranları giderek artan veri kümelerinin kategorizasyonunda daha belirgindir. Karşılaştımalı analizler, yeni maliyet duyarlı algoritmalarla parasal kayıplardaki azalmanın verilmiş Özgüllük değerlerinde % 33.7'ye ulaşabildiğini göstermektedir.

Anahtar Kelimeler: Örneğe-Özgü, Kredi Ödememe Olasılığı, Lojistik Regresyon, Grafik Sinir Ağları, Grafik Dikkat Ağları, Yapay Sinir Ağları, XGBoost, AdaBoost

To My Dear Family

ACKNOWLEDGMENTS

The patience, positive attitudes, and support of my thesis supervisors have greatly encouraged me and fostered a continuous desire to search for knowledge and improve myself throughout the thesis preparation process. I express my deep gratitude to my thesis supervisor, Prof. Dr. Ayşe Sevtap Selçuk-Kestel, for her never-ending support, guidance, and valuable advice during the development and preparation of this thesis. I am thankful to my thesis co-supervisor Prof. Dr. Ş. Kasırga Yıldırak for inspiring me to select and focus on the main topic, which has become a genuine research interest that can be further developed.

In addition, I would like to acknowledge the valuable supportive suggestions of Prof. Dr. Ceylan Talu Yozgatlıgil and Assoc. Prof. Dr. Furkan Başer during the thesis committee meetings. Their insightful reviews have expanded and contributed to the comparative analysis of my research topic.

I would like to thank Prof. Dr. Ömür Uğur and Prof. Dr. Çağdaş Hakan Aladağ for their valuable comments which have further improved my thesis.

I am very thankful for the support of my colleagues Ayfer, Ayşegül, Tuğba, Ekin, and Suat Bey for their motivation at work.

Special thanks to my mother, father, Sister Gülhan, and Brother Ersin for their assistance and encouragement.

Finally, I am grateful to Medet and my sons, for their support, encouragement, and understanding, and for their willingness to postpone family events during this demanding period of my academic journey. Their sacrifice and support have been invaluable.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

AdaBoost	Adaptive Boosting
ANN	Artifical Neural Network
CatBoost	Categorical Features Supporting Gradient Boosting
GAT	Graph Attention Network
GNN	Graph Neural Network
k-NN	K nearest neighbors
LGD	Loss Given Default
LightGBM	Light Gradient-Boosting Machine
LR	Logistic Regression
XGBoost	Extreme Gradient Boosting
RUSBoost	Random Under Sampling Boosting
SA	Simulated Annealing
SMOTEBoost	Synthetic Minority Over-sampling Technique Boosting

CHAPTER 1

INTRODUCTION

Credit risk analysts have extensively been using machine learning (ML) techniques to assess the financial risks in the financial system with various studies in the literature [46]. The availability of a wide range of historical data on bank loans allows the good use of machine learning techniques in such binary classification problems and credit scoring.

The number of defaults in loan data may be severely low and disproportionate to non-defaults, especially during the expansion and peak phases of the economic cycle. The skewed class distributions certainly pose the problem of class imbalance, where well-known machine learning techniques fail to develop the statistical inferences in depth which are necessary to identify who is likely to default. In this context, the problem of class imbalance appears as the first obstacle to be solved for a balanced detection of all classes by the classification algorithms.

The other critical issue arises in the minimization of misclassification costs in credit risk classification. Inaccurate estimates of binary classification in credit data do not result in similar cost amounts for all classes. Defaults on loans create losses for banks in terms of the principal and the interest payments, but the cost of the rejection of the loan for the customer who would have repaid if not rejected is only an interest income loss for the banks. As a result, the misclassification in credit risk management also creates a cost-sensitive learning problem. Class imbalance and cost-sensitive learning problems appear as two types of asymmetries in machine learning applications in credit risk management. Based on these obstacles, we question how ML can be incorporated into the estimation of default risk with high performance.

The proposed cost-sensitive classification design is built on a 2-stage algorithm level modeling; risk/cost parameters are estimated in the first part, and they are used to re-weight the data instances in three ordinary classification algorithms in the second part. New instance-specific cost parameters are calculated taking into account each individual's default risk level.

The re-weighting of the samples with default probabilities is rationalized by the fact that the riskiest ones should be out-weighted, and higher default probabilities of this class ensure assigning higher weights to these instances. Conversely, the non-risky class standing as the majority class should be lower-weighted, and their relatively lower default probabilities decrease the importance given to this class. Furthermore, the relative ratio of total default risk probabilities is used to re-weight the instances and to strengthen the minority class.

This study consists of 5 main sections; Section 2 underlies the importance of consumer credit default risk. Section 3 introduces the new method, the set-up of the 2-stage algorithm design, the generation of an instance-specific cost parameter with logistic regressions, GNNs and GATs with sub-datasets in the first algorithm stage, and the embedment of instance-specific cost parameters in the original algorithms selected as ensemble learning machines of Adaptive Boosting (AdaBoost) [23] and Extreme Gradient Boosting (XGBoost) [14], and Artificial Neural Networks (ANN) of which backpropagation developed by [94]. Section 4 presents the empirical data level classification results with comparisons of existing algorithms and the proposed cost-sensitive models. Section 5 concludes with a summary of the method proposed and its comparative success in detecting minority class instances.

1.1 Literature Review

In the literature, a number of hybrid methods have been proposed to address the issue of imbalanced data in consumer credit risk determination, along with comparative studies on decision rules for the classification of risky applicants.

Early methods, such as MetaCost [18], 'Thresholding' [76], and cost-sensitive decision rules based on Quinlan's C4.5 decision tree model and Bayes classifier [101], are well known early cost-sensitive algorithms.

In addition to these early methods, example-dependent cost-sensitive SVM algorithm designs have been proposed for cost-sensitive classification problems [12, 60, 41], while [9] generated a cost-sensitive method for fraud detection in credit cards based on real financial costs using Bayes minimum risk. [70] design a predictive model called ‘Cost-Sensitive Online Multiple Kernel Classification’ for data sets incrementally and sequentially updated in data streams.

Compared with the FICO credit scoring system which is a human expert-based system, [62] highlight the success of machine-learning approaches and emphasize the superior predictive success of the deep neural networks and XGBoost algorithm. Credit risk classification models with ANN are designed with the use of different network structures ([95],[103], [7], [6], [40], [61], [10]).

The study of [95] analyzes the performances of five neural network models of multi-layer perceptron, a mixture of experts, radial basis function, learning vector quantization, and fuzzy adaptive resonance on credit data. The multi-layer network model developed by [103] focuses on credit scoring and designs an Average Random Sampling method for the class imbalance problem in the data pre-processing stage. [7] develop a credit scoring model with a multi-layer perceptron (MP) network and evaluate the loss with the mean square errors and the misclassified costs which are found lower when they are compared with the values in linear regression and discriminant analysis. [6] analyze the customers’ missed credit card payments and generate a probabilistic model indicating customer behavioral scores called bidirectional Long-Short Term Memory (LSTM) neural network model. Recurrent Neural Networks (RNN) used generally with time series data are claimed to have vanishing gradients in the training stage which makes it difficult to capture the long-term dependencies. To use bidirectional LSTM, time steps of the input are required, and credit card data for sequential payment transactions are used in the dataset [6].

The classification algorithm design of [40] is based on a feed-forward artificial neural network (ANN) with cross entropy and mean square error losses for credit datasets. [61] investigate the performances of logistic regression, SVM, gradient boosting, random forest deep feed-forward neural network, and XGBoosting on a credit data separated as 2009-2013 as financial crisis period and 2014-2018 post-financial crisis period. The last years of each period are used as test data and the default rates are reported to be 4.16 % for the first period and 3.44 % for the second period. XG-Boosting, and neural networks as non-linear models are evaluated as outperforming the linear ones in most of the performance metrics. Model-Agnostic Meta-Learning (MAML) is famous as being one of the few-shot algorithms in neural networks and it has one inner loop determining the initial meta-parameters of the model and one outer loop in which the model is tested on the query data and the meta-parameters are updated. Based on MAML, [10] offer a MetaBalance algorithm in which outer-loop and inner-loop loss functions are decoupled to make different class balancing strategies possible. The algorithm is tested on different imbalanced data sets like image classification, credit card fraud detection, loan default prediction, and facial recognition data sets. The comparison of MetaBalance with random over-sampling, under-sampling, SMOTE, SVMSmote, Edited Nearest Neighbors (EEN), and Cluster Centroids (CC) indicates an improvement in ROC-AUC values, and the over-fitting problem is prevented with this new algorithm.

The study of [8] proposes cost-sensitive logistic regression with a logistic cost-lost function in which the coefficients of logistic regressions are estimated from the expected loss function. [36] choose the average expected cost as the objective function and use gradient-based optimization in the estimation of a cost-sensitive logistic model.

On the other hand, the introduction of XGBoost by [14] is appreciated due to its faster processing speed, minimum memory usage, and superior performance compared to other learning systems. XGBoost is known as a scalable end-to-end tree boosting system developed on gradient tree boosting and it is a solution for the over-fitting problem. Various comparative analyses for credit risk prediction conducted with different algorithms conclude with the superior performance of XGBoost ([97], [33], [96], [83], [89], [47], [68], [75], [59], [86], [61]).

The sequential ensemble credit scoring model developed by [97] uses XGBoost and selects its parameters with Bayesian hyper-parameter optimization. [97] utilize Tree-structured Parzen estimator (TPE) as Bayes optimization algorithm offered by [11].

The model offered, XGBoost-TPE, is tested on four credit data sets and found to be superior when compared with the algorithms of LR, NN, DT, SVM, bagging-NN, bagging-DT, AdaBoost-DT, AdaBoost-NN, and RF.

The study of [96] proposes Easy-SMT algorithm (EasyEnsemble+SMOTE+XGBoost) which combines the oversampling method SMOTE with the ensemble-based method EasyEnsemble. [83] investigate various algorithms for multi-class classification with an emphasis on modifications for imbalanced data, including AdaBoost.MH, SAMME (a special variant of AdaBoost), LogitBoost [24], GradientBoost, XGBoost, LightGBM [43], CatBoost (Gradient Boost capable of processing categorical variables), SMOTEBoost, RUSBoost, MEBoost (different weak classifiers are mixed), AdaCost, AdaC1, AdaC2, and AdaC3. CatBoost is proved to be the first algorithm with the highest performance metrics and SMOTEBoost is the second and XGBoost and LogitBoost are the third algorithms in terms of $MAUC$ (average of AUC), G_{mean}, and $MMCC$ values. [89] analyze the performance metrics of Imbalance-XGBoost (Weighted-XGBoost or Focal-XGBoost) with 5 imbalanced datasets. $F1-score$ and Matthews Correlation Coefficient (MCC) are evaluated as more proper measures than the accuracy in imbalance data, and they are reported as higher in Imbalance-XGBoost when they are compared with the ordinary XGBoost.

The comparative study of [47] evaluating the classification performance of XGBoost with logistic regression and tree-based models on credit data underlines the superiority of XGBoost. [34] propose the Adaptive Swarm Optimization (APSO) algorithm to optimize the hyper-parameters of random forest, while [68] use APSO to determine the optimal values of multiple XGBoost hyper-parameters for imbalance datasets. [59] generate a new hybrid method called SMOTEXGBoost and claim that increasing the minority class with this method rather than duplicating prevents the over-fitting problem. [61] compare the performances of logistic regression, SVM, gradient boosting, random forest deep feed-forward neural network, and XGBoosting on a credit data separated as 2009-2013 as financial crisis period and 2014-2018 post-financial crisis period. The last years of each period are used as test data and the default rates are reported to be 4.16 % for the first period and 3.44 % for the second period. The non-linear models, XGBoosting, and neural networks are evaluated as outperforming the linear ones in most of the performance metrics.

In their comparative study to analyze the performance of five well-known base classifiers (neural network, decision tree, logistic regression, Naïve Bayes, and support vector machine) in credit scoring, [49] state the supremacy of ensemble learning algorithms (random forest, AdaBoost, XGBoost, LightGBM [43], and Stacking), except for AdaBoost. [31] explore the superior classification ability to boost algorithms compared with other ensemble-learning methods and neural-network methods. [28] state that their comparative analysis demonstrates the success of stochastic gradient boosting, which outperformed decision trees and random forests.

The hybrid model proposed by [54] combine under-sampling and AdaBoost, and generate EasyEnsemble and BalanceCascade which are evaluated to be superior to other algorithms in the classification of 16 imbalance datasets with several performance metrics. [74] offer random under-sampling (RUS) technique as a simpler, faster, and favorable technique in terms of performance when compared with the synthetic minority oversampling technique (SMOTE) offered by Chawla et al [13] and later improved by [32] as BorderlineSMOTE and by [38] as Modified SMOTE. Sampling methods of RUS and SMOTE are also combined with AdaBoost and they are called RUSBoost and SMOTEBoost, respectively[38]. [33] strengthen the BalanceCascade in terms of maintaining the class ratio and adjusting the imbalance ratio of positive and negative classes in each iteration of the training data. Random forest and XGBoost are selected as the base classifiers and Particular Swarm Optimization (PSO) is applied to the optimal combination of the base classifiers in the final classification function. In the context of a search for the data imbalance problem, [96] propose a new algorithm called Easy-SMT (EasyEnsemble+SMOTE+XGBoost) which is a combination of an oversampling method, SMOTE, and ensemble-based method, EasyEnsemble. [59] generate a hybrid method of SMOTE and XGBoost called SMOTEXGBoost.

An improved version of the random forest offered by [105] assigns weights to decision trees in the forest during tree aggregation for prediction and out-of-bag errors to facilitate the calculation of the weights. [90] apply ensemble learning with the logistic regression as a base classifier to large unbalanced credit data (the ratio of defaults to full payments is 1:14). [22] generate a new algorithm called Non-linear Boosting (NLB) similar to boosted neural networks in which the embedding layer is learned with boosting. Different from gradient boosting (GB), in NLB, weak learners are iteratively processed and fewer and weaker base classifiers with fewer iterations and non-linear combinations of them in the final function can make the classification of the inputs in NLB.

A comparative study of [21] investigates the performances of Logistic regression (LR), support vector machines (SVM), and multilayer perceptrons (MLP) in imbalanced data classification. Random, Condensed Nearest-Neighbor (CNN), Edited Nearest-Neighbor (ENN), Tomek's Links, One-side-selection (OSS), NearMiss-1, and NearMiss-2 are used as the under-sampling methods. On the other hand, Random, Subclass-based, SMOTE, and Adaptive synthetic (ADASYN) are the over-sampling methods analyzed in the study. At the algorithm level, Easy Ensemble (sample size adjustments made in AdaBoost), Bagging, Gradient boosting decision trees as ensemble learning, and weighted loss functions (cross-entropy loss for LR and MLP, hinge loss for SVM) as cost-sensitive approaches are all evaluated. In addition, kernel-based SVM (Gaussian) and XGBoost (with a scale adjustment for the weights of the minority class) as model-specific algorithms which cannot be integrated with ensemble learning are also involved in the comparative analysis. The gradient-boosting decision trees are stated to give the best results in two of three data sets.

The classification study of [102] undertakes 56 imbalanced datasets with 14 ensemble algorithms designed with the dynamic selection strategy. A comparative analysis of the prediction capability of the ensemble classifications like AdaBoostNC, AdaC2M1, FuzzyImbECOC, MHDDTECOC, HDDTOVA, ImECOCdense, ImECOCOVA, ImECOCsparse, MCHDDT, MultiIMAO, MultiIMOAHO, MultiIMOVA, MultiIMOVO, and Piboost are made with the multi-class imbalance datasets and the most of these algorithms are found to have better prediction results when the dynamic selection strategy is applied. Furthermore, [102] design a patch learning in the scheme of a dynamic selection ensemble classification with AdaC2M1 as the global classifier and one-class SVM as the patch classifier.

The study of [50] offers to reshape the standard cross entropy loss and create a focal loss function to detect a sparse set of hard examples in the existence of a class imbalance problem. [89] re-formulate the loss functions of weighted cross-entropy and focal loss with imbalance parameters and call the new algorithm as Imbalance-XGBoost. [86] introduce Modified Focal Loss in which the class losses are balanced with the inverse of class weights. [85] introduce an imbalance parameter from Weighted-Cross Entropy Loss to the Focal Loss in XGBoost and name it Modified Focal Loss. [85] claim that the proposed model based on the XGBoost algorithm is superior to Vanilla XGBoost, Weighted-Imbalance XGBoost, and Imbalanced XGBoost with Focal Loss.

The comparative analysis conducted by [64] undertakes the imbalanced data and compare the performances of 2 data re-sampling methods, namely SMOTE and Deep Belief Network (DBN), and 2 successful cost-sensitive methods, namely focal loss and weighted loss. Deep Belief Network is described as the unsupervised probabilistic deep learning model composed of Restricted Boltzmann Machines (RBM) which takes the inputs from the hidden layer of the previous RBM. AUC values indicate that the data pre-processing methods of SMOTE and DBN are proven to improve the performances of Logistic Regression and XGBoost. [63] and [53] minimize the focal loss functions in imbalanced credit data sets. The use of focal loss as a solution for imbalanced data is also common in image and signal processing, and semantic segmentation ([37], [51], [99], [66]).

The hybrid method of [17] combines the simulated annealing (SA) strategy as a data level under-sampling and different algorithms like discriminant analysis, SVM, decision tree, and k-NN. The simulated annealing strategy, one of the meta-heuristic techniques, selects an optimal subset of the majority class instances and converges to the global optimum. The experimental analysis is conducted on 51 real-world imbalanced datasets and the evaluation metrics used G_{mean} and F-score indicate a superior performance for the SA algorithm.

Graph representation learning based on the main setting of information transfer between data instances is one of the successfully evolving techniques in deep learning algorithms in recent years. Graph Neural Networks (GNNs) and Graph Attention Networks (GATs) are also used for the estimations of expected default probabilities of the instances. GNNs and GATs formulate the data instances in a graph structure and the main motivation is to strengthen each data instance's features with very close neighbor data instances ([27],[104],[30]).

Graph Convolutional Networks (GCN) offered by [93] aggregates and encodes the nodes' features with neighbors' information directly and train the weight parameters on the same weight matrix. [56] underline the significance of the structure of combining the node embeddings and offer a pooling of the node features based on graph Fourier transform in GCNs. [58] focus on the graphs with a dynamic structure with changing vertices and edges, and propose a combination of Long Short-Term Memory networks and GCNs. [16] generate graph clustering algorithm and restrict the neighborhood with Cluster-GCN algorithm as a solution for the computational difficulty of large-scale data sets.

Underlying the fact that some neighbors' features can be more critical to the node and the attention coefficients between each node and its neighbors can be included to strengthen the information embedding, [98] emphasize the existence of different relations between nodes and their neighbors and generate GAT with multi-view network embeddings with an attention mechanism.

An advanced deep learning design introduced by [78] combines GCN and RNN with an attention mechanism for credit scoring. [45] apply GCN for credit default prediction and undertake three separate categories as loan information, credit history information, and soft information to identify the relationships between the borrowers.

Graph Sample and Aggregate (GraphSAGE) proposed by [29] emphasizes the notion that a node can have different graph structures besides the selected neighbors for the training stage and claim the inclusion of unseen graphs with sampling the neighbors. The neighbors are uniformly sampled and the messages can be aggregated with different architectural functions like a mean aggregator, LSTM aggregator, and pooling aggregator [29].

The study of [26] offers a new attention mechanism with a new edge matrix as edge feature enhanced GNNs and it is different from the one of the ordinary design of GNNs, since it is multidimensional and indicates the edges between the features of the inputs. [42] embeds the weight and distance functions of kNN in GNN and designs a kNNGNN model. A node has more than one edge set and messages are updated to find the optimal kNN distance and weight parameters in GNN design. [100] propose GNNEXPLAINER as a new approach to explain the predictions of GNNs which can be complex due to the graph structure and feature embedding. GNNEXPLAINER undertakes the trained GNN with its predictions and generates a subgraph of the input data indicating small subsets of node features being effective in the predictions.

The ordinary graph design offered by [71] is expanded with the vertex and edge sets, and node embeddings with the inclusion of coordinate embeddings as squared distances between two inputs, and the new structure is called Equivariant Graph Neural Networks. [15] focus on capturing a better graph structure with better node embeddings and insert cosine similarity learning function in the graph learning process to reach better graph topology and introduce 'Iterative Deep Graph Learning'.

Camouflage-Resistant GNN developed by [19] has fraudster camouflages as masking features and the relations in fraud detection. Different from most of the studies, [19] also utilize a different similarity measure called label-aware similarity measure estimated with one-layer perceptron in which the node label is predicted and the distance between prediction results is used as the similarity metric for the neighbor selection.

A GNN model designed by [106] incorporates a hierarchical node selection algorithm to detect vulnerable nodes which have a high potential to attack security systems in the Internet of Things (IoT) technologies. [39] introduce Embedding Clustering-based Optimization (ECO), Graph Reconstruction-based Optimization (GRO), and Hard Sample-based Knowledge Distillation methods for imbalanced data problems. [81] focus on the imbalanced-graph structured data in Reinforcement Learning and generate the model of Reinforced Contrastive GNNs to detect anomalies.

The study of [92] designs a GNN of GNNs for imbalanced data with the use of GNNs in the first level and the topological similarity and different kernel functions for the graphs in Graph Isomorphism Network (GIN) in the second level. [91] generate synthetic minority nodes and apply cost-sensitive learning with a Gumbel distribution (type-I generalized extreme value distribution) as an activation function for the imbalanced data. [55] generate sub-graphs with the sampled nodes and edges for fraud detection in imbalanced data.

1.2 The Aim of The Study

In the literature, studies dealing with instance-specific cost-sensitive classification algorithms are very few. The base of MetaCost [18] and cost matrix analysis by [101] and example dependent cost-sensitive SVM by [41], the studies of [8] and [36] based on logistic regressions all generate example dependent cost sensitive algorithm designs with the estimations of expected costs which are generally inserted in the algorithms as the monetary burden of the misclassification.

The main aim of this study is to introduce new cost-sensitive detection methods for credit risk classification when traditional algorithms fail to classify in the presence of severe class imbalance. In the literature, cost sensitivity is extensively applied with re-weighting on a class basis in classification algorithms. The instances or the errors resulting from predictions of these instances in the minority class are out-weighted with pre-determined constants defined on no specific rule or a grid search is applied for the optimal class weight. Instance-dependent cost-sensitive classification studies are very few and focus on expected nominal costs calculated with credit amount disbursed. It is important to notice that the nominal monetary costs might be misleading since every high credit amount does not mean high risk. The applicant should also be evaluated with his/her other financial features and relative comparisons of monetary indicators should be made. There is no well-established methodology in the literature that designs a classification algorithm built on sample-specific cost/risk indicators based on expected default probabilities. This study raises the question of whether it is possible to create an instance-specific cost parameter that reflects the default risk level of the instances and contributes to well-known algorithms in the process of diagnosing defaults in imbalanced credit data sets.

This study questions whether it is possible to improve the estimation of the default probabilities in the ordinary models of AdaBoost, XGBoost, and ANN with instance-specific data weights generated with default probabilities in sub-datasets. Logistic regression, GNN, and GAT models are generated with sub-train data sets which are equally distributed classes, and the maximums of the default probabilities of all data instances are used for the proposed cost/risk parameters. The improvement with new parameters in ordinary models can be observed with higher correct classification rates for both classes and with the identification of the riskiest applicants since their weight will be relatively higher in the given data set due to their higher default probabilities estimated in sub-datasets. The empirical findings of eight credit data set prove that the proposed instance-specific cost parameters can outperform the existing algorithms if the default probabilities in sub-datasets are successfully estimated. The success of the models in the first stage depends on logistics models with higher R^2 values, not over-fitted GNN and GAT models, and higher *Sensitivity* and *Specificity* values both for the train and test data.

CHAPTER 2

PRELIMINARIES

2.1 Borrower's Credit Default Risk

Consumer credits are one of the main crucial demand side supporting dynamics in the economies as a source of liquidity. They facilitate the purchase of goods and services, the repayment of existing debts, and prevent sharp reductions in household demand when they encounter liquidity crunches. During boom periods of business cycles, consumer credits tend to increase while they decrease during economic contractions. Repayment of consumer credits is also linked to the soundness and the course of the economy, and unexpected shocks can adversely affect the productive side of the economy. Shocks in the economy can deeply force the aggregate output to drop and increase unemployment, which can further cause credit defaults.

On the other hand, a high number of defaults in credit repayments can also cause a widespread crisis in the economies. Massive, uncontrolled risky credit disbursements along with their leverage with securities in the bonds market caused financial crises in 2007 with sub-prime mortgage crises which turned out to be a global financial crisis and spread to the real sector creating a subsequent deep recessionary period.

Consumer credits constitute one of the highly profitable balance sheet items for the lenders which are generally commercial banks or other financial institutions. However, it can be the riskiest balance sheet item for lenders if the default risks are not properly measured. The incorrect assessment of default causes monetary loss for the lender. Thus, this very fact should be taken into serious consideration when choosing the classification approach and this is where we offer a novel cost-sensitive approach.

Credit applicant assessments should certainly be done with close monitoring of the expectations in the economy such as the response and the trend of the macro-variables to existing and expected new policies and corresponding actions of the policymakers. In the contractionary periods of the economy, the risky sectors can be determined, and lenders can avoid lending to people working in these sectors, and other precautionary measures like setting lower limits to credit amounts, charging higher interest rates, and increasing the asset value demanded mortgages can be taken.

During the expansion periods, the economies, the interest rates tend to be relatively lower if the inflation is low in a country, and as a result, people are more willing to borrow because the cost of borrowing is lower. Domestic banks can borrow from other financial institutions abroad relatively easily due to the country's well-performing macro indicators like higher records in Gross Domestic Product (GDP) leading to higher real growth rates, government spending, aggregate investments and employment, lower inflation, and lower current account deficit, and these all lead excess liquidity and credit supply as well. The defaults in consumer credits also appear relatively lower in booming periods of the business cycle when compared with the contraction and trough periods. During the expansion periods, lenders have difficulty identifying risky borrowers as the number of defaults is very few. Lenders such as private commercial banks and financial institutions face the difficulty of identification of risky borrowers as the ratio of defaults to non-defaults in credit data sets becomes low in the economic expansion periods. The imbalanced data problem in credit data sets requires analysts to focus on more advanced default recognition models for the detection of the borrowers who are to default.

Personal information collection becomes very important for imbalanced credit data sets. The advanced classification techniques assign a probabilistic estimate for the default potentials of the credit applicants based on the personal attributes reported. Financial institutions as lenders can also put the financial history of the borrower under strict surveillance by checking the regularities and delays in credit card payments delays, previous credit payments, bank account check-ins, and salary payments. To extract more features besides the reflections on application reports, creditors can also search non-reported data sources, such as social media activity, online behavior, and social network to provide a more detailed view of an individual's creditworthiness.

2.2 Machine Learning Performance Metrics for Imbalanced Credit Data Sets

Assume Y denotes the binary class label of the borrower, where the non-default class is labeled as O, otherwise as 1. Let $(x_{i1}, ...x_{iF}) \in X_i$ and X_i represent the feature set of the borrower i, where $i \in N$, N, and F are the number of the borrowers and the features, respectively, of a given credit data set. $h(\theta, X, Y)$: $X \rightarrow (0, 1)$ represents the prediction of the selected classification algorithm with the loss minimizing hyperparameter θ.

There are several metrics available for analyzing classification performances of Y. The performance metrics can be computed with the four classification results reported in the confusion matrix table, as given in Table 2.1, in which the positive class and the negative class indicate the default class and the negative class, respectively:

Table 2.1: Confusion Matrix

		True Classes	
		Positive $(Y = 1)$	Negative $(Y = 0)$
Predicted Classes	Positive $(Y = 1)$	True Positives (TPs)	False Positives (FPs)
	Negative $(Y = 0)$	False Negatives (FNs)	True Negatives (TNs)

The correctly predicted borrowers of the default class are called True Positives, TPs, and the correctly predicted borrowers of the non-default class are called True Negatives, TNs. On the other hand, the falsely predicted borrowers of the default class are called False Negatives, FNs, and the falsely predicted borrowers of the non-default class are called False Positives, FPs.

The asymmetric class distributions in imbalanced data sets complicate the learning parameters of the algorithms to focus on the default class since the non-defaults dominate the loss functions in classification problems of credit data sets. This study aims to address the issue of imbalanced class size in credit data sets by achieving balanced correct classification results in both classes using offered cost/risk parameters in classification algorithms. However, most of the performance metrics are reliable and indicative of the actual success of the prediction task only when the class sizes in the data sets are relatively balanced. The most commonly used metric is accuracy, which represents the ratio of correct predictions to all data instances. The accuracy score can be misleading when the data classes are imbalanced, as the majority class instances will dominate the score, while the correct classification of minority class instances may be quite low. In credit data classification tasks, the priority is to detect the default class, as financial institutions incur a significant real monetary cost when an applicant fails to repay their loan. Nevertheless, classification algorithms should aim to identify both classes as effectively as possible. Therefore, performance metrics of imbalanced credit data sets are chosen to be fair for both classes, such as True Positive Rate or *Sensitivity* and True Negative Rate or *Specificity*.

$$\textit{Sensitivity} = \frac{TP}{TP + FN}, \tag{2.1}$$

$$\textit{Specificity} = \frac{TN}{TN + FP}. \tag{2.2}$$

Classification algorithms such as AdaBoost and XGBoost assign direct class labels and make the binary classification with a threshold of 0.5 for the probability estimations [23], [97]. Algorithms might not yield close *Sensitivity* and *Specificity* values at a threshold of 0.5, and probability estimates may require a threshold adjustment for final classification. Therefore, the first classification successes of all algorithms are compared with the geometric mean of *Sensitivity* $\times$ *Specificity*:

$$G_{mean} = (\textit{Sensitivity} \times \textit{Specificity})^{1/2}. \tag{2.3}$$

Different from other classification tasks, the monetary Loss Given Defaults, *LGD*, with predetermined *Specificity* value is evaluated as the real performance metric in credit data classification comparisons. *LGD* is calculated as the sum of the defaulting borrower-specific losses. That is, *LGD* is exactly the total credit amounts disbursed to the borrowers categorized as FNs in the confusion matrix and expressed as:

$$\textit{LGD} = \sum_{k=1}^{FN} Credit\ \ Amount_k \tag{2.4}$$

when $h(\theta, X_k, y_k) = 0$ and $Y_k = 1$ for $\forall\ k \in (1, .., FN)$.

Moreover, the expected misclassified cost, $MC_{Y=1}$, for all classification algorithms used in the experimental stage is computed as:

$$MC_{Y=1} = Credit\ \ Amount_{(h(\theta,X,Y)=0|Y=1)} \times P(\hat{Y} = 0 \mid Y = 1) \times P(Y = 1). \tag{2.5}$$

CHAPTER 3

PROPOSED METHODOLOGY

This section presents the methodology on how instance-specific cost/risk parameters are generated in three selected machine learning algorithms. The reason behind the necessity of the generation of these parameters lies behind the biased total loss minimization of the ordinary algorithms in the presence of class imbalance. The proposed cost/risk parameters aim to balance loss minimization in terms of class basis and increase the detection of the minority (default) class instances which will further reduce the monetary loss. Since the real financial loss in credit data sets stems from the misclassification of the minority/default class instances. This study is accomplished by two stages at which the first one contains two major steps to estimate the parameters and the second one embeds these outputs to minimize the binary classification costs in the imbalanced credit data sets.

Given N borrowers or data instances in a given credit data set and F features or attributes, let $x_{ij_{(i=1,...,N,J=1,...,F)}}$ for $\forall\ x_{ij} \in X$ are the predictors for $\forall\ i \in N$ and $\forall\ j \in F$. The binary classes is represented by $y_i \in Y = (0,1)$ for $\forall\ i \ \epsilon\ N$ are such that $y_i = 1$ is the default class instance while $y_i = 0$ is the non-default class instance. $h(\theta, X, Y)$: $X \rightarrow (0,1)$ is the selected binary classification algorithm and $\hat{Y} = h(\theta_1, ..., \theta_n, X, Y)$ denotes the predictions of this classification algorithm with n learning parameters θ. The total loss is defined as $\mathcal{L}(Y, \hat{Y})$.

The classification algorithms, $h(\theta_1, ..., \theta_n, X, Y)$, all optimize the learning parameters, $\theta_1, ..., \theta_n$, with the objective of minimization of pre-determined loss function, $\mathcal{L}(Y, \hat{Y})$. In binary classification problems, the losses can be minimized for both classes if the class sizes are almost equal. The loss function might not require an external factor to divert the parameters to minimize the losses for both classes. The commonly selected loss function in binary classification tasks is the cross entropy loss or the negative binomial loss given as ([52], [35], [82], [69], [89], [57]):

$$\mathcal{L}(Y,\hat{Y}) = -\frac{1}{N}\sum_{i=1}^{N}\Bigg[y_i \log \hat{y}_i + (1-y_i)\log(1-\hat{y}_i)\Bigg]. \tag{3.1}$$

Let $\mathcal{L}_i(y_i, \hat{y}_i = h(\theta, x_i, y_i))$ be the prediction loss of the specific borrower x_i. The sum of the misclassification losses for the defaults and the non-defaults are demonstrated as $\sum_{h(\theta,X_i,y_i)=0 \text{ and } y_i=1} \mathcal{L}_j(y_i, h(\theta, X_i, y_i))$ and $\sum_{h(\theta,X_j,y_j)=1 \text{ and } y_j=0} \mathcal{L}_j(y_j, h(X_j, y_j))$, respectively. We define the corresponding total loss as:

$$\mathcal{L}(Y, h(\theta, X, Y)) = \sum_{\forall k \epsilon N \text{ for } h(\theta,X_k,Y_k)\neq Y_k} \mathcal{L}_k(y_k, h(\theta, x_k, y_k)) \tag{3.2}$$

$$\begin{aligned}\mathcal{L}(Y, h(\theta, X, Y)) = &\sum_{h(\theta,x_i,y_i)=0 \text{ and } y_i=1} \mathcal{L}_i(y_i, h(\theta, x_i, y_i)) + \\ &\sum_{h(\theta,x_j,y_j)=1 \text{ and } y_j=0} \mathcal{L}_j(y_j, h(\theta, x_j, y_j)).\end{aligned} \tag{3.3}$$

In imbalanced data sets, however, the loss function becomes biased toward the majority class due to the dominance of the majority class loss. Therefore, we introduce the condition as:

$$\sum_{h(\theta,X_i,y_i)=0 \text{ and } Y_i=1} \mathcal{L}_j(y_i, h(\theta, X_i, y_i)) \;<\; \sum_{h(\theta,X_j,y_j)=1 \text{ and } y_j=0} \mathcal{L}_j(y_j, h(\theta, X_j, y_j)). \tag{3.4}$$

As a result, the algorithms with the objective of total loss minimization generally cannot successfully solve the classification problem for the minority class. If there is a direct feature specific to a minority class, the classification algorithm can separate the instances of the minority class based on this feature and the skewed distribution of the class sizes cease to be a problem for the loss function. The lack of information and insufficient past data records usually prevent the determination of the minority class with just features. This study focuses on the binary classification problem for imbalanced data sets which can not be solved with only the selected features and requires a deep analysis of the classification algorithms.

The motivation behind the generation of the instance-specific cost parameter, $C_i \; \forall \; i \; \epsilon \; N$, lies behind balancing the loss function in terms of class basis given as:

$$\mathcal{L}(Y, h(\theta, X, Y)) = \sum_{h(\theta, X_i, y_i)=0 \text{ and } y_i=1} C_i \mathcal{L}_i(y_i, h(\theta, X_i, y_i)) + \sum_{h(X_j, y_j)=1 \text{ and } y_j=0} C_j \mathcal{L}_j(y_j, h(\theta, X_j, y_j)). \quad (3.5)$$

Here $\forall \; C_i$ for $\forall \; i \in N$ is expected to ensure a well-recognized minority class as well as the majority class in classification tasks.

3.1 Use of Default Probabilities as Instance-Specific Cost/Risk Parameters

A well-estimated instance-specific cost/risk parameter aims to reflect the real default possibility of the borrower to the loss function of the classification algorithm. If the default probability of the borrower is considerably high, then its contribution to the loss function should be relatively high and the optimization process of loss minimization should end with the decision to reject this customer.

Let $P(D_i)$ define the probability of default for borrower i where $i \in N$ and then, the expected default probability, $\mathbb{E}[P(D_i)] = h(X_{i1}, ..., X_{iF}, Y)$, is estimated with the features reported to the lender and it gives a probabilistic estimate of whether the loan will be paid off. The use of the default probability as an instance-specific default risk level indicator appears as the first option in the search for an instance-specific metric that is rational for balancing the classes for two reasons. Firstly, it is a direct measure signaling the relative risk embedded in the applicant attributes, and its estimations are instance-specific. Secondly, the default classes are generally the minority classes in imbalanced credit data sets, and the default probabilities are expected to be higher for the default classes when compared with the ones of non-default class instances. For this reason, we define the expected values of default probabilities as $\mathbb{E}[P(D)]$ as follows:

$$\mathbb{E}[P(D)_{Y=1}] > \mathbb{E}[P(D)_{Y=0}] \quad (3.6)$$

$$\text{which is equivalent to } h(X \mid Y = 1) > h(X \mid Y = 0) \quad (3.7)$$

The boosting of the minority class instances is crucial in the sense that the loss function should treat both classes equally and produce balanced classification results. Therefore, the use of the expected default probabilities stands as serving the purpose of the cost/risk parameter for boosting the weights of the minority class and reducing the weight of the majority class to balance the loss function.

The existence of a severe class imbalance problem also arises as a problem in ordinary classification algorithms in which the default probabilities of minority classes can not be estimated on a well-distributed higher default probability range. Therefore, default probability estimations are conducted with small subsets generated with minority class and sampled majority class. In this step, the determination of class ratio in subsets appears as an important decision rule since the subsets generated should reflect as much information for both classes and not cause to miss significant information loss. Moreover, the selected sampling size of the non-default class also should lead the algorithm to recognize the minority class instances and assign them higher probabilities which are to be used for boosting purposes of this class.

The determination of the class ratio of the sub-datasets, the sampling method, and the default probability estimation techniques are the first steps of the generation of instance-specific cost/risk parameters. Rejection sampling based on the credit amounts of the applicants is preferred as the sampling method. Sub-datasets with different class ratios are generated, they are modeled with selected algorithms, and default probability estimation results are compared to determine an optimal class ratio. Logistic regressions (LRs) and Simulated Annealing stochastic process is used for the analysis of the optimal class ratio of sub-datasets.

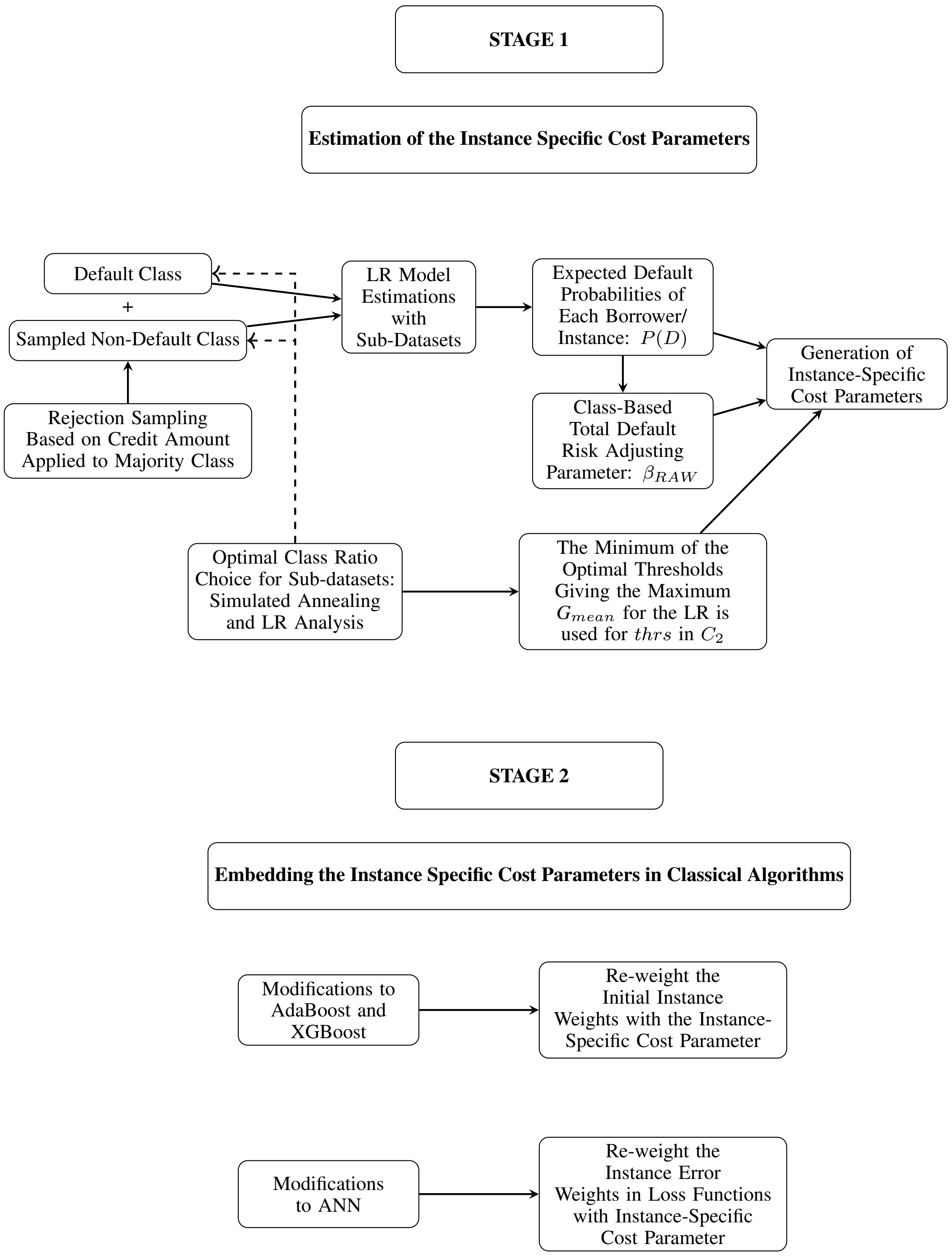
LRs, Graph Neural Networks (GNNs), and Graph Attention Networks (GATs) are used for the default probability estimations. There are several reasons for the selection of these estimation methods. Firstly, the coefficients of the data inputs in LRs can be easily evaluated for the feature selections. The statistically insignificant features in LR models are excluded to facilitate the computation task for all algorithms and increase the explanatory power of the models. The estimations of the models using the train data and making predictions for the test data instances with the LRs are very fast. The over-fitting problem in LRs is not critically observed as it is observed in ANN and other boosting algorithms. After estimating cost parameters with LR models, feed-forward ANNs are also used for the default probability estimations in sub-datasets. However, the estimations of ANN are not fast as LRs and their prediction results are observed not to surpass the results of LRs. On the other hand, the reason for the use of GNNs and GATs is that they have different approaches to each data instance by concatenating each instance's information with their close neighbor information to strengthen the explanatory power of the models. The motivation behind the use of these clustering-based deep learning of these models is that they are relatively more complex and are expected to catch a higher level of information for a borrower, and therefore, they can better reflect the real default probabilities of the borrowers. We have witnessed that clustering-based deep learning techniques have improved the reflection of the real default probabilities in some data sets when compared with LRs.

After the estimation of default probabilities, the cost parameters specified on an instance basis are constructed with the target of obtaining balanced losses for the algorithms. This study offers a methodology of two stages. Stage 1: The cost parameters are generated. Stage 2: Parameters obtained in Stage 1 are embedded in the ordinary algorithms of ANN, AdaBoost, and XGBoost. Algorithm 1 and Figure 3.1 summarize the proposed methodology of the cost parameter generation process with the use of LR. Furthermore, keeping the optimal class ratio analysis the same, the stages of the proposed methodology using GNNs and GATs for the default probability estimations are described in Algorithm 2 and Figure 3.2.

Algorithm 1 Instance-Specific Cost Sensitive Algorithms

The Cost Parameters are Estimated with LRs

```
Require: The predictors are $N$ borrowers with $F$ features, $X^{N\times F}$, with the binary classes,
         $Y^{N\times 1}$, represented with $Y=0$ and $Y=1$ for the non-defaults and defaults, respectively.
 1: Initialize the First Step: Generation of instance-specific cost parameters with sub-datasets.
 2: Determine the optimal class ratio, cr, for sub-datasets: cr= $\frac{\text{Non-defaults}}{\text{Defaults}}$ .
 3: for $cr = 0.5$ to $4$ do
 4:     for $t = 1$ to $I$ do
            Sample $cr\times$ Number of Defaults borrowers from non-defaults by rejection sampling.
            Generate sub-datasets, $(X_s, Y_s)$, with the sampled non-defaults and all default instances.
            Estimate LR models: $h_{LR}(\Theta, X_s, Y_s) = \frac{1}{1+e^{-(\Theta_0+\sum_i^N \Theta_i X_s)}}$
 5:     end for
 6: end for
 7: Analyze the optimal threshold, $thrS$, giving the maximum G$_{mean}$ with G$_{mean}$ and R$^2$ values and
    decide the optimal class ratio.
 8: Estimate LR models with $Nsim$ sub-datasets to get a probability estimation for each borrower.
 9: for $t = 1$ to $Nsim$ do
        Apply rejection sampling to non-defaults keeping the optimal class ratio.
        Generate sub-datasets, $(X_s, Y_s)$, with all defaults and sampled non-defaults.
        Estimate LR models: $h_{LR}(\Theta, X_s, Y_s) = \frac{1}{1+e^{-(\Theta_0+\sum_i^N \Theta_i X_s)}}$.
10: end for
11: Set the expected default probability, $E[Y_i \mid X_i]$, as the maximum default probability estimations,
    $P(D_{LR})_i$, in LR models for each borrower $i$:
        $P(D_{LR})_i = \max\left\{h_{LR_1}(\Theta_1, X_i, Y_i), h_{LR_2}(\Theta_2, X_i, Y_i), ..., h_{LR_{NSim}}(\Theta_{NSim}, X_i, Y_i)\right\}$
12: Compute the total class-based default risk adjusting parameter, $\beta_{RAW}$:
        $\beta_{RAW} = \frac{\sum_i E[P(Y_i \mid X_i)]\mathbb{1}_{Y=0}}{\sum_j E[P(Y_j \mid X_j)]\mathbb{1}_{Y=1}}$ for default class instances, $Y = 1$
        $\beta_{RAW} = 1$ for non-default class instances, $Y = 0$
13: Generate the instance-specific cost parameters, $C_1 = (P(D_{LR}) \times \beta_{RAW})^m$,
        $C_2 = [\exp(\frac{P(D_{LR})\times\beta_{RAW}}{thrs} - 1)]^m$ and $C_3 = \exp(P(D_{LR})) \times (\beta_{RAW})^m$
14: Initialize the second step: Embed the instance-specific cost parameters in AdaBoost, XGBoost,
    and ANN algorithms.
```


STAGE 1
Estimation of the Instance Specific Cost Parameters
Default Class
+
Sampled Non-Default Class
Rejection Sampling Based on Credit Amount Applied to Majority Class
LR Model Estimations with Sub-Datasets
Expected Default Probabilities of Each Borrower/ Instance: $P(D)$
Class-Based Total Default Risk Adjusting Parameter: β_{RAW}
Generation of Instance-Specific Cost Parameters
Optimal Class Ratio Choice for Sub-datasets: Simulated Annealing and LR Analysis
The Minimum of the Optimal Thresholds Giving the Maximum G_{mean} for the LR is used for $thrs$ in C_2
STAGE 2
Embedding the Instance Specific Cost Parameters in Classical Algorithms
Modifications to AdaBoost and XGBoost
Re-weight the Initial Instance Weights with the Instance-Specific Cost Parameter
Modifications to ANN
Re-weight the Instance Error Weights in Loss Functions with Instance-Specific Cost Parameter
Figure 3.1: Instance-Specific Cost Sensitive Algorithms Design with LRs

Algorithm 2 Instance-Specific Cost Sensitive Algorithms

The Cost Parameters are Estimated with GNNs and GATs

```
Require: The predictors are $N$ borrowers with $F$ features, $X^{N\times F}$, with the binary classes,
         $Y^{N\times 1}$, represented with $Y = 0$ and $Y = 1$ for the non-defaults and defaults, respectively.
 1: Initialize the First Step: Generation of instance-specific cost parameters with sub-datasets.
 2: Determine the optimal class ratio, $cr$, for sub-datasets: $cr = \frac{\text{Non-defaults}}{\text{Defaults}}$ .
 3: for $cr = 0.5$ to $4$ do
 4:     for $t = 1$ to $I$ do
            Sample $cr\times$ Number of Defaults borrowers from non-defaults by rejection sampling.
            Generate sub-datasets, $(X_s, Y_s)$, with the sampled non-defaults and all default instances.
            Estimate LR models: $h_{LR}(\Theta, X_s, Y_s) = \frac{1}{1+e^{-(\Theta_0+\Sigma_i^S \Theta_i X_s)}}$
 5:     end for
 6: end for
 7: Analyze the optimal threshold, $thrS$, giving the maximum G$_{mean}$ with G$_{mean}$ and R$^2$ values and
    decide the optimal class ratio.
 8: Estimate GNNs and GATs with $Nsim$ sub-datasets to get a default probability for each borrower.
 9: for $t = 1$ to $Nsim$ do
        Apply rejection sampling to non-defaults and defaults keeping the optimal class ratio.
        Generate sub-datasets, $(X_s, Y_s)$, with sampled defaults and sampled non-defaults.
        Estimate GNN models: $\hat{Y_s} = h_{GNN}(\theta_1, \theta_2, \theta_3, \eta_1, \eta_2, \eta_3, \sigma_1, \sigma_2, \sigma_3, X_s, Y_s)$
        Estimate GAT models: $\hat{Y_s} = h_{GAT}(a, W, \varphi, X_s, Y_s)$
10: end for
11: Set the expected default probability, $E[Y_i \mid X_i]$, as the maximum default probability estimations,
    $P(D_{GNN})_i$ and $P(D_{GAT})_i$, for each borrower $i$ as:
        $P(D_{GNN})_i = \max\left\{h_{GNN_1}(..., X_i, Y_i), ..., h_{GNN_{NSim}}(..., X_i, Y_i)\right\}$
        $P(D_{GAT})_i = \max\left\{h_{GAT_1}(..., X_i, Y_i), ..., h_{GAT_{NSim}}(..., \varphi_{NSim}, X_i, Y_i)\right\}$
12: Compute the total class-based default risk adjusting parameter, $\beta_{RAW}$:
        $\beta_{RAW} = \frac{\sum_i E[P(Y_i \mid X_i)]_{\mathbb{1}_{Y=0}}}{\sum_j E[P(Y_j \mid X_j)]_{\mathbb{1}_{Y=1}}}$ for default class instances, $Y = 1$
        $\beta_{RAW} = 1$ for non-default class instances, $Y = 0$
13: Generate the instance-specific cost parameters, $C_1 = (P(D_{GNN}) \times \beta_{RAW})^m$,
        $C_2 = [\exp(\frac{P(D_{GNN})\times\beta_{RAW}}{thrs} - 1)]^m$ and $C_3 = \exp(P(D_{GNN})) \times (\beta_{RAW})^m$
14: Generate the instance-specific cost parameters, $C_1 = (P(D_{GAT}) \times \beta_{RAW})^m$,
        $C_2 = [\exp(\frac{P(D_{GAT})\times\beta_{RAW}}{thrs} - 1)]^m$ and $C_3 = \exp(P(D_{GAT})) \times (\beta_{RAW})^m$
15: Initialize the Second Step: Embed the instance-specific cost parameters in AdaBoost, XGBoost,
    and ANN algorithms.
```

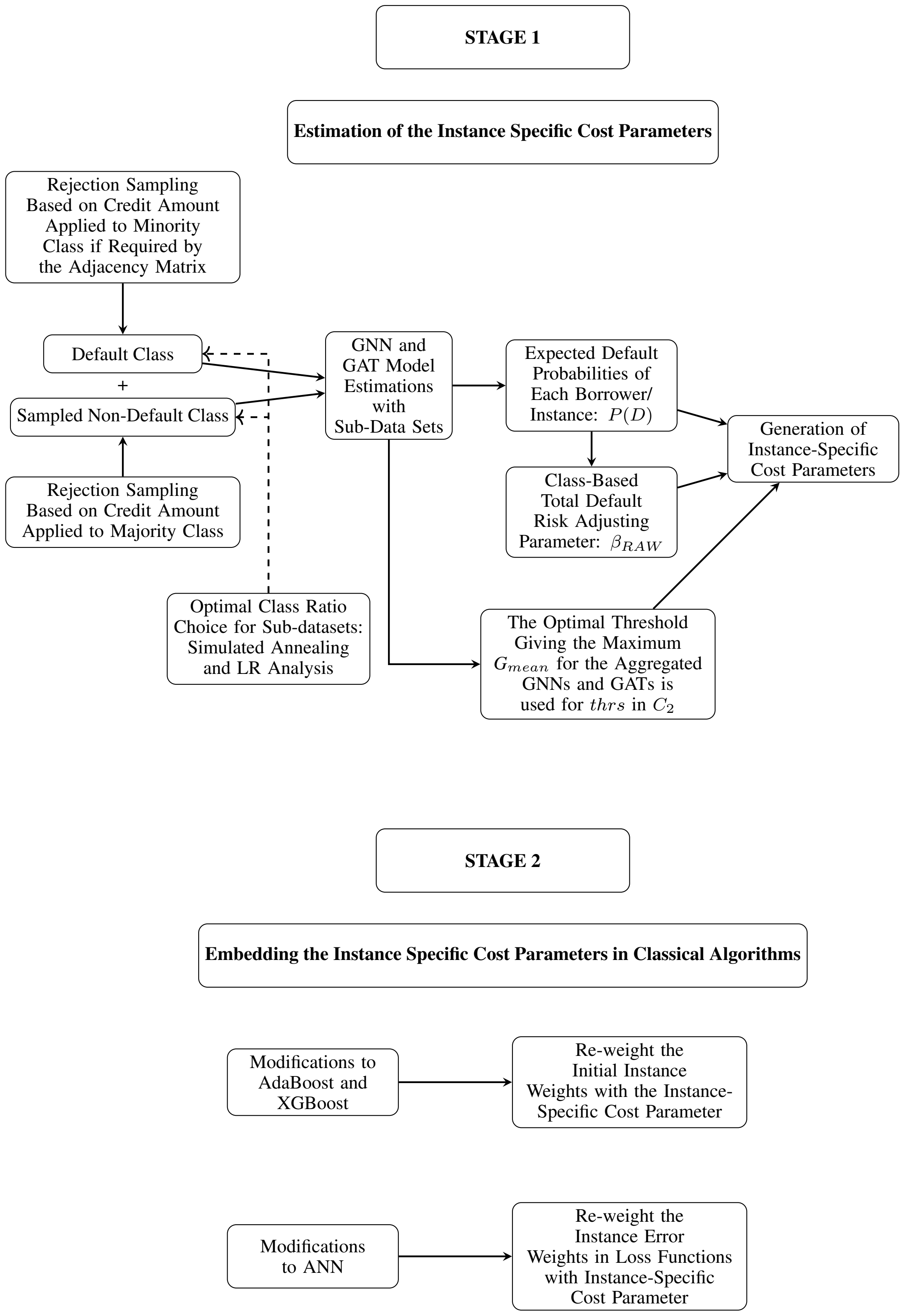

STAGE 1
Estimation of the Instance Specific Cost Parameters
Rejection Sampling Based on Credit Amount Applied to Minority Class if Required by the Adjacency Matrix
Default Class
+
Sampled Non-Default Class
Rejection Sampling Based on Credit Amount Applied to Majority Class
Optimal Class Ratio Choice for Sub-datasets: Simulated Annealing and LR Analysis
GNN and GAT Model Estimations with Sub-Data Sets
Expected Default Probabilities of Each Borrower/ Instance: $P(D)$
Class-Based Total Default Risk Adjusting Parameter: β_{RAW}
Generation of Instance-Specific Cost Parameters
The Optimal Threshold Giving the Maximum G_{mean} for the Aggregated GNNs and GATs is used for $thrs$ in C_2
STAGE 2
Embedding the Instance Specific Cost Parameters in Classical Algorithms
Modifications to AdaBoost and XGBoost
Re-weight the Initial Instance Weights with the Instance-Specific Cost Parameter
Modifications to ANN
Re-weight the Instance Error Weights in Loss Functions with Instance-Specific Cost Parameter

Figure 3.2: Instance-Specific Cost Sensitive Algorithms Design with GNNs and GATs

3.2 Generation of Sub-data Sets for Default Probability Estimations

The sampling method and the optimal class ratio are significant for the well-estimated models with superior predictive parameters. Rejection sampling based on the credit amounts disbursed to the applicants is applied to ensure the instances in the samples reflect the distribution of credit amounts to all applicants.

The class ratio of the sub-sampled data sets appears as an important problem to be solved since the default probabilities will change as the sub-sampled majority class dominates the minority class. Different class ratios of sub-datasets are analyzed with the prediction results of LR models and the optimal threshold analysis conducted with the Simulated Annealing Stochastic (SA) process. The values of the target function in SA can be the best (minimum in SA process) for a wide range of thresholds. Therefore, the reason for the use of a stochastic process for the class ratio analysis is to capture the range of the optimal thresholds minimizing the objective function in SA.

3.2.1 Rejection Sampling

The sub-datasets should reflect the best sample with similar distributions of credit amounts when compared with the complete data set. Therefore, a rejection sampling method based on $X_{CA}=\frac{CreditAmount}{max(CreditAmount)}$, is applied for each data set.

Rejection sampling can be preferred when there is no pre-defined density function for a variable. The target density q of X_{CA}, is evaluated with a candidate density g, supported by X_g, of which density is already built-in and matches closely with q and serves as $X_q \subset X_g$. To ensure that g covers q with heavier tails, the condition given in Equation 3.8 should be satisfied [77]:

$$M = \underset{x \in X_q}{sup} \frac{q(x)}{g(x)} < \infty \tag{3.8}$$

The basic application of the rejection sampling can be described with $A \sim U(0,1)$ and $X \sim g(x)$. If $A \leq \frac{q(x)}{Mg(x)}$ holds, X is accepted, otherwise, it is rejected. The probability of X being accepted is $1/M$ and the smaller M, the higher the chance of it being accepted [77].

$$\begin{aligned}P(X_{accepted}) &= P(A \leq \frac{q(x)}{Mg(x)}) \\ &= \int P(A \leq \frac{q(x)}{Mg(x)}|X = x)g(x)dx \\ &= \int \frac{q(x)}{Mg(x)}g(x)dx \\ &= \frac{1}{M}\end{aligned}$$

The samples accepted with rejection sampling also have the same distribution with the target density q, $F(z) = \int_{-\infty}^{z} q(x)dx$ [77].

$$\begin{aligned}P(X \leq z|X_{accepted}) &= \frac{P(X \leq z, X_{accepted})}{P(X_{accepted})} \\ &= \frac{P(X \leq z, X_{accepted})}{1/M} \\ &= M\mathbb{E}_g\left[\mathbb{E}\left[\mathbb{1}\{x \leq z\}\mathbb{1}\left\{A \leq \frac{q(x)}{Mg(x)}\right\}|X = x\right]\right] \\ &= M\mathbb{E}_g\left[\mathbb{1}\{X \leq z\}E\left[\mathbb{1}\left\{A \leq \frac{q(x)}{Mg(x)}\right\}|X = x\right]\right] \\ &= M\mathbb{E}_g\left[\mathbb{1}\{X \leq z\}\frac{q(x)}{Mg(x)}\right] \\ &= \int_{-\infty}^{\infty} \mathbb{1}\{x \leq z\}\frac{q(x)}{g(x)}g(x)dx \\ &= \int_{-\infty}^{z} q(x)dx \\ &= F(z)\end{aligned} \tag{3.9}$$

This shows that such a shift of the candidate density,g, to generate the target density, q, does not influence the distribution.

3.2.2 Stochastic Approach to Optimal Class Ratio: Simulated Annealing

Simulated Annealing (SA) is inspired by the annealing of metallurgy in which the atoms rapidly move around the whole space when the temperature is the highest but as the temperature falls the moves of the atoms decrease and they take place in a more stable place [65], [25]. SA is a meta-heuristic algorithm with a stochastic global search design and it is easily used for the optimization of non-linear objective functions. SA changes the initial assigned solution with its neighbor points and if the new point is lower than the previous, SA accepts this new point. If the neighbor point is worse than the previous result, the algorithm may still accept the new one with some probability, and its estimation is known as a cooling strategy calculated with an objective cost function. In the initial steps when the temperature is the highest, the algorithm moves on the points to a large extent, but as the temperature decreases the search scale narrows and the algorithms stop at the global minimum.

SA is used for the determination of optimal the class ratio for the sub-datasets. These sub-datasets, (X_s, Y_s) where s is the size of the sub-data ($s < N$), are generated using a targeted algorithm based on the default and non-default cases. Sub-samples with different class ratios are formed and all of them are estimated with the LR models. Since the attributes of data are not completely independent, the conditional default probabilities can not be estimated with the Bayesian risk classifier. Therefore, well-known LR, $h_{LR}(\Theta, X_s, Y_s)$, is used to estimate the default probabilities as:

$$h_{LR}(\Theta, X_s, Y_s) = \frac{1}{1 + e^{-(\Theta_0 + \sum_i^N \Theta_i X_s)}} \tag{3.10}$$

$$\log \frac{p_i}{1 - p_i} = \theta_0 + \sum_j^F \theta_j X_{ij} \text{ and} \tag{3.11}$$

$$p_i = \mathbb{E}[Y_i \mid X_i] = \frac{1}{1 + e^{-(\Theta_0 + \sum_i^N \Theta_i X_i)}} \tag{3.12}$$

where p_i is the expected default probability of borrower i and $\frac{p_i}{1-p_i}$ denotes the odds of the probability of default against the probability of non-default for the borrower i.

The features with statistically significant coefficients are included in all models and the models with higher R^2 are selected. After the estimation of an LR model, negative of G$_{mean}$ (-G_{mean}) minimization is set as an objective function in SA and the optimal thresholds maximizing G$_{mean}$ are analyzed together with the estimated R^2s.

The elements and the stages of this optimization process are described as follows: For every threshold, $thr \in \Omega$, $\Omega \in (0, 1)$ is the solution space and $G(thr) = -G_{mean}(thr)$, $\mid G(thr) \mid < \infty$ and $G(thr) : \Omega \to \mathbb{R}$ is the objective function. The aim is to find the global minimum $thrS^*$: $G(thr) => G(thrS^*)$ for $\to \forall\ thr \in \Omega$, where $N(thr)$ is the neighborhood function for $thr \in \Omega$.

The simulated annealing optimization process starts with an initial random $thr \in \Omega$. The objective function moves to a neighbor threshold, $thr^{'}$, randomly selected or selected on a certain pre-defined rule. The neighbor $thr^{'}$ is accepted based on the Metropolis acceptance criterion [73]:

$$P(thrS^{'}_{accepted}) = \begin{cases} e^{\left[-[G(thr^{'})-G(thr)]/T_k\right]} \text{ if } G(thr^{'}) - G(thrS) > 0 \\ 1 \text{ if } G(thr^{'}) - G(thr) \leq 0 \end{cases} \tag{3.13}$$

T_k is defined as the temperature parameter and k represents the iteration, $T_k > 0$ for $\forall\ k$ and $\underset{k\to\infty}{lim} T_k = 0$. $thrS^* \in \Omega$ is the optimal threshold holding the system in state at $T = T^*$ with the probability following Boltzmann distribution [87]:

$$P(thr_{Optimal} = thrS^* \mid T = T^*) = \frac{\exp(-G(thrS^*)/T_k)}{\sum\limits_{thr''\in\Omega} \exp(-G(thr'')/T_k)} \tag{3.14}$$

where $thr^{''}$ represents the all thresholds accessible in the solution space Ω.
$g(thr, thrS^*)$ is defined as the probability of $thrS$ as a solution among the neighbors and $\sum\limits_{thrS^{'}\in N(thrS)} g_k(thrS, thrS^*) = 1$ for $\forall\ thrS \in \Omega$, $k = 1, 2, ...$, the stochastic transition probability matrix P_k can be defined as:

$$P_k(thr, thrS^*) = \begin{cases} g_k(thr, thrS^*)e^{(-\triangle_{thr,thrS^*/T_k})}, & thrS^* \in N(thr), thrS^* \neq thr \\ 0, & thrS^* \notin N(thr), thrS^* \neq thr \\ 1 - \sum\limits_{\substack{thr''\in N(thrS) \\ thr''\neq thrS^*}} P_k(thrS, thrS^*), & thrS^* = thr \end{cases}$$

The SA algorithm starts with the initial parameters of $thr_0 = 0.5$, *lower bound*=0.1 and *upper bound*=0.9, and $T_0 = 100$ with annealing schedule $T = T_0 * 0.95^k, k_i = \log \frac{T_0 max_j(s_j)}{T_i s_i}$ where k_i is the annealing parameter for component i and s_i = gradient of objective in direction i times difference of bounds in direction i. *Reanneal Interval* is 50 and the stopping criteria is selected to be *Function Tolerance* annealing process is set to continue as the condition of $(thr_{t+1}) - G(thr_t) \leq 1e - 6$ holds.

Simulation of LR estimations with at least 3,000 sub-data is conducted. The subsets include all minority class instances and majority class instances sampled with the rejection sampling method. Each data instance in the non-default class is almost used in at least one of the model estimations and all instances can appear in more than one model and have more than one default probability.

Sub-samples with different class ratios are selected from 0.5 to 3.75 for the sampled majority/minority ratio. Figures 3.3-3.5 display the changes in the optimal thresholds, $thrS$, maximum G_{mean}, and R^2 values for eight credit data set subject to the experimental analysis of this study. The optimal thresholds in simulations are observed to decrease as the class ratio increases in the samples. The maximum G_{mean} values also decrease considerably in the first three data sets as the class ratio increases. As the non-defaults class size exceeds the default class size more than 1.5 times, the ordinary R^2 and the adjusted R^2 in LR estimations are also observed to decrease significantly.

The classification results of LR estimations with the G_{mean} maximizing threshold are evaluated with the *Sensitivity* and the *Specificity* values for different class ratios. As the majority/minority ratio increases in sample data sets, correctly classified minority class instances decrease with the optimal threshold choice giving maximum G_{mean}. The aim is to detect the minority class and not severely deteriorate recognition of the majority class instances. Therefore, the class ratio is selected as 1 and the number of majority class instances is equally included as the minority class instances in the LR estimations. The maximums of default probabilities in LR model simulations are also assigned to each loan applicant for the instance-specific cost parameter. The minimum of optimal thresholds, $thrs$, is also used in the design of cost parameters.

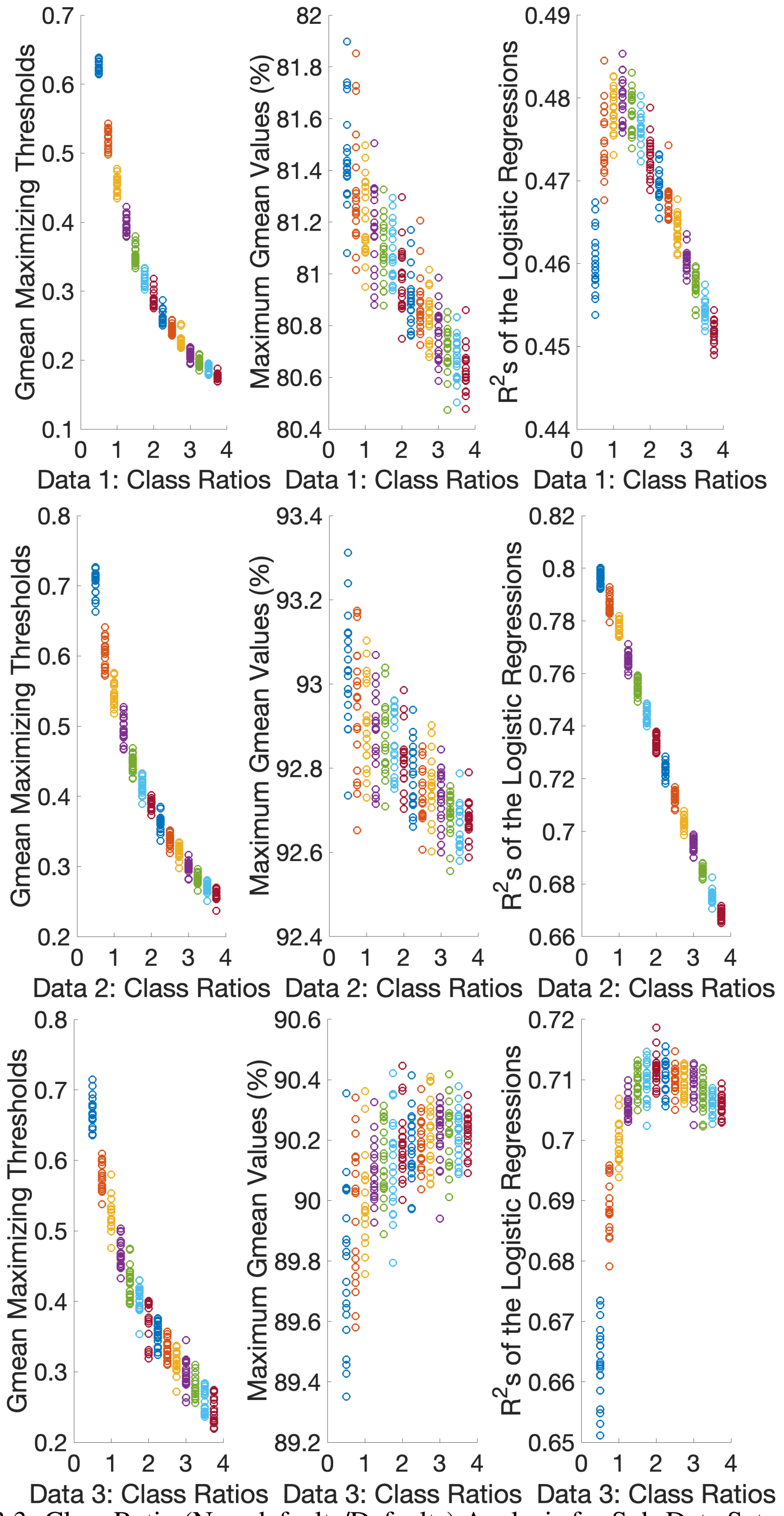


Figure 3.3: Class Ratio (Non-defaults/Defaults) Analysis for Sub-Data Sets with LR and the Optimal Thresholds Estimated by SA

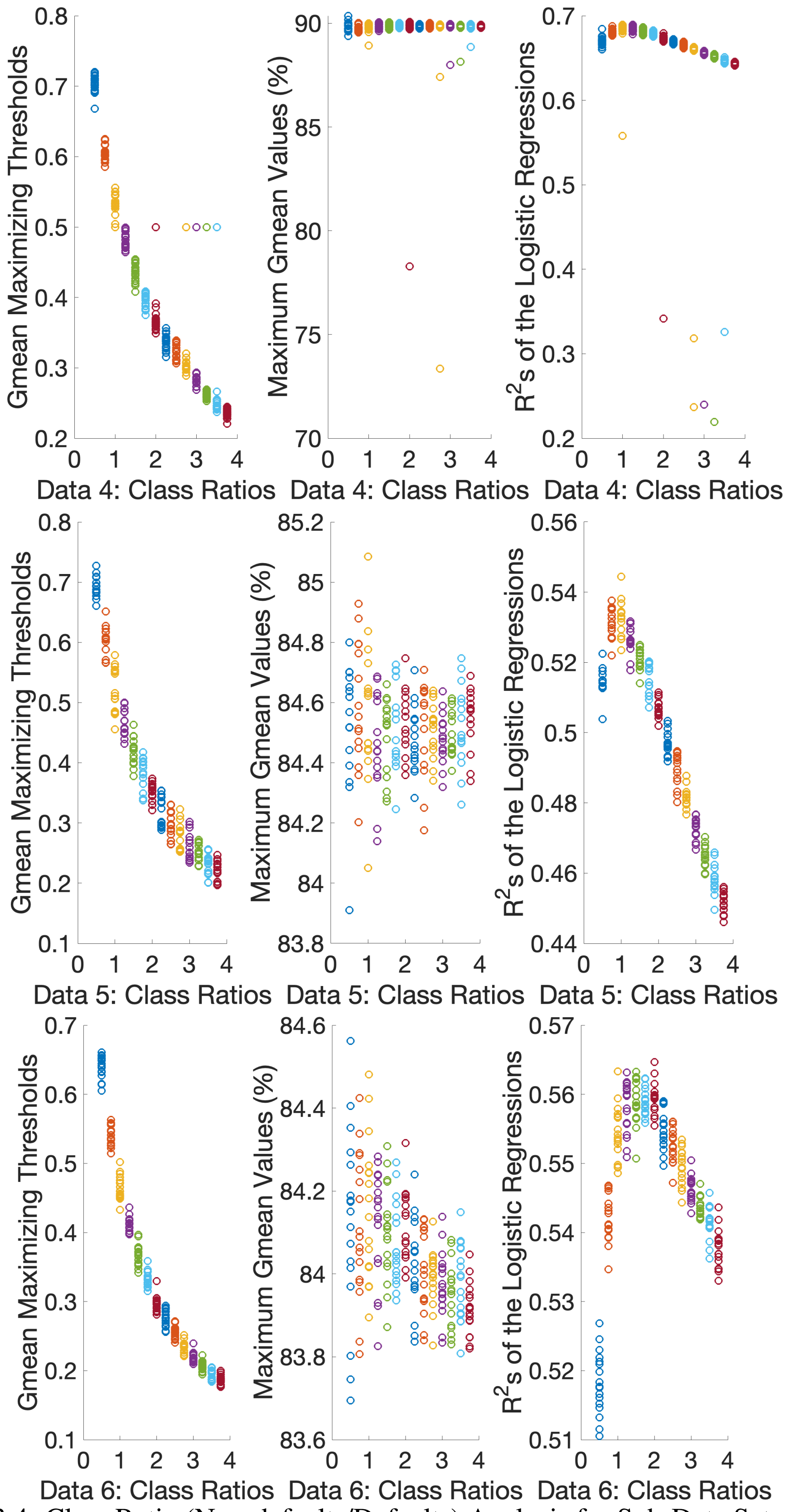


Figure 3.4: Class Ratio (Non-defaults/Defaults) Analysis for Sub-Data Sets with LR and the Optimal Thresholds Estimated by SA

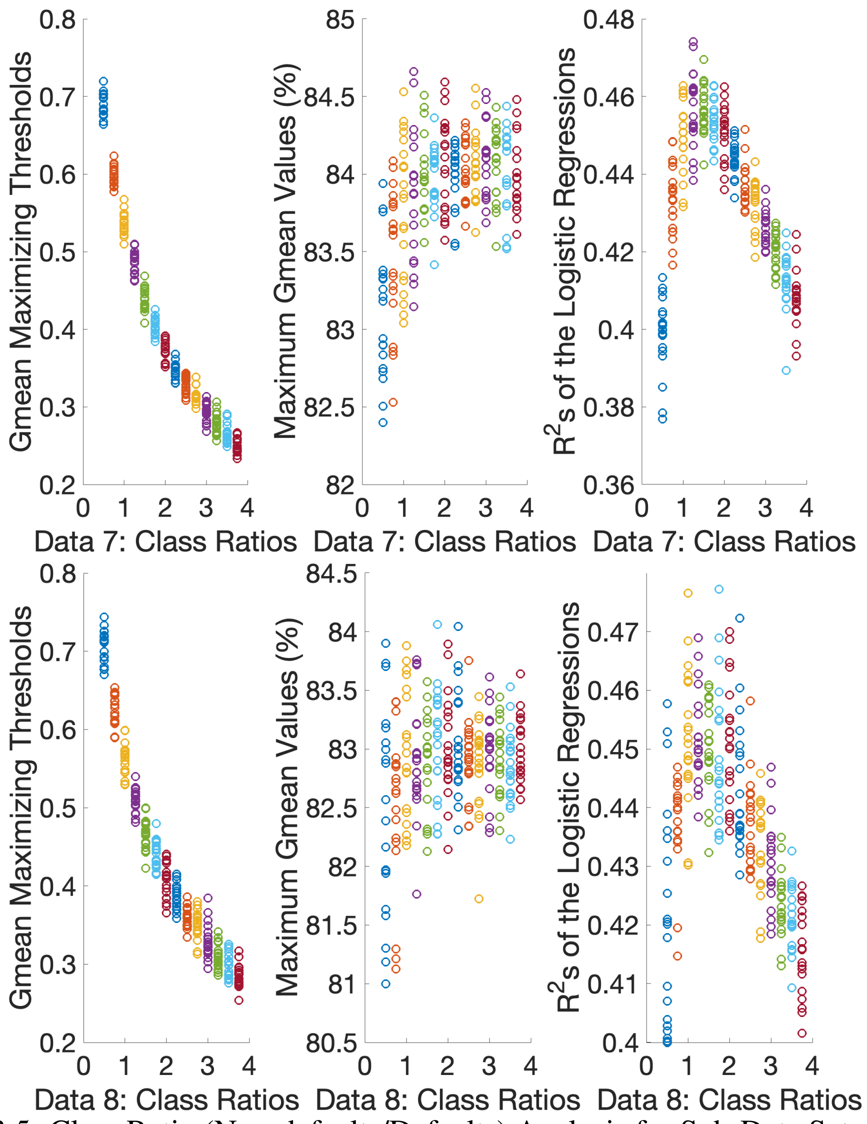


Figure 3.5: Class Ratio (Non-defaults/Defaults) Analysis for Sub-Data Sets with LR and the Optimal Thresholds Estimated by SA

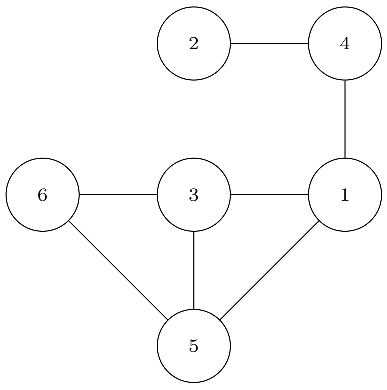
3.2.3 Graph Neural Networks

A graph in Graph Neural Network (GNN) is represented with nodes and neighbors defined on pre-determined distance metrics. The basic idea is that a node collects information from its neighbors, combines it with its own features, and sends the aggregated data to the neural network model. If the class labels are utilized in the training stage, GNNs are modeled in the supervised setting, and if some of the labels are used and most of them not, then the GNNs are evaluated as semi-supervised learning ([27],[104],[30]).

The data in GNN is processed as formatted graph data named ‘a graph embedding’. The graph embeddings are vectors of numerical values representing the relevant information of the neighbor data instances in addition to the nodes’ own features.

The nodes are defined in a finite Vertex Set, V, and their neighbors are defined in the Edge Set, E. A node $\nu \in V$ has an edge with $u \in V$, if $\forall u \in N(\nu)$. A graph G is demonstrated as $G = (V, E)$.

Figure 3.6 illustrates data instances with six nodes and with their connected neighbors on a simple graph. 1^{st} node has edges with all other nodes except node 6 and 2^{nd} only has an edge with node 4. A is the adjacency matrix and shows which node has an edge with which node if node i is connected to node k, A_{ik} is 1, and if it is not connected to the node j, A_{ij} is 0. Moreover, $A_{ii} = 0$ for $\forall\ i \in V$.



$$A = \begin{bmatrix} 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}$$

Figure 3.6: Nodes, Edges and Adjacency Matrix

Let z_ν define the encoder function, $ENC : V \rightarrow \mathbb{R}^d$, which maps the data instances with their aggregated neighbor data information, $\nu \in V$, to vector embeddings z_ν and they represent the embedding of the node ν, $ENC(v) = z_\nu$, [30]. The node embedding of data instance ν at level k is generally formulated as h_ν^k. There is a graph embedding representation of every node as h_ν^k for $\forall\ \nu \in V$, for every node embedding level $k \in R$. The working process of how the GNN works can be described with equations 3.15– 3.17.

$$h_\nu^0 = x_\nu \tag{3.15}$$

$$h_\nu^k = \sigma([A_k.AGG(h_u^{k-1}, \forall u \in N(\nu)), B_k h_\nu^{k-1}]), \forall k \in 1, 2, ..., K \tag{3.16}$$

$$h_\nu^k = \sigma(W_k \sum_{\in N(\nu)} \frac{h_u^{k-1}}{|N(\nu)|}), B_k h_\nu^{k-1}) \text{ if } AGG(h_u^{k-1}) = \sum_{u \in N(\nu)} \frac{h_u^{k-1}}{|N(\nu)|}$$

$$z_\nu = h_\nu^K \tag{3.17}$$

X is the input set, $X \in R^{NxF} \rightarrow R$, N is the number of the inputs and F is the number of features. In the 0^{th} level node embedding, the representation of the node ν is the numerical feature vector of the data instance, $h_\nu^0 = x_\nu$, $x_\nu \in X$ and $\nu \in N$. h^k with $k > 0$ is the representation for inner node embeddings. The superscript of h^k, k, indicates at what level the embedding is. The increase in k means information is transformed to a node ν from the neighbors, $\forall\ u \in N(\nu)$, and their consecutive neighbors, and finally, the message transfer stops after k^{th} level neighbors. Figure 3.7 displays node embeddings of a GNN model with 3 message passing levels with 5 neighbors for each node.

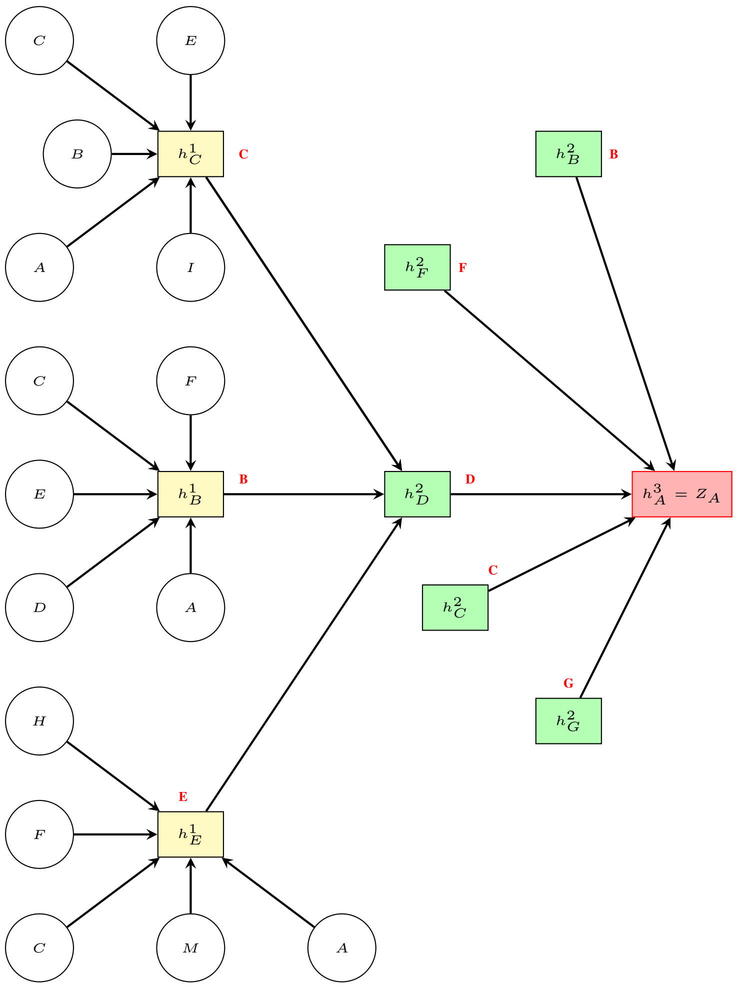


Figure 3.7: Illustration of Node Embeddings in GNN with kNN=5 for node A, z_A

A simple neighbor information aggregation function can be defined as a function averaging the neighbor messages and is given as:

$$AGG(h_u^{k-1}) = \sum_{u \in N(\nu)} \frac{h_u^{k-1}}{|N(\nu)|}. \tag{3.18}$$

We define $\sigma(.)$ as a non-linear transformation function for the aggregated neighbor messages in the previous level, $AGG(h_u^{k-1})$, and the node embedding from the previous layer, h_ν^{k-1}. W_k and B_k are trained weight parameters and shared for all nodes, therefore, they can be used for any new node in test data. The final layer representation for the node embedding is the $z_\nu = h_\nu^K$.

The computation of the aggregation and the transfer of the neighbor messages layer by layer is not an easy task. The use of the adjacency matrix A and the normalization degree matrix D facilitates the information embeddings from the neighbors in GNN operations.

$$AGG(h_u^{k-1}) = \sum_{u \in N(\nu)} \frac{h_u^{k-1}}{|N(\nu)|} \rightarrow H^k = D^{-1}AH^{k-1} where, \tag{3.19}$$

$$H^{k-1} = [h_1^{k-1}, h_2^{k-1}, ..., h_n^{k-1}] \tag{3.20}$$

The study of [93] proposes a direct graph embedding structure with the convolution of the weighted aggregation of the node's own features and its neighbor's features in Graph Convolution Networks (GCN). The adjacency matrix, A, is normalized with the degree matrix D as $\hat{D}^{-1/2}\hat{A}\hat{D}^{-1/2}$ in both of the GNN and GCN. The difference of GCN demonstrates itself in the adjacency matrix, $\tilde{A} = A + I_N$. GCN treats all neighbors and the node itself equally and there is only one W_k parameter matrix for optimization.

$$GNN : h_\nu^k = \sigma([\hat{D}^{-1/2}\hat{A}\hat{D}^{-1/2}XW_k + XB_k])\ \forall k \in 1, 2, ..., K \tag{3.21}$$

$$GCN : h_\nu^k = \sigma([\hat{D}^{-1/2}\tilde{A}\hat{D}^{-1/2}XW_k])\ \forall k \in 1, 2, ..., K \tag{3.22}$$

The decoder function, $DEC : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}^+$, predicts the relationship between the nodes such as z_ν and z_u as follows:

$$DEC(z_\nu, z_u) \approx S[z_u, z_\nu] \tag{3.23}$$

$$\mathcal{L} = \sum_{u,\nu \in X} \ell(DEC(z_u, z_\nu), S[z_u, z_\nu]). \tag{3.24}$$

The aim of the optimization process in GNN is to minimize the loss between the prediction and the real similarity measure of $S[z_u, z_\nu]$ [30].

3.2.3.1 Link Prediction

The adjacency matrix requires the determination of the linkages between the data instances. The neighborhoods of data instances are analyzed with k-means clustering with different distance metrics like Euclidean, Cityblock, Chebychev, Minkowski, Cosine, Hamming, and Jaccard. Figures 3.8-3.10 demonstrate which distance metric brings the same class instances for the data instances. The graphs in the right column display the non-defaults in the 30 nearest neighbors of non-default instances as percentages and the left ones indicate the defaults in the 30 nearest neighbors of default instances as percentages. The reason for the 30 neighbors is that increasing the neighbors beyond 30 generally causes over-fitting problems in the train data in the experimental stage.

In most of the data sets, the cosine distance metric among the seven metrics is observed to capture higher default neighbors in the closest 30 neighbors of the default class borrowers. Cosine distance [67] displayed with green dots in Figures 3.8- 3.10 brings more than $70\ \%$ of the neighbors as the default neighbors for the default class borrowers for most of the data sets. Therefore, the linkages between the data instances are constructed with respect to the cosine distance in all data sets in the experimental stage. It is defined as:

$$\begin{aligned}\text{Cosine Distance} &= 1 - \text{Cosine Similarity} \\ &= 1 - \frac{A.B}{\|A\|\|B\|} = 1 - \frac{\sum_i^n A_i B_i}{\sqrt{\sum_i^n A_i^2}\sqrt{\sum_i^n B_i^2}}\end{aligned} \tag{3.25}$$

The neighbors in GNN are also weighted inversely with their cosine distance to the nodes and the contributions of the data instances become inversely proportional to their distance. In the training stage, GNNs with non-weighted adjacency matrices are observed to over-fit easily and the prediction performances for the test data are observed to be significantly different from the training data results. Weighting the neighbor messages with respect to their distances is observed to improve the GNN estimations of sub-datasets.

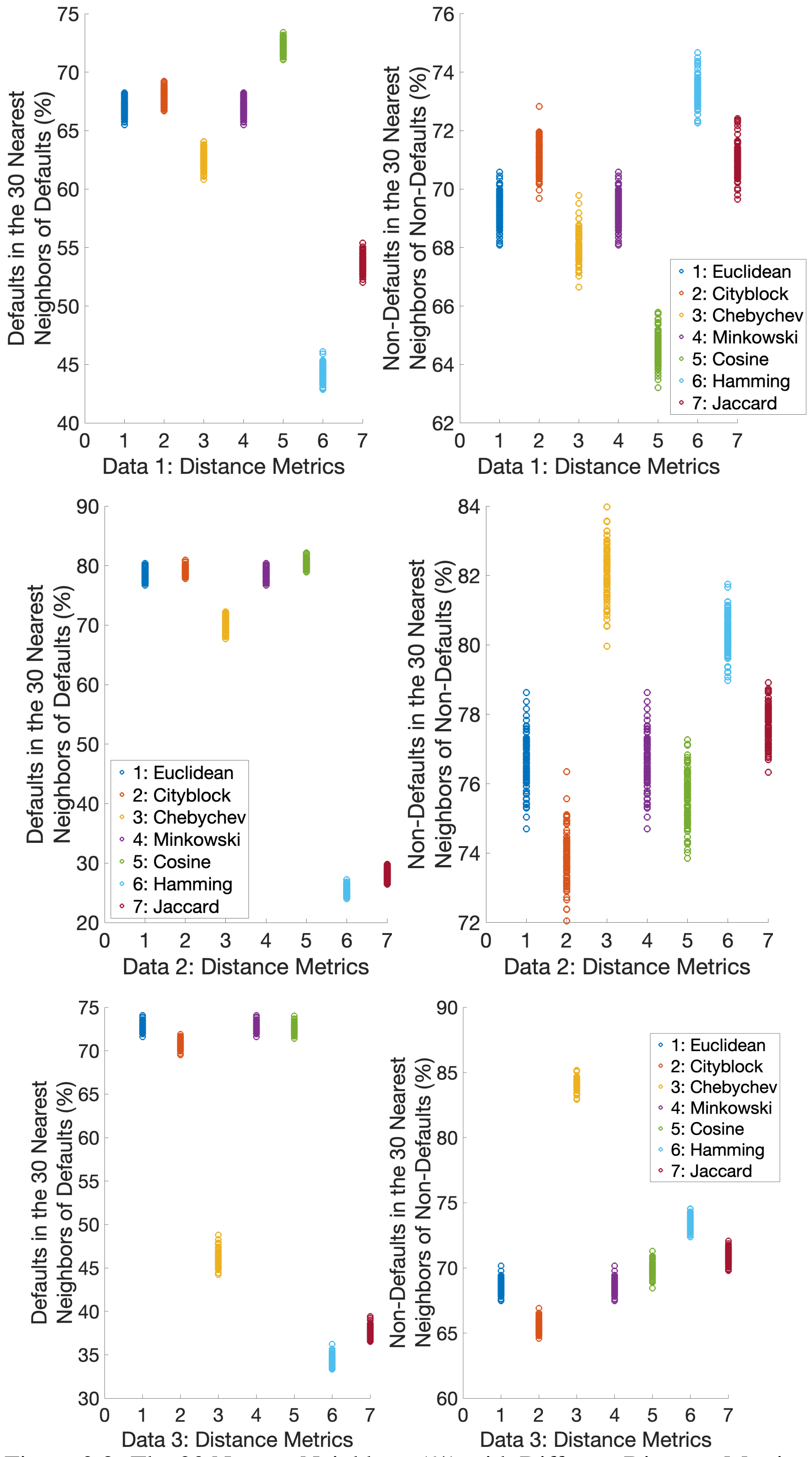


Figure 3.8: The 30 Nearest Neighbors (%) with Different Distance Metrics

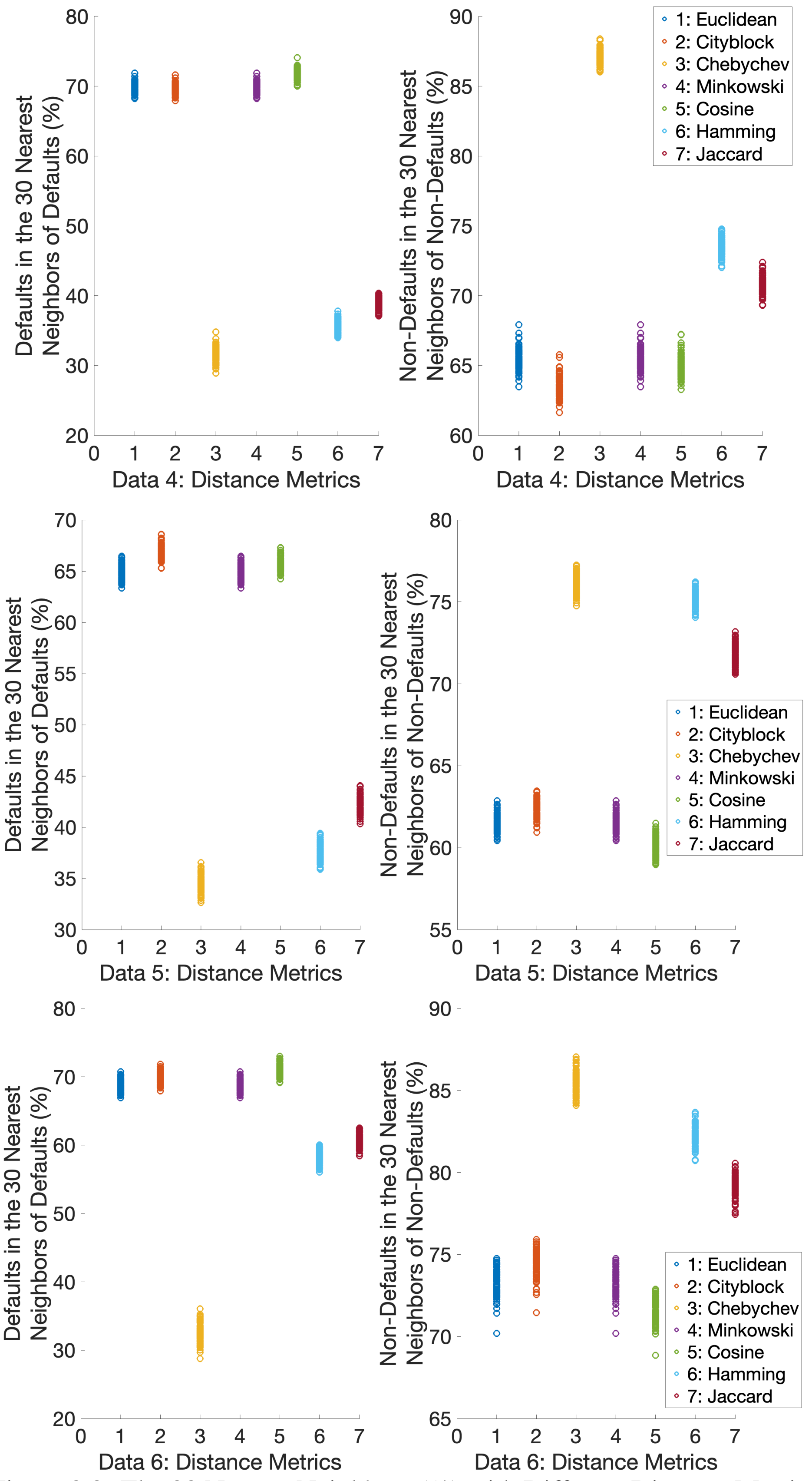


Figure 3.9: The 30 Nearest Neighbors (%) with Different Distance Metrics

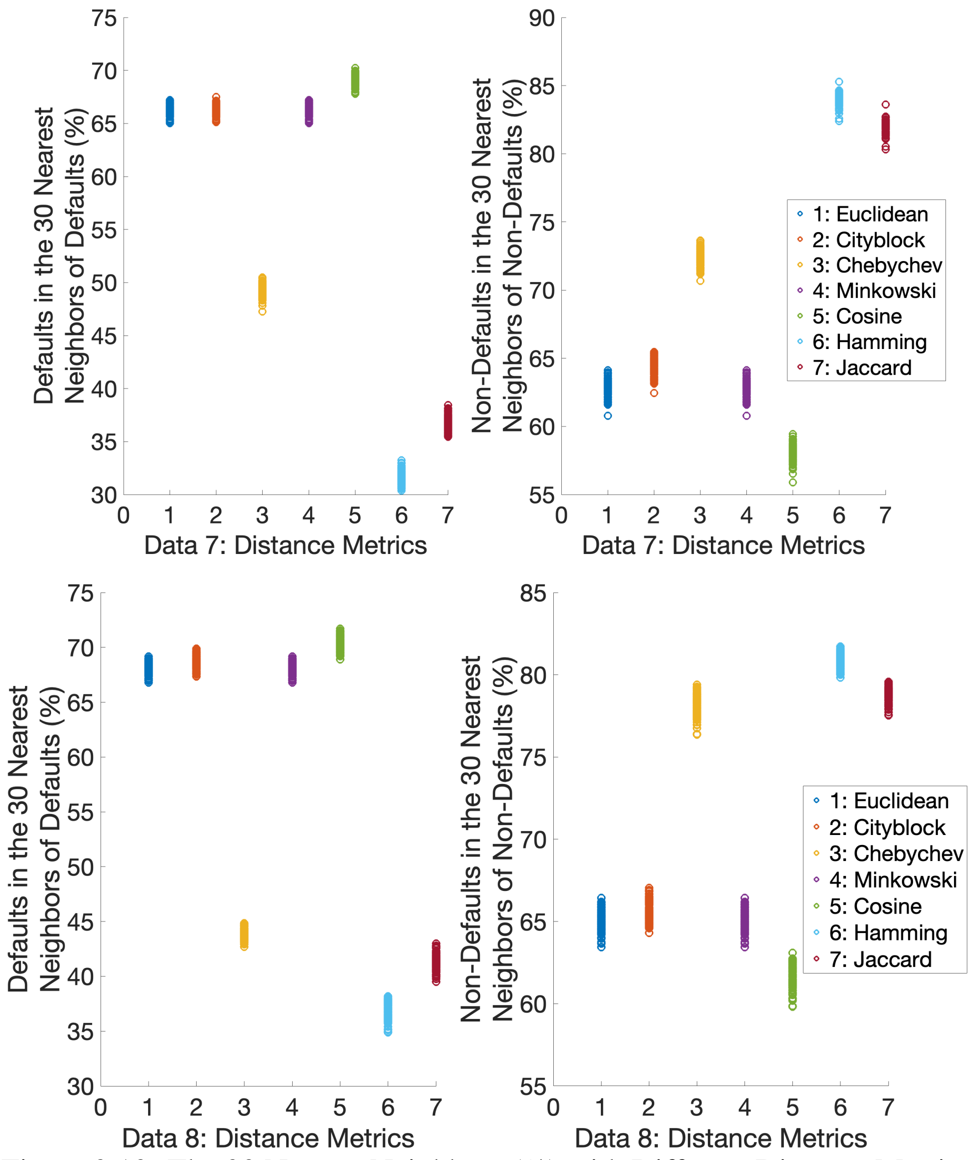


Figure 3.10: The 30 Nearest Neighbors (%) with Different Distance Metrics

3.2.3.2 GNN Model Design for Sub-Data Sets

The data instances in imbalanced data can not be transformed to graph-structured node embeddings, since the neighborhoods of the minority class instances are extensively dominated by the majority class instances, and GNNs can not perform, and neither GATs nor GCNs. Therefore, the dominant class instances should be sparse enough for setting up graph designs enabling information transformation with well-functioning vertex and edge sets for the minority class instances.

Sub-data sets with predictors, X_s^{SxF}, and their corresponding class labels, Y_s^{SxF}, which have the input size S and the feature size F are generated with a class ratio of 1. Given the predictors as x_{ij} where $\forall\ i \in S,\ j \in F,\ X_i$ represents the feature set of the borrower i. The class labels are indicated by $y_i \in Y_s = (0,1)$ for the borrower i. Both classes are sampled for the large data sets to enable Matlab to fulfill the adjacency matrix operations. Rejection sampling based on the credit amount or the random sampling methods is utilized with the consideration of the test data results.

The GNN estimation process starts with the definition of the graph structure defined as $G = (X_s^G, V^G, E^G)$. Let $E(V_i)$ denote the edge set, for $\forall\ i \in S$ and V_i represents the vertex set of the borrower i. $E(V_{ik}) = 1$ if $X_k \in N(X_i)$, otherwise 0, for $\forall\ k \in S$.

V^G is constructed with the Cosine Distance in k-NN means clustering. The optimal numbers for the neighbors are determined in the empirical model estimations considering the model performances in the test data and the over-fitting problem in train data. The empirical test data findings of GNN estimations are observed to improve when the neighbors' information is weighted inversely with the cosine distance and then concatenated with the target data instance. Therefore, the normalized weighted adjacency matrix $\tilde{A_w}$ is used in all data sets and given as:

$$\tilde{A_w} = \frac{A \times W_{Cos}}{\sum\limits_{j \in N(i)} W_{Cos_i}} \tag{3.26}$$

$$\text{where } W_{Cos} = \frac{1}{\text{Cosine Distance}} \tag{3.27}$$

The node embeddings are all indicated with the matrix Z_ℓ, ℓ is the node embedding level (embedding layer), where $Z_1 = X$ and $h_i^0 = X_i$ for $\forall\ i \in X_s$.

$$Z_\ell = \begin{bmatrix} h_1^{\ell-1} & h_\nu^{\ell-1} & \cdots & h_N^{\ell-1} \end{bmatrix}^T \tag{3.28}$$

Initially, GNN models for all data sets in the empirical stage are designed with 3 message-passing levels described:

$$Z_1 = X_s \tag{3.29}$$

$$Z_2 = \tilde{A}_w \times Z_1 \times \theta_1, \quad Z_2 = \sigma_1(Z_2 + Z_1 \times \eta_1), \tag{3.30}$$

$$Z_3 = \tilde{A}_w \times Z_2 \times \theta_2, \quad Z_3 = \sigma_2(Z_3 + Z_2 \times \eta_2), \tag{3.31}$$

$$Z_4 = \tilde{A}_w \times Z_3 \times \theta_3, \quad Z_4 = Z_4 + Z_3 \times \eta_3, \tag{3.32}$$

$$\hat{Y_s} = \sigma_3(Z_4). \tag{3.33}$$

GNN model searches the optimal values for the learning parameters of θ_1, θ_2, θ_3, η_1, η_2 and η_3, and the optimal activation functions, $\sigma_1(.)$, $\sigma_2(.)$, $\sigma_3(.)$, with the target of loss minimization. The sampling size of X_s, the sampling method, the iteration numbers of the model, the learning rate, and the number of estimated GNNs for the simulation are all significant parameters for a good fit of the estimation of the default probabilities of credit applicants.

Let $\hat{Y_s}$ be defined as:

$$\begin{aligned} \hat{Y_s} &= h_{GNN}(\theta_1, \theta_2, \theta_3, \eta_1, \eta_2, \eta_3, \sigma_1, \sigma_2, \sigma_3, X_s, Y_s) \\ &= \sigma_3(\tilde{A}_w \times \sigma_2(\tilde{A}_w \times \sigma_1(\tilde{A}_w \times X \times \theta_1 + X_s \times \eta_1) \times \theta_2 \\ &+ \sigma_1(\tilde{A}_w \times X_s \times \theta_1 + X_s \times \eta_1) \times \eta_2) \times \theta_3 \\ &+ \sigma_2(\tilde{A}_w \times \sigma_1(\tilde{A}_w \times X \times \theta_1 + X_s \times \eta_1) \times \theta_2 \\ &+ \sigma_1(\tilde{A}_w \times X_s \times \theta_1 + X \times \eta_1) \times \eta_2) \times \eta_3). \end{aligned} \tag{3.34}$$

Sigmoid function, $\frac{1}{1+e^{-(\theta_0 + \sum_i^N (\theta_i * X_i)}}$ has proved to be the optimal activation function for all node embeddings for all data sets. The final node embedding/output layer activation function for the binary classification is the "softmax" function and it is calculated as $\sigma_3(Z_{4_{ij}}) = \text{softmax}(Z_{4_{ij}}) = \frac{\exp(Z_{4_{ij}})}{\sum\limits_{k \in Z_{4_i}} \exp(Z_{4_{ik}})}$. The algorithm design of GNN with 3 node embeddings is displayed in Figure 3.11 as an illustration.

GNN model estimation process, h_{GNN} for sub-datasets, (X_s, Y_s), initializes the system with randomly assigned learning parameters, θ_1, θ_2, θ_3, η_1, η_2 and η_3 σ_1, σ_2, σ_3 and the model searches to reach to minimum loss between the target Y_s and the estimated default probability, $h_{GNN}(\theta_1, \theta_2, \theta_3, \eta_1, \eta_2, \eta_3, X_s, Y_s)$ with "Adam Optimizer". The objective loss function for this classification task is the cross entropy loss func-

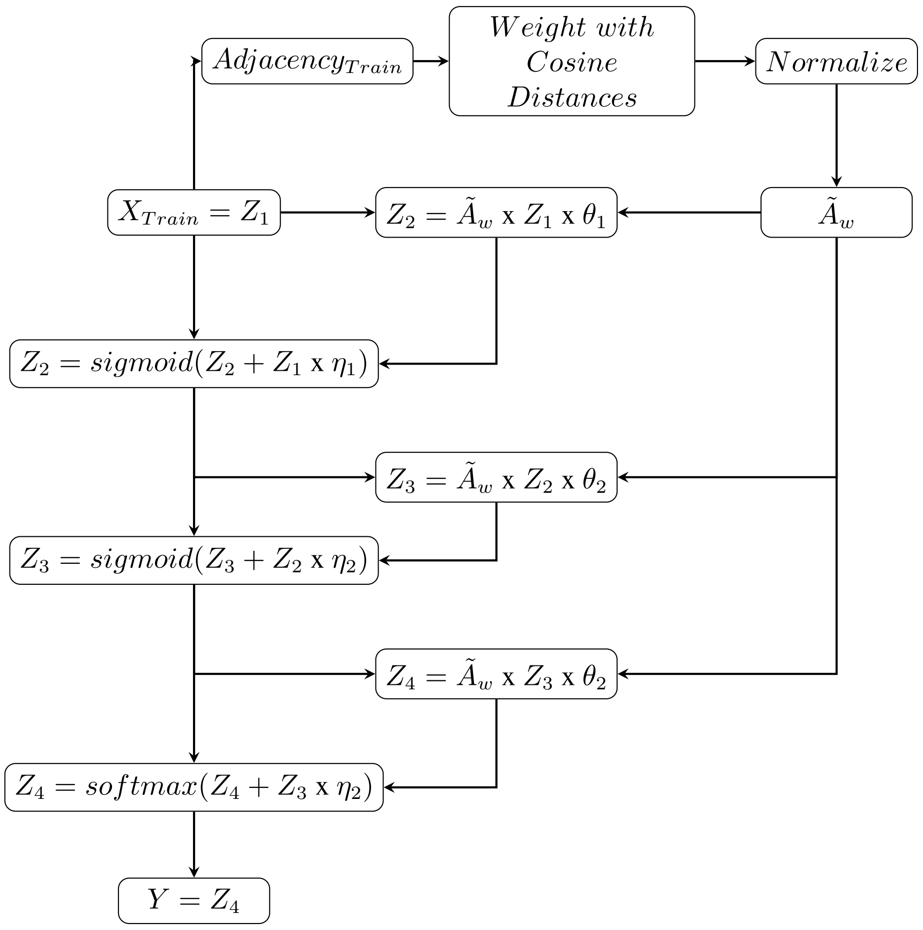

$Adjacency_{Train}$
$Weight\ with\ Cosine\ Distances$
$Normalize$
$X_{Train} = Z_1$
$Z_2 = \tilde{A}_w$ x Z_1 x θ_1
$\tilde{A}_w$
$Z_2 = sigmoid(Z_2 + Z_1$ x $\eta_1)$
$Z_3 = \tilde{A}_w$ x Z_2 x θ_2
$Z_3 = sigmoid(Z_3 + Z_2$ x $\eta_2)$
$Z_4 = \tilde{A}_w$ x Z_3 x θ_2
$Z_4 = softmax(Z_4 + Z_3$ x $\eta_2)$
$Y = Z_4$

Figure 3.11: Design of GNN Models

tion, $\mathcal{L}(Y_s, h_{GNN}(\theta_1, \theta_2, \theta_3, \eta_1, \eta_2, \eta_3, X_s, Y_s))$, given as:

$$\mathcal{L}(Y_s, h_{GNN}(\theta_1, \theta_2, \theta_3, \eta_1, \eta_2, \eta_3 X_s, Y_s)) \\ = -\frac{1}{S}\sum_{n=1}^{S}\left[Y_n \log \hat{Y}_n + (1 - Y_n)\log(1 - \hat{Y}_n)\right] \tag{3.35}$$

"Adam Optimizer", a stochastic adaptive moment optimization process, utilizes the first and second-moment estimations of the gradient, m_t and v_t, respectively. The gradient vector of the loss function is computed at the initial iteration $t = 0$ and the learning parameters are updated with moment estimates and the initially set learning rate δ in each subsequent iteration. The biased first and second moment estimates, m_t and v_t, are converted to bias-corrected estimates of $\hat{m_t}$ and $\hat{v_t}$ with the parameters of β_1 and β_2. $\beta_1 = 0.9$ and $\beta_2 = 0.999$ are set in all optimization processes. The update continues until the parameters converge.

$$\nabla\mathcal{L} = \nabla\mathcal{L}(y, h_{GNN}(\theta_1,\theta_2,\theta_3,\eta_1,\eta_2,\eta_3,X_s,y))$$

$$= \begin{bmatrix} \frac{\partial\mathcal{L}}{\partial\theta_1}(y, h_{GNN}(\theta_1,\theta_2,\theta_3,\eta_1,\eta_2,\eta_3,X_s,y)) \\ \frac{\partial\mathcal{L}}{\partial\theta_2}(y, h_{GNN}(\theta_1,\theta_2,\theta_3,\eta_1,\eta_2,\eta_3X_s,y)) \\ \frac{\partial\mathcal{L}}{\partial\theta_3}(y, h_{GNN}(\theta_1,\theta_2,\theta_3,\eta_1,\eta_2,\eta_3,X_s,y)) \\ \frac{\partial\mathcal{L}}{\partial\eta_1}(y, h_{GNN}(\theta_1,\theta_2,\theta_3,\eta_1,\eta_2,\eta_3,X_s,y)) \\ \frac{\partial\mathcal{L}}{\partial\eta_2}(y, h_{GNN}(\theta_1,\theta_2,\theta_3,\eta_1,\eta_2,\eta_3,X_s,y)) \\ \frac{\partial\mathcal{L}}{\partial\eta_3}(y, h_{GNN}(\theta_1,\theta_2,\theta_3,\eta_1,\eta_2,\eta_3,X_s,y)) \end{bmatrix} \tag{3.36}$$

$$m_t = \beta_1 \times m_{(t-1)} + (1-\beta_1)\times\nabla\mathcal{L} \tag{3.37}$$

$$v_t = \beta_2 \times v_{t-1} + +(1-\beta_2)\times\nabla\mathcal{L}_t^2 \tag{3.38}$$

$$\hat{m_t} = m_t/(1-\beta_1^t) \tag{3.39}$$

$$\hat{v_t} = v_t/1-\beta_2^t \tag{3.40}$$

$$\begin{bmatrix} \theta_{1_t} \\ \theta_{2_t} \\ \theta_{3_t} \\ \eta_{1_t} \\ \eta_{2_t} \\ \eta_{3_t} \end{bmatrix} = \begin{bmatrix} \theta_{1_{t-1}} \\ \theta_{2_{t-1}} \\ \theta_{3_{t-1}} \\ \eta_{1_{t-1}} \\ \eta_{2_{t-1}} \\ \eta_{3_{t-1}} \end{bmatrix} - \delta \times \frac{\hat{m_t}}{\sqrt{\hat{v_t}+\epsilon}} \tag{3.41}$$

The node embeddings in GNNs are reduced if the over-fitting problem with the considerable divergence between train and test data predictions is observed in data sets. The increase in the node embeddings, iterations, learning rates, and assigning superfluous learning parameters can cause the train data to learn and memorize itself in detail, but the parameters of this kind of model can not identify the new data instances in test data. Therefore, it is vital to avoid over-fitting when focusing on loss minimization, and the gap between the performance metrics of *Sensitivity*$_{Train}$-*Specificity*$_{Train}$ and *Sensitivity*$_{Test}$-*Specificity*$_{Test}$ should be minimum. This rule actually increases the instance-specific cost parameters in the final classification algorithms.

It is necessary to underline the fact that all test data used in this study also have unbalanced classes. Generating graph structures within test data instances does not support the basic designs of GNNs, the dominant class will be the neighbors mostly surrounding the data instances. This problem can be tackled if each test data instance is evaluated to appear one by one not as a group. In this context, the test data graph structure can be established with the search of neighbors in the existing train data instances. The neighbors of one test data instance are searched through the instances of sub-datasets of the train data. Generation of node embeddings of test data instances within the equally distributed train data sets solves the difficulty and impossibility of prediction of test data through graph structures.

The medians of the predictions of all data instances in 1,000 GNN model simulations are also used for classification. The aggregated GNN prediction for the borrower i, $\mathbb{E}(Y_i \mid X_i)_{Agg.\ GNNs}$, is computed as:

$$\begin{aligned}\mathbb{E}(Y_i \mid X_i)_{Agg.\ GNNs} =& \text{median}\Big\{ h_{GNN_1}(....,X_i,Y_i), h_{GNN_2}(....,X_i,Y_i),\\ & ...., h_{GNN_{1000}}(....,X_i,Y_i)\Big\},\end{aligned} \tag{3.42}$$

The threshold for the classification is determined as the G_{mean} maximizing threshold in train data and it is also applied to the medians of the predictions of the test data.

3.2.4 Graph Attention Networks

Graph Attention Networks (GATs) are developed with the consideration of the features of some neighbors can be more explanatory for the node. Attention coefficients are incorporated in GNN models to uncover the explanatory power of the neighbor not by its whole feature set but by a specific feature. Therefore, $e_{\nu u}$ for $\forall \nu$ and $\forall u$ $\in V$, are computed with the attention mechanism $a : R^{FxF} \to R$ and linear weight matrix, $W \in R^{FxF}$ as:

$$e_{\nu u} = a(W_k h_u^{k-1}, W_k h_\nu^{k-1}) \tag{3.43}$$

$$\alpha_{\nu u} = \text{softmax}_\nu(e_{\nu u}) = \frac{\exp(e_{\nu v})}{\sum_{k \in N(\nu)} \exp(e_{\nu k})} \tag{3.44}$$

$$\alpha_{\nu u} = \frac{\exp(\text{LeakyRELU}(\vec{a}[W\vec{h_\nu} W\vec{h_u}]))}{\sum_{k \in N_\nu} \exp(\text{LeakyRELU}(\vec{a}[W\vec{h_\nu} W\vec{h_u}]))} \tag{3.45}$$

$\alpha_{\nu u}$ are the normalized attention weights and they are the byproducts of an attention mechanism a, and they linearly combine the features of two inputs [88]. The adjacency matrix in GAT is utilized in the normalization process of the attention weights, and the softmax normalization with the data instances only in the neighborhood is applied. The node embedding of ν at the embedding level ℓ in the GAT model is:

$$h_\nu^\ell = \sigma\left(\sum_{\forall u \in N(\nu)} \alpha_{\nu u} W h_u^{\ell-1}\right) \tag{3.46}$$

GATs can be extended to multi-head attention with K-independent attention learning weight matrices to make the learning process stable as shown:

$$h_\nu^\ell = \overset{K}{\underset{k=1}{\|}} \sigma\left(\sum_{\forall u \in N(\nu)} \alpha_{\nu u}^\ell W^k h_u^{\ell-1}\right) \tag{3.47}$$

The estimation outputs of the multi-head mechanism are finally concatenated or simply averaged. Figure 3.13 displays multi-head node embedding with K=3 for level $k+1$ in a GAT model. $\overset{K}{\underset{k=1}{\|}}$ sign indicates the concatenation of the multi-head attention, it can be a pre-defined function or simply an average function.

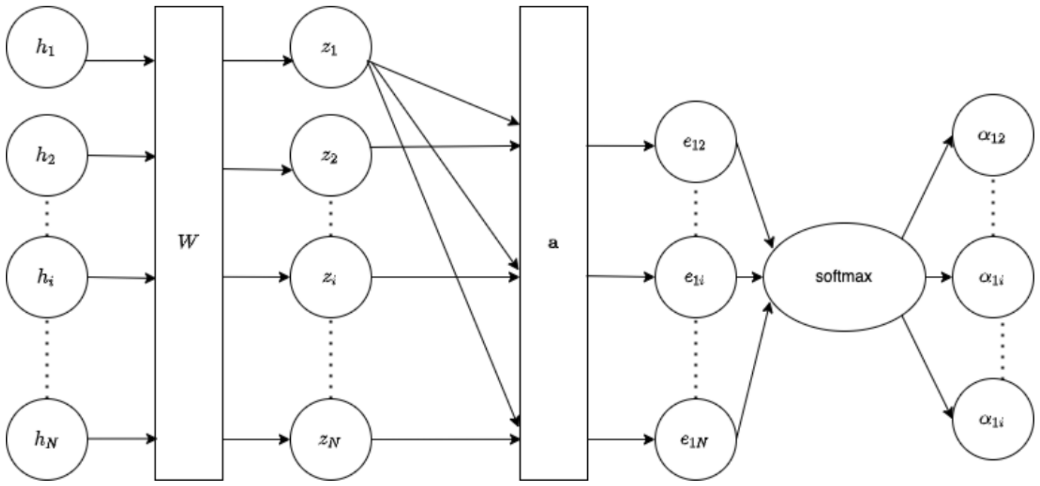


Figure 3.12: Attention Weights in GATs

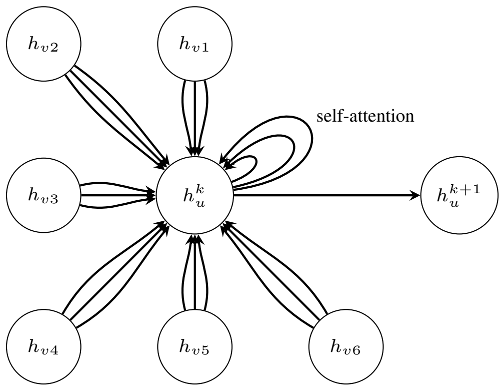


Figure 3.13: Multi-head Attention GAT

3.2.4.1 GAT Model Design for Sub-Data Sets

The predictors and their corresponding classes, (X_s, Y_s), in sub-datasets are defined as in GNN models. The parameters in GAT models can be listed as the size of the sub-datasets, the method of sampling, the neighbors generating the adjacency matrix, A, for the graph structure $G = (X^G, V^G, E^G)$ and well-performing model design with the activation functions and their learning parameters, node embedding levels, learning rate, δ, and the optimal iteration numbers. Different from the GNN model design, attention coefficients and multi-head attention interactions with mean function are included in GATs. The node embeddings are all indicated with the matrix Z_ℓ, ℓ is the node embedding level (embedding layer), where $Z_1 = X$ and $h_i^0 = X_i$ for $\forall\ i \in X$, expressed as:

$$Z_\ell = \begin{bmatrix} h_1^{\ell-1} & h_\nu^{\ell-1} & \cdots & h_N^{\ell-1} \end{bmatrix}^T . \tag{3.48}$$

The initial GAT estimations are conducted with three levels of node embeddings with the given model design in Equations 3.49-3.53. The attention coefficients and the linear weights in the first, second, and third level of node embeddings are $\alpha_{1_{NxN}}$, $W_{1_{FxF}}$, $\varphi_{1_{FxF}}$, $\alpha_{2_{NxN}}$, $W_{2_{FxF}}$, $\varphi_{2_{FxF}}$ and $\alpha_{3_{NxN}}$, $W_{1_{Fx2}}$, $\varphi_{3_{Fx2}}$, respectively.

$$Z_1 = X_s, \tag{3.49}$$

$$Z_2 = \sigma_{11}(\alpha_1 \times Z_1 \times W_1) + Z_1 \times \varphi_1 \ , Z_2 = \sigma_{12}(Z_2), \tag{3.50}$$

$$Z_3 = \sigma_{21}(\alpha_2 \times Z_2 \times W_2) + Z_2 \times \varphi_2 \ , Z_3 = \sigma_{22}(Z_3), \tag{3.51}$$

$$Z_4 = \sigma_{31}(\alpha_2 \times Z_3 \times W_3) + Z_3 \times \varphi_3, \tag{3.52}$$

$$\hat{Y}_s = \text{softmax}(Z_4). \tag{3.53}$$

GAT models with 3 layers are observed to learn the train data successfully due to learning parameters assigned to the feature-matching mechanism of each data instance. However, the optimal learning parameters of the train data models can not detect the test data instances, the performance metrics are observed to be very low for the test data predictions. The over-fitting problem of GAT models is directly solved by decreasing the node embedding levels to one and changing the sampling method from rejection sampling to random sampling for some data sets, and the changing sub-data sizes also contributed to test data prediction performances. All GAT models are designed with one node embedding levels but the other parameters such as the number of the neighbors, activation functions before the output layer and etc are determined as specific to each data set.

The set-up of the attention mechanism starts with the computation of non-normalized attention coefficients, e_{ij} for $\forall\ i \in X_s$ and for $\forall\ j \in N(i)$:

$$e_{ij} = (X_i W_{1_{Fx2}}, X_j W_{1_{Fx2}})\vec{a}_{2x2} \tag{3.54}$$

$$e_{\vec{NxN}} = [(X_{\vec{NxF}} \cdot W_{\vec{Fx2}}) \cdot a_{1_{\vec{2x1}}}] + [(X_{\vec{NxF}} \cdot W_{\vec{Fx2}}) \cdot a_{2_{\vec{2x1}}}]^T \tag{3.55}$$

$$e_{\vec{NxN}} = \begin{bmatrix} a\vec{1}_1[W_{11} \times X_{11} + W_{21} \times X_{12} + ... + W_{F1} \times X_{1F}] + a\vec{1}_2[W_{12} \times X_{11} + ... + W_{F2} \times X_{1F}] \\ a\vec{1}_1[W_{11} \times X_{21} + W_{22} \times X_{22} + ... + W_{F1} \times X_{2F}] + a\vec{1}_2[W_{12} \times X_{21} + ... + W_{F2} \times X_{2F}] \\ a\vec{1}_1[W_{11} \times X_{31} + W_{22} \times X_{32} + ... + W_{F1} \times X_{3F}] + a\vec{1}_2[W_{12} \times X_{31} + ... + W_{F2} \times X_{3F}] \\ \vdots \\ a\vec{1}_1[W_{11} \times X_{N1} + W_{22} \times X_{N2} + ... + W_{F1} \times X_{NF}] + a\vec{1}_2[W_{12} \times X_{N1} + ... + W_{F2} \times X_{NF}] \end{bmatrix}$$

$$+ \begin{bmatrix} a\vec{2}_1[W_{11} \times X_{11} + W_{21} \times X_{12} + ... + W_{F1} \times X_{1F}] + a\vec{2}_2[W_{12} \times X_{11} + ... + W_{F2} \times X_{1F}] \\ a\vec{2}_1[W_{11} \times X_{21} + W_{22} \times X_{22} + ... + W_{F1} \times X_{2F}] + a\vec{2}_2[W_{12} \times X_{21} + ... + W_{F2} \times X_{2F}] \\ a\vec{2}_1[W_{11} \times X_{31} + W_{22} \times X_{32} + ... + W_{F1} \times X_{3F}] + a\vec{2}_2[W_{12} \times X_{31} + ... + W_{F2} \times X_{3F}] \\ \vdots \\ a\vec{2}_1[W_{11} \times X_{N1} + W_{22} \times X_{N2} + ... + W_{F1} \times X_{NF}] + a\vec{2}_2[W_{12} \times X_{N1} + ... + W_{F2} \times X_{NF}] \end{bmatrix}^T \tag{3.56}$$

$$\begin{aligned} e_{11} = {} & a\vec{1}_1[W_{11} \times X_{11} + W_{21} \times X_{12} + ... + W_{F1} \times X_{1F}] + a\vec{1}_2[W_{12} \times X_{11} + W_{22} \times X_{12} + ... + W_{F2} \times X_{1F}] \\ & + a\vec{2}_1[W_{11} \times X_{11} + W_{21} \times X_{12} + ... + W_{F1} \times X_{1F}] + a\vec{2}_2[W_{12} \times X_{11} + W_{22} \times X_{12} + ... + W_{F2} \times X_{1F}] \end{aligned} \tag{3.57}$$

$$\begin{aligned} e_{21} = {} & a\vec{1}_1[W_{11} \times X_{21} + W_{22} \times X_{22} + ... + W_{F1} \times X_{2F}] + a\vec{1}_2[W_{12} \times X_{21} + W_{22} \times X_{22} + ... + W_{F2} \times X_{2F}] \\ & + a\vec{2}_1[W_{11} \times X_{11} + W_{21} \times X_{12} + ... + W_{F1} \times X_{1F}] + a\vec{2}_2[W_{12} \times X_{11} + W_{22} \times X_{12} + ... + W_{F2} \times X_{1F}] \end{aligned} \tag{3.58}$$

$$\begin{aligned} e_{31} = {} & a\vec{1}_1[W_{11} \times X_{31} + W_{22} \times X_{32} + ... + W_{F1} \times X_{3F}] + a\vec{1}_2[W_{12} \times X_{31} + W_{22} \times X_{32} + ... + W_{F2} x X_{3F}] \\ & + a\vec{2}_1[W_{11} \times X_{11} + W_{21} \times X_{12} + ... + W_{F1} \times X_{1F}] + a\vec{2}_2[W_{12} \times X_{11} + W_{22} \times X_{12} + ... + W_{F2} \times X_{1F}] \end{aligned} \tag{3.59}$$

$$\vdots$$

$$\begin{aligned} e_{NN} = {} & a\vec{1}_1[W_{11} \times X_{N1} + W_{22} \times X_{N2} + ... + W_{F1} \times X_{NF}] + a\vec{1}_2[W_{12} \times X_{N1} + + ... + W_{F2} \times X_{NF}] \\ & + a\vec{2}_1[W_{11} \times X_{N1} + W_{22} \times X_{N2} + ... + W_{F1} \times X_{NF}] + a\vec{2}_2[W_{12} \times X_{N1} + ... + W_{F2} \times X_{NF}] \end{aligned} \tag{3.60}$$

$$\alpha_{ij} = \frac{\exp(\text{LeakyRELU}(e_{ij})}{\sum_{k \in N_i} \exp(\text{LeakyRELU}(e_{ik}))} \tag{3.61}$$

$$h_i^1 = \sigma_1(\sum_{\forall j \in N(i)} \alpha_{ij} W_1 X_j) \tag{3.62}$$

$$h_i^1 = \overset{K}{\underset{k=1}{\|}} \frac{1}{K} \sum_{\forall j \in N(i)} \alpha_{ij}^1 W_k^1 h_{j_{k-1}}^1 \tag{3.63}$$

One-level node embedding, Z_2, and the prediction class, $\hat{Y_s}$, are computed as:

$$Z_1 = X_s, \tag{3.64}$$

$$Z_2 = \sigma_1(\alpha * Z_1 \times W_1) + Z_1 \times \varphi, \quad Z_2 = \sigma_2(Z_2), \tag{3.65}$$

$$\hat{Y_s} = \text{softmax}(Z_2). \tag{3.66}$$

Activation functions improving the performance metrics are an exponential linear unit (elu) and rectified linear unit reLU function, and the final output layer activation function is the softmax function for the binary classification. The algorithm design for GAT is displayed in Figure 3.14. The details of the components in the flowchart are expressed in equations 3.57-3.69.

$$\text{elu} : \sigma(X_i) = X_i \text{ for } X_i \geq 0 \text{ and} \tag{3.67}$$

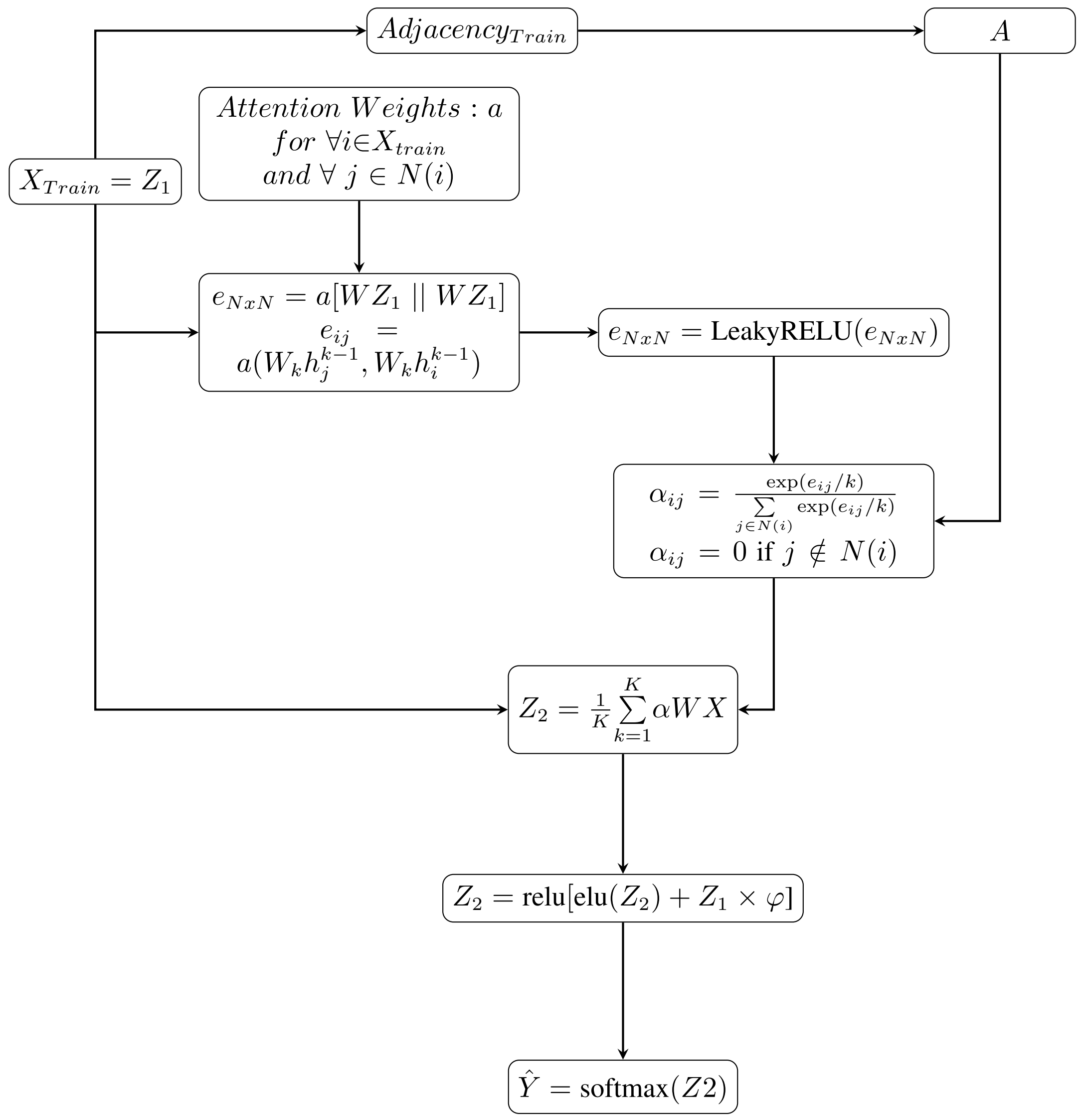
$$\sigma(X_i) = \alpha(\exp(X_i) - 1) \text{ for } X_i < 0$$

$$\text{reLU} : \sigma^{'}(X_i) = \max(0, X_i) \tag{3.68}$$

$$\text{softmax} : \sigma^{"}(Z_{\ell_{ij}}) = \frac{\exp(Z_{\ell_{ij}})}{\sum\limits_{k \in Z_{\ell_i}} \exp(Z_{ik})} \tag{3.69}$$

GAT estimation process, $\hat{Y} = h_{GAT}(a, W, \varphi, X_s, Y_s)$ for sub-datasets, (X_s, Y_s), initializes the system with randomly assigned learning parameters, $a_{1_{2x2}}$, $W_{1_{Fx2}}$, $\varphi_{1_{Fx2}}$ and the model searches to reach to minimum loss between the target Y_s and the estimated default probability, $\hat{Y_s}$, with Adam Optimizer. The objective loss function for this classification task is the cross entropy loss function and it is given as:

$$\mathcal{L}(Y_s, h_{GAT}(a, W, \varphi, X_s, Y_s)) = -\frac{1}{S}\sum_{n=1}^{S}\Bigg[Y_n \log \hat{Y}_n + (1 - Y_n) \log(1 - \hat{Y}_n)\Bigg] \tag{3.70}$$



$Adjacency_{Train}$
A
$Attention\ Weights : a$ $for\ \forall i \in X_{train}$ $and\ \forall\ j \in N(i)$
$X_{Train} = Z_1$
$e_{NxN} = a[WZ_1 \mid\mid WZ_1]$ $e_{ij} = a(W_k h_j^{k-1}, W_k h_i^{k-1})$
$e_{NxN} = \text{LeakyRELU}(e_{NxN})$
$\alpha_{ij} = \frac{\exp(e_{ij}/k)}{\sum_{j \in N(i)} \exp(e_{ij}/k)}$ $\alpha_{ij} = 0 \text{ if } j \notin N(i)$
$Z_2 = \frac{1}{K}\sum_{k=1}^{K} \alpha W X$
$Z_2 = \text{relu}[\text{elu}(Z_2) + Z_1 \times \varphi]$
$\hat{Y} = \text{softmax}(Z2)$

Figure 3.14: Design of GAT Models

The loss function is minimized with "Adam Optimizer", a stochastic adaptive moment optimization process, as in GNN models. The first and second-moment estimations of the gradient and their bias-corrected estimates, m_t, v_t, $\hat{m_t}$, and $\hat{v_t}$ are used, respectively. The gradient vector of the loss function, $\nabla\mathcal{L}(y, h_{GAT}(a, W, \varphi, X_s, y))$, is computed at the initial iteration $t = 0$ and in each subsequent iterations the learning parameters updated with moment estimates.

$$\nabla\mathcal{L}(y, h_{GAT}(a, W, \varphi, X_s, y)) = \begin{bmatrix} \frac{\partial\mathcal{L}}{\partial a_1}(y, h_{GAT}(a, W, \varphi, X_s, y)) \\ \frac{\partial\mathcal{L}}{\partial W_1}(y, h_{GAT}(a, W, \varphi, X_s, y)) \\ \frac{\partial\mathcal{L}}{\partial \varphi_1}(y, h_{GAT}(a, W, \varphi, X_s, y)) \end{bmatrix} \tag{3.71}$$

$$m_t = \beta_1 \times m_{(t-1)} + (1-\beta_1) \times \nabla\mathcal{L} \tag{3.72}$$

$$v_t = \beta_2 \times v_{t-1} + (1-\beta_2) \times \nabla\mathcal{L}_t^2 \tag{3.73}$$

$$\hat{m_t} = m_t/(1-\beta_1^t) \tag{3.74}$$

$$\hat{v_t} = v_t/1-\beta_2^t \tag{3.75}$$

$$\begin{bmatrix} a_t \\ W_t \\ \varphi_t \end{bmatrix} = \begin{bmatrix} a_{t-1} \\ W_{t-1} \\ \varphi_{t-1} \end{bmatrix} - \delta \times \frac{\hat{m_t}}{\sqrt{\hat{v_t} + \in}} \tag{3.76}$$

The learning rate, δ, is specific to each data set. β_2. $\beta_1 = 0.9$ and $\beta_2 = 0.999$ are set in all optimization processes. The update continues until the parameters converge.

Loss minimization is not the only crucial point in selecting the optimal initial parameters in GAT model estimations. GAT models can easily over-fit and the loss in the train data predictions can converge to zero. The performance of these neural network designs is controlled by the prediction performances in validation and test data. The imbalance problem in test data prevents the conversion of it to graph-structured data. Therefore, as in GNNs, the node embeddings of each test data instance are generated in the sub-datasets of the train data. The initial parameters and the design of the models are selected such as the divergence in the metrics of *Sensitivity*$_{Train}$-*Specificity*$_{Train}$ and *Sensitivity*$_{Test}$-*Specificity*$_{Test}$ are the minimum. The strictly overfitting models in the train data are eliminated since the default probability estimations of these models are observed to produce unsatisfying instance-specific cost parameters. The simulations of GAT models are conducted with 1,000 sub-datasets. Moreover, the GAT model estimations are aggregated and medians of the estimations are used for the aggregated GAT model results. The expected default probability of borrower i, $\mathbb{E}(Y_i \mid X_i)_{Agg.\ GATs}$ is computed as:

$$\begin{aligned}\mathbb{E}(Y_i \mid X_i)_{Agg.\ GATs} = &\text{median}\Big\{ h_{GAT_1}(....,X_i,Y_i), h_{GAT_2}(....,X_i,Y_i), \\ &....,h_{GAT_{1000}}(....,X_i,Y_i)\Big\},\end{aligned} \tag{3.77}$$

G_{mean} maximizing threshold in the train data classification is applied to test data for the classification of the aggregated GAT results.

3.3 Generation of Instance-Specific Cost/Risk Parameter

The usual implementation for the class imbalance problem is re-weighting all instances in the minority class with the class ratio (majority class size/minority class size). [44], [86], and [48] offer balancing the classes with the cross multiplication with the class ratio. The latest studies of [89] and [86] dealing with imbalance data and XGBoost assign weights to lose functions on a class basis. This class-balancing approach for solving the imbalanced data problem inspired this study to determine whether classes could be balanced with instance-specific weights instead of some fixed numbers like class ratio. The instance-specific weight should represent the characteristics of this instance, in other words, it should carry some embedded information about itself with its features as a whole and should not be weighted as a constant number determined by a trial error method.

In this study, the expected default probabilities of each instance are selected as a risk indicator to generate cost-sensitive algorithms. The rationale behind the choice of expected default probability for the re-weighting adjustment can be explained with four reasons. Firstly, the expected default probability is a risk level identifier which is a more distinct, more precise measure to link the individual applicant's features directly to his/her potential class with a certain probability. Secondly, the expected default probability can arrange the weights in a convenient ranking for the instances. If the applicant has a higher expected default probability due to his/her features, the classification algorithm will attach a higher weight and give priority to this borrower for his/her correct classification. Thirdly, if the default instance is falsely classified, the error correction for this instance will take a higher weight in the following iterations of the algorithm. Finally, expected default probability is not a nominal monetary indicator, and the comparisons between instances do not require conversions of nominal values to real values. Therefore, instance-specific cost parameter is generated based on the default probabilities of each applicant.

In this part of the study, the generation of three cost/risk parameters with the use of default probability estimates is explained. In addition to default probability estimates, the other parameters required for a balanced classification evolved during the empirical studies of eight data sets. Moreover, cost parameters are all first generated with

LRs, GNN and GAT model estimations are conducted with the question of whether the cost parameters can be improved further with deep learning methods.

3.3.1 Cost Parameter C_1

The borrower or instance-specific risk weight for borrower i, X_i, is represented with $\acute{r_i}$ and it is calculated with the expected default probabilities of instances, $P(D)_i = \mathbb{E}[Y_i \mid X_i]$, and it might arise as one of the candidates for the cost parameter that is being searched. That is,

$$\acute{r_i} = \frac{\mathbb{E}[Y_i \mid X_i]}{\sum_i \mathbb{E}[Y \mid X]} \tag{3.78}$$

The default probabilities of a borrower i, $P(D_{LR})_i$, $P(D_{GNN})_i$ and $P(D_{GAT})_i$, are selected to be the maximums of the default probability observed in the LR, GNN, and GAT model estimations conducted with the sub-datasets, respectively. The numbers of the generated sub-data set are represented with $NSim$. The medians of the default probabilities are also selected but the maximums of the default probabilities of the borrowers are observed to be more effective in the generation of successful cost parameters.

$$\begin{aligned}
P(D_{LR})_i =& \mathbb{E}[Y_i \mid X_i] \\
=& \max \Big\{ h_{LR_1}(\Theta_1, X_i, Y_i), h_{LR_2}(\Theta_2, X_i, Y_i), ..., \\
& h_{LR_{NSim}}(\Theta_{NSim}, X_i, Y_i) \Big\} \qquad (3.79) \\
P(D_{GNN})_i =& \mathbb{E}[Y_i \mid X_i] \\
=& \max \Big\{ h_{GNN_1}(\theta_{1_1}, \theta_{1_2}, \theta_{1_3}, \eta_{1_1}, \eta_{1_2}, \eta_{1_3}, X_i, Y_i), \\
& h_{GNN_2}(\theta_{2_1}, \theta_{2_2}, \theta_{2_3}, \eta_{2_1}, \eta_{2_2}, \eta_{2_3}, X_i, Y_i), ..., \\
& h_{GNN_{NSim}}(\theta_{NSim}, \theta_{NSim_1}, \theta_{NSim_2}, \theta_{NSim_3}, \\
& \eta_{NSim_1}, \eta_{NSim_2}, \eta_{NSim_3}, X_i, Y_i) \Big\} \qquad (3.80)
\end{aligned}$$

$$
\begin{aligned}
P(D_{GAT})_i =& \mathbb{E}[Y_i \mid X_i] \\
=& \max\Big\{ h_{GAT_1}(a_1, W_1, \varphi_1, X_i, Y_i), h_{GAT_2}(a_2, W_2, \varphi_2, X_i, Y_i), ..., \\
& h_{GAT_{NSim}}(a_{NSim}, W_{NSim}, \varphi_{NSim}, X_i, Y_i)\Big\}
\end{aligned} \tag{3.81}
$$

A well-operating cost-sensitive algorithm should ensure balanced recognition/classification of both classes. Nevertheless, inserting $\acute{r}$ as an instance-specific cost parameter to algorithms and re-weighting all data with this parameter is observed to improve the detection of the minority class but not sufficiently, since the majority class keeps dominating the loss function in all credit data sets subject to experimental study. The potential reason is evaluated such that the sum of default probabilities of non-default instances continues to dominate the sum of default probabilities of the default class. In other words, the number of instances in the dominant class is so high that the total error caused by default class instances remains small as a percentage of the total error. The class equalization in terms of default risks and total error in lost functions are not attained with only $\acute{r}$.

It becomes clear what has been required for the construction of new cost parameters when the ratios of total default risks of classes are still so high. [86] offer equalizing the class losses with cross multiplication of class ratio in Modified Focal Loss (L_{MF}) with ϕ parameter. ϕ parameter is inserted in the loss function as the inverse of class weights to balance the class losses and it is described as [86]:

$$
L_{MF} = -\sum_{i=1}^{N} \phi Y_i (1-\hat{Y}_i)^{\gamma} \log(\hat{Y}_i) + (1-Y_i)\hat{Y}_i^{\gamma} \log(1-\hat{y}_i) \tag{3.82}
$$

where $\phi = \dfrac{1}{N}\displaystyle\sum_{i=1}^{N} \beta_i$, and N is the data size

$$
\text{and here, } \beta_i = \begin{cases} \frac{\textit{Positives}+\textit{Negatives}}{\textit{Positives}} \text{ for } Y = 1, \\ \frac{\textit{Positives}+\textit{Negatives}}{\textit{Negatives}} \text{ for } Y = 0. \end{cases}
$$

Here, *Positives* is the class size with the label $Y = 1$ and *Negatives* shows the class size with the label $Y = 0$.

L_{MF} inspires about the incompleteness of the cost parameter, and the necessity of an additional adjustment for balancing the classes in terms of classed-based total risks.

In other words, the ratio of class-based total risk, (sum of default probabilities in majority class)/(sum of default probabilities in minority class), can be used to balance the class-based risk levels and the loss functions. Consequently, a new '$Risk\ Adjusting\ Weight'$ (RAW) parameter, β_{RAW}, is proposed to equalize the total cost/risk of both classes.

$$\beta_{RAW} = \begin{cases} \frac{\sum_i \mathbb{E}[P(Y_i|X_i)]\mathbb{1}_{Y=0}}{\sum_j \mathbb{E}[P(Y_j|X_j)]\mathbb{1}_{Y=1}} \text{ for } Y = 1, \\ 1 \text{ for } Y = 0. \end{cases} \tag{3.83}$$

First cost parameter, C_1 is obtained with the multiplication of β_{RAW} and $P(D)$ to equalize the sum of the total probability of defaults on a class basis. The empirical findings indicate β_{RAW} values are low if the class ratios are not high, but they are considerably high for the data sets with severely unbalanced classes. Moreover, the distributions of the probability predictions, $P(D)_i$, and their divergence from the real class labels can change with the model used and β_{RAW} values also change but not with a high deviation. The empirical findings indicate that high β_{RAW} values can over-boost the minority class while the low β_{RAW} values might be insufficient to balance the minority class losses. Therefore, m parameter, $0 < m < \infty$, is introduced to control the effects of the $P(D)$ and β_{RAW}. m is specific to each data set and the classification algorithm used and it has to be optimized with the objective function that is maximized or minimized. The first cost/risk parameter for borrower i is defined as:

$$C_{1(i)} = (P(D)_i \times \beta_{RAW_i})^m \tag{3.84}$$

3.3.2 Cost Parameter C_2 and C_3

In the literature, most of the earlier studies dealing with cost-sensitive classification algorithms have been conducted with the modifications to Adaptive Boosting (AdaBoost) algorithm. AdaBoost's error correction design with exponential weight updating function has inspired this study for the formulation of the second and the third instance-specific cost/risk parameters, C_2 and C_3, respectively.

AdaBoost algorithm, one specific method of boosting, is listed among the top ten successful algorithms. [72] set up an error correction system based on the weak classifiers, $h(x)$ and weight updating mechanism focusing on false predictions with boosted weights for their corrections in the final aggregated prediction function, $H(x)$. AdaBoost first selects a base learner/weak classifier which has the prediction success of at least $1/2$ and it assigns equal weights to every instance in the data. After the observation of false predictions, AdaBoost algorithm makes iterative calls, modifies the initial weights and increases the weights of false predictions in the next iteration, and decreases the truly classified instances.

The elements of AdaBoost can be described with the given predictors and the classes of the data with size N as $(x_1, y_1), ...(x_N, y_N)$, where $x_i \in X$ is the input set, $y_i \in (-1, +1)$ represents the class for the data instance i. $t \in (1, T)$ is the number of iterations. $h_t(x)$: $X \rightarrow (-1, 1)$ represents the prediction of the weak learner at iteration t and it is the weak learner like a decision stump. $f^t(x) = \alpha_t h(x)$ denotes the corrected prediction of $h_t(x)$ with the weight D^t. $\varepsilon_t = Pr_{i_{\sim D^t}}[h_t(x_i) \neq y_i] = \sum\limits_{i_{h_t(x_i) \neq y_i}} D^t(i)$ is the error at iteration t. $H(x) = sign\left(\sum\limits_{t=1}^{T} \alpha_t h_t(x)\right)$ is the final prediction. Here, $D_1(i) = 1/N$ is the initial weights for $\forall\, i \in (1, 2, ..., N)$ and weight updating mechanism of AdaBoost, D_{t+1}, is computed as:

$$H_{t-1}(x_i) = \alpha_1 h_1(x_i) + \alpha_2 h_2(x_i) + ... + \alpha_{t-1} h_{t-1}(x_i) \tag{3.85}$$

$$H_t(x_i) = H_{t-1}(x_i) + \alpha_{t-1} h(x_i) \tag{3.86}$$

$$\varepsilon_t = \sum_i \varepsilon[H_{t-1}(x_i) + \alpha h(x_i)] \tag{3.87}$$

$$\varepsilon_t = \sum_{i=1}^{N} e^{-y_i H_m(x_i)} = \sum_{i=1}^{N} e^{-y_i H_{m-1}(x_i)} e^{-y_i \alpha_m h_m(x_i)} \tag{3.88}$$

$$w_i^{(1)} = 1 \text{ and } w_i^{(m)} = e^{-y_i H_{m-1}(x_i)} \text{ for } m > 1 \tag{3.89}$$

$$\varepsilon = \sum_{i=1}^{N} w_i^{(m)} e^{-y_i \alpha_m h_m(x_i)} \tag{3.90}$$

$$\varepsilon = \sum_{y_i = h_m(x_i} w_i^{(m)} e^{-\alpha_m} + \sum_{y_i \neq h_m(x_i} w_i^{(m)} e^{\alpha_m} \tag{3.91}$$

$$= \sum_{i=1}^{N} w_i^{(m)} e^{-\alpha_m} + \sum_{y_i \neq h_m(x_i} w_i^{(m)} (e^{\alpha_m} - e^{-\alpha_m}) \tag{3.92}$$

$$\frac{\partial \varepsilon}{\partial \alpha_m} = \frac{\partial \sum\limits_{y_i = h_m(x_i} w_i^{(m)} e^{-\alpha_m} + \sum\limits_{y_i \neq h_m(x_i} w_i^{(m)} e^{\alpha_m}}{\partial \alpha_m}$$

$$\alpha = \frac{1}{2}\left(\frac{\sum\limits_{y_i = h_m(x_i} w_i^{(m)}}{\sum\limits_{y_i \neq h_m(x_i} w_i^{(m)}}\right) \tag{3.93}$$

$$\varepsilon_m = \sum_{y_i \neq h_m(x_i} \frac{w_i^{(m)}}{\sum\limits_{i=1}^{N} w_i^{(m)}} \tag{3.94}$$

$$\alpha_m = \frac{1}{2} ln \left(\frac{1 - \varepsilon_m}{\varepsilon_m}\right) \tag{3.95}$$

$$D^{t+1}(i) = \frac{D^t(i)}{Z_t} \times \begin{cases} \exp(-\alpha_t \text{ if } h_t(x_i) = y_i \\ \exp(\alpha_t \text{ if } h_t(x_i) \neq y_i \end{cases} \tag{3.96}$$

$$D^{t+1}(i) = \frac{D^t(i)\exp(-\alpha_t y_i h_t(x_i))}{Z_t} \tag{3.97}$$

where Z_t is the normalization factor, $Z_t = \sum\limits_{i=1}^{N} D^t(i)\exp(-\alpha_t y_i h_t(x_i))$, so D^{t+1} becomes a distribution.

The weight update equations of the original AdaBoost algorithm and its cost-sensitive modifications are displayed in Table 3.1. In the literature, it is observed that there is no specific rule defining what cost parameter should be in cost-sensitive modifications of AdaBoost, they usually focus on creating a cost ratio based on no rule but setting higher any constant number for all the instances in minority class which is generally based on trial error basis. The cost parameter C_i is inserted in the original weight equation of D_{t+1} as a direct multiplier, with its exponential or both and with several other functional forms.

Table 3.1: Cost Sensitive Modifications of AdaBoost

$\alpha_t = \frac{1}{2}\log(\frac{\sum_{i,y_i=h_t(x_i)} D^t(i)}{\sum_{i,y_i \neq h_t(x_i)} D^t(i)}$, D^{t+1}: Weights at $t+1$	
AdaBoost [72]:	$D^{t+1}(i) = \frac{D^t(i)\exp(-\alpha_t h_t(x_i) y_i}{Z_t}$
AdaC1 [79]:	$D^{t+1}(i) = \frac{D^t(i)\exp(-\alpha_t C_i h_t(x_i) y_i}{Z_t}$
AdaC2 [80]:	$D^{t+1}(i) = \frac{C_i D^t(i)\exp(-\alpha_t h_t(x_i) y_i}{Z_t}$
AdaC3 [80]:	$D^{t+1}(i) = \frac{C_i D^t(i)\exp(-\alpha_t C_i h_t(x_i) y_i}{Z_t}$
AdaCost [20]:	$D^{t+1}(i) = D^t(i)\exp(-\alpha_t h_t(x_i) y_i \beta_{sgn}(h_t(x_i), y_i)$
CSB1 [84]:	$D^{t+1}(i) = \frac{C_{sgn}(h_t(x_i), y_i) D^t(i)\exp(-y_i h_t(x_i)}{Z_t}$
CSB2 [84]:	$D^{t+1}(i) = \frac{C_{sgn}(h_t(x_i), y_i) D^t(i)\exp(-\alpha_t y_i h_t(x_i)}{Z_t}$

The second cost Parameter, C_2, is built on the first cost parameter and AdaBoost error correcting weight updating process. In the original AdaBoost algorithm, the weight updating parameter, $D^{t+1}(i)$, for each instance is used and the weights of truly classified instances are reduced and the weights of the false classified instances are increased in the next iteration.

$$D^{t+1}(i) = \frac{D^t(i)\exp(-\alpha_t h_t(x_i)y_i}{Z_t} \text{ where,} \tag{3.98}$$

$$\alpha_t = \frac{1}{2}\log(\frac{\sum_{i,y_i=h_t(x_i)} D^t(i)}{\sum_{i,y_i\neq h_t(x_i)} D^t(i)} \tag{3.99}$$

where, y_i is the class label and $h_t(x_i)$ is the base classifier prediction.

The cost parameter in AdaC1 [79] is used to strengthen the exponential power of the error correcting α parameter as $D^{t+1}(i) = \frac{D^t(i)\exp(-\alpha_t C_i h_t(x_i)y_i}{Z_t}$. Based on AdaC1, exponentially scaled risk parameters of C_2 and C_3 are computed. Re-weighting with only $\exp(\acute{(}PD)$ does not result in balanced classification like $\acute{r}$ and the reinforcement of β_{RAW} is found critically important in C_2 and C_3.

Different from the original AdaBoost algorithm, C_2 is designed such that it now also interferes with the weights on a class basis, it assigns weights lower than 1 to the instances of the non-default (majority) class and weights higher than 1 to the instances in the default (minority) class. A threshold parameter, $thrs$, is inserted in C_2 which is the minimum of optimal thresholds maximizing G_{mean} estimated in a simulated annealing process conducted with LRs of samples with a class ratio of 1. On the other hand, $thrs$ is the G_{mean} maximizing threshold for the medians of probability estimations in the aggregated GNN and GAT models.

The second weight updating instance-specific cost parameter for borrower i, C_2, is designed as:

$$C_{2(i)} = [\exp(\frac{P(D)_i \times \beta_{RAW_i}}{thrs} - 1)]^m, 0 < m < \infty \tag{3.100}$$

$$C_{2(i)} = [\exp(\frac{C_{1(i)}}{thrs} - 1)]^m \tag{3.101}$$

$thrs$ acts as a breaking point since the main intuition for $\exp(\frac{P(D)\times\beta_{RAW}}{thrs} - 1)$ to make it greater than 1 for the default class and less than 1 for the non-default class which ensures $C_2 < 1$for $Y = 0$ and $C_2 > 1$ for $Y = 1$. Moreover, m in C_2 stands for the control of excess weighting for the default class, since exponentially scaling raises too high weights for these instances leading the non-defaults to become minority class. According to the data level experiments, m changes on a data basis, if the imbalance is severely high, β_{RAW} is also seriously high causing m to be relatively lower.

$$P(D) = h(parameters, X_i, Y_i) \tag{3.102}$$

$$P(D_{LR}) = \max\{h_{LR_1}(..., X_i, Y_i), ..., h_{LR_{NSim}}(..., X_i, Y_i)\} \tag{3.103}$$

$$P(D_{GNN}) = \max\{h_{GNN_1}(..., X_i, Y_i), .., h_{GNN_{NSim}}(..., X_i, Y_i)\} \tag{3.104}$$

$$P(D_{GAT}) = \max\{h_{GAT_1}(..., X_i, Y_i), .., h_{GAT_{NSim}}(..., X_i, Y_i)\} \tag{3.105}$$

$$1 < C_{2(i)} < \infty \text{ for } thrs < P(D) \text{ and } 1 < \beta_{RAW}$$

$$0 < C_{2(i)} < 1 \text{ for } P(D) < thrs \text{ and } \beta_{RAW} = 1$$

The third Cost Parameter, C_3, is generated with the multiplication of exponentially scaled default probabilities and β_{RAW}. Re-weighting with $\exp(P(D) \times \beta_{RAW})$ over-boosts the minority class and the loss function becomes biased towards this class instances. Therefore, scaling only $P(D)$ exponentially and multiplying it with m powered β_{RAW}, where $0 < m < \infty$ and m controls excess weighting, generates a well-functioning third cost/risk parameter for borrower i, C_3:

$$C_3(i) = \exp(P(D_i)) \times (\beta_{RAW_i})^m \tag{3.106}$$

It is crucial to underline the fact that if a default instance appears to have very low default probability due to the insufficient identification of the features reported, the cost parameter might not strongly boost its weight above 1, but the error correcting mechanism of AdaBoost, XGBoost, and ANN are expected to fortify the detection power of the cost-sensitive algorithms.

3.3.3 Can Expected Nominal Costs be Instance-Specific Cost Parameter?

The credit amount approved by the bank is certainly the actual real loss if the borrower defaults and it should be one of the indicators defining the risk level of the loan. The cost parameter could be defined as the expected default costs of the applicants. However, the use of nominal values of credit amounts might be misleading in designating the real expected risk level of applicants. As a simple example, a consumer with the default probability of 0.1 and 1,000 units of credit amount has an expected cost of 100 units. On the other hand, a consumer with a default probability of 0.9 and 100 units of credit amount has an expected cost of 90 units. This example indicates that cost-sensitive algorithms designed on the expected nominal losses, even if they are converted to relative ratios, can be misleading. Since nominal credit amounts can lead the less risky applicants to be evaluated as riskier than they are when the comparisons are made in terms of nominal expected losses. In fact, there are several attributes asked to be reported by the applicant and the high credit amount can be ignorable if the applicant already belongs to a high-income group and has other satisfying conditions.

In the data experimental stage, expected costs for each instance in the trained data are calculated with estimated default probabilities and credit amounts as:

$$\mathbb{E}[Cost_i] = \mathbb{E}[Y_i = 1 \mid X_i] \times \textit{Credit Amount}_i. \tag{3.107}$$

Besides different logarithmic transformations of each expected cost, different cost ratios like $\frac{\mathbb{E}[Cost_i]}{\sum_i \mathbb{E}[Cost]}, \frac{\mathbb{E}[Cost_i]}{max\mathbb{E}[Cost]}, \frac{\mathbb{E}[Cost_i]}{min\mathbb{E}[Cost]}, \frac{\mathbb{E}[Cost_i]-min\mathbb{E}[Cost]}{max\mathbb{E}[Cost]-min\mathbb{E}[Cost]}$ are calculated and included as a cost parameter in the estimations of AdaC1, AdaC2, and AdaC3. However, the results are unsatisfactory and even worse than the original AdaBoost algorithm.

The cost sensitivity parameter should be generated in real terms indicating the real risk of the applicant. In fact, in most of the data sets, the default probabilities for each instance are estimated with the attributes of the nominal credit amounts, the ratio of nominal credit amount to nominal income, or the ratio of annuity to income. Default probabilities in LRs already carry the information about the credit amounts. Therefore, in the cost-sensitive setting, the cost parameter is constructed on the estimated default probability which is also accepted as the indicator of the risk level for each applicant.

3.3.4 The Generation of Cost-Sensitive Algorithms

The instance-specific cost/risk parameters are embedded in boosting algorithms of AdaBoost and XGboost, and ANN models to improve the minority class weight in the loss functions and lead to balanced classification results for both classes. The choice of these algorithms can be reasoned with three basic points. Firstly, very early cost-sensitive modifications in algorithms exist in AdaBoost studies. The generation of exponentially scaled cost parameters has been inspired by the weight-update mechanism of AdaBoost and the new parameters are first tested on AdaBoost. Secondly, XGBoost is evaluated as the most successful and the fastest algorithm in most of the empirical studies in recent years. Therefore, the impact of the offered parameters is also analyzed with XGBoost. Thirdly, ANN models allow very complex non-linear model designs which might increase the classification performances beyond the other algorithms, and the effect of the new cost parameters are analyzed with these neural network models.

3.3.4.1 Risk Adjusted Cost Sensitive AdaBoost-1

The initial weights for each instance in the original AdaBoost are attained equally as $1/N, N$ is the sample size [72] and [44] propose initial weight adjustment in AdaBoost which is based on the asymmetry in class ratios. In this study, as a first modification to the original AdaBoost, the new initializing weight vector (cost-based instance weights), is changed from $D^0 = 1/N$ to D^0_{RAW}. It is calculated with the default probability of instances, $P(D)$, and risk-adjusted class weights of the instances, β_{RAW}. Since total weights of instances should sum up to 1, $P(D)_i \times \beta_{RAW_i}$ is divided by the total adjusted cost, $\sum_i^N P(D)_i \times \beta_{RAW_i}$. This modified version is called 'Risk-Adjusted Cost Sensitive AdaBoost-1' (RCS-AdaB1) and is expressed as:

$$D^0_{RAW}(i) = \frac{(P(D)_i \times \beta_{RAW_i})^m}{\sum_i^N (P(D)_i} \times \beta_{RAW_i})^m \text{ or } D^0_{RAW_i} = \frac{C_{1(i)}}{\sum_i^N C_{1(i)}} \qquad (3.108)$$

The progress in RCS-Ada1 takes place with $(X_1, Y_1), ..., (X_N, Y_N)$ predictors and classes where $X_i \in X^{NxF}, Y_i \in Y = (-1, 1)$, N is the data size, F is the feature size and base learner of $h_t \rightarrow Y$. I is the number of iterations, $t = 1$ to I and Z_t is the normalization factor. The algorithm initializes the instances with $D^0 = D^0_{RAW}$, and α and D^{t+1} are updated as in the original AdaBoost as follows:

$$D^0(i) = D^0_{RAW}(i) \tag{3.109}$$

$$\alpha_t = \frac{1}{2}\log(\frac{\sum_{i,y_i=h_t(x_i)} D^t(i)}{\sum_{i,y_i\neq h_t(x_i)} D^t(i)}) \tag{3.110}$$

$$D^{t+1}(i) = \frac{D^t(i)\exp(-\alpha_t h_t(x_i)y_i)}{Z_t} \tag{3.111}$$

$$H(x) = sign(\sum_t \alpha_t h_t(x)) \tag{3.112}$$

3.3.4.2 Risk Adjusted Cost Sensitive AdaBoost-2

Beyond the initial weight adjustments, the cost-sensitive modifications of AdaBoost are mostly focused on adjusting instance weights, D_{t+1}, with a cost parameter C_i in each iteration as given in Table 3.1. Similar to these interventions in each iteration, weights are also adjusted in each iteration with a cost parameter C_2 to strengthen the algorithm in the recognition of default classes. However, inserting C_2 in the updating weight equation leads to a severe over-weighting for the default class in the first 10 iterations and a decrease in true classifications of non-defaults (True Negative Rate: TN_{rate}) significantly. Therefore, the effect of this cost parameter, C_2, is reduced with a parameter of ψ in the subsequent iterations to prevent a reduction in the specificity and to keep strengthening the sensitivity rate without distorting the specificity rate in all data sets. This can be expressed as:

$$D^{t+1}(i) = \frac{D^t(i)C_2\exp(-\alpha_t h_t(x_i)y_i}{Z_t} \text{ where,} \tag{3.113}$$

$$C_2^{t=1} = [\exp(\frac{P(D)\times\beta_{RAW}}{thrs} - 1)]^m \text{ for } t = 1 \text{ and} \tag{3.114}$$

$$C_2^{t+1} = (C_2^t)^\psi \text{ for } t > 1$$

The progress in RCS-Ada2 takes place with $(x_1, y_1), ..., (x_m, y_m)$ predictors and classes where $x_i \in X$, $y_i \in Y = (-1, 1)$ and base learner of $h_t \rightarrow Y$. I is the number of iterations, $t = 1$ to I and Z_t is the normalization factor. The algorithm initializes the instances with $D^0 = D^{RAW_0}$, and α and D_{t+1} are updated as in the Equations 3.116-3.117. The instance weight update equation in the second modified algorithm, Risk Adjusted Cost Sensitive AdaBoost-2 (RCS-Ada2) is:

$$D^0(i) = D^0_{RAW}(i) \tag{3.115}$$

$$\alpha_t = \frac{1}{2}\log(\frac{\sum_{i,y_i=h_t(x_i)} D^t(i)}{\sum_{i,y_i\neq h_t(x_i)} D^t(i)}) \tag{3.116}$$

$$D^{t+1}(i) = \frac{D^t(i)C_2\exp(-\alpha_t h_t(x_i)y_i}{Z_t} \text{ where,} \tag{3.117}$$

$$H(x) = sign(\sum_t \alpha_t h_t(x)) \tag{3.118}$$

3.3.5 Risk Adjusted Cost Sensitive XGBoost

XGBoost is designed on traditional gradient boosting with the addition of regularization parameters to the loss function [14]. Defined as a scalable end-to-end tree boosting system, XGBoost has an efficient calculation for parallel tree learning with a novel sparsity-aware algorithm for sparse data.

XGBoost classification is a process with predictors-classes: $(X_1, Y_1), ..., (X_N, Y_N)$ predictors and classes where $X_i \in X^{NxF}, Y_i \in Y = (0, 1)$, N is the data size, F is the feature size and it proceeds with K additive decision tree classifier functions, $f(x)$. The prediction for Y_i is $\hat{Y}_i = \phi(X_i) = \sum_{k=1}^{K} f_k(X_i), f_k \in \mathcal{F}$ where $\mathcal{F} = \{f(x) = w_{q(x)}\}(q : \mathbb{R}^m \to T, w \in \mathbb{R}^T)$ is defined as the space of classification trees [14]. q is the structure of each tree mapping an example to the corresponding leaf index and T is the number of leaves in the tree. Each f_k represents an independent tree structure q with leaf weights w. The leaf weights are computed with the loss minimization target function with regularization parameters and they are specific to leaves and are added to previous predictions for the new prediction. $\mathcal{L}^t$ is the loss function at iteration t and it is the sum of the borrower-specific prediction losses, $l(Y_i, \hat{Y}_i^{(t-1)})$, where i represents the borrower i, $i \in N$:

$$\mathcal{L}^t = \sum_i^N l(Y_i, \hat{Y}_i^{(t-1)} + f_t(X_i)) + \Omega(f_t) \text{ where } \Omega(f) = \gamma T + \frac{1}{2}\lambda|w| \tag{3.119}$$

To make a faster optimization, the loss function is computed with second-order Taylor approximation:

$$\mathcal{L}^t \approx \sum_{i}^{N}[l(Y_i, \hat{Y}_i^{(t-1)}) + \frac{\partial l(Y_i, \hat{y}_i^{(t-1)})}{\partial \hat{Y}^{(t-1)}} f_t(X_i) + \frac{1}{2}\frac{\partial^2 l(Y_i, \hat{Y}_i^{(t-1)})}{\partial l(Y_i, \hat{Y}_i^{(t-1)})^2} f_t^2(X_i)] + \Omega(f_t) \tag{3.120}$$

$$\mathcal{L}^t \approx \sum_{i}^{N}[l(Y_i, \hat{Y}_i^{(t-1)}) + g_i f_t(X_i) + \frac{1}{2} h_i f_t^2(X_i)] + \Omega(f_t) \tag{3.121}$$

The instance set of leaf j is defined with $I_j = \{i \mid q(x_i) = j\}$ and re-formulates the loss function with weight w_j for the leaf j [14]:

$$\mathcal{L}^t \approx \sum_{i}^{N}[l(Y_i, \hat{Y}_i^{(t-1)}) + \sum_{j=1}^{T}[(\sum_{i \in I_j} g_i) w_j + \frac{1}{2}(\sum_{i \in I_j} h_i + \lambda) w_j^2] + \gamma T \tag{3.122}$$

The optimal weight for weight j of a tree structure $q(x)$, w_j:

$$\frac{\partial \mathcal{L}}{\partial w_j} = \sum_{i \in I_j} g_i + (\sum_{i \in I_j} h_i + \lambda) w_j = 0,\ w_j = -\frac{\sum_{i \in I_j} g_i}{\sum_{i \in I_j} h_i + \lambda} \tag{3.123}$$

The loss function can be re-formulated as the Equation 3.124:

$$\mathcal{L}^t = \sum_{i}^{N}[l(Y_i, \hat{Y}_i^{(t-1)}) - \frac{1}{2}\sum_{j=1}^{T}\frac{(\sum_{i \in I_j} g_i)^2}{\sum_{i \in I_j} h_i + \lambda} + \gamma T \tag{3.124}$$

$$Score = -\frac{1}{2}\sum_{j=1}^{T}\frac{(\sum_{i \in I_j} g_i)^2}{\sum_{i \in I_j} h_i + \lambda} + \gamma T \tag{3.125}$$

The second part of the loss function, $Score$, computed with the sum of gradients and hessian values of the instances split on the same leaf is used as a scoring function for the quality of a tree with $q(x)$ with the optimal w_j. The split candidates for the nodes are evaluated with a greedy algorithm which initializes the process with a single leaf and makes iterative branch additions to the tree. The instances in the right and left sides of the nodes after the split are evaluated with the loss function given in Equation 3.126.

$$\mathcal{L}_{split} = \frac{1}{2}\Bigg[\sum_{j=1}^{T}\frac{(\sum_{i \in I_L} g_i)^2}{\sum_{i \in I_L} h_i + \lambda} + \sum_{j=1}^{T}\frac{(\sum_{i \in I_R} g_i)^2}{\sum_{i \in I_R} h_i + \lambda} - \sum_{j=1}^{T}\frac{(\sum_{i \in I} g_i)^2}{\sum_{i \in I} h_i + \lambda}\Bigg] - \gamma \tag{3.126}$$

Exact greedy algorithm for split finding is computed from $k = 1$ to m as: $G \leftarrow \sum\limits_{i \in I} g_i$, $H \leftarrow \sum\limits_{i \in I} h_i$ and $G_L \leftarrow 0, H_L \leftarrow 0$

All instances in each leaf, $\forall j \in I$, are sorted and the gradients and hessians are computed as $G_L \leftarrow G_L + g_i, H_R \leftarrow H_L + h_j, G_R \leftarrow G - G_L, H_R \leftarrow H - H_L$

score $\leftarrow$ max(score, $\frac{G_L^2}{H_L+\lambda} + \frac{G_R^2}{H_R+\lambda} - \frac{G^2}{H+\lambda}$)

where $f(x)$ is the split with the maximum score.

The instance-specific cost-sensitive modifications of XGBoost with the estimated cost parameters are generated by re-weighting the cross entropy loss function becomes:

$$\mathcal{L}(Y, \phi(f(x) = w_{q(X)}), K)) = -\frac{1}{N}\sum_{n=1}^{N} C_i \Bigg[Y_n \log \hat{Y}_i + (1 - Y_i) \log(1 - \hat{Y}_i) \Bigg] \tag{3.127}$$

$$+ \gamma T + \frac{1}{2} * \lambda \sum_{j=1}^{T} \hat{Y}_j^{\,2}$$

The cost-sensitive modifications of XGBoost are called Risk-Based Cost-Sensitive XGBoost (RCS-XGB1) if C= C_1 RCS-XGB2 if C_2 and RCS-XGB3 if C_3.

$p_i = \hat{Y}_i$ is the default probability of an instance and $p_i/(1 - p_i)$ is the odds of the probability of default against the probability of non-default for the borrower i. Cross entropy loss function can be defined with the $\log(odds)$ and the update of the predictions, $\hat{Y}$, in each iteration t can be defined as:

$$\mathcal{L}(Y_i, p_i) = -C_i \Bigg[Y_i \log(p_i) + (1 - Y_i) \log(1 - p_i) \Bigg], \tag{3.128}$$

$$\mathcal{L}(Y_i, \log(odds)_i) = \hat{Y}) = -C_i \Bigg[Y_i \log(odds)_i + (1 - Y_i) \log(1 - e^{\log(odds)_i}) \Bigg], \tag{3.129}$$

$$\text{where } \frac{\partial \mathcal{L}(Y, \log(odds)_i}{\partial \log(odds)_i} = C_i \times (-Y_i + \frac{e^{\log(odds)_i}}{1 + e^{\log(odds)_i}})$$

$$g_i = -C_i(Y_i - \hat{Y}_i) \tag{3.130}$$

$$\frac{\partial^2 \mathcal{L}(Y_i, \log(odds)_i}{\partial(\log(odds)_i)^2} = C_i \times \frac{e^{\log(odds)_i}}{1 + e^{\log(odds)_i}} \times \frac{1}{1 + e^{\log(odds)_i}} \tag{3.131}$$

$$h_i = C_i \times \hat{Y}_i \times (1 - \hat{Y}_i)$$

$$w_j = -\frac{\sum\limits_{i \in I_j} g_i}{\sum\limits_{n \in I_j} h_i + \lambda} = \frac{\sum\limits_{i \in I_j} C_i(Y_i - \hat{Y}_i)}{\sum\limits_{i \in I_j} C_i \times \hat{Y}_i \times (1 - \hat{Y}_i) + \lambda} \tag{3.132}$$

where λ is the regulation parameter.

$$\log(odds)_i^{t+1} = \log(odds)_i^t + \delta \times w_i^{t+1} \tag{3.133}$$

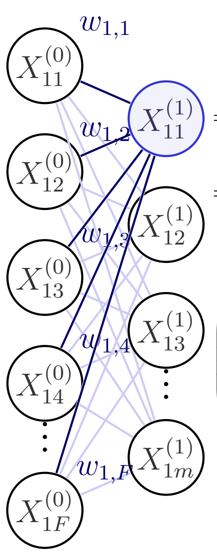
$$= \log(odds)_i^{t+1} + \delta \times \frac{\sum\limits_{i \in I_j} C_i(Y_i - \hat{Y}_i)}{\sum\limits_{i \in I_j} C_i \times \hat{Y}_i \times (1 - \hat{Y}_i) + \lambda}$$

$$\hat{Y}_i^{t+1} = \frac{e^{\log(odds)_i^t}}{1 + \log(odds)_i^{t+1}} \tag{3.134}$$

3.3.6 Risk Adjusted Cost Sensitive ANN

A multilayer perceptron feedforward artificial neural network (ANN) can be designed with different layer and activation function choices for binary classification, but the output layer of the network is the function of Softmax $(X_{ij}) = \frac{\exp(X_{ij})}{\sum_{k \in X_i} \exp(X_{ik})} \in [0, 1]$. The nonlinear transformations of the input data start with randomly assigned weights, W, and the selected activation functions, $\sigma(.)$, as illustrated with the first hidden layer operation for data instance X_1^0 as given in Figure 3.15. The non-linearity increases as the hidden layers and the neurons in each layer increase.

ANN classification process for binary classification of defaults and non-defaults in credit data sets are set up with the predictors and classes: $(X_1, Y_1), ..., (X_N, Y_N)$ where $X_i \in X^{NxF}, Y_i \in Y = (0, 1)$, N is the data size, F is the feature size. A feedforward neural network with the non-linear function
$\hat{Y} = h_{ANN}(Y, X, \theta_1, \theta_2, ..., \theta_\ell, \sigma_1(.), \sigma_2(.), ..., \sigma_\ell(.))$ is designed with $\ell - 1$ hidden layers and optimal neurons in layers, activation functions for each hidden layers, $\sigma_1(.)$, $\sigma_2(.)$,..., $\sigma_{\ell-1}(.)$ and output layer with softmax function, $\sigma_\ell(.)$. The network starts with random weights, $W = [W_0, W_1, W_2, ..., W_\ell]$ and a pre-determined learning rate,



$$= \sigma\left(w_{1,1}X^{(0)}_{11} + \ldots + w_{1,F}X^{(0)}_{1F} + b^{(0)}_1\right)$$
$$= \sigma\left(\sum_{i=1}^{F} w_{1,i}X^{(0)}_{1i} + b^{(0)}_1\right)$$

$$\begin{pmatrix} X^{(1)}_1 \\ X^{(1)}_2 \\ \vdots \\ X^{(1)}_m \end{pmatrix}_{mxN} = \sigma\left[\begin{pmatrix} w_{1,1} & \ldots & w_{1,F} \\ w_{2,1} & \ldots & w_{2,F} \\ \vdots & \vdots & \ddots & \vdots \\ w_{m,1} & \ldots & w_{m,F} \end{pmatrix}_{mxF} \begin{pmatrix} X^{(0)}_{11} & X^{(0)}_{21} & \cdots & X^{(0)}_{N1} \\ X^{(0)}_{12} & X^{(0)}_{22} & \cdots & X^{(0)}_{N2} \\ \vdots \\ X^{(0)}_{1F} & X^{(0)}_{2F} & \cdots & X^{(0)}_{NF} \end{pmatrix}_{FxN} + \begin{pmatrix} b^{(0)}_1 \\ b^{(0)}_2 \\ \vdots \\ b^{(0)}_m \end{pmatrix}\right]$$

$$X^{(1)} = \sigma\left(\mathbf{W}^{(0)}X^{(0)} + \mathbf{b}^{(0)}\right)$$

Figure 3.15: First Layer Interactions in ANN

δ. The weights corresponding to each layer are updated at each iteration as they propagate back into the neural network to adjust the weight parameters to minimize loss by the amounts allowed by loss gradient descent and learning rate.

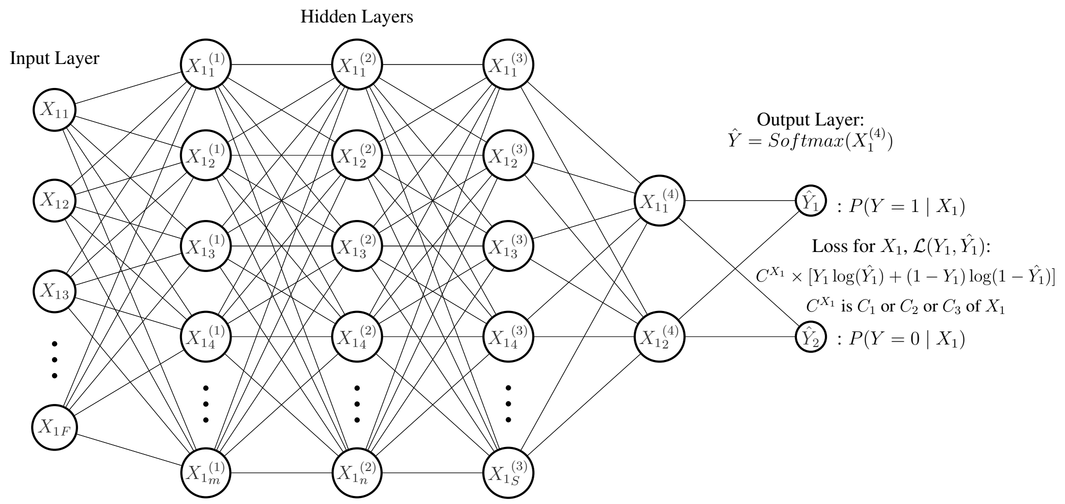
$$X^{(1)} = \sigma_1\left(\mathbf{W}^{(0)}X^{(0)} + \mathbf{b}^{(0)}\right) \tag{3.135}$$
$$X^{(2)} = \sigma_2\left(\mathbf{W}^{(1)}X^{(1)} + \mathbf{b}^{(1)}\right) \tag{3.136}$$
$$\vdots$$
$$X^{(\ell-1)} = \sigma_{\ell-1}\left(\mathbf{W}^{(\ell-2)}X^{(\ell-2)} + \mathbf{b}^{(\ell-2)}\right) \tag{3.137}$$
$$\hat{Y} = \sigma_\ell(X^{(\ell-1)}) \text{ where ,} \tag{3.138}$$
$$\sigma_\ell(X^{(\ell)}) = \frac{\exp(X_{ij})}{\sum_{k\in X_i}\exp(X_{ik})} \in [0,1]$$

The study of [89] utilizes ‘Weighted Cross Entropy Loss’, given in Equation 3.139, and calls the new algorithm as ‘Imbalance-XGBoost’. α parameter in the weighted cross entropy loss function is the imbalance parameter, and α greater than 1 lead to higher misclassification cost for the instances belonging to class 1. On the other hand, α less than 1 causes the loss to be dominated by the falsely predicted instances of class 0.

$$\mathcal{L}_{WCE}(Y, h(Y,X), \alpha) = -\sum_{i=1}^{m} \alpha Y_i \log(\hat{Y}_i) + (1-Y_i)\log(1-\hat{Y}_i) \tag{3.139}$$

ANN algorithms for binary classification can be specialized as instance-specific cost-sensitive with changing α in $\mathcal{L}_{WCE}$ with the new instance-specific cost parameters,

C_1, C_2 and C_3. The cost parameters modify the instance-specific errors as illustrated in Figure 3.16.



Hidden Layers
Input Layer
Output Layer: $\hat{Y} = Softmax(X_1^{(4)})$
$\hat{Y}_1$: $P(Y = 1 \mid X_1)$
Loss for X_1, $\mathcal{L}(Y_1, \hat{Y}_1)$:
$C^{X_1} \times [Y_1 \log(\hat{Y}_1) + (1 - Y_1)\log(1 - \hat{Y}_1)]$
C^{X_1} is C_1 or C_2 or C_3 of X_1
$\hat{Y}_2$: $P(Y = 0 \mid X_1)$

Figure 3.16: Multilayer perceptron feedforward ANN

The new cost-sensitive loss cross entropy loss function, $\mathcal{L}_{CS-CEL}$, for ANN classification algorithm design, $\hat{Y} = h_{ANN}(.)$, is:

$$h_{ANN}(.) = h_{ANN}(Y, X, W_1, W_2, ..., W_\ell, \sigma_1(.), \sigma_2(.), ..., \sigma_\ell(.)) \quad (3.140)$$

$$\mathcal{L}_{CS\text{-}CE}(Y, h_{ANN}(.), C) = -\sum_{i=1}^{m} C_i(Y_i \log(\hat{Y_i}) + (1 - Y_i)\log(1 - \hat{Y_i})). \quad (3.141)$$

which uses instance-specific C, and the algorithm is called as RCS-ANN1 if C=C_1, RCS-ANN2 if C=C_2, and RCS-ANN3 if C=C_3. The weight update takes place with the gradient descent of the loss function and it is computed as:

$$\nabla\mathcal{L}_{CS\text{-}CE}(Y, h_{ANN}(.), C) = \begin{bmatrix} \frac{\partial\mathcal{L}_{CS-CE}(Y,h_{ANN}(.),C)}{\partial h_{ANN}(X)} \times \frac{\partial h_{ANN}(X)}{W_1} \\ \frac{\partial\mathcal{L}_{CS-CE}(Y,h_{ANN}(.),C)}{\partial h_{ANN}(X)} \times \frac{\partial h_{ANN}(X)}{W_2} \\ \vdots \\ \frac{\partial\mathcal{L}_{CS-CE}(Y,h_{ANN}(.),C)}{\partial h_{ANN}(X)} \times \frac{\partial h_{ANN}(X)}{W_\ell} \end{bmatrix} \quad (3.142)$$

$$\begin{bmatrix} W_{1_t} \\ W_{2_t} \\ \vdots \\ W_{\ell_t} \end{bmatrix} = \begin{bmatrix} W_{1_{t-1}} \\ W_{2_{t-1}} \\ \vdots \\ W_{\ell_{t-1}} \end{bmatrix} - \delta \times \nabla\mathcal{L}_{CS\text{-}CE}(Y, h_{ANN}(.), C) \quad (3.143)$$

Focal loss function has also been extensively used in classifications of imbalanced data [50, 37, 51, 99, 66, 63, 53]. ANN models can also be modified as cost-sensitive algorithms with the use of focal loss in the final classification layer. In addition to re-weighting cross-entropy loss, the ANN classification algorithm, $\hat{Y} = h_{ANN}(.)$, is also transformed into cost-sensitive by re-weighting the focal loss, L_{CS-F}, with the use of instance-specific cost parameters, $C = C_1$ and $C = C_3$.

$$h_{ANN}(.) = h_{ANN}(Y, X, W_1, W_2, ..., W_\ell, \sigma_1(.), \sigma_2(.), ..., \sigma_\ell(.)) \tag{3.144}$$

$$\mathcal{L}_{CS-F}(Y, h_{ANN}(.), C, \gamma_1, \gamma_2) = -\sum_{i=1}^{m} C_i \Big[Y_i(1 - \hat{Y}_i)^{\gamma_1} \log(\hat{Y}_i) + \\ (1 - Y_i)\hat{Y}_i^{\gamma_2} \log(1 - \hat{Y}_i) \Big] \tag{3.145}$$

The update of the weights with the gradients of the focal loss function is expressed as:

$$\nabla \mathcal{L}_{CS-F}(Y, h_{ANN}(.), C) = \begin{bmatrix} \frac{\partial \mathcal{L}_{CS-F}(Y, h_{ANN}(.), C)}{\partial h_{ANN}(X)} \times \frac{\partial h_{ANN}(X)}{W_1} \\ \frac{\partial \mathcal{L}_{CS-F}(Y, h_{ANN}(.), C)}{\partial h_{ANN}(X)} \times \frac{\partial h_{ANN}(X)}{W_2} \\ \vdots \\ \frac{\partial \mathcal{L}_{CS-F}(Y, h_{ANN}(.), C)}{\partial h_{ANN}(X)} \times \frac{\partial h_{ANN}(X)}{W_\ell} \end{bmatrix} \tag{3.146}$$

$$\begin{bmatrix} W_{1_t} \\ W_{2_t} \\ \vdots \\ W_{\ell_t} \end{bmatrix} = \begin{bmatrix} W_{1_{t-1}} \\ W_{2_{t-1}} \\ \vdots \\ W_{\ell_{t-1}} \end{bmatrix} - \delta \times \nabla \mathcal{L}_{CS-F}(Y, h_{ANN}(.), C) \tag{3.147}$$

CHAPTER 4

EMPIRICAL DATA ANALYSIS

In this chapter, the effect of new cost parameters on three common algorithms of AdaBoost, XGBoost, and ANN are empirically tested by using eight credit data sets. These data sets are selected due to their certain diverse characteristics such as different features, loan issue years, and scales for the monetary attributes such as the credit amount and the income of the borrower due to the differences in the countries of the lenders and the loan issue years. The countries of the borrowers are not reported in all data sets but some data sets report the states of the borrower as a location and some of them have no information about where the borrower is located. The monetary units for credit amounts are not specifically defined in the data description files. The diversity of information in the datasets and their public availability is the reason why these datasets were chosen for empirical purposes. These data sets are summarized in Table 4.1 with information on their sizes, feature numbers, class ratio, sub-data set sizes, and the number of simulations for LR, GNN, and GAT models. It is also important to note that the class distributions, the ratio of non-default instances to default instances, are kept the same for the train and the test data set in all classifications.

The data sets exhibit varying imbalanced class ratios in their original forms, but some of them have been further imbalanced without distorting their original distributions to investigate how the impact of the proposed parameters changes as the imbalance becomes more severe. The deleted default instances are selected with random sampling and they are all reported in data information features tables. Histograms of the credit amounts of non-defaults and defaults are also graphed except the Data Set 5: Freddie Mac Single Family Loan Data in which loan amounts are not reported. The

Table 4.1: Data Sets

Data Sets	Data Size[1]	Number of Features[2]	Class Ratio	Sample Size[3]/ Num. of LRs[4]	Sample Size[3]/Num. of GNNs-GATs[4]
Home Credit Risk D. Data	307,217	30	11.4:1	34,726/3,000	3,000/1,000
Risky Loans Data	231,285	13	8.8:1	33,060/3,000	3,000/1,000
Irish Loan Data$_{Y=2013}$	124,381	12	15.0:1	10,888/5,000	4,400/1,000
Irish Loan Data$_{Y=2014}$	235,626	12	9.5:1	31,550/5,000	3,000/1,000
Irish Loan Data$_{Y=2015}$	421,092	11	33.8:1	16,936/3,000	3,000/1,000
Freddie Mac S. F. Loans	500,137	12	26.8:1	25,188/4,000	3,000/1,000
SBA N. Loans[5]$_{Y=2004,2005,2006}$	164,021	8	40.0:1	5,602/10,000	3,600/1,000
SBA N. Loans[5]$_{Y=2002,2003}$	95,932	8	50.0:1	2,634/10,000	2,634/1,000

[1]Source: Kaggle.com.
[1]The size of the data used might change compared to the original data size due to reducing the default instances to increase the class imbalance.
[2]Number of features found to be explanatory and used in the classification models
[3]The data size of the subsets generated with sampling.
[4]The number of estimated LR, GNN, and GAT models for the estimation of default probabilities.
[5]A loan data set served by U.S. Small Business Administration.

correlation coefficients between the numerical variables of the features are presented in Appendix A tables A.1-A.8.

The optimization of the hyperparameters in ANN models is conducted with the choice of optimal hidden layer sizes, activation functions, loss functions, training functions, iterations, and proposed m parameter. The optimal parameters are selected with the main goal of maximum the G$_{mean}$ values with the maximum $Sensitivity$ and *Specificity* values in test data. The optimal layer sizes, iteration numbers, and m parameters are observed to change in all data sets but $sigmoid$ transfer function, cross-entropy, and focal loss functions are the common parameters for the optimal models. Moreover, resilient backpropagation is found to be the optimal function for network training to update weight and bias values among the other functions of quasi-Newton backpropagation, conjugate gradient backpropagation with Powell-Beale restarts, conjugate gradient backpropagation with Fletcher-Reeves updates, conjugate gradient backpropagation with Polak-Ribiére update, gradient descent with adaptive learning rate backpropagation, gradient descent with momentum backpropagation, gradient descent with momentum and adaptive learning rate backpropagation, Levenberg-Marquardt backpropagation, one-step, secant backpropagation, and scaled conjugate gradient backpropagation.

The optimal hyper-parameters of the boosting algorithms are searched through an expanded grid search of the predetermined ranges for the learning rates, depth from a root to leaf, the sub-sample ratio of columns, maximum delta value, number of estimators, and the proposed new parameters of m. The objective is to reach maximum G_{mean} with maximum *Sensitivity* and *Specificity* values in test data predictions in all data sets. The optimized hyper-parameters of all models are reported after the data information tables. Moreover, detailed information about the sub-dataset simulations with the models of LR, GNN, and GAT models are reported in the first sections of the tables reporting the performances of the algorithms.

To evaluate and compare the success of the classification algorithms, the medians of the aggregated predictions of model simulations of LR, GNNs, and GATs are reported as well as cost-insensitive algorithms such as AdaBoost, XGBoost, ANN, CatBoost, and LightGBM. In addition, the prediction results of boosting algorithms that are specifically designed for imbalanced data, such as SMOTEBoost, RUSBoost, and XGBoost scaled with class ratio, XGBoost$_{scaled}$, and CatBoost scaled with class ratio, CatBoost$_{scaled}$, are presented for comparative purposes.

The cost/risk parameters in cost-sensitive algorithms are defined as C_{LR}, C_{GNN} and C_{GAT} if the default probabilities are estimated with LRs, GNNs, and GATs, respectively. In the data analysis section, the abbreviations for risk-based cost-sensitive algorithms (RCS) have also been removed to make it easier to read the algorithm comparisons and discussions. The proposed cost-sensitive algorithms also have LR, GNN, and GAT as sub-scripts to indicate which model is used in the estimation of default probabilities such as ANN2$_{LR}$, AdaB1$_{GNN}$, XGB3$_{GAT}$ and etc.

The class predictions of default probabilities of all algorithms are the classification of the threshold 0.5 for AdaBoost and XGBoost algorithms. Thus, the same threshold is applied for ANN classifications. The class predictions of the most successful algorithms are converted to probability estimates and the results are analyzed with thresholds ranging between 0.4 and 0.6. Additionally, k-50 fold cross-validations are conducted for the competing algorithms, and the medians of the thresholds that maximize G_{mean} for the validation data sets are also applied to the test data predictions, which are also reported in the classification results tables. Furthermore, k-10 fold cross-validations of the cost-sensitive XGBoost algorithms are given in Appendix B tables B.1-B.8.

The most successful ordinary algorithms are found to be XGBoost$_{scaled}$, CatBoost$_{scaled}$ in all data sets, and RUSBoost is also successful in two data sets, and they are compared with the most successful cost-sensitive algorithms which are found to be the modifications of XGBoost in all data sets. Comparative analyses are reflected in four main graphs for each cost-sensitive XGBoost for seven data sets. The first graph demonstrates the sensitivity-specificity values of successful ordinary algorithms with thresholds of 0.4-0.6, while the second one displays the *Sensitivity* and *Specificity* values for the successful cost-sensitive XGBoost algorithms. The third graph indicates how the *LGD* changes as the threshold changes for the selected prominent algorithms. The data information specifically marked on the first three graphs shows the *Sensitivity* and *Specificity* values maximizing G_{mean} with their corresponding thresholds. The final graph, on the other hand, discloses how the non-default class predictions, *Specificity* values, change as the *LGD* value increases for the thresholds of 0.4-0.6.

4.1 DATA SET 1: Home Credit Default Risk Data

Unlike the others, Home Credit Default Risk data is the only set specific to housing loans and has detailed information tables that report the bank account history of the applicants which is retrieved from kaggle.com website [2]. The number of features are deleted if they are not reported by many applicants. Six numerical features not reported by 41,519 applicants and one numerical feature unreported by 60,965 applicants, as well as 44 numerical features unreported by more than 148,000, are deleted. Thirty features are found explanatory for the models and after defining categorical variables with dummy variables, the data has numerical 67 numerical features. Table 4.2 displays the details about the size and type of the data set and the explanatory features selected for the classification algorithms.

Home Credit Default Risk Data reports the credit amounts disbursed to the applicants but does not give its monetary unit. Figure 4.1 demonstrates the histograms of the credit amounts of non-defaults and defaults and the exponentially decaying distributions are found to be expressed best by Gamma($\alpha = 1.8$, $\beta = 0.05$) with $M = 1$ for the rejection sampling.

Table 4.2: Home Credit Default Risk Data

Data Information				
Original Data File Name: Application Train				
Original Data Size	Data Size After the Cleansing	Number of Categorical Variables	Number of Numerical Variables	Class Ratio
307,511	307,217	16	104	11.4:1
30 features found explanatory for the models, they are:				
Credit Amount - Monthly Income Amount - Age - Days After Credit ID Given - Work Phone Region Rating Client - External Source$_2$ - Social Circle$_{60}$ Days Last Phone Change - Flag Document$_3$ Flag Document$_5$ - Flag Document$_6$ - Flag Document$_9$ - Flag Document$_{13}$ - Flag Document$_{15}$ Flag Document$_{16}$ - Flag Document$_{18}$ - Contract Type-Gender - Flag Own Car - Income Type Family Status - Occupation Type - Emergency State - Interest Rate - Term Goods Price/Credit Amount - Credit Card Debt/Monthly Income - Days Employed $\times$ Age Interest Rate^2 $\times$Term				

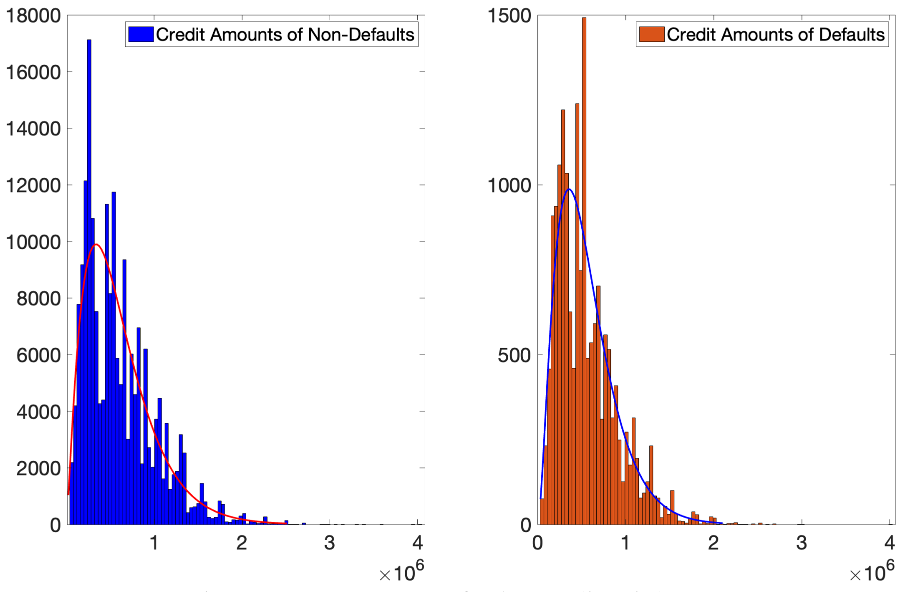


Figure 4.1: Home Default Credit Risk Data
Histograms of Credit Amounts

Tables 4.3-4.5 present the optimized hyperparameters of ANN, AdaBoost, and XGBoost with the target of maximum G_{mean} values in the test data. The common characteristic which can be a comparison base is the iteration number which is the smallest in cost-sensitive XGBoost algorithms generated with LR models. Also, the optimal value of m is found to be as $m > 1$ for C_1 and C_3, while $m < 1$ for C_2 for all cost-sensitive algorithms.

Table 4.3: Home Credit Default Risk Data: ANN Optimized Hyperparameters

Activation Functions:	Sigmoid
Learning Rate:	0.001 and 0.01 for ANN with Focal Loss and Cross Entropy Loss, respectively.
Number of Hidden Layers:	32 16 8 2
Training Function:	Resilient Back Propagation
Increment to weight change:	1.2
Decrement to weight change:	0.5
Initial weight change:	0.07
Maximum weight change:	50.0
Classification Threshold	0.5

	Focal Loss Function					Cross Entropy Loss Function		
	Iterations	m	α	γ_1	γ_2	Iterations	m	thr
ANN	750	-	1	1	1	1000	-	-
RCS-ANN1$_{LR}$	368	1.7	C_1	0.1	2	1000	1.82	-
RCS-ANN2$_{LR}$	-	-	-	-	-	1000	0.7	0.42
RCS-ANN3$_{LR}$	350	1.7	C_3	0.1	2	1000	1.83	-
RCS-ANN1$_{GNN}$	880	1.48	C_1	0.1	2	1000	1.38	-
RCS-ANN2$_{GNN}$	-	-	-	-	-	1000	0.55	0.45
RCS-ANN3$_{GNN}$	800	1.7	C_3	0.1	2	1000	1.8	-
RCS-ANN1$_{GAT}$	900	1.48	C_1	0.1	2	1000	1.4	-
RCS-ANN2$_{GAT}$	-	-	-	-	-	1000	0.6	0.43
RCS-ANN3$_{GAT}$	823	1.72	C_3	0.1	2	1000	1.75	-

Table 4.4: Home Credit Default Risk Data: AdaBoost Optimized Hyperparameters

	Iterations	m	thr
RCS-AdaB1$_{LR}$	1000	1.75	-
RCS-AdaB2$_{LR}$	1000	0.22	0.42
RCS-AdaB1$_{GNN}$	1000	1.37	-
RCS-AdaB2$_{GNN}$	1000	0.13	0.45
RCS-AdaB1$_{GAT}$	1000	1.4	-
RCS-AdaB2$_{GAT}$	1000	0.12	0.43

Table 4.5: Home Credit Default Risk Data: XGBOOST Optimized Hyperparameters

Evaluation Metric: Log Loss/Cross Entropy Loss Function.

	Learning Rate:	Depth from a Root to Leaf	Sub-sample Ratio of Columns	Maximum Delta	Number of Estimators	Scale Positive Weight	m	$thrs$
XGBoost	0.05	6	1	1	1000	-	-	-
CatBoost	0.05	6	-	-	400	-	-	-
LightGBM	0.05	-	1	0	800	-	-	-
XGBoost$_{Scaled}$	0.05	6	1	1	300	11.4	-	-
CatBoost$_{Scaled}$	0.05	6	-	-	400	11.4	-	-
RCS-XGB1$_{LR}$	0.1	7	0.6	0	91	C_1	1.75	-
RCS-XGB2$_{LR}$	0.1	10	0.5	3	55	C_2	0.78	0.42
RCS-XGB3$_{LR}$	0.1	7	0.6	1	91	C_3	1.8	-
RCS-XGB1$_{GNN}$	0.05	4	1	3	600	C_1	1.35	-
RCS-XGB2$_{GNN}$	0.025	10	0.4	3	188	C_2	0.55	0.45
RCS-XGB3$_{GNN}$	0.025	6	0.4	6	613	C_3	1.7	-
RCS-XGB1$_{GAT}$	0.1	5	0.4	3	302	C_1	1.4	-
RCS-XGB2$_{GAT}$	0.025	10	0.3	2	188	C_2	0.52	0.43
RCS-XGB3$_{GAT}$	0.025	6	0.4	5	603	C_3	1.7	-

	Sampling Strategy	Random State	k Neighbors	Number of Estimators
SMOTHEBoost	Only Minority	$np.random$ generator of Python	5	1000
RUSBoost	Only Majority Class	$np.random$ generator of Python	-	1000

The algorithm classification results for the train and test data are reported in tables 4.6-4.8. Table 4.6 informs about the train and test data sizes, class ratio, and simulations with LR, GNN, and GAT models and also it presents the cost-insensitive algorithm results and cost-sensitive ANN model results. The cost-insensitive algorithms have higher *Specificity* values but lower *Specificity* values, therefore, *LGD* values are not reported for these ordinary algorithms. ANN3$_{LR}$ and ANN1$_{GAT}$ algorithms estimated with cross-entropy loss function are observed to have lower *LGD* values when compared with all cost-sensitive ANNs.

Table 4.7 displays the classification performances of boosting algorithms of SMOTE-Boost, RUSBoost, XGBoost$_{scaled}$, CatBoost$_{scaled}$ and cost-sensitive boosting algorithms with the first results of the algorithms classified with the threshold of 0.5. The prediction performances of these boosting algorithms are found to be superior to the other algorithms reported in Table 4.6 due to lower losses given default values in test data.

It is shown that borrower (instance)-specific risk-based cost-sensitive modifications of XGBoost have relatively lower *LGD* values for test data, but the Specificity values of XGBoost$_{scaled}$ and CatBoost$_{scaled}$ are the highest allowing a threshold move for improving *Sensitivity*. Therefore, the prediction results for *Sensitivity*-*Specificity* and *LGD* values are analyzed with the thresholds of 0.4-0.6 where the maximums of G_{mean} values are also analyzed in the test data. A detailed comparative analysis of the results displayed with Figures 4.2, 4.3 and 4.4 indicates that XGB1$_{LR}$ and XGB3$_{LR}$ are the best-performing algorithms with their higher G_{mean} values and lower *LGD* values given in the third graph of Figure 4.2. Moreover, *LGD* values for given *Specificity* values as shown with the fourth graph in Figure 4.2 are also relatively lower when compared with the other successful algorithms of XGBoost$_{scaled}$ and CatBoost$_{scaled}$. The monetary unit is not reported in the data set but given the maximum G_{mean} values in test data, XGB3$_{LR}$ saves 5.9 million loss with G_{mean}/*LGD*: $92.48/257.0$ million when compared with the CatBoost$_{scaled}$ algorithm with G_{mean}/*LGD*: $92.34/262.9$ million.

Table 4.6: Home Credit Default Risk Data
Performances of Algorithms

Train Data Size=$215,052$, Test Data Size=$92,165$, Train Data/Test Data = $7/3$
Non-defaults/Defaults=11.4
Numbers of LR/GNN/GAT Models with Sub-Data Sets = $3,000/1,000/1,000$
Sub-data Sizes for LR/GNN/GAT Models with Sub-Data Sets=$34,726/3,000/3,000$
LR: Models with $R^2_{ordinary} >= 0.450$, $R^2_{adjusted} >= 0.450$, $R^2_{AdjustedGeneralized} >= 0.530$
LR: $thrs = 0.42$ for C_2
Levels of Node Embeddings in GNN/GAT: 3/1
β_{RAW} used in C_2 estimated with LR/GNN/GAT: $4.536/4.202/4.275$
GNN: Iterations= 120, $knn = 5$, $lR = 0.01$ $thrs = 0.45$ for C_2
GAT: Iterations=100, $knn = 25$, $K = 1$, $lR = 0.02$, $thrs = 0.43$ for C_2
Threshold for ANNs: 0.5

Cost-Insensitive Algorithms

	TRAIN DATA			TEST DATA		
Algorithms	Sens.	Spec.	G_{mean}	Sens.	Spec.	G_{mean}
LR	80.1	80.1	80.1	79.6	80.0	79.8
GNN	84.0	83.1	83.5	78.0	84.6	81.2
GAT	82.7	82.8	82.8	81.0	81.2	81.1
ANN$CrossEntropyLoss$	79.0	99.9	88.9	77.3	99.8	87.8
ANN$FocalLoss$	47.0	99.0	68.2	47.7	99.4	68.9
AdaBoost	75.8	99.9	87.1	75.8	99.9	87.0
XGBoost	87.6	100	93.6	79.6	100	89.1
CatBoost	79.7	100	89.3	79.2	100	89.0
LightGBM	85.5	100	92.5	79.5	100	89.1

Cost-Sensitive ANNs with Focal Loss Function

	TRAIN DATA			TEST DATA					
Algorithms	Sens.	Spec.	G_{mean}	Sens.	Spec.	G_{mean}	AUC	$MC^{x10^6}_{Y=1}$	Loss Given Default$^{x10^6}$
RCS-ANN1$_{LR}$	80.7	76.9	78.8	80.2	77.0	778.6	87.2	62.619	904.004
RCS-ANN2$_{LR}$	81.0	81.4	81.2	80.1	81.4	80.8	89.4	62.829	890.009
RCS-ANN1$_{GNN}$	82.3	82.7	82.5	81.3	82.7	82.0	90.7	60.064	871.144
RCS-ANN2$_{GNN}$	81.5	80.8	81.2	80.7	80.8	80.8	89.5	59.406	859.424
RCS-ANN1$_{GAT}$	81.5	82.8	82.2	80.4	82.8	81.6	90.1	64.059	909.052
RCS-ANN2$_{GAT}$	82.4	83.2	82.8	81.4	83.1	82.3	91.3	55.439	823.943

Cost-Sensitive ANNs with Cross Entropy Loss Function

	TRAIN DATA			TEST DATA					
Algorithms	Sens.	Spec.	G_{mean}	Sens.	Spec.	G_{mean}	AUC	$MC^{x10^6}_{Y=1}$	Loss Given Default$^{x10^6}$
RCS-ANN1$_{LR}$	91.4	87.6	89.5	89.0	87.5	88.2	96.4	12.458	390.208
RCS-ANN2$_{LR}$	91.3	91.0	91.1	88.1	90.7	89.4	97.1	13.635	400.272
RCS-ANN3$_{LR}$	93.4	89.6	91.5	90.6	89.5	90.0	97.4	8.843	317.713
RCS-ANN1$_{GNN}$	90.7	89.8	90.2	88.1	89.5	88.8	96.5	14.764	430.121
RCS-ANN2$_{GNN}$	91.0	89.6	90.3	88.4	89.5	88.9	96.7	10.292	392.460
RCS-ANN3$_{GNN}$	92.5	89.5	91.0	89.6	89.3	89.5	97.0	10.061	372.298
RCS-ANN1$_{GAT}$	91.5	89.9	90.7	88.6	89.6	89.1	96.9	12.952	294.211
RCS-ANN2$_{GAT}$	92.0	89.3	90.7	89.7	89.3	89.5	97.2	9.259	361.372
RCS-ANN3$_{GAT}$	92.5	89.4	90.9	89.5	89.3	89.4	97.0	10.355	367.144

Table 4.7: Home Credit Default Risk Data
Performances of Algorithms

Boosting Algorithms for Imbalanced Data

	TRAIN DATA			TEST DATA					
	Sens.	Spec.	G_{mean}	Sens.	Spec.	G_{mean}	AUC	$MC^{x10^6}_{Y=1}$	Loss Given Default$^{x10^6}$
SMOTEBoost	78.4	99.4	88.3	78.4	95.0	92.3	98.0	76.828	703.047
RUSBoost	91.1	93.3	92.2	89.9	93.3	91.6	98.2	17.321	342.016
XGBoost$_{scaled}$	92.6	95.5	92.2	89.3	95.3	92.2	98.5	13.369	366.439
CatBoost$_{scaled}$	92.2	95.3	93.7	89.6	95.1	92.3	98.5	12.694	353.570
RCS-Ada1$_{LR}$	90.9	90.4	90.6	90.2	90.3	90.3	97.2	10.839	359.906
RCS-Ada2$_{LR}$	91.0	90.4	90.7	90.3	90.3	90.3	97.3	9.789	382.748
RCS-AdaB1$_{GNN}$	91.3	91.3	91.3	90.5	91.2	90.9	97.8	10.917	348.958
RCS-AdaB2$_{GNN}$	91.6	90.8	91.2	90.9	90.8	90.8	97.8	9.621	330.008
RCS-Ada1$_{GAT}$	91.3	91.4	91.3	90.7	91.2	91.0	97.9	10.676	346.629
RCS-Ada2$_{GAT}$	91.5	90.9	91.2	90.7	90.9	90.8	97.9	10.080	338.941
RCS-XGB1$_{LR}$	94.8	93.1	93.9	92.0	92.9	92.5	98.5	7.999	272.514
RCS-XGB2$_{LR}$	96.3	92.3	94.2	91.9	91.7	91.8	98.3	6.236	295.471
RCS-XGB3$_{LR}$	94.9	92.9	93.9	92.2	92.8	92.5	98.5	7.470	265.944
RCS-XGB1$_{GNN}$	93.3	93.9	93.6	90.8	93.7	92.2	98.4	10.645	334.653
RCS-XGB2$_{GNN}$	96.0	93.8	94.9	91.3	93.4	92.4	98.4	8.244	294.549
RCS-XGB3$_{GNN}$	95.2	94.4	94.8	90.7	94.3	92.5	98.5	10.033	321.553
RCS-XGB1$_{GAT}$	94.6	94.0	94.3	90.6	93.7	92.1	98.4	10.659	341.157
RCS-XGB2$_{GAT}$	95.5	94.2	94.9	90.8	93.7	92.3	98.3	9.236	307.336
RCS-XGB3$_{GAT}$	94.2	94.0	94.1	91.1	93.9	92.5	98.5	9.940	317.118

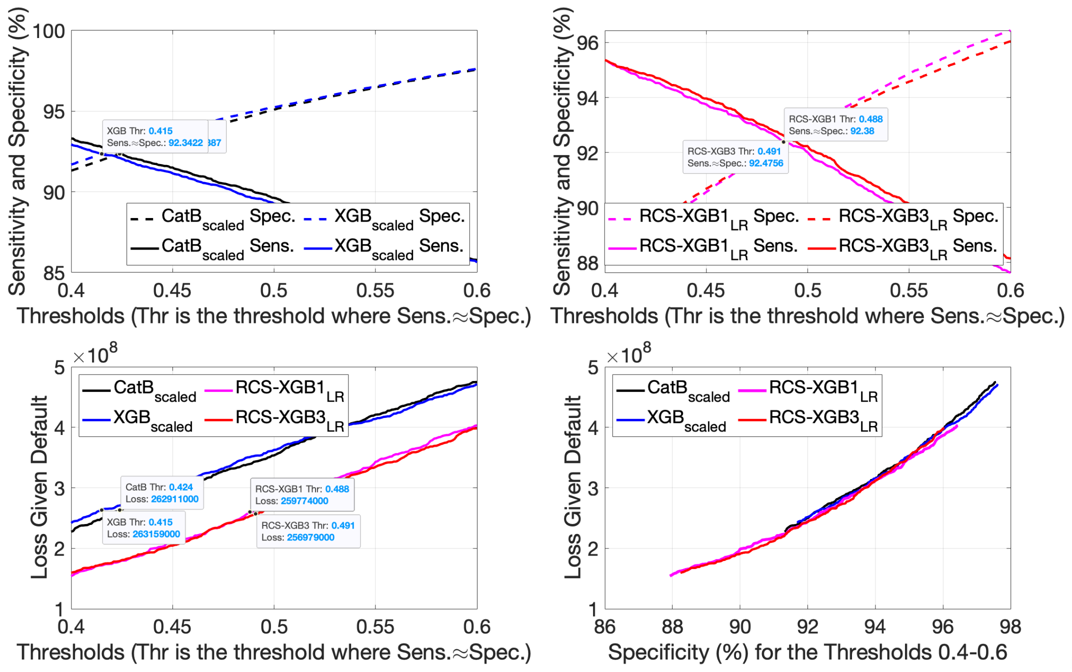


Figure 4.2: Cost Parameters are Estimated with LRs
Sensitivity-Specificity and *LGD* with Different Thresholds in Test Data

The cost-sensitive parameters estimated with GNN and GAT models are not successful as the ones estimated with the LR. Comparing with CatBoost$_{scaled}$ which reaches its maximum G_{mean} at 92.3 with 262.9 million *LGD*, XGB2$_{GNN}$ saves 715 thousand with G_{mean}/LGD: $92.25/262.2$ million and XGB2$_{GAT}$ saves only 106 thousand with G_{mean}/LGD: $92.18/262.8$ million. The reason is that the discrepancies in train and test data predictions are minimum in the LR models, and the LR does not cause an over-fitting problem for any of the sub-data models as the aggregated median values of LR also prove these facts. Despite the several different designs and node embedding levels for GNN and GAT models, their cost-sensitive parameter estimates could not be improved further. The empirical findings indicate that the success of the parameter estimates increases if *Sensitivity-Specificity* values in test data exceed $85-85$ % in sub-data model simulations.

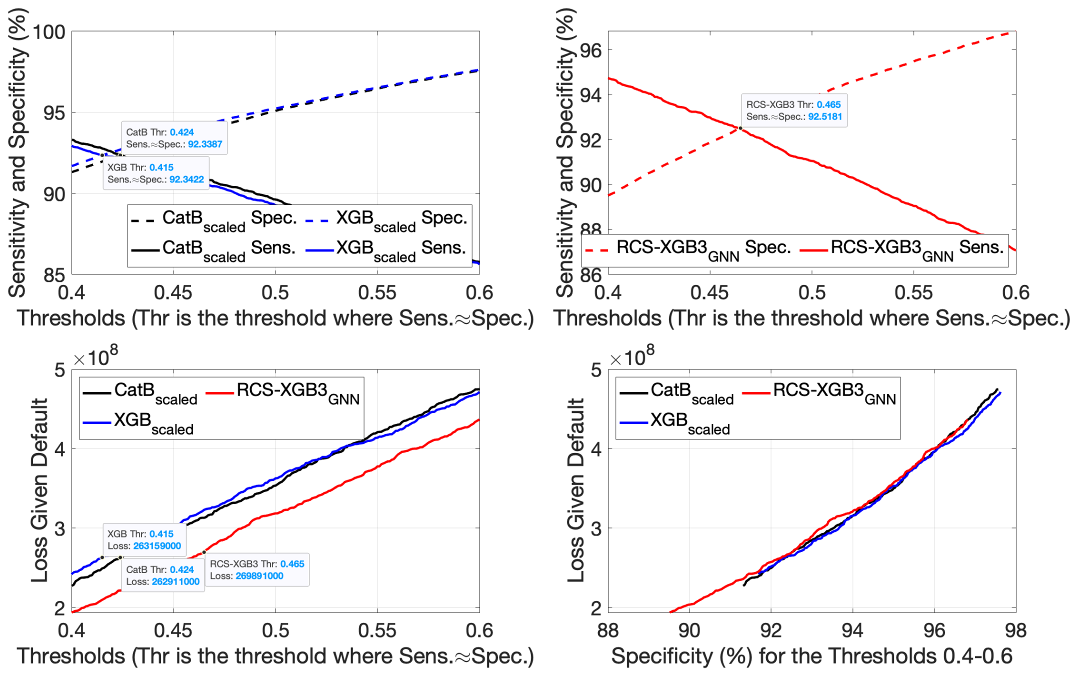


Figure 4.3: Cost Parameters are Estimated with GNNs
Sensitivity-Specificity and *LGD* with Different Thresholds in Test Data

A threshold decision for class labeling might be given for a fixed *Specificity* and the corresponding *LGD* might be the secondary importance or for a fixed *Sensitivity* with the maximum *LGD*. Considering almost balanced correct classification percentages for both classes, G_{mean} maximization thresholds of validation data of k-50 fold are computed, and the medians of the thresholds are applied for the test data classification. Table 4.8 presents how the balanced classification thresholds of validation data

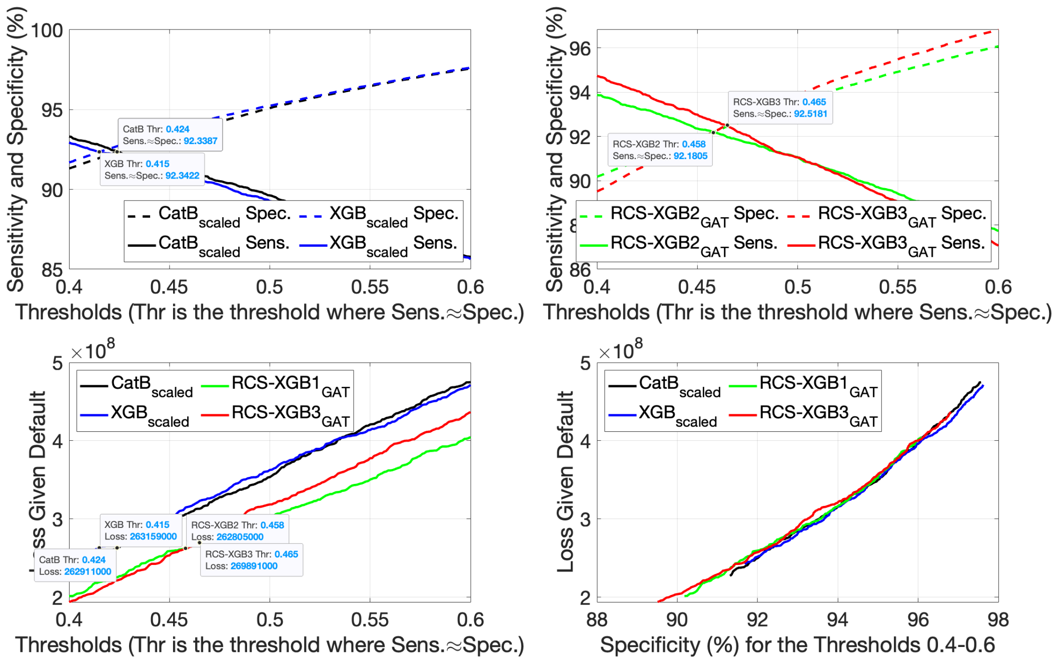


Figure 4.4: Cost Parameters are Estimated with GATs
Sensitivity-Specificity and *LGD* with Different Thresholds in Test Data

result in *Sensitivity-Specfcity* and *LGD* for the test data.

Table 4.8: Home Credit Default Risk Data
Competing Performances of Algorithms After a Threshold Adjustment

	TRAIN DATA			TEST DATA					
Algorithms$_{Weights^1,thr^2}$	Sens.	Spec.	G$_{mean}$	Sens.	Spec.	G$_{mean}$	AUC	MC$^{x10^6}_{Y=1}$	Loss Given Default$^{x10^6}$
XGBoost$_{ClassRatio,thr=0.422}$	95.4	92.7	94.1	92.2	92.6	92.4	98.5	6.130	268.489
CatBoost$_{ClassRatio,thr=0.425}$	94.9	92.5	93.7	92.3	92.4	92.4	98.5	6.206	263.844
RCS-XGB1$_{LR,thr=0.492}$	95.1	92.7	93.9	92.3	92.6	92.4	98.5	7.369	263.430
RCS-XGB2$_{LR,thr=0.498}$	96.3	92.2	94.2	91.9	91.7	91.8	98.3	6.162	294.405
RCS-XGB3$_{LR,thr=0.494}$	95.1	92.6	93.9	92.4	92.6	92.5	98.5	7.044	259.416
RCS-XGB1$_{GNN,thr=0.461}$	94.8	92.2	93.5	92.2	92.1	92.1	98.4	7.407	291.060
RCS-XGB2$_{GNN,thr=0.465}$	96.5	92.7	94.6	92.2	92.3	92.3	98.4	6.189	262.460
RCS-XGB3$_{GNN,thr=0.446}$	96.7	92.4	94.5	92.4	92.3	92.3	98.5	6.551	272.286
RCS-XGB1$_{GAT,thr=0.454}$	96.0	92.1	94.0	92.1	91.9	92.0	98.4	7.248	296.761
RCS-XGB2$_{GAT,thr=0.453}$	96.6	92.6	94.6	92.2	92.1	92.2	98.3	6.175	261.653
RCS-XGB3$_{GAT,thr=0.460}$	95.6	92.4	94.0	92.7	92.3	92.5	98.5	6.204	261.589

$^1P(D_{LR})$, $P(D_{GNN})$ and $P(D_{GAT})$ in weights are the default probabilities of the instances estimated with LR, GNNs, and GATs, respectively.
^{2}Optimal Threshold: It is the median of G_{mean} maximizing thresholds of validation data sets predicted in k-50 fold cross validations.

4.2 DATA SET 2: Risky Loans Data

Risky Loans Data contains 74 features of which 2 variables are identity numbers of applicants, 5 variables are in date format and 1 is the status of the loan. This data set consists of 887,379 instances with the non-defaults, the default, and current loans. Current loans are deleted in the data set and non-defaults (Fully Paid) and defaults (Default + Charged Off + Does not meet the credit policy. Status: Charged Off) are chosen for the study in the data set. the source of the data is kaggle.com [4]. The ratio of non-defaults to defaults is about 4.4:1. In fact, to make the data set more skewed in class distributions, half of the default instances are deleted so that the class ratio becomes 8.80, as the main goal of this study is to investigate the classification in imbalance data sets.

The reason for the significant decrease in the feature size is that some of the features are repeated in different formats in the data set such as categorical variables are also reported in dummy variables format. Thirteen features are found explanatory for the models of which 2 categorical features are defined with dummy variables and 11 numerical features generate 33 numerical variables for the classification of this data set. Table 4.9 presents the size and feature types of the data set and the explanatory features selected for the classification algorithms.

Table 4.9: Risky Loans Data

Data Information				
Original Data File Name: Loan				
Original Data Size	Data Size After the Cleansing	Number of Categorical/ String Variables	Number of Numerical Variables	Class Ratio
254,951	231,285	9/17	40	8.8
13 features are found explanatory for the models, they are:				
Credit Amount - Term - Interest Rate The ratio of Monthly Debt to All Debt Obligations - The number of Inquiries in the Past 6 Months (except auto and mortgage) - Number of Derogatory Public Records - Interest Received to Date Late Fees Received to Date - Last Payment Amount of Credit - Purpose of Credit- Days Between the Last Credit Use and This Credit Issue Year of the Credit - Ratio of Annuity to Income				

Risky Loans Data reports the credit amounts disbursed to the applicants but does not give its monetary unit. Figure 4.5 demonstrates the histograms of the credit amounts of non-defaults and defaults and their distributions are found to be expressed best by Gamma($\alpha = 2.6$, $\beta = 0.14$) with $M = 1$ for the rejection sampling.

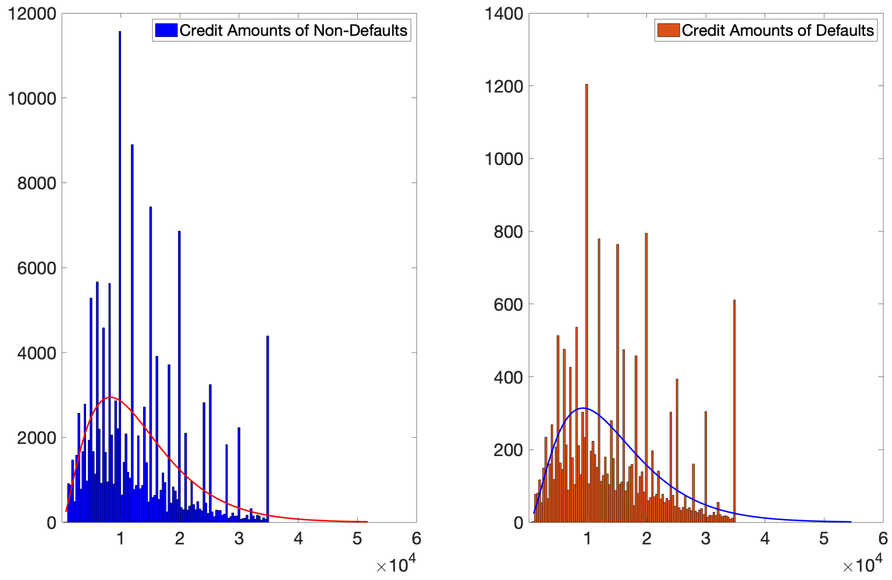


Figure 4.5: Risky Loans Data
Histograms of Credit Amounts

The optimized hyperparameters of ANN, AdaBoost, and XGBoost are reported in Tables 4.10-4.12. One of the general comparison basis is the optimal sub-sample ratios of columns which are observed to be less than for the best-performing instance-specific cost-sensitive XGBoost algorithms. The optimal delta values of these algorithms are also found to be greater than zero. Moreover, the optimal iteration numbers in cost-sensitive ANNs are observed to be relatively higher. Also, the optimal value of m for C_1, C_2 and C_3 is greater than 1 for all cost-sensitive algorithms, except RCS-ANN2$_{GAT}$.

Table 4.10: Risky Loans Data: ANN Optimized Hyperparameters

Activation Functions:	Sigmoid
Learning Rate:	0.001 and 0.01 for ANN with Focal Loss and Cross Entropy Loss, respectively.
Number of Hidden Layers:	32 16 8
Training Function:	Resilient Back Propagation
Increment to weight change:	1.2
Decrement to weight change:	0.5
Initial weight change:	0.07
Maximum weight change:	50.0
Classification Threshold	0.5

	Focal Loss Function					**Cross Entropy Loss Function**		
	Iterations	m	α	γ_1	γ_2	Iterations	m	$thrs$
ANN	700	-	1	1	1	1000	-	-
RCS-ANN1$_{LR}$	1800	2.4	C_1	0.1	2	1000	2	-
RCS-ANN2$_{LR}$	-	-	-	-	-	1000	1.24	0.52
RCS-ANN3$_{LR}$	1400	2.85	C_3	0.1	2	2500	2.55	-
RCS-ANN1$_{GNN}$	1000	1.35	C_1	0.1	2	1000	1.4	-
RCS-ANN2$_{GNN}$	-	-	-	-	-	1000	1.8	0.462
RCS-ANN3$_{GNN}$	1000	1.65	C_3	0.1	2	1000	1.4	-
RCS-ANN1$_{GAT}$	1634	1.38	C_1	0.1	2	1000	1.25	-
RCS-ANN2$_{GAT}$	-	-	-	-	-	1200	0.5	0.485
RCS-ANN3$_{GAT}$	1410	1.65	C_3	0.1	2	1500	1.4	-

Table 4.11: Risky Loans Data: AdaBoost Optimized Hyperparameters

	Iterations	m	thrs
RCS-AdaB1$_{LR}$	500	2.28	-
RCS-AdaB2$_{LR}$	500	0.5	0.52
RCS-AdaB1$_{GNN}$	500	1.2	-
RCS-AdaB2$_{GNN}$	500	0.075	0.462
RCS-AdaB1$_{GAT}$	500	1.25	-
RCS-AdaB2$_{GAT}$	500	0.09	0.485

Table 4.12: Risky Loans Data: XGBOOST Optimized Hyperparameters

Evaluation Metric: Log Loss/Cross Entropy Loss Function.

	Learning Rate:	Depth from a Root to Leaf	Sub-sample Ratio of Columns	Maximum Delta	Number of Estimators	Scale Positive Jobs	m	$thrs$
XGBoost	0.1	6	1	1	1000	-	-	-
CatBoost	0.1	6	-	-	1000	-	-	-
LightGBM	0.1	6	-	-	1000	-	-	-
XGBoost$_{Scaled}$	0.025	7	0.6	3	180	8.8	-	-
CatBoost$_{Scaled}$	0.1	7	-	-	185	8.8	-	-
RCS-XGB1$_{LR}$	0.1	7	0.6	3	105	C_1	3.2	-
RCS-XGB2$_{LR}$	0.1	7	0.8	1	300	C_2	2.5	0.52
RCS-XGB3$_{LR}$	0.1	7	0.6	3	82	C_3	3.75	-
RCS-XGB1$_{GNN}$	0.05	7	0.6	2	198	C_1	1.78	-
RCS-XGB2$_{GNN}$	0.1	7	0.8	5	237	C_2	1.0	0.462
RCS-XGB3$_{GNN}$	0.05	8	0.5	2	145	C_3	2.1	-
RCS-XGB1$_{GAT}$	0.1	7	0.6	3	162	C_1	2	-
RCS-XGB2$_{GAT}$	0.1	7	0.8	1	200	C_2	1	0.485
RCS-XGB3$_{GAT}$	0.05	6	0.8	5	185	C_3	2.1	-

	Sampling Strategy	Random State	k Neighbors	Number of Estimators
SMOTHEBoost	Only Minority	$np.random$ generator of Python	5	1000
RUSBoost	Only Majority Class	$np.random$ generator of Python	-	1000

Table 4.13 presents the train and test data details such as the data sizes, class ratio, and simulations with LR, GNN, and GAT models. It also reports the cost-insensitive algorithm results and cost-sensitive ANN model results. The cost-insensitive algorithms have higher *Specificity* values but lower *Specificity* values, therefore, *LGD* values are not reported for these ordinary algorithms. ANN3$_{LR}$ and ANN2$_{GNN}$ algorithms estimated with cross-entropy loss function are observed to have lower *LGD* values when compared with all cost-sensitive ANNs.

The results of boosting algorithms which are observed to be superior to the cost-sensitive ANN algorithms are reported in Table 4.14. SMOTEBoost, RUSBoost, XGBoost$_{scaled}$, CatBoost$_{scaled}$, and cost-sensitive boosting algorithms are classified with the threshold of 0.5. The test data predictions in this table result in lower *LGD* and make these boosting algorithms superior to the other algorithms. Instance-specific risk-based cost-sensitive modifications of XGBoost have higher *G*$_{mean}$ values and relatively lower *LGD* for the test data, but the *Specificity* value of CatBoost$_{scaled}$ with 96.3 is the highest and a threshold adjustment will certainly improve the *Sensitivity*. Therefore, a threshold analysis is conducted to observe how *Sensitivity*, *Specificity*, and *LGD* values change with the thresholds of 0-4-0.6. The reason for the choice of this range for the thresholds is that the maximums of *G*$_{mean}$ values are observed in this range in k-50 fold cross-validations.

The best performing cost-sensitive algorithms giving the maximum *G*$_{mean}$ values and the lowest *LGD* values given the *Specificity* values are analyzed with figures 4.6, 4.7 and 4.8. The monetary unit is not reported in the data set, therefore, *LGD* are reported in unit values. Comparing the results with the *G*$_{mean}$/*LGD*: $96.0/3.744$ million reported by CatBoost$_{scaled}$ as the best performing algorithm among the previously developed classification algorithms, XGB3$_{LR}$ saves 119.1 thousand loss with *G*$_{mean}$/*LGD*: $96.1/3.625$ million, XGB2$_{GNN}$ reduces the loss by 160.0 thousand loss with *G*$_{mean}$/*LGD*: $96.2/3.584$ million XGB3$_{GNN}$ decreases the loss by 101.1 thousand loss with *G*$_{mean}$/*LGD*: $96.1/3.643$ million and XGB2$_{GAT}$ leads a decrease in the loss by 156.7 thousand loss with *G*$_{mean}$/*LGD*: $96.1/3.588$ million.

Table 4.13: Risky Loans Data
Performances of Algorithms

Train Data Size=$161,884$, Test Data Size= $69,401$, Train Data/Test Data = $7/3$
Non-defaults/Defaults=8.8
Numbers of LR/GNN/GAT Models with Sub-Data Sets= $3,000/1,000/1,000$
Sub-data Sizes for LR/GNN/GAT Models with Sub-Data Sets= $33,060/3,000/3,000$
Levels of Node Embeddings in GNN/GAT: 1/1
β_{RAW} used in C_2 estimated with LR/GNN/GAT: $1.857/2.805/2.849$
LR: Models with $R^2_{ordinary} >= 0.668$, $R^2_{adjusted} >= 0.668$, $R^2_{AdjustedGeneralized} >= 0.758$
LR:$thrs = 0.52$ for C_2
GNN: Iterations= 25, $knn = 5$, $lR = 0.01$, $thrs = 0.462$ for C_2
GAT: Iterations=80, $knn = 25$, $K = 1$, $lR = 0.02$, $thrs = 0.485$ for C_2
Threshold for ANNs: 0.5

Cost-Insensitive Algorithms

	TRAIN DATA			TEST DATA		
Algorithms	Sens.	Spec.	G_{mean}	Sens.	Spec.	G_{mean}
LR	90.3	90.2	90.2	90.2	90.2	90.2
GNN	89.9	90.0	89.9	89.8	88.1	89.0
GAT	90.4	89.7	90.0	89.7	88.0	88.8
ANN$CrossEntropyLoss$	83.2	97.1	89.1	81.8	96.9	89.1
ANN$FocalLoss$	72.3	97.6	84.0	71.4	97.5	83.4
AdaBoost	73.1	98.0	84.6	72.5	98.0	84.3
XGBoost	97.0	99.7	98.4	89.7	99.0	94.2
CatBoost	92.9	99.3	96.0	88.9	99.0	93.8
LightGBM	95.9	99.0	97.4	90.0	98.4	94.0

Cost-Sensitive ANNs with Focal Loss Function

	TRAIN DATA			TEST DATA					
Algorithms	Sens.	Spec.	G_{mean}	Sens.	Spec.	G_{mean}	AUC	$MC^{x10^6}_{Y=1}$	Loss Given Default$^{x10^6}$
RCS-ANN1$_{LR}$	93.2	92.9	93.0	92.5	92.8	92.6	97.3	0.149	5.153
RCS-ANN1$_{LR}$	93.6	93.8	93.7	93.3	93.8	93.5	97.6	0.118	4.846
RCS-ANN1$_{GNN}$	93.2	93.5	93.3	92.9	93.4	93.2	97.5	0.142	5.228
RCS-ANN2$_{GNN}$	93.4	93.3	93.4	93.3	93.3	93.3	97.4	0.118	4.665
RCS-ANN1$_{GAT}$	93.6	94.1	93.9	93.3	94.0	93.7	97.7	0.122	4.980
RCS-ANN2$_{GAT}$	93.8	93.8	93.8	93.6	93.7	93.7	97.6	0.100	4.409

Cost-Sensitive ANNs with Cross Entropy Loss Function

	TRAIN DATA			TEST DATA					
Algorithms	Sens.	Spec.	G_{mean}	Sens.	Spec.	G_{mean}	AUC	$MC^{x10^6}_{Y=1}$	Loss Given Default$^{x10^6}$
RCS-ANN1$_{LR}$	94.0	93.6	93.8	92.6	93.6	93.1	97.6	0.112	5.903
RCS-ANN2$_{LR}$	94.6	93.6	94.1	93.4	93.5	93.4	97.5	0.077	5.250
RCS-ANN3$_{LR}$	95.6	94.5	95.0	94.7	94.3	94.5	98.0	0.041	3.896
RCS-ANN1$_{GNN}$	95.1	93.6	94.4	94.1	93.4	93.8	97.8	0.070	4.916
RCS-ANN2$_{GNN}$	95.2	94.2	94.7	94.5	94.0	94.3	97.7	0.041	3.780
RCS-ANN3$_{GNN}$	95.1	94.5	94.8	93.9	94.4	94.1	98.0	0.063	4.557
RCS-ANN1$_{GAT}$	94.7	94.4	94.6	93.6	94.4	93.8	97.9	0.079	5.308
RCS-ANN2$_{GAT}$	94.7	94.4	94.6	93.8	94.4	94.1	97.7	0.061	4.323
RCS-ANN3$_{GAT}$	95.3	94.4	94.8	94.1	94.3	94.2	97.7	0.058	4.580

Table 4.14: Risky Loans Data: Performances of Algorithms

	Boosting Algorithms for Imbalanced Data								
	TRAIN DATA			TEST DATA					
	Sens.	Spec.	G_{mean}	Sens.	Spec.	G_{mean}	AUC	$MC_{Y=1}^{x10^6}$	Loss Given Default$^{x10^6}$
SMOTEBoost	89.9	94.8	92.3	89.2	94.9	92.0	97.7	0.495	9.199
RUSBoost	95.7	92.4	94.0	95.3	92.4	93.9	97.9	0.090	3.845
XGBoost_{scaled}	97.3	95.4	96.3	95.9	95.3	95.6	98.9	0.050	3.664
CatBoost_{scaled}	99.7	98.3	99.0	95.7	96.3	96.0	99.1	0.051	4.061
RCS-Ada1_{LR}	93.1	93.1	93.1	92.6	93.2	92.9	97./	0.148	6.582
RCS-Ada2_{LR}	93.3	93.2	93.3	93.0	93.4	93.2	97.8	0.129	6.320
RCS-AdaB1_{GNN}	93.4	93.1	93.2	93.1	93.3	93.2	97.9	0.124	6.125
RCS-AdaB2_{GNN}	93.6	92.8	93.3	93.2	93.0	93.1	97.9	0.119	5.942
RCS-Ada1_{GAT}	93.2	93.2	93.2	92.8	93.4	93.1	97.9	0.132	6.266
RCS-Ada2_{GAT}	93.7	93.0	93.4	93.3	93.2	93.2	97.9	0.115	5.701
RCS-XGB1_{LR}	97.9	95.2	96.5	96.0	94.0	95.5	98.8	0.044	3.526
RCS-XGB2_{LR}	98.7	96.2	97.4	96.5	95.7	96.1	99.2	0.026	3.466
RCS-XGB3_{LR}	98.4	96.1	97.2	96.4	95.9	96.1	99.1	0.034	3.361
RCS-XGB1_{GNN}	98.5	95.5	97.0	96.5	95.3	95.9	99.1	0.035	3.608
RCS-XGB2_{GNN}	99.0	96.8	97.9	96.3	96.0	96.2	99.2	0.028	3.441
RCS-XGB3_{GNN}	98.8	96.1	97.5	96.3	95.9	96.1	99.0	0.037	3.386
RCS-XGB1_{GAT}	99.0	96.0	97.5	96.4	95.6	96.0	99.2	0.034	3.690
RCS-XGB2_{GAT}	98.5	96.0	97.3	96.5	95.6	96.1	99.2	0.027	3.253
RCS-XGB3_{GAT}	98.0	95.5	96.7	96.4	95.4	95.9	99.1	0.035	3.378

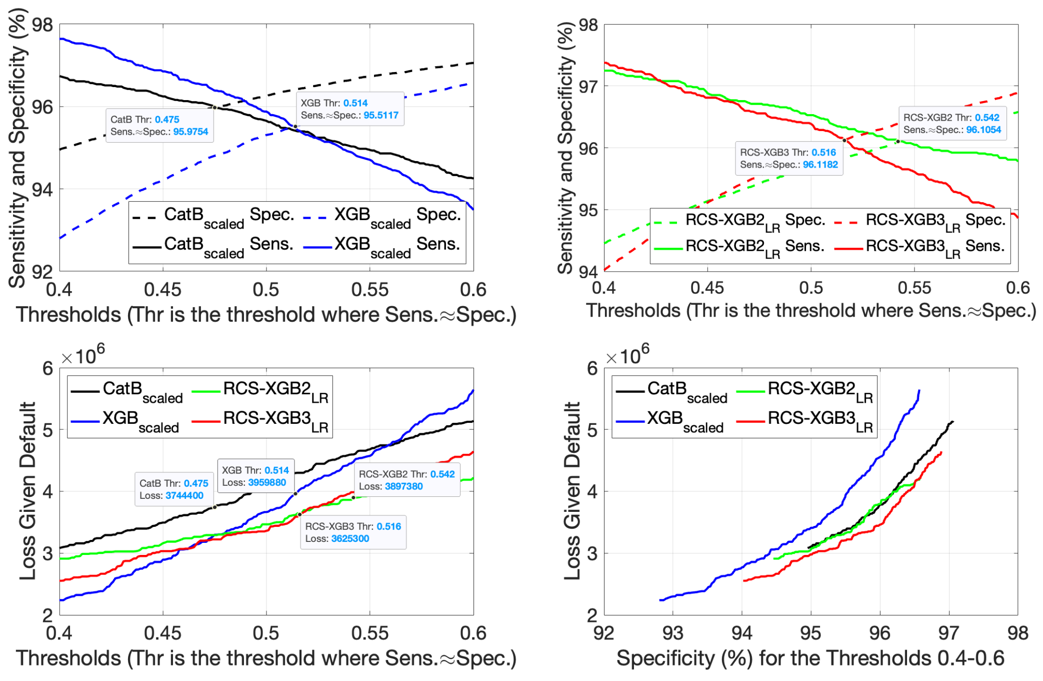


Figure 4.6: Cost Parameters are Estimated with LRs
Sensitivity-Specificity and *LGD* with Different Thresholds in Test Data

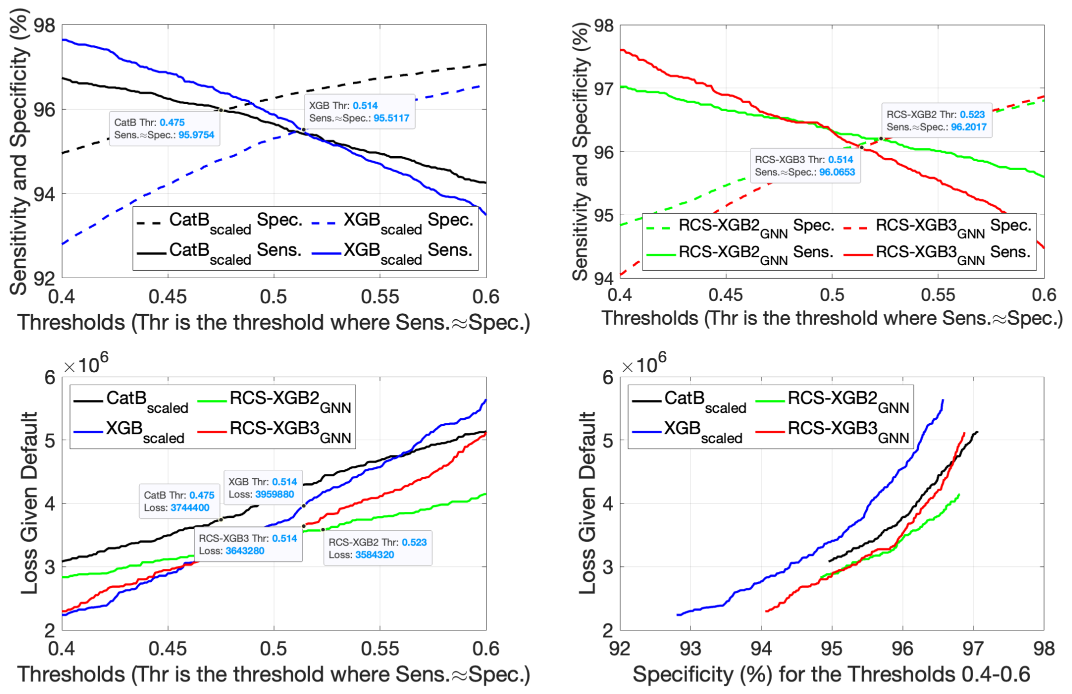


Figure 4.7: Cost Parameters are Estimated with GNNs
Sensitivity-Specificity and *LGD* with Different Thresholds in Test Data

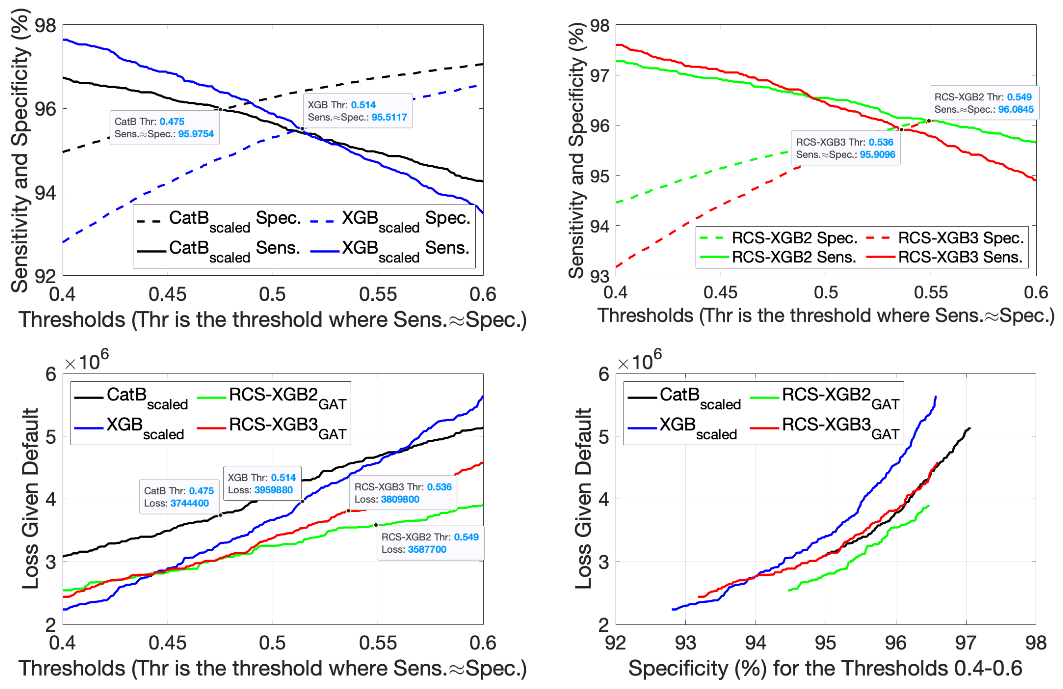


Figure 4.8: Cost Parameters are Estimated with GATs
Sensitivity-Specificity and *LGD* with Different Thresholds in Test Data

The proposed cost parameters, $C_{3_{(P(D_{LR}))}}$, $C_{2_{(P(D_{GNN}))}}$ and $C_{2_{(P(D_{GAT}))}}$, improve XG-Boost modifications with lower *LGD* values for given *Specificity* values. Considering almost balanced classification performances for both classes, a threshold adjustment is made with an optimal threshold choice determined with validation data sets. G_{mean} maximization thresholds of validation data of k-50 fold are computed, and the medians of the thresholds are applied for the test data classification. Table 4.15 presents how the G_{mean} maximizing thresholds of validation data result in test data predictions of *Sensitivity*-$Specifcity$ and *LGD* for the test data. The results indicate that the proposed cost-sensitive models with C_2 and C_3 leads lower *LGD* and $C_{2_{(P(D_{GNN}))}}$ appears as the most successful instance-specific cost parameter.

Table 4.15: Risky Loans Data
Competing Performances of Algorithms After a Threshold Adjustment

	TRAIN DATA			TEST DATA					
Algorithms$_{Weights^1, thr^2}$	Sens.	Spec.	G$_{mean}$	Sens.	Spec.	G$_{mean}$	AUC	MC$^{x10^6}_{Y=1}$	Loss Given Default$^{x10^6}$
XGBoost$_{scaled,thr=0.517}$	97.0	95.6	96.3	95.4	95.6	95.5	98.9	0.066	4.048
CatBoost$_{scaled,thr=0.477}$	99.7	98.2	99.0	96.0	96.0	96.0	99.1	0.042	3.774
RCS-XGB1$_{LR,thr=0.528}$	95.6	92.5	92.5	95.6	95.4	95.5	98.8	0.055	3.829
RCS-XGB2$_{LR,thr=0.525}$	97.6	95.7	96.7	96.3	95.9	96.1	99.1	0.046	3.749
RCS-XGB3$_{LR,thr=0.517}$	98.2	96.2	97.2	96.1	96.1	96.1	99.1	0.042	3.639
RCS-XGB1$_{GNN,thr=0.541}$	98.1	96.2	97.1	95.9	95.9	95.9	99.1	0.049	4.016
RCS-XGB2$_{GNN,thr=0.522}$	98.9	97.0	97.9	96.2	96.2	96.2	99.2	0.032	3.572
RCS-XGB3$_{GNN,thr=0.514}$	98.7	96.3	97.5	96.1	96.1	96.1	99.0	0.044	3.643
RCS-XGB1$_{GAT,thr=0.533}$	98.8	96.5	97.6	96.2	96.1	96.1	99.2	0.042	3.997
RCS-XGB2$_{GAT,thr=0.516}$	98.4	96.2	97.3	96.4	95.8	96.1	99.2	0.031	3.359
RCS-XGB3$_{GAT,thr=0.526}$	97.6	95.9	96.7	96.1	95.8	95.9	99.1	0.044	3.642

[1] $P(D_{LR})$, $P(D_{GNN})$ and $P(D_{GAT})$ in weights are the default probabilities of the instances estimated with LR, GNNs, and GATs, respectively.
[2] Optimal Threshold: It is the median of G_{mean} maximizing thresholds of validation data sets predicted in k-50 fold cross-validations.

4.3 DATA SET 3: Irish Loan Data$_{2013}$

The next three data set belongs to Irish Loan Data and it is analyzed separately for the years 2013, 2014, and 2015. Irish Loan Data is the largest credit data set in this empirical analysis with 3,387,379 credit data instances starting with credit issue dates in 2007 and ending in 2015. The reason for selecting data on a yearly basis is that the defaults in 2007 and the following years are dominant in the data set, and this might be the effect of the global mortgage crisis. Therefore, the analysis could not be made with the use of all data set instances. Moreover, these years are chosen since the data sets become more homogeneous so that LR, GNN, and GAT models of sub-datasets of each year's data can be estimated better.

The data source is available at: kaggle.com [3]. The ratio of non-defaults/defaults is observed to be 6.45 for the credits issued in 2013 in the original Irish Loan Data. The default class instances are deleted with random selection to increase the imbalance in class sizes for this year and the class ratio is adjusted to be 15. Moreover, the reason for the significant decrease in the feature size is that most of the features are reported both in categorical and numerical data formats. Moreover, some features are stated in a general and more specific format like loan issue date, loan issue year and etc. Ten features are found explanatory for the models of which three categorical features are defined with dummy variables and seven numerical features generate twenty-five numerical variables for the classification of this data set. Table 4.16 demonstrates data-specific properties with the explanatory features selected for the classification algorithms.

Table 4.16: Irish Loan Data$_{2013}$

Data Information				
Original Data File Name: Irish Loan Data				
Original Data Size	Data Size After the Cleansing	Number of Categorical Variables	Number of Numerical Variables	Class Ratio
134,755	124,381	10	19	15
10 features found explanatory for the models, they are:				
Credit Amount - Term - Interest Rate - Employment Length - Grade - Total Principal Received to Date - Region - Days Between the Credit Issue and Final Date - Ratio of Annuity to Income - Home Ownership				

Irish Loan Data$_{2013}$ reports the credit amounts disbursed to the applicants but does not give its monetary unit. Figure 4.9 demonstrates the histograms of the credit amounts of non-defaults and defaults and their distributions are found to be expressed best by Gamma($\alpha = 2.6$, $\beta = 0.14$) with $M = 1.2$ for the rejection sampling.

Tables 4.17-4.19 summarize the optimized hyperparameters of ANN, AdaBoost, and XGBoost with the target of maximum G_{mean} values for the test data. The optimal hyperparameters change as the cost parameters change but the optimal sub-sample ratios of columns are observed to be 1 and the optimal delta values of most of these algorithms are also found to be greater than zero for all XGBoost algorithms. Moreover, the optimal iteration numbers in all RCS3-XGB are relatively higher, but they

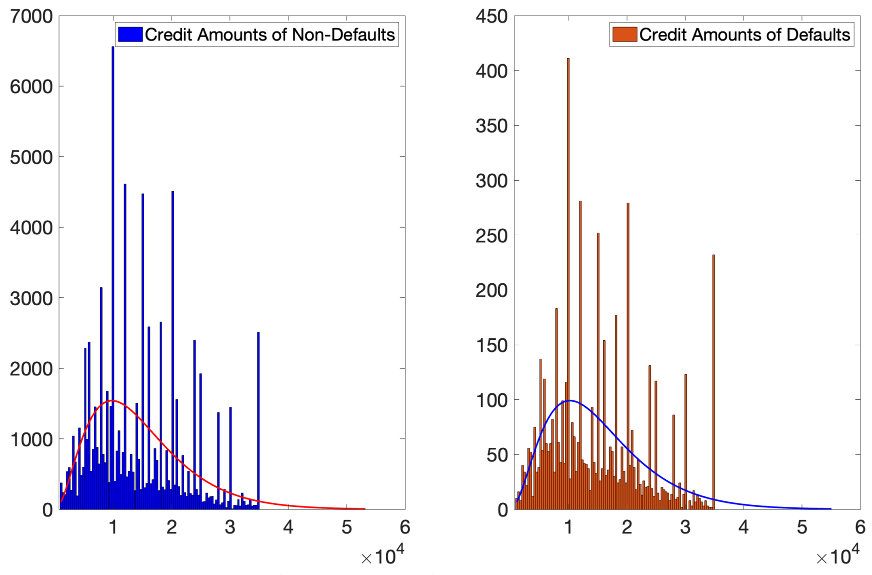


Figure 4.9: Irish Loan Data$_{2013}$
Histograms of Credit Amounts

are relatively lower in all RCS-ANN2 and RCS-ANN3 estimated with the cross entropy loss function. Also, the optimal value of $m > 1$ for C_1 and C_3, while $m < 1$ 1 for C_2 for all cost-sensitive algorithms.

Data-specific information such as train and test data sizes, class ratio, and simulations with LR, GNN, and GAT models are reported in Table 4.20. The cost-insensitive algorithm results and cost-sensitive ANN model results are also presented in this table. The cost-insensitive algorithms have higher *Specificity* values but lower *Specificity* values, therefore, *LGD* values are not reported for these ordinary algorithms. Instance-specific cost-sensitive ANN algorithms estimated with the focal loss function have higher G_{mean} values when compared with the ones estimated with the cross-entropy loss function. ANN3$_{LR}$, ANN3$_{GNN}$ and ANN3$_{GAT}$ using focal loss function are observed to have lower *LGD* values when compared with all cost-sensitive ANNs.

Table 4.17: Irish Loan Data$_{2013}$: ANN Optimized Hyperparameters

Activation Functions:	Sigmoid
Learning Rate:	0.001 and 0.01 for ANN with Focal Loss and Cross Entropy Loss, respectively.
Number of Hidden Layers:	64 32 8
Training Function:	Resilient Back Propagation
Increment to weight change:	1.2
Decrement to weight change:	0.5
Initial weight change:	0.07
Maximum weight change:	50.0
Classification Threshold	0.5

	Focal Loss Function					**Cross Entropy Loss Function**		
	Iterations	m	α	γ_1	γ_2	Iterations	m	thr
ANN	500	-	1	1	1	2000	-	-
RCS-ANN1$_{LR}$	800	2	C_1	0.1	2	1000	1.75	-
RCS-ANN2$_{LR}$	-	-	-	-	-	50	0.5	0.49
RCS-ANN3$_{LR}$	1000	1.75	C_3	0.1	2	100	1.8	-
RCS-ANN1$_{GNN}$	1000	1.85	C_1	0.1	2	1000	1.75	-
RCS-ANN2$_{GNN}$	-	-	-	-	-	50	0.5	0.46
RCS-ANN3$_{GNN}$	1000	1.75	C_3	0.1	2	100	1.8	-
RCS-ANN1$_{GAT}$	1200	1.7	C_1	0.1	2	1000	1.85	-
RCS-ANN2$_{GAT}$	-	-	-	-	-	50	0.5	0.44
RCS-ANN3$_{GAT}$	1000	1.76	C_3	0.1	2	100	1.86	-

Table 4.18: Irish Loan Data$_{2013}$: AdaBoost Optimized Hyperparameters

	Iterations	m	thr
RCS-AdaB1$_{LR}$	1000	1.35	-
RCS-AdaB2$_{LR}$	1500	0.125	0.49
RCS-AdaB1$_{GNN}$	1000	1.45	-
RCS-AdaB2$_{GNN}$	1000	0.167	0.46
RCS-AdaB1$_{GAT}$	1000	1.45	-
RCS-AdaB2$_{GAT}$	2000	0.167	0.44

Table 4.19: Irish Loan Data$_{2013}$: XGBOOST Optimized Hyperparameters

Evaluation Metric: Log Loss/Cross Entropy Loss Function.

	Learning Rate:	Depth from a Root to Leaf	Sub-sample Ratio of Columns	Maximum Delta	Number of Estimators	Scale Positive Weight	m	$thrs$
XGBoost	0.1	6	1	1	1000	-	-	-
CatBoost	0.1	8	-	-	1000	-	-	-
LightGBM	0.05	-	-	-	2000	-	-	-
XGBoost$_{Scaled}$	0.025	6	1	3	184	15	-	-
CatBoost$_{Scaled}$	0.025	8	-	-	195	15	-	-
RCS-XGB1$_{LR}$	0.005	8	1	2	250	C_1	1.26	-
RCS-XGB2$_{LR}$	0.01	8	1	5	372	C_2	0.66	0.49
RCS-XGB3$_{LR}$	0.02	4	1	3	1358	C_3	2.1	-
RCS-XGB1$_{GNN}$	0.01	8	1	0	650	C_1	1.7	-
RCS-XGB2$_{GNN}$	0.05	8	1	5	180	C_2	0.75	0.46
RCS-XGB3$_{GNN}$	0.02	4	1	3	1165	C_3	2	-
RCS-XGB1$_{GAT}$	0.01	8	1	3	534	C_1	1.7	-
RCS-XGB2$_{GAT}$	0.05	8	1	3	170	C_2	0.73	0.44
RCS-XGB3$_{GAT}$	0.02	4	1	3	1270	C_3	2.05	-

	Sampling Strategy	Random State	k Neighbors	Number of Estimators
SMOTHEBoost	Only Minority	$np.random$ generator of Python	5	1500
RUSBoost	Only Majority Class	$np.random$ generator of Python	-	1500

Table 4.20: Irish Loan Data$_{2013}$
Performances of Algorithms

Train Data Size=$87,050$, Test Data Size=$37,331$, Train Data/Test Data = 15
Non-defaults/Defaults=15
Numbers of LR/GNN/GAT Models with Sub-Data Sets=5,000/1,000/1,000
Sub-data Sizes for LR/GNN/GAT Models with Sub-Data Sets=10,888/4,400/4,400
Levels of Node Embeddings in GNN/GAT: 15
β_{RAW} estimated with LR/GNN/GAT: $4.75/4.68/4.84$
LR: R$^2_{ordinary}$ $>= 0.7079$, R$^2_{adjusted}$ $>= 0.7076$ and R$^2_{AdjustedGeneralized}$ $>= 0.7613$
LR: $thrs = 0.49$ for C_2
GNN: Iterations$=$ 100, $knn = 5$, $lR = 0.01$, $thrs = 0.46$ for C_2
GAT: Iterations=150, $knn = 15$, $K = 1$, $lR = 0.02$, $thrs = 0.44$ for C_2
Threshold for ANNs: 0.5

Cost-Insensitive Algorithms

	TRAIN DATA			TEST DATA		
Algorithms	Sens.	Spec.	G$_{mean}$	Sens.	Spec.	G$_{mean}$
LR	86.2	86.3	86.2	86.0	86.2	86.1
GNN	88.6	89.4	89.0	89.6	88.2	88.9
GAT	88.8	87.5	88.3	88.0	87.7	87.9
ANN$CrossEntropyLoss$	82.2	99.8	90.6	77.0	99.3	87.5
ANN$FocalLoss$	67.4	99.9	82.1	69.1	99.9	83.1
AdaBoost	82.0	99.9	90.5	81.4	99.9	90.2
XGBoost	99.1	100	99.6	89.3	100	94.5
CatBoost	91.2	99.9	95.5	88.7	99.9	94.2
LightGBM	99.9	100	99.9	89.4	99.9	94.5

Cost-Sensitive ANNs with Focal Loss Function

	TRAIN DATA			TEST DATA					
Algorithms	Sens.	Spec.	G$_{mean}$	Sens.	Spec.	G$_{mean}$	AUC	MC$^{x10^6}_{Y=1}$	Loss Given Default$^{x10^6}$
RCS-ANN1$_{LR}$	92.6	93.3	92.8	92.5	93.1	92.8	97.7	0.052	2.638
RCS-ANN2$_{LR}$	93.0	93.0	93.0	93.0	92.9	92.9	97.7	0.045	2.501
RCS-ANN1$_{GNN}$	93.1	92.1	92.6	93.1	92.2	92.6	97.2	0.046	2.615
RCS-ANN2$_{GNN}$	93.1	92.8	93.0	93.0	92.8	92.9	97.7	0.045	2.462
RCS-ANN1$_{GAT}$	92.5	92.9	92.7	92.6	92.8	92.7	97.4	0.049	2.705
RCS-ANN2$_{GAT}$	93.1	92.8	93.0	93.1	92.8	92.9	97.7	0.043	2.411

Cost-Sensitive ANNs with Cross Entropy Loss Function

	TRAIN DATA			TEST DATA					
	Sens.	Spec.	G$_{mean}$	Sens.	Spec.	G$_{mean}$	AUC	MC$^{x10^6}_{Y=1}$	Loss Given Default$^{x10^6}$
RCS-ANN1$_{LR}$	94.7	94.2	94.4	87.7	93.6	90.6	98.3	0.072	4.281
RCS-ANN2$_{LR}$	90.6	91.6	91.1	91.0	91.9	91.4	95.8	0.045	3.492
RCS-ANN3$_{LR}$	93.2	92.5	92.9	92.3	92.5	92.4	97.6	0.045	2.698
RCS-ANN1$_{GNN}$	96.6	93.3	94.9	88.1	92.4	90.2	98.8	0.073	4.462
RCS-ANN2$_{GNN}$	91.3	91.5	91.4	90.8	91.5	91.2	96.2	0.072	3.202
RCS-ANN3$_{GNN}$	92.4	92.5	93.9	92.3	93.5	92.4	97.4	0.040	2.674
RCS-ANN1$_{GAT}$	97.0	93.1	95.0	89.1	92.4	90.7	98.8	0.065	4.121
RCS-ANN2$_{GAT}$	90.7	90.3	90.4	90.6	90.4	90.5	96.1	0.069	3.200
RCS-ANN3$_{GAT}$	94.5	91.9	93.2	92.5	91.8	92.2	97.8	0.037	2.563

Table 4.21: Irish Loan Data$_{2013}$
Performances of Algorithms

Boosting Algorithms for Imbalanced Data

	TRAIN DATA			TEST DATA					
	Sens.	Spec.	G$_{mean}$	Sens.	Spec.	G$_{mean}$	AUC	MC$_{Y=1}^{x10^6}$	Loss Given Default$^{x10^6}$
SMOTEBoost	88.1	99.9	93.8	88.8	99.9	94.2	98.8	0.230	4.107
RUSBoost	94.3	95.9	93.1	94.0	95.7	94.9	98.8	0.064	2.141
XGBoost$_{scaled}$	94.8	94.9	94.8	93.4	94.8	94.1	98.5	0.034	2.418
CatBoost$_{scaled}$	93.6	94.3	94.0	93.4	94.2	93.8	98.2	0.039	2.545
RCS-Ada1$_{LR}$	92.5	92.5	92.5	92.6	92.3	92.4	97.9	0.045	2.413
RCS-Ada2$_{LR}$	93.2	93.8	93.5	93.4	93.5	93.4	98.3	0.032	2.184
RCS-Ada1$_{GNN}$	93.6	94.3	93.9	93.8	94.0	93.9	98.5	0.033	2.157
RCS-Ada2$_{GNN}$	94.6	94.4	94.5	94.6	94.2	94.4	98.8	0.021	1.805
RCS-Ada1$_{GAT}$	93.4	93.8	93.6	93.9	93.6	93.7	98.4	0.030	2.031
RCS-Ada2$_{GAT}$	94.7	93.2	93.9	94.6	93.1	93.8	98.5	0.022	1.781
RCS-XGB1$_{LR}$	93.5	93.4	93.5	92.2	93.3	92.8	97.6	0.060	2.605
RCS-XGB2$_{LR}$	95.3	93.7	94.5	93.5	93.5	93.5	98.0	0.026	2.567
RCS-XGB3$_{LR}$	97.0	95.5	96.3	94.5	95.3	94.3	98.8	0.024	1.980
RCS-XGB1$_{GNN}$	97.4	93.0	95.2	93.0	92.9	92.9	98.0	0.039	2.519
RCS-XGB2$_{GNN}$	97.6	95.9	96.7	94.0	95.3	94.7	98.5	0.018	2.227
RCS-XGB3$_{GNN}$	96.5	95.6	96.0	94.5	95.5	95.0.	98.7	0.025	2.040
RCS-XGB1$_{GAT}$	97.3	95.6	96.4	93.4	95.2	94.3	98.3	0.033	2.320
RCS-XGB2$_{GAT}$	97.3	95.6	96.5	93.9	95.1	94.5	98.5	0.021	2.263
RCS-XGB3$_{GAT}$	96.8	95.7	96.2	94.4	95.6	95.0	98.8	0.026	2.035

In a comparison with the cost-sensitive ANN algorithms, the results of outperforming SMOTEBoost, RUSBoost, XGBoost$_{scaled}$, CatBoost$_{scaled}$, and cost-sensitive boosting algorithms classified with the threshold of 0.5 are shown in Table 4.21. RUSBoost has the highest G_{mean} value and the XGBoost$_{scaled}$ comes second when compared with all other algorithms, but they are not superior to all cost-sensitive algorithms. Instance-specific risk-based cost-sensitive modifications of AdaBoost and XGBoost have relatively lower *LGD* values for the test data, but the Specificity values of XGBoost$_{scaled}$ is higher than its *Sensitivity* value and a threshold move for improving the classification of the defaults is required. Therefore, the prediction results for *Sensitivity*, *Specificity*, and *LGD* values are analyzed with the thresholds of 0-4-0.6 where the maximums of G_{mean} values are also observed in k-50 fold cross-validations.

The last graphs of Figures 4.10, 4.11 and 4.12 display how *LGD* values change as *Specificity* changes. The success of cost-sensitive algorithms has lower *LGD* values given *Specificity* values for the thresholds of 0.4-0.6.

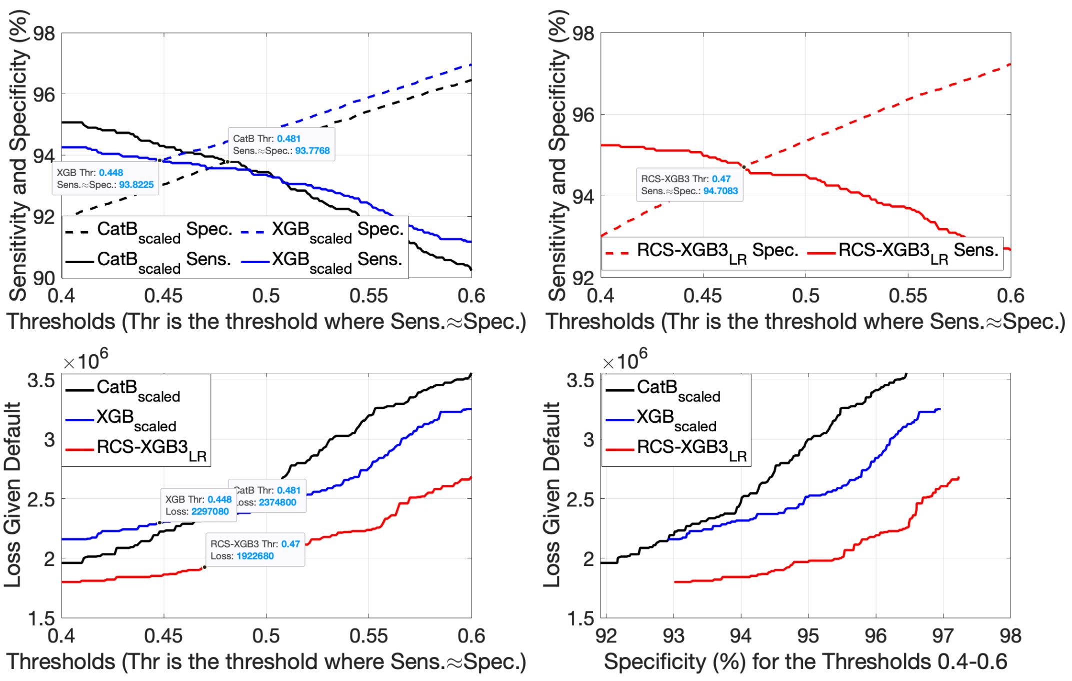


Figure 4.10: Cost Parameters are Estimated with LRs
Sensitivity-Specificity and *LGD* with Different Thresholds in Test Data

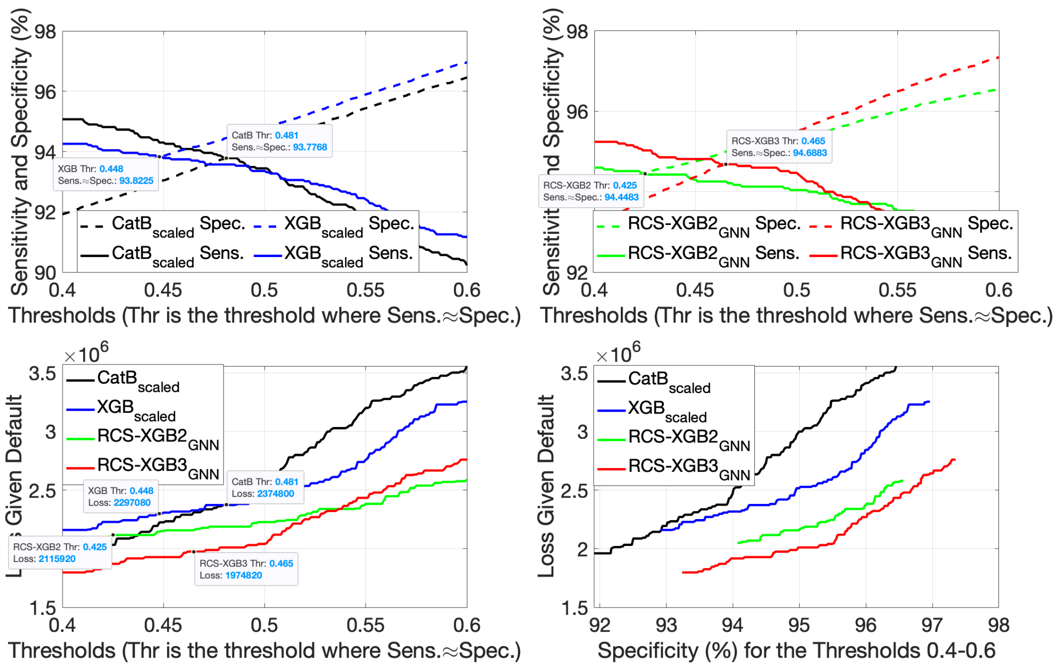


Figure 4.11: Cost Parameters are Estimated with GNNs
Sensitivity-Specificity and *LGD* with Different Thresholds in Test Data

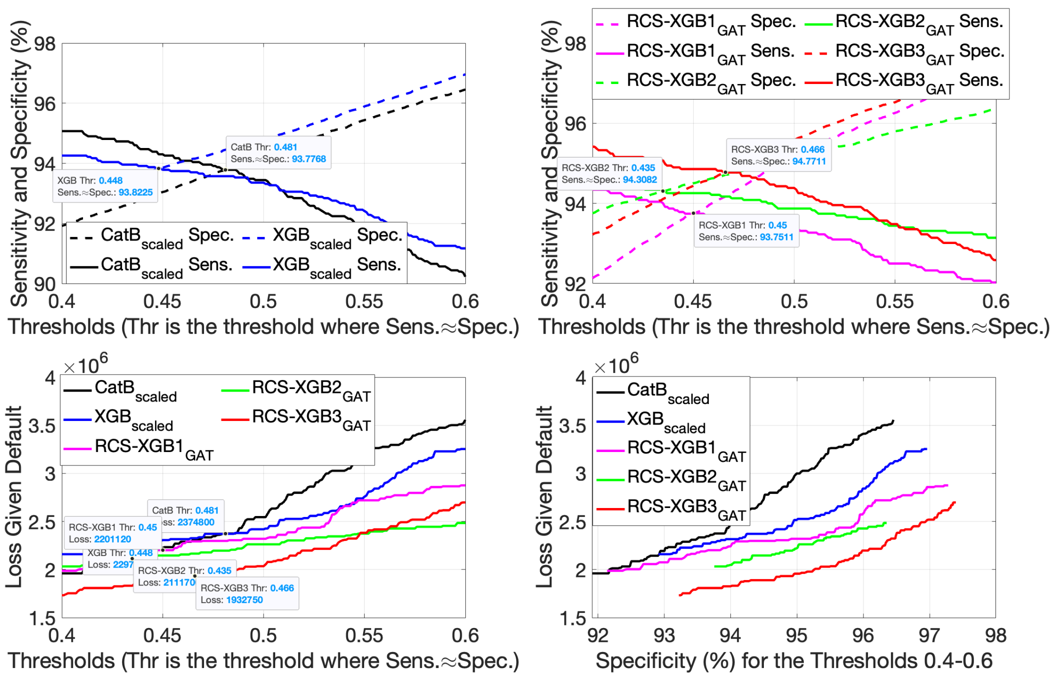


Figure 4.12: Cost Parameters are Estimated with GATs
Sensitivity-Specificity and *LGD* with Different Thresholds in Test Data

The default probability predictions of RUSBoost are not dispersed in the interval [0,1], but they appear in a very narrow interval [0.45-0.55] as Figure 4.13 displays.

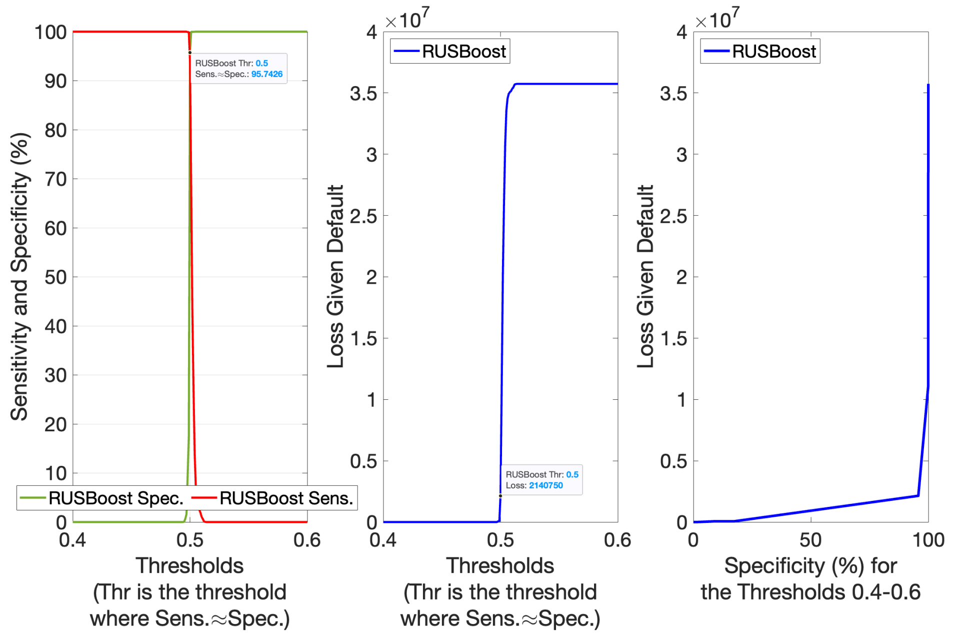


Figure 4.13: RUSBoost *Sensitivity-Specificity* and *LGD* with Different Thresholds in Test Data

XGBoost$_{scaled}$ can reach to its maximum G_{mean} value in the test data predictions with G_{mean}/LGD: $93.8/2.297$ million loss and other proposed cost-sensitive algorithms can have higher G_{mean}values with lower *LGD*. For example, XGB3$_{LR}$ saves 374.4 thousand with G_{mean}/LGD: $94.7/1.922$ million loss, XGB3$_{GNN}$ reduces the loss 322.3 thousand with G_{mean}/LGD: $94.7/1.975$ million loss, XGB2$_{GNN}$ decreases the loss by 181.2 thousand with G_{mean}/LGD: $94.4/2.116$ million loss, XGB3$_{GAT}$ leads the loss to drop by 364.3 thousand with G_{mean}/LGD: $94.8/1.933$ million loss, and XGB2$_{GAT}$ saves 185.4 thousand with G_{mean}/LGD: $94.3/2.112$ million loss. Finally, the less effective cost-sensitive algorithm appears as XGB1$_{GAT}$ and it decreases the loss by 96.0 thousand with G_{mean}/LGD: $93.8/2.201$ million loss.

Cross-validations are conducted with k-50 fold for the threshold analyses. The optimal thresholds maximizing G_{mean} in the validation data sets are evaluated and the medians of the optimal thresholds are applied for the test data. Table 4.22 presents the classification results with new thresholds. AdaB1$_{GNN}$, AdaB2$_{GNN}$, AdaB1$_{GAT}$ and AdaB2$_{GAT}$ presented in Table 4.21 and XGB3$_{LR}$, XGB2$_{GNN}$, XGB3$_{GNN}$, XGB1$_{GAT}$, XGB2$_{GAT}$ and XGB3$_{GAT}$ presented in Table 4.22 all result in lower *LGD* values with higher *Specicificty* values when compared to XGBoost$_{scaled}$ and CatBoost$_{scaled}$.

Table 4.22: Irish Loan Data$_{2013}$
Competing Performances of Algorithms After a Threshold Adjustment

	TRAIN DATA			TEST DATA					
Algorithms$_{Weights^1,thr^2}$	Sens.	Spec.	G$_{mean}$	Sens.	Spec.	G$_{mean}$	AUC	MC$^{x10^7}_{Y=1}$	Loss Given Default$^{x10^7}$
XGBoost$_{ClassRatio,thr=0.429}$	95.6	93.5	94.6	94.0	93.4	93.7	98.4	0.0252	2.234
CatBoost$_{ClassRatio,thr=0.468}$	94.3	93.5	93.9	94.0	93.5	93.8	98.2	0.287	2.310
RCS-XGB1$_{LR,thr=0.466}$	94.3	92.1	93.2	92.8	91.9	92.4	97.6	0.050	2.436
RCS-XGB2$_{LR,thr=0.463}$	95.6	93.1	94.3	93.9	93.0	93.4	98.0	0.021	2.452
RCS-XGB3$_{LR,thr=0.450}$	97.7	94.4	96.0	95.0	94.2	94.6	98.8	0.020	1.864
RCS-XGB1$_{GNN,thr=0.486}$	97.5	92.6	95.0	93.1	92.5	92.8	98.0	0.036	2.472
RCS-XGB2$_{GNN,thr=0.383}$	98.0	94.3	96.1	94.6	93.8	94.2	98.5	0.012	2.028
RCS-XGB3$_{GNN,thr=0.437}$	97.2	94.2	95.7	94.9	94.1	94.5	98.7	0.020	1.917
RCS-XGB1$_{GAT,thr=0.434}$	97.9	93.6	95.7	94.0	93.3	93.7	98.3	0.024	2.111
RCS-XGB2$_{GAT,thr=0.373}$	97.9	93.8	95.9	94.7	93.4	94.0	98.5	0.011	1.958
RCS-XGB3$_{GAT,thr=0.44}$	97.5	94.3	95.9	94.9	94.2	94.6	98.8	0.020	1.880

[1] $P(D_{LR})$, $P(D_{GNN})$ and $P(D_{GAT})$ in weights are the default probabilities of the instances estimated with LR, GNNs, and GATs, respectively.
[2] Optimal Threshold: It is the median of G_{mean} maximizing thresholds of validation data sets predicted in k-50 fold cross-validations.

4.4 DATA SET 4: Irish Loan Data$_{2014}$

The ratio of non-defaults/defaults is observed to be 9.54 for the credits issued in 2014 in the original Irish Loan Data. Similar to Irish Loan Data$_{2013}$, ten explanatory features of three are categorical features and seven are the numerical features and they generate 25 numerical variables. Table 4.23 shows details of the data with the explanatory features selected for the classification algorithms.

Table 4.23: Irish Loan Data$_{2014}$

Data Information				
Original Data File Name: Irish Loan Data				
Original Data Size	Data Size After the Cleansing	Number of Categorical Variables	Number of Numerical Variables	Class Ratio
235,628	235,626	10	19	9.54
10 features are found explanatory for the models, they are:				
Credit Amount-Term-Interest Rate-Employment Length-Grade-Total Principal Received to Date-Region-Days Between the Credit Issue and Final Date-Ratio of Annuity to Income-Home Ownership				

Irish Loan Data$_{2014}$ reports the credit amounts disbursed to the applicants but does not give its monetary unit. Figure 4.14 demonstrates the histograms of the credit amounts of non-defaults and defaults and their distributions are found to be expressed best by Gamma($\alpha = 2.2$, $\beta = 0.15$) with $M = 1.5$ for the rejection sampling.

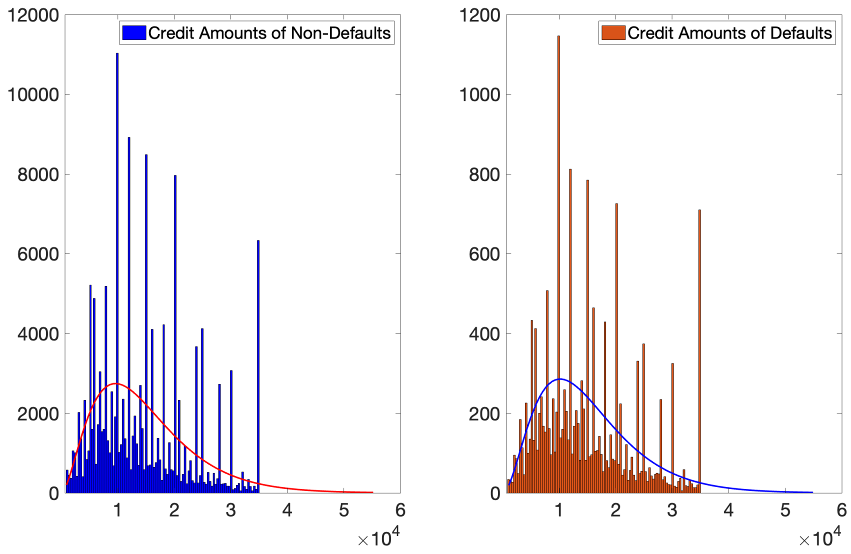


Figure 4.14: Irish Loan Data$_{2014}$: Histograms of Credit Amounts

Tables 4.24-4.26 present the hyperparameters of ANN, AdaBoost, and XGBoost and they are optimized with the target of maximum G_{mean} values in the test data. The iteration numbers of cost-sensitive ANN are the highest when compared with cost-sensitive XGBoost algorithms. The optimal learning rate and the delta values of most of XGBoost algorithms are the common characteristics of cost-sensitive XGBoost algorithms.

Table 4.24: Irish Loan Data$_{2014}$: ANN Optimized Hyperparameters

Activation Functions:	Sigmoid
Learning Rate:	0.001 and 0.01 for ANN with Focal Loss and Cross Entropy Loss, respectively.
Number of Hidden Layers:	64 32 8
Training Function:	Resilient Back Propagation
Increment to weight change:	1.2
Decrement to weight change:	0.5
Initial weight change:	0.07
Maximum weight change:	50.0
Classification Threshold	0.5

	Focal Loss Function					**Cross Entropy Loss Function**		
	Iterations	m	α	γ_1	γ_2	Iterations	m	$thrs$
ANN	1500	-	1	1	1	2000	-	-
RCS-ANN1$_{LR}$	500	1.95	C_1	0.1	2	1000	2.5	-
RCS-ANN2$_{LR}$	-	-	-	-	-	1000	1.2	0.5
RCS-ANN3$_{LR}$	500	2	C_3	0.1	2	1000	2.85	-
RCS-ANN1$_{GNN}$	500	1.65	C_1	0.1	2	1000	2.2	-
RCS-ANN2$_{GNN}$	-	-	-	-	-	1000	1	0.56
RCS-ANN3$_{GNN}$	500	1.7	C_3	0.1	2	1000	2.5	-
RCS-ANN1$_{GAT}$	500	1.4	C_1	0.1	2	1000	1.9	-
RCS-ANN2$_{GAT}$	-	-	-	-	-	1000	0.7	0.6
RCS-ANN3$_{GAT}$	500	1.4	C_3	0.1	2	1000	2	-

Table 4.25: Irish Loan Data$_{2014}$: AdaBoost Optimized Hyperparameters

	Iterations	m	$thrs$
RCS-A1$_{LR}$	500	1.68	-
RCS-A2$_{LR}$	500	0.25	0.5
RCS-A1$_{GNN}$	500	1.6	-
RCS-A2$_{GNN}$	500	0.225	0.56
RCS-A1$_{GAT}$	500	1.3	-
RCS-A2$_{GAT}$	500	0.1	0.6

Table 4.27 reports the train and test data sizes, class ratio, and simulations with LR, GNN, and GAT models. It also presents prediction results of the cost-insensitive algorithms and cost-sensitive ANN models The cost-insensitive algorithms have higher *Specificity* values but lower *Specificity* values, therefore, *LGD* values are not reported for these ordinary algorithms. Instance-specific cost-sensitive ANN algorithms estimated with the cross entropy loss function have higher G_{mean} values when compared with the ones estimated with the focal loss function. ANN3$_{LR}$, ANN2$_{GNN}$,

Table 4.26: Irish Loan Data$_{2014}$: XGBOOST Optimized Hyperparameters

Evaluation Metric: Log Loss/Cross Entropy Loss Function.

	Learning Rate:	Depth from a Root to Leaf	Sub-sample Ratio of Columns	Maximum Delta	Number of Estimators	Scale Positive Weight	m	$thrs$
XGBoost	0.1	6	1	1	2000	-	-	-
CatBoost	0.1	7	1	-	2000	-	-	
LightGBM	0.1	-	-	-	2000	-	-	
XGBoost$_{Scaled}$	0.1	7	1	2	53	9.5	-	-
CatBoost$_{Scaled}$	0.05	7	1	-	220	9.5	-	-
RCS-XGB1$_{LR}$	0.05	7	1	1	200	C_1	2	-
RCS-XGB2$_{LR}$	0.05	7	1	0	260	C_2	1.25	0.5
RCS-XGB3$_{LR}$	0.05	8	0.9	5	148	C_3	2.8	-
RCS-XGB1$_{GNN}$	0.05	7	1	1	99	C_1	1.7	-
RCS-XGB2$_{GNN}$	0.05	7	0.8	2	110	C_2	0.8	0.56
RCS-XGB3$_{GNN}$	0.05	7	0.8	4	117	C_3	2.25	-
RCS-XGB1$_{GAT}$	0.05	8	1	4	89	C_1	1.7	-
RCS-XGB2$_{GAT}$	0.05	7	0.8	2	302	C_2	0.8	0.6
RCS-XGB3$_{GAT}$	0.05	7	1	3	309	C_3	2.25	-

	Sampling Strategy	Random State	k Neighbors	Number of Estimators
SMOTHEBoost	Only Minority	$np.random$ generator of Python	5	2000
RUSBoost	Only Majority Class	$np.random$ generator of Python	-	2000

ANN3$_{GNN}$ and ANN2$_{GAT}$ using cross entropy loss function are observed to have lower *LGD* values when compared with all cost-sensitive ANNs.

Table 4.28 displays the classification performances of boosting algorithms which are observed to be superior to the cost-sensitive ANN algorithms due to lower *LGD* values in test data. The prediction probabilities of SMOTEBoost, RUSBoost, XGBoost$_{scaled}$, CatBoost$_{scaled}$, and cost-sensitive boosting algorithms are classified with the threshold of 0.5. RUSBoost has the highest *G$_{mean}$* value and the lowest *LGD* values when compared with all other ordinary algorithms. Instance-specific risk-based cost-sensitive modifications of XGBoost have relatively lower *LGD* values for test data, but the *Specificity* values of RUSBoost and XGBoost$_{scaled}$ are higher allowing a threshold move for improving *Sensitivity*. Therefore, the prediction results for *Sensitivity*, *Specificity*, and *LGD* values are analyzed with the thresholds of 0-4-0.6 where the maximums of *G$_{mean}$* values are also observed in the test data.

Table 4.27: Irish Loan Data$_{2014}$
Performances of Algorithms

Train Data Size=$164,938$, Test Data Size=$70,688$, Train Data/Test Data = $7/3$
Non-defaults/Defaults=9.5
Numbers of LR/GNN/GAT Models with Sub-Data Sets=5000/1000/1000
Sub-data Sizes for LR/GNN/GAT Models with Sub-Data Sets=31,550/3,000/3,000
Levels of Node Embeddings in GNN/GAT: 2/1
β_{RAW} estimated with LR/GNN/GAT: $2.676/2.751/3.641$
LR: $R^2_{ordinary} >= 0.663$, $R^2_{adjusted} >= 0.663$ and $R^2_{AdjustedGeneralized} >= 0.737$
LR: $thrs = 0.5$ for C_2
GNN: Iterations= 80, $knn = 10$, $lR = 0.02$, $thrs = 0.56$ for C_2
GAT: Iterations=100, $knn = 25$, $K = 1$, $lR = 0.02$, $thrs = 0.6$ for C_2
Threshold for ANNs: 0.5

Cost-Insensitive Algorithms

	TRAIN DATA			TEST DATA		
Algorithms	Sens.	Spec.	G_{mean}	Sens.	Spec.	G_{mean}
LR	87.0	87.0	87.0	87.6	86.7	87.2
GNN	91.4	86.5	88.9	88.1	88.2	88.2
GAT	88.3	88.1	88.2	88.4	86.7	87.5
ANN$CrossEntropyLoss$	83.7	99.9	91.4	82.7	99.7	90.1
ANN$FocalLoss$	60.3	98.5	77.1	61.3	98.5	77.7
AdaBoost	74.7	99.8	86.3	75.2	99.7	86.6
XGBoost	95.4	100	97.7	83.7	99.9	91.4
CatBoost	89.2	100	94.4	83.5	99.9	91.4
LightGBM	99.7	100	99.8	83.6	99.9	91.4

Cost-Sensitive ANNs with Focal Loss Function

	TRAIN DATA			TEST DATA					
Algorithms	Sens.	Spec.	G_{mean}	Sens.	Spec.	G_{mean}	AUC	$MC^{x10^6}_{Y=1}$	Loss Given Default$^{x10^6}$
RCS-ANN1$_{LR}$	88.2	90.0	89.1	89.1	89.9	89.5	95.2	0.339	12.428
RCS-ANN2$_{LR}$	89.9	89.4	89.7	90.6	89.3	89.9	95.6	0.246	10.378
RCS-ANN1$_{GNN}$	89.4	89.9	89.7	90.2	89.8	90.0	95.6	0.315	11.642
RCS-ANN2$_{GNN}$	89.1	90.0	89.5	89.6	89.9	89.7	95.5	0.319	11.563
RCS-ANN1$_{GAT}$	90.5	89.3	89.9	91.1	89.2	90.2	95.7	0.255	10.372
RCS-ANN2$_{GAT}$	89.7	89.5	89.6	90.3	89.3	89.8	95.5	0.275	10.697

Cost-Sensitive ANNs with Cross Entropy Loss Function

	TRAIN DATA			TEST DATA					
	Sens.	Spec.	G_{mean}	Sens.	Spec.	G_{mean}	AUC	$MC^{x10^6}_{Y=1}$	Loss Given Default$^{x10^6}$
RCS-ANN1$_{LR}$	91.4	91.8	91.6	90.1	91.5	90.8	96.5	0.196	11.296
RCS-ANN2$_{LR}$	92.4	89.9	91.1	91.9	89.7	90.8	96.7	0.092	9.736
RCS-ANN3$_{LR}$	93.7	91.1	92.4	91.8	90.8	91.3	97.0	0.150	8.947
RCS-ANN1$_{GNN}$	92.6	92.4	92.0	90.9	91.0	90.9	96.8	0.209	10.342
RCS-ANN2$_{GNN}$	93.1	91.0	92.0	91.8	90.6	91.2	97.0	0.118	9.328
RCS-ANN3$_{GNN}$	93.0	91.9	92.4	91.6	91.6	91.6	97.1	0.159	9.396
RCS-ANN1$_{GAT}$	93.2	91.3	92.3	91.0	91.1	91.1	96.8	0.217	9.798
RCS-ANN2$_{GAT}$	93.2	91.0	92.1	91.6	90.7	91.2	97.0	0.126	8.975
RCS-ANN3$_{GAT}$	93.5	90.4	92.0	91.6	90.2	90.9	97.0	0.172	9.115

Table 4.28: Irish Loan Data$_{2014}$
Performances of Algorithms

Boosting Algorithms for Imbalanced Data

	TRAIN DATA			TEST DATA					
	Sens.	Spec.	G$_{mean}$	Sens.	Spec.	G$_{mean}$	AUC	MC$_{Y=1}^{x10^6}$	Loss Given Default$^{x10^6}$
SMOTEBoost	81.0	98.9	89.5	81.7	98.9	89.9	97.5	1.728	18.931
RUSBoost	91.4	93.7	92.6	91.6	93.7	92.6	97.9	0.362	8.609
XGBoost$_{scaled}$	93.0	92.9	93.0	91.7	92.5	92.1	97.6	0.164	9.064
CatBoost$_{scaled}$	92.1	92.0	92.1	91.9	91.9	91.9	97.5	0.161	8.927
RCS-Ada1$_{LR}$	90.5	90.2	90.3	90.8	89.9	90.4	96.5	0.195	9.201
RCS-Ada2$_{LR}$	90.7	91.0	90.8	90.9	90.8	90.9	96.8	0.184	9.497
RCS-AdaB1$_{GNN}$	91.0	91.0	91.0	91.3	90.8	91.0	97.1	0.212	9.298
RCS-AdaB2$_{GNN}$	91.2	91.2	91.2	91.6	91.1	91.3	97.3	0.172	9.052
RCS-Ada1$_{GAT}$	91.3	91.3	91.3	91.7	91.2	91.4	97.3	0.184	8.980
RCS-Ada2$_{GAT}$	91.2	91.6	91.4	91.5	91.5	91.5	97.4	0.178	9.001
RCS-XGB1$_{LR}$	91.0	90.2	90.6	88.8	89.6	89.2	96.1	0.361	11.292
RCS-XGB2$_{LR}$	93.9	92.5	93.2	92.1	92.1	92.1	97.5	0.095	8.971
RCS-XGB3$_{LR}$	95.9	92.3	94.4	92.2	92.2	92.2	97.8	0.152	8.263
RCS-XGB1$_{GNN}$	91.3	91.3	91.3	90.2	90.9	90.6	96.8	0.300	10.373
RCS-XGB2$_{GNN}$	93.2	92.6	92.9	92.0	92.2	92.1	97.4	0.121	8.705
RCS-XGB3$_{GNN}$	93.6	92.7	93.1	92.1	92.3	92.2	97.6	0.156	8.602
RCS-XGB1$_{GAT}$	94.7	92.7	93.7	92.0	92.1	92.1	97.6	0.183	8.498
RCS-XGB2$_{GAT}$	95.1	92.9	94.0	92.2	92.3	92.3	97.8	0.132	8.577
RCS-XGB3$_{GAT}$	97.0	92.7	94.8	92.5	92.1	92.3	97.8	0.148	8.112

The success of cost-sensitive algorithms with lower *LGD* values given *Specificity* values are demonstrated with the last graphs of Figures 4.15, 4.16 and 4.17. In addition, although RUSBoost has the highest G_{mean} values, default probability predictions of this algorithm do not result in a dispersed in the interval [0,1] but they appear in a very narrow interval [0.45-0.55] as Figure 4.18 displays.

The comparisons with XGBoost$_{scaled}$ (G_{mean}/LGD: 91.9/8.853 million) for the case of maximum G_{mean} values in the test data demonstrate that XGB3$_{LR}$ saves 590.7 thousand with G_{mean}/LGD: 92.2/8.263 million, XGB2$_{GNN}$ reduces the loss by 190.3 thousand with G_{mean}/LGD: 92.1/8.663 million, XGB3$_{GNN}$ leads a decrease in the loss by 306.1 thousand with G_{mean}/LGD: 92.1/8.547 million, XGB1$_{GAT}$ saves 440.1 thousand with G_{mean}/LGD: 92.1/8.413 million , XGB2$_{GAT}$ decreases the loss by 290.3 thousand with G_{mean}/LGD: 92.3/8.563 million, and the most successful algorithm, XGB3$_{GAT}$, saves 594 thousand with G_{mean}/LGD: 92.3/8.230 million.

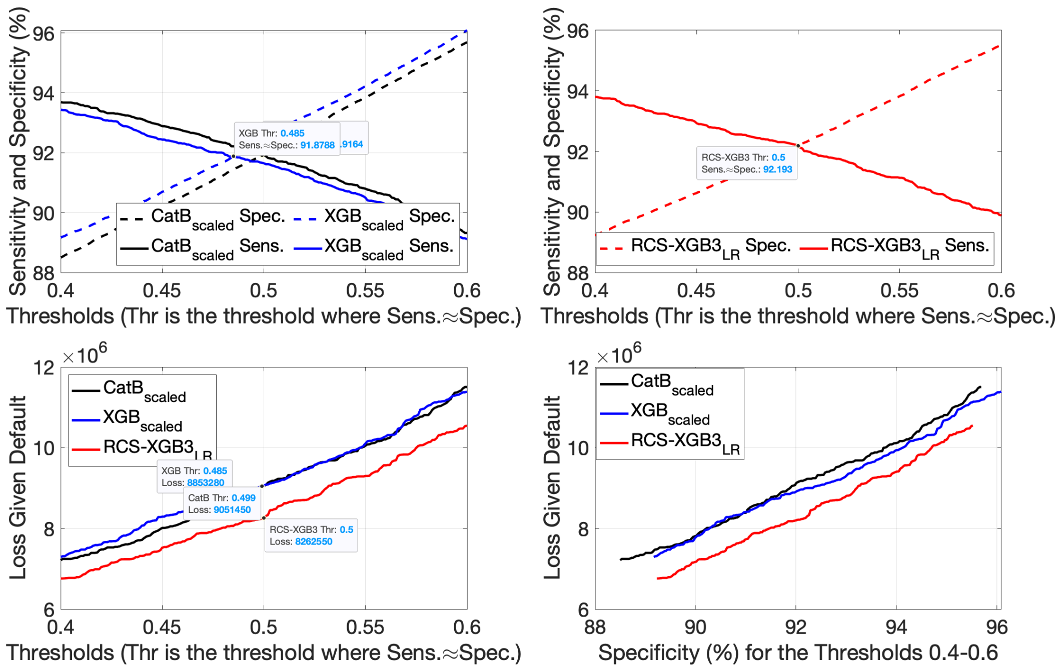


Figure 4.15: Cost Parameters are Estimated with LRs
Sensitivity-Specificity and *LGD* with Different Thresholds in Test Data

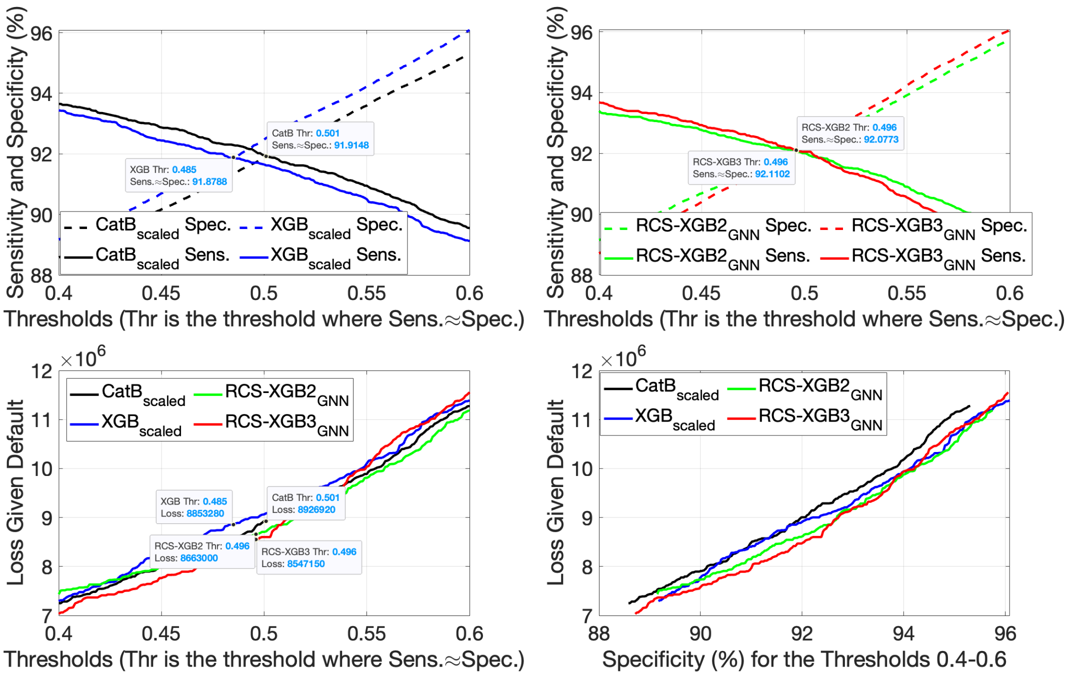


Figure 4.16: Cost Parameters are Estimated with GNNs
Sensitivity-Specificity and *LGD* with Different Thresholds in Test Data

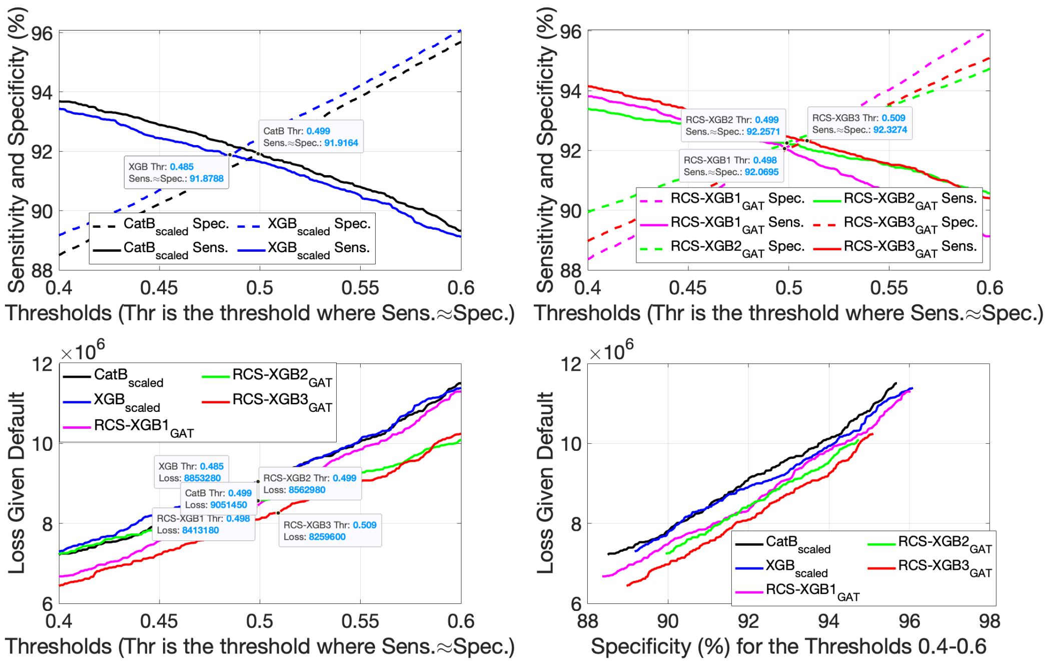


Figure 4.17: Cost Parameters are Estimated with GATs
Sensitivity-Specificity and *LGD* with Different Thresholds in Test Data

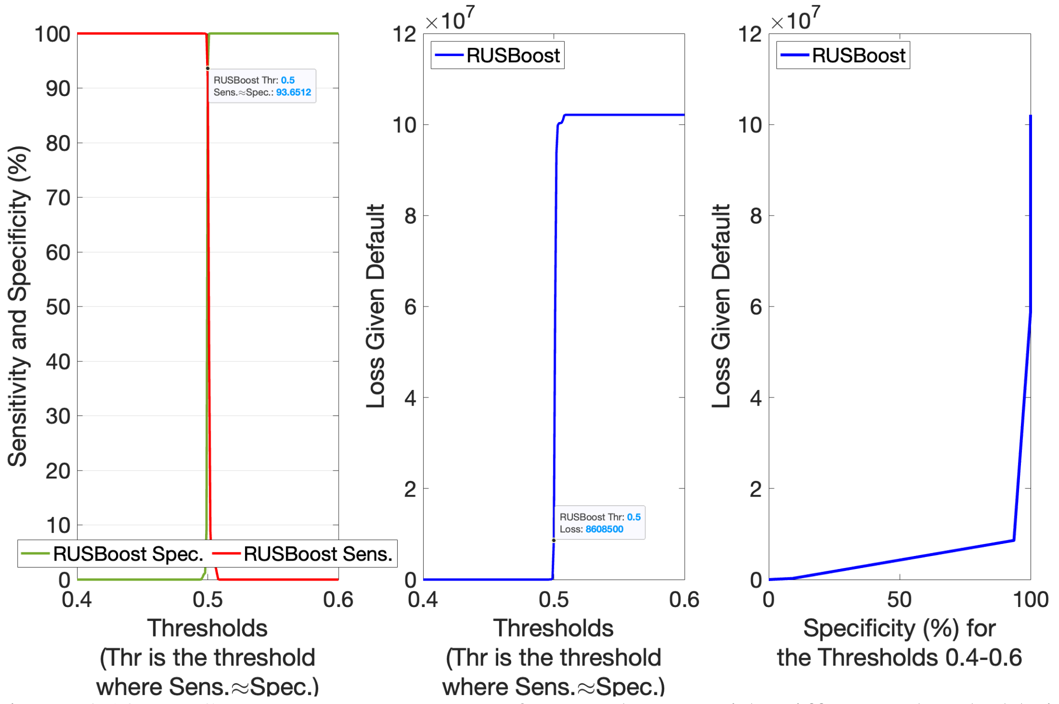


Figure 4.18: RUSBoost *Sensitivity-Specificity* and *LGD* with Different Thresholds in Test Data

A threshold adjustment in test data predictions can be made with the optimal thresholds maximizing G_{mean} values in the validation data sets in k-50 fold validations. The medians of the optimal thresholds are applied for the test data classification and Table 4.29 presents test data results. XGB3$_{LR}$, XGB2$_{GNN}$, XGB3$_{GNN}$, XGB1$_{GAT}$, XGB2$_{GAT}$ and XGB3$_{GAT}$ result in lower *LGD* values with given $Specicificty$ values when compared to XGBoost$_{scaled}$ and CatBoost$_{scaled}$. The success of the cost parameters estimated by GAT models can be attributed to their superior predictions in test data when compared with LR estimation results and they do not over-fit in the train data as GNN models as its aggregated results in Table 4.27 display.

Table 4.29: Irish Loan Data$_{2014}$
Competing Performances of Algorithms After a Threshold Adjustment

	TRAIN DATA			TEST DATA					
Algorithms$_{Weights^1, thr^2}$	Sens.	Spec.	G$_{mean}$	Sens.	Spec.	G$_{mean}$	AUC	MC$^{x10^6}_{Y=1}$	Loss Given Default$^{x10^6}$
XGBoost$_{ClassRatio,thr=0.471}$	93.7	91.8	92.8	92.1	91.4	91.8	97.6	0.139	8.651
CatBoost$_{ClassRatio,thr=0.493}$	92.3	91.8	91.1	92.2	91.6	91.9	97.5	0.149	8.713
RCS-XGB1$_{LR,thr=0.475}$	91.7	89.3	90.5	89.5	88.6	89.0	96.1	0.304	10.661
RCS-XGB2$_{LR,thr=0.480}$	94.1	92.1	93.1	92.3	91.6	91.9	97.5	0.088	8.829
RCS-XGB3$_{LR,thr=0.492}$	96.0	92.6	94.3	92.3	91.9	92.1	97.8	0.146	8.180
RCS-XGB1$_{GNN,thr=0.475}$	92.0	90.1	91.1	90.9	89.6	90.3	96.8	0.246	9.660
RCS-XGB2$_{GNN,thr=0.480}$	93.5	91.9	92.7	92.3	91.6	91.9	97.4	0.108	8.467
RCS-XGB3$_{GNN,thr=0.492}$	93.8	92.3	93.1	92.2	91.9	92.1	97.6	0.148	8.469
RCS-XGB1$_{GAT,thr=0.494}$	94.7	92.5	93.6	92.2	91.9	92.1	97.6	0.171	8.289
RCS-XGB2$_{GAT,thr=0.480}$	95.3	92.4	93.8	92.5	91.8	92.1	97.8	0.118	8.299
RCS-XGB3$_{GAT,thr=0.490}$	97.1	92.4	94.7	92.6	91.7	92.1	97.8	0.143	8.036

[1] $P(D_{LR})$, $P(D_{GNN})$ and $P(D_{GAT})$ in weights are the default probabilities of the instances estimated with LR, GNNs, and GATs, respectively.
[2] Optimal Threshold: It is the median of G_{mean} maximizing thresholds of validation data sets predicted in k-50 fold cross-validations.

4.5 DATA SET 5: Irish Loan Data$_{2015}$

The ratio of non-defaults/defaults is found to be 33.81 for the credits issued in 2015 in the original Irish Loan Data. Different than years 2013 and 2014, 11 features are found explanatory for the models of which 2 categorical features defined with dummy variables and 9 numerical features generate 21 numerical variables for the classification of this data set. Table 4.30 displays the data size and the features evaluated as explanatory for the classification algorithms.

As in Data Set 4 and 5, Irish Loan Data$_{2015}$ reports the credit amounts disbursed to the applicants but does not give its monetary unit. The credit amounts of non-defaults

Table 4.30: Irish Loan Data$_{2015}$

Data Information				
Original Data File Name: Irish Loan Data				
Original Data Size	Data Size After the Cleansing	Number of Categorical Variables	Number of Numerical Variables	Class Ratio
421,092	420,577	10	19	33.81

11 features are found explanatory for the models, they are:

Credit Amount - Term - Interest Rate - Employment Length - Grade - Total Payments- Total Principal Received to Date - Region - Days Between the Credit Issue and Final Date - Ratio of Annuity to Income - Ratio of Loan to Annual Income

and defaults and their distributions are expressed best by Gamma($\alpha = 2.2$, $\beta = 0.15$) with $M = 1.5$ for the rejection sampling (Figure 4.19).

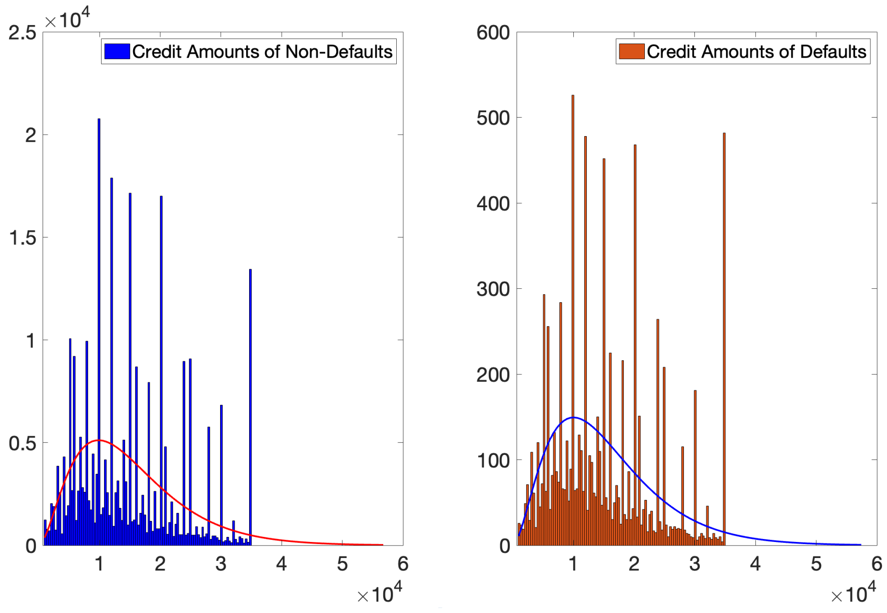


Figure 4.19: Irish Loan Data$_{2015}$
Histograms of Credit Amounts

Tables 4.31-4.33 present the optimized hyperparameters of ANN, AdaBoost, and XG-Boost with the target of maximum G_{mean} values in the test data. Common values for hyperparameters can be specified as cost-sensitive ANNs using focus loss function have the highest optimal iteration numbers and cost-sensitive XGBoost algorithms have the lowest. Similarly, the optimal value of m are found to be as $m > 1$ for C_1 and C_3, while $m < 1$ for C_2 for all cost-sensitive algorithms.

Table 4.31: Irish Loan Data$_{2015}$: ANN Optimized Hyperparameters

Activation Functions:	Sigmoid
Learning Rate:	0.001 and 0.01 for ANN with Focal Loss and Cross Entropy Loss, respectively.
Number of Hidden Layers:	64 32 8
Training Function:	Resilient Back Propagation
Increment to weight change:	1.2
Decrement to weight change:	0.5
Initial weight change:	0.07
Maximum weight change:	50.0
Classification Threshold	0.5

	Focal Loss Function					**Cross Entropy Loss Function**		
	Iterations	m	α	γ_1	γ_2	Iterations	m	thr
ANN	1500	-	1	1	1	1000	-	-
RCS-ANN1$_{LR}$	1500	1.25	C_1	0.1	2	1000	1.15	-
RCS-ANN2$_{LR}$	-	-	-	-	-	100	0.2	0.5
RCS-ANN3$_{LR}$	1500	1.25	C_3	0.1	2	500	1.34	-
RCS-ANN1$_{GNN}$	1500	1.18	C_1	0.1	2	1000	1.15	-
RCS-ANN2$_{GNN}$	-	-	-	-	-	100	0.2	0.5
RCS-ANN3$_{GNN}$	1500	1.25	C_3	0.1	2	500	1.34	-
RCS-ANN1$_{GAT}$	1500	1.25	C_1	0.1	2	1000	1.15	-
RCS-ANN2$_{GAT}$	-	-	-	-	-	100	0.2	0.54
RCS-ANN3$_{GAT}$	1500	1.25	C_3	0.1	2	500	1.34	-

Table 4.32: Irish Loan Data$_{2015}$: AdaBoost Optimized Hyperparameters

	Iterations	m	thr
RCS-AdaB11$_{LR}$	500	1.25	-
RCS-AdaB2$_{LR}$	500	0.33	0.5
RCS-AdaB1$_{GNN}$	500	1.25	-
RCS-AdaB2$_{GNN}$	1000	0.33	0.5
RCS-AdaB1$_{GAT}$	500	1.132	-
RCS-AdaB2$_{GAT}$	500	0.2	0.54

Table 4.33: Irish Loan Data$_{2015}$: XGBOOST Optimized Hyperparameters

Evaluation Metric: Log Loss/Cross Entropy Loss Function.

	Learning Rate:	Depth from a Root to Leaf	Sub-sample Ratio of Columns	Maximum Delta	Number of Estimators	Scale Positive Weight	m	$thrs$
XGBoost	0.1	6	1	1	2000	-	-	-
CatBoost	0.05	6	-	-	2000	-	-	-
LightGBM	0.05	-	-	-	2000	-	-	-
XGBoost$_{Scaled}$	0.025	6	1	3	184	33.8	-	-
CatBoost$_{Scaled}$	0.05	6	-	-	268	33.8	-	-
RCS-XGB1$_{LR}$	0.05	7	0.5	0	76	$C_1(P(D_{LR}))$	1.352	-
RCS-XGB2$_{LR}$	0.1	6	1	0	40	$C_2(P(D_{LR}))$	0.224	0.5
RCS-XGB3$_{LR}$	0.05	6	0.6	5	78	$C_3(P(D_{LR}))$	1.356	-
RCS-XGB1$_{GNN}$	0.1	6	0.6	4	66	$C_1(P(D_{GNN}))$	1.36	-
RCS-XGB2$_{GNN}$	0.1	6	1	0	50	$C_2(P(D_{GNN}))$	0.222	0.5
RCS-XGB3$_{GNN}$	0.1	6	1	4	68	$C_3(P(D_{GNN}))$	1.45	-
RCS-XGB1$_{GAT}$	0.05	7	0.6	2	130	$C_1(P(D_{GAT}))$	1.28	-
RCS-XGB2$_{GAT}$	0.1	6	1	0	50	$C_2(P(D_{GAT}))$	0.187	0.54
RCS-XGB3$_{GAT}$	0.1	6	1	0	72	$C_3(P(D_{GAT}))$	1.34	-

	Sampling Strategy	Random State	k Neighbors	Number of Estimators
SMOTHEBoost	Only Minority	$np.random$ generator of Python	5	1000
RUSBoost	Only Majority Class	$np.random$ generator of Python	-	1000

The detailed information for the train and test data sizes, class ratio, and simulations with LR, GNN, and GAT models are all reported in Table 4.34. The prediction results of the cost-insensitive algorithm and cost-sensitive ANN model are also given in this table. The cost-insensitive algorithms have higher *Specificity* values but lower *Specificity* values, therefore, *LGD* values are not reported for these ordinary algorithms. Instance-specific cost-sensitive ANN algorithms estimated with the focal loss function have higher G_{mean} values when compared with the ones estimated with the focal loss function. Table 4.35 shows the classification performances of boosting algorithms with the threshold of 0.5 which are observed to be superior to the cost-sensitive ANN algorithms.

Table 4.35 presents the classification performances of boosting algorithms offered for imbalanced data problems and their prediction results are observed to be higher for *Sensitivity* and *Specificity* values when compared with the results reported in Table 4.34. The reported class predictions of SMOTEBoost, RUSBoost, XGBoost$_{scaled}$, CatBoost$_{scaled}$ and cost-sensitive boosting algorithms are the first prediction results of the algorithms classified with the threshold of 0.5.

The last graphs of Figures 4.20 and 4.21 clearly demonstrate the success of cost-sensitive algorithms with lower *LGD* values given *Specificity* values for the thresholds of 0.4-0.6. RUSBoost has the lowest *LGD* value but lower *Specificity* and the XGBoost$_{scaled}$ has the second lower *LGD* but higher *Specificity* when compared with all ordinary algorithms. XGBoost$_{scaled}$ can reach to the highest $G_{mean} = 87.0$ with $LGD = 7.832$ million in test data predictions and the comparisons with XGBoost$_{scaled}$ for the case of maximum G_{mean} values in the test data demonstrate that XGB3$_{GNN}$ saves 44.6 thousand with G_{mean}/*LGD*: $87.3/7.788$ million, XGB2$_{GAT}$ decreases the loss by 129.6 thousand with G_{mean}/*LGD*: $86.9/7.703$ million and XGB3$_{GAT}$ leads a decrease in the monetary loss by 233 thousand with G_{mean}/*LGD*: $87.3/7.599$ million.

G_{mean} maximizing optimal thresholds in the validation data sets are computed in k-50 fold validations. The medians of the thresholds are applied for the test data classification and Table 4.36 presents how the new thresholds affect the test data prediction results. XGB3$_{GNN}$, XGB2$_{GAT}$ and XGB3$_{GAT}$ result in lower *LGD* values with given *Specicificty* values when compared to XGBoost$_{scaled}$ and CatBoost$_{scaled}$.

Table 4.34: Irish Loan Data$_{2015}$
Performances of Algorithms

Train Data Size=$294,249$, Test Data Size=$126,328$, Train Data/Test Data = $7/3$
Non-defaults/Defaults=33.8
Numbers of LR/GNN/GAT Models with Sub-Data Sets=3000/1000/1000
Sub-data Sizes for LR/GNN/GAT Models with Sub-Data Sets=16,936/3,000/3,000
Levels of Node Embeddings in GNN/GAT: 2/1
β_{RAW} estimated with LR/GNN/GAT: $12.41/13.79/13.03$
LR: R$^2_{ordinary}$ $>= 0.51$, R$^2_{adjusted}$ $>= 0.51$ and R$^2_{AdjustedGeneralized}$ $>= 0.59$
LR: $thrs = 0.49$ for C_2
GNN: Iterations$= 100$, $knn = 5$, $lR = 0.02$ $thrs = 0.5$ for C_2
GAT: Iterations=150, $knn = 15$, $K = 2$, $lR = 0.02$, $thrs = 0.54$ for C_2
Threshold for ANNs: 0.5

Cost-Insensitive Algorithms

	TRAIN DATA			TEST DATA		
Algorithms	Sens.	Spec.	G$_{mean}$	Sens.	Spec.	G$_{mean}$
LR	82.5	82.7	82.6	80.8	82.9	81.8
GNN	86.3	82.1	84.2	80.8	84.1	82.5
GAT	84.7	84.3	84.5	83.5	82.2	82.8
ANN$CrossEntropyLoss$	37.3	99.9	61.0	35.3	99.9	59.4
ANN$FocalLoss$	0.7	100	8.37	0.8	100	8.94
AdaBoost	37.3	99.9	61.0	35.3	99.9	59.4
XGBoost	82.3	100	90.7	59.9	100	77.4
CatBoost	60.9	100	78.0	58.7	100	76.6
LightGBM	69.6	100	83.4	59.6	100	77.2

Cost-Sensitive ANNs with Focal Loss Function

	TRAIN DATA			TEST DATA					
Algorithms	Sens.	Spec.	G$_{mean}$	Sens.	Spec.	G$_{mean}$	AUC	MC$^{x10^6}_{Y=1}$	Loss Given Default$^{x10^6}$
RCS-ANN1$_{LR}$	84.2	84.2	84.2	83.2	84.2	83.7	90.0	0.548	10.258
RCS-ANN3$_{LR}$	84.0	84.4	82.7	83.5	84.4	83.5	90.4	0.587	10.685
RCS-ANN1$_{GNN}$	83.1	85.2	84.1	81.6	85.2	83.4	90.0	0.710	11.432
RCS-ANN3$_{GNN}$	83.8	84.7	84.2	82.4	84.6	83.5	90.4	0.610	10.912
RCS-ANN1$_{GAT}$	87.9	81.3	84.5	87.0	81.2	84.0	90.0	0.377	8.083
RCS-ANN3$_{GAT}$	86.2	82.9	84.5	85.0	82.8	83.9	90.3	0.448	9.197

Cost-Sensitive ANNs with Cross Entropy Loss Function

	TRAIN DATA			TEST DATA					
	Sens.	Spec.	G$_{mean}$	Sens.	Spec.	G$_{mean}$	AUC	MC$^{x10^6}_{Y=1}$	Loss Given Default$^{x10^6}$
RCS-ANN1$_{LR}$	84.9	85.5	85.2	80.9	85.5	83.1	89.9	0.500	11.868
RCS-ANN2$_{LR}$	80.9	85.6	82.3	78.9	85.6	82.2	88.8	0.409	13.648
RCS-ANN3$_{LR}$	87.0	82.9	84.9	83.7	82.9	83.3	89.0	0.329	9.788
RCS-ANN1$_{GNN}$	86.8	83.8	81.9	80.3	83.5	81.9	91.3	0.536	12.630
RCS-ANN2$_{GNN}$	80.1	85.6	83.2	78.9	85.6	82.2	89.0	0.409	13.648
RCS-ANN3$_{GNN}$	87.0	82.9	84.9	83.7	82.9	83.3	90.9	0.329	9.788
RCS-ANN1$_{GAT}$	84.3	88.0	86.1	78.5	87.9	83.1	92.7	0.687	13.095
RCS-ANN2$_{GAT}$	80.9	85.6	83.2	78.9	85.6	82.2	89.0	0.408	13.648
RCS-ANN3$_{GAT}$	87.0	82.9	84.9	83.7	82.9	83.3	90.9	0.329	9.788

Table 4.35: Irish Loan Data$_{2015}$
Performances of Algorithms

Boosting Algorithms for Imbalanced Data

	TRAIN DATA			TEST DATA					
	Sens.	Spec.	G_{mean}	Sens.	Spec.	G_{mean}	AUC	$MC_{Y=1}^{x10^6}$	Loss Given Default$^{x10^6}$
SMOTEBoost	59.6	97.3	76.2	59.9	97.3	76.3	93.6	4.778	23.838
RUSBoost	88.0	84.8	86.4	87.9	84.9	86.4	94.5	0.432	7.156
XGBoost$_{scaled}$	88.2	87.1	87.6	87.0	87.0	87.0	94.6	0.268	7.832
CatBoost$_{scaled}$	87.6	87.0	87.3	87.0	87.0	87.0	94.7	0.282	8.016
RCS-AdaB1$_{LR}$	85.8	85.7	85.7	86.0	85.6	85.8	94.1	0.302	8.033
RCS-AdaB2$_{LR}$	85.9	85.8	85.8	86.2	85.8	86.0	94.1	0.269	8.199
RCS-AdaB1$_{GNN}$	86.3	85.7	86.0	86.5	85.6	86.1	94.5	0.317	7.952
RCS-AdaB2$_{GNN}$	85.8	86.3	86.0	86.2	86.2	86.2	94.1	0.303	8.195
RCS-AdaB1$_{GAT}$	86.1	86.1	86.1	86.3	86.1	86.2	94.3	0.337	8.238
RCS-AdaB2$_{GAT}$	86.6	85.6	86.1	86.9	85.6	86.2	94.3	0.293	7.801
RCS-XGB1$_{LR}$	89.5	87.1	88.3	87.0	87.0	87.0	94.4	0.334	7.980
RCS-XGB2$_{LR}$	88.8	84.6	86.7	86.9	84.6	85.8	92.0	0.238	8.765
RCS-XGB3$_{LR}$	88.1	87.0	87.5	86.9	86.9	86.9	94.3	0.309	8.134
RCS-XGB1$_{GNN}$	90.4	87.0	88.7	87.1	87.0	87.0	94.8	0.310	8.130
RCS-XGB2$_{GNN}$	89.4	86.0	87.6	87.2	85.9	86.5	94.2	0.143	7.837
RCS-XGB3$_{GNN}$	90.5	87.2	88.8	87.5	87.1	87.3	95.0	0.251	7.684
RCS-XGB1$_{GAT}$	92.9	87.5	90.2	87.1	87.3	87.2	94.8	0.322	8.172
RCS-XGB2$_{GAT}$	90.8	87.1	89.0	87.0	86.6	86.8	94.5	0.197	7.562
RCS-XGB3$_{GAT}$	89.7	86.7	88.2	87.7	86.6	87.3	95.0	0.254	7.415

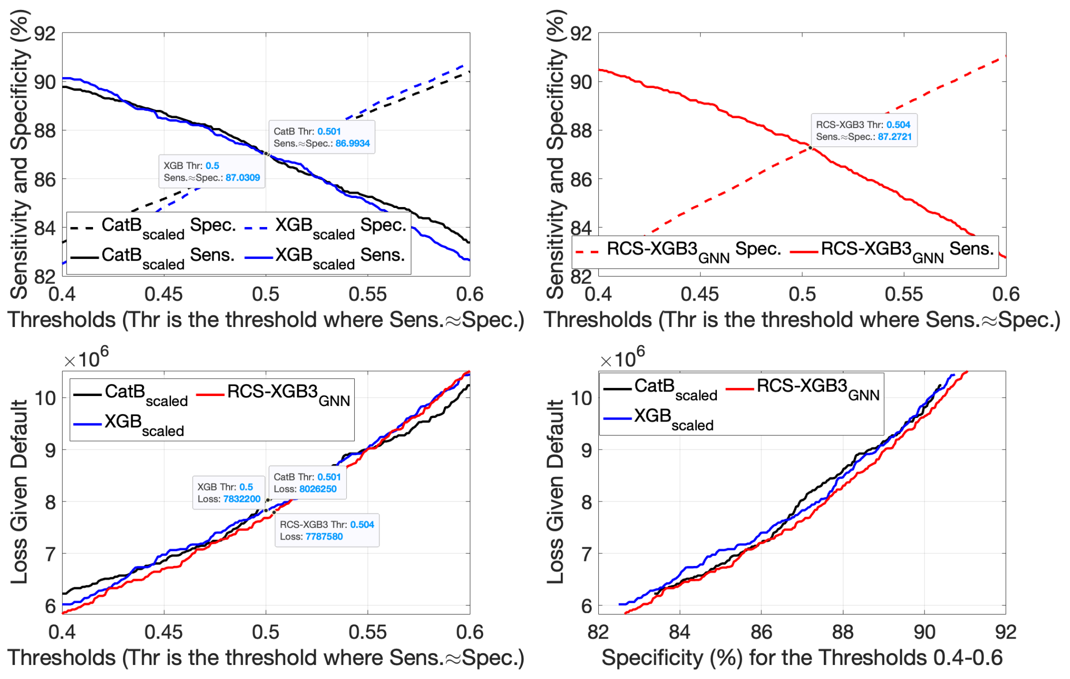


Figure 4.20: Cost Parameters are Estimated with GNNs
Sensitivity-Specificity and *LGD* with Different Thresholds in Test Data

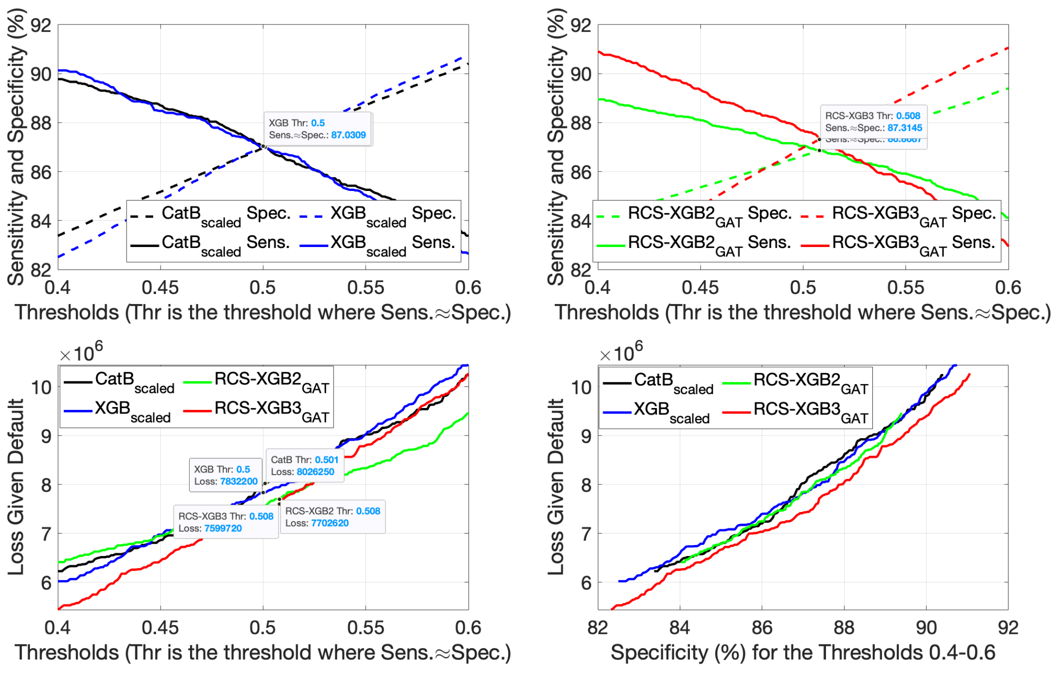


Figure 4.21: Cost Parameters are Estimated with GATs
Sensitivity-Specificity and *LGD* with Different Thresholds in Test Data

Table 4.36: Irish Loan Data$_{2015}$
Competing Performances of Algorithms After a Threshold Adjustment

	TRAIN DATA			TEST DATA					
Algorithms$_{Weights^1, thr^2}$	Sens.	Spec.	G$_{mean}$	Sens.	Spec.	G$_{mean}$	AUC	MC$^{x10^6}_{Y=1}$	Loss Given Default$^{x10^6}$
XGBoost$_{ClassRatio, thr=0.488}$	88.6	86.6	87.6	87.4	86.5	87.0	94.6	0.247	7.630
CatBoost$_{ClassRatio, thr=0.495}$	87.9	86.8	87.3	87.3	86.8	87.0	94.7	0.260	7.762
RCS-XGB1$_{LR, thr=0.49}$	90.0	86.7	88.3	87.2	86.6	86.9	94.4	0.318	7.830
RCS-XGB2$_{LR, thr=0.55}$	87.9	85.4	86.7	86.0	85.4	85.7	92.0	0.304	9.433
RCS-XGB3$_{LR, thr=0.485}$	88.6	86.4	87.5	87.3	86.4	86.9	94.3	0.281	7.845
RCS-XGB1$_{GNN, thr=0.489}$	90.8	86.5	88.6	87.6	86.4	87.0	94.8	0.280	7.526
RCS-XGB2$_{C_2(P(D_{GNN})), thr=0.53}$	88.9	86.6	87.7	86.6	86.5	86.6	94.2	0.176	8.208
RCS-XGB3$_{GNN, thr=0.488}$	91.0	86.7	88.8	87.8	86.6	87.2	95.0	0.232	7.461
RCS-XGB1$_{GAT, thr=0.484}$	93.4	86.7	90.0	87.6	86.5	87.0	94.8	0.292	7.898
RCS-XGB2$_{GAT, thr=0.498}$	90.9	87.0	88.9	87.0	86.6	86.8	94.5	0.197	7.562
RCS-XGB3$_{GAT, thr=0.495}$	89.7	86.6	88.1	87.9	86.7	87.3	95.0	0.241	7.286

[1] $P(D_{LR})$, $P(D_{GNN})$ and $P(D_{GAT})$ in weights are the default probabilities of the instances estimated with LR, GNNs, and GATs, respectively.
[2] Optimal Threshold: It is the median of G_{mean} maximizing thresholds of validation data sets predicted in k-50 fold cross validations.

4.6 DATA SET 6: Freddie Mac Single Family Loan Data

Freddie Mac Single Family Loan Data is the largest credit data set among the other 7 data sets. It contains 500,137 credit applicants with 26 features in other credit data sets in this empirical analysis. The data set is obtained from kaggle.com [1]. The number of explanatory variables for the models is 14 of which 8 categorical features are defined with dummy variables and 6 numerical features generate 108 numerical variables for the classification of this data set. Table 4.37 gives brief summary of the data set and the attributes evaluated as explanatory for the classification algorithms.

Table 4.37: Freddie Mac Single Family Loan Data

Data Information				
Original Data File Name: loan level 500k				
Original Data Size	Data Size After the Cleansing	Number of Categorical Variables	Number of Numerical Variables	Class Ratio
500,137	500,137	14	12	26.8
14 features are found explanatory for the models, they are:				
Credit Score - Interest Rate - Mortgage Insurance Percentage - Occupancy Status - Original Debt to Income Ratio - Original Loan to Value - Property State - Loan Purpose - Number of Borrowers - Loan Servicer Name - Prepaid Status - Loan Issue Date (First Payment Date)				

Tables 4.38-4.40 present the hyperparameters of the models which are optimized with the objective of maximum G_{mean} values in the test data. Common values for hyperparameters can be specified as cost-sensitive ANNs using focal loss function have the highest optimal iteration numbers and cost-sensitive XGBoost algorithms have the lowest. All cost-sensitive XGBoost algorithms have a sub-sample ratio of columns less than 1 and a maximum delta higher than 0. Furthermore, the optimizations indicate $m > 1$ for C_1 and C_3, while $m < 1$ for C_2 for all cost-sensitive algorithms.

Table 4.41 informs about the train and test data sizes, class ratio, and simulations with LR, GNN, and GAT models. It also presents the cost-insensitive algorithm results and cost-sensitive ANN model results. Table 4.49 shows the classification performances of boosting algorithms with the threshold of 0.5 which are observed to be superior to the cost-sensitive ANN algorithms.

Table 4.38: Freddie Mac Single Family Loan Data
ANN Optimized Hyperparameters

Activation Functions:	Sigmoid
Learning Rate:	0.001 and 0.01 for ANN with Focal Loss and Cross Entropy Loss, respectively.
Number of Hidden Layers:	32 16 8
Training Function:	Resilient Back Propagation
Increment to weight change:	1.2
Decrement to weight change:	0.5
Initial weight change:	0.07
Maximum weight change:	50.0
Classification Threshold	0.5

	Focal Loss Function					**Cross Entropy Loss Function**		
	Iterations	m	α	γ_1	γ_2	Iterations	m	$thrs$
ANN	1000	-	1	1	1	100	-	-
RCS-ANN1$_{LR}$	1000	1.64	C_1	0.1	2	500	1.65	-
RCS-ANN2$_{LR}$	-	-	-	-	-	500	0.4	0.44
RCS-ANN3$_{LR}$	1200	1.68	C_3	0.1	2	500.	1.68	-
RCS-ANN1$_{GNN}$	1000	1.45	C_1	0.1	2	500	1.44	-
RCS-ANN2$_{GNN}$	-	-	-	-	-	500	0.245	0.467
RCS-ANN3$_{GNN}$	1000	1.54	C_1	0.1	2	500.	1.52	-
RCS-ANN1$_{GAT}$	1000	1.425	C_1	0.1	2	500	1.44	-
RCS-ANN2$_{GAT}$	-	-	-	-	-	500	0.275	0.44
RCS-ANN3$_{GAT}$	1000	1.6	C_3	0.1	2	400	1.56	-

Table 4.39: Freddie Mac Single Family Loan Data:
AdaBoost Optimized Hyperparameters

	Iterations	m	$thrs$
RCS-AdaB1$_{LR}$	1000	1.525	-
RCS-AdaB2$_{LR}$	200	0.1	0.44
RCS-AdaB1$_{GNN}$	1000	1.46	-
RCS-AdaB2$_{GNN}$	300	0.075	0.467
RCS-AdaB1$_{GAT}$	500	1.35	-
RCS-AdaB2$_{GAT}$	300	0.06	0.44

Table 4.40: Freddie Mac Single Family Loan Data
XGBOOST Optimized Hyperparameters

Evaluation Metric: Log Loss/Cross Entropy Loss Function.

	Learning Rate:	Depth from a Root to Leaf	Sub-sample Ratio of Columns	Maximum Delta	Number of Estimators	Scale Positive Weights	m	$thrs$
XGBoost	0.1	6	0.5	0	1000	-	-	-
CatBoost	0.05	6	0	1000	-	-	-	
LightGBM	0.05	-	-	1000	-	-	-	
XGBoost$_{Scaled}$	0.1	6	0.5	0	72	26.8	-	-
CatBoost$_{Scaled}$	0.1	6	1	0	65	26.8	-	-
RCS-XGB1$_{LR}$	0.1	8	0.5	1	61	C_1	1.55	-
RCS-XGB2$_{LR}$	0.1	9	0.5	1	56	C_2	0.36	0.44
RCS-XGB3$_{LR}$	0.1	7	0.51	1	61	C_3	1.65	-
RCS-XGB1$_{GNN}$	0.1	7	0.49	1	58	C_1	1.4	-
RCS-XGB2$_{GNN}$	0.1	7	0.49	1	56	C_2	0.24	0.467
RCS-XGB3$_{GNN}$	0.1	7	0.49	1	59	C_3	1.5	-
RCS-XGB1$_{GAT}$	0.1	7	0.51	1	63	C_1	1.4	-
RCS-XGB2$_{GAT}$	0.11	7	0.51	1	61	C_2	0.27	0.44
RCS-XGB3$_{GAT}$	0.12	8	0.51	1	38	C_3	1.58	-

	Sampling Strategy	Random State	k Neighbors	Number of Estimators
SMOTHEBoost	Only Minority	$np.random$ generator of Python	5	1000
RUSBoost	Only Majority Class	$np.random$ generator of Python	-	1000

Table 4.41: Freddie Mac Single Family Loan Data
Performances of Algorithms

Train Data Size=$350,096$, Test Data Size=$150,041$, Train Data/Test Data = $7/3$
Non-defaults/Defaults=26.8
Numbers of LR/GNN/GAT Models with Sub-Data Sets=$4000/1000/800$
Sub-data Sizes for LR/GNN/GAT Models with Sub-Data Sets=25,188/3,000/3,000
LR: Models with $R^2_{Ordinary} >= 0.550$, ordinary $R^2_{Adjusted} >= 0.550$, $R^2_{AdjustedGeneralized} >= 0.640$
LR: $thrs = 0.44$ for C_2
Levels of Node Embeddings in GNN/GAT: 1/1
β_{RAW} used in C_2 estimated with LR/GNN/GAT: $8.669/9.696/8.464$
GNN: Iterations= 30, $knn = 5$, $lR = 0.01$ $thrs = 467$ for C_2
GAT: Iterations=100, $knn = 15$, $K = 1$, $thrs = 0.44$ for C_2
Threshold for ANNs: 0.5

Cost-Insensitive Algorithms

	TRAIN DATA			TEST DATA		
Algorithms	Sens.	Spec.	G$_{mean}$	Sens.	Spec.	G$_{mean}$
LR	83.9	83.9	83.9	84.1	83.6	83.9
GNN	84.0	84.1	84.0	83.4	83.8	83.6
GAT	85.6	85.4	85.5	83.5	84.3	83.9
ANN$CrossEntropyLoss$	47.5	99.5	68.8	47.8	99.4	69.0
ANN$FocalLoss$	47.7	99.5	68.9	48.5	99.5	69.5
AdaBoost	53.2	98.9	72.6	54.1	98.8	73.1
XGBoost	53.2	99.7	72.8	48.0	99.4	69.1
CatBoost	49.4	99.6	70.2	48.1	99.5	69.2
LightGBM	50.8	99.6	71.1	48.1	99.5	69.2

Cost-Sensitive ANNs with Focal Loss Function

	TRAIN DATA			TEST DATA			
Algorithms	Sens.	Spec.	G$_{mean}$	Sens.	Spec.	G$_{mean}$	AUC
RCS-ANN1$_{LR}$	85.5	83.8	84.6	84.1	83.6	83.9	92.2
RCS-ANN2$_{LR}$	86.9	84.5	85.7	83.0	84.3	83.7	92.2
RCS-ANN1$_{GNN}$	87.1	83.6	85.3	83.8	83.4	83.6	92.1
RCS-ANN2$_{GNN}$	86.7	84.3	85.5	83.5	84.1	83.8	92.2
RCS-ANN1$_{GAT}$	86.8	84.3	85.5	83.4	84.0	83.7	92.0
RCS-ANN2$_{GAT}$	86.6	84.5	85.5	83.3	84.2	83.8	92.2

Cost-Sensitive ANNs with Cross Entropy Loss Function

	TRAIN DATA			TEST DATA			
	Sens.	Spec.	G$_{mean}$	Sens.	Spec.	G$_{mean}$	AUC
RCS-ANN1$_{LR}$	86.6	83.4	85.0	82.9	83.3	83.1	90.8
RCS-ANN2$_{LR}$	84.9	84.4	84.6	83.4	84.3	83.8	92.1
RCS-ANN3$_{LR}$	87.6	82.3	84.9	83.3	82.0	82.7	91.3
RCS-ANN1$_{GNN}$	87.0	82.4	83.4	82.1	82.6	90.9	91.1
RCS-ANN2$_{GNN}$	86.1	84.0	85.0	83.7	83.8	83.8	92.1
RCS-ANN3$_{GNN}$	86.7	83.6	85.1	82.7	83.2	83.0	91.4
RCS-ANN1$_{GAT}$	86.9	82.0	84.6	82.6	82.2	82.4	90.3
RCS-ANN2$_{GAT}$	85.8	84.0	84.9	83.5	83.8	83.7	91.9
RCS-ANN3$_{GAT}$	86.4	83.7	85.0	83.1	83.5	83.3	91.6

Table 4.42: Freddie Mac Single Family Loan Data
Performances of Algorithms

	Boosting Algorithms for Imbalanced Data						
	TRAIN DATA			TEST DATA			
	Sens.	Spec.	G_{mean}	Sens.	Spec.	G_{mean}	AUC
SMOTEBoost	80.4	87.5	83.9	79.8	87.4	83.5	91.6
RUSBoost	80.4	87.5	83.9	79.8	87.4	83.5	91.7
XGBoost$_{scaled}$	81.2	89.1	85.1	79.2	89.0	84.0	92.3
CatBoost$_{scaled}$	79.6	88.7	84.0	79.5	88.6	83.9	92.2
RCS-Ada1$_{LR}$	82.3	83.6	82.9	82.9	83.5	83.2	91.3
RCS-Ada2$_{LR}$	82.7	83.2	82.9	83.4	83.2	83.3	91.4
RCS-AdaB1$_{GNN}$	83.5	83.3	83.4	83.8	83.2	83.5	91.6
RCS-AdaB2$_{GNN}$	83.5	83.2	83.3	83.9	83.2	83.6	91.7
RCS-Ada1$_{GAT}$	83.0	83.4	83.2	83.5	83.4	83.4	91.5
RCS-Ada2$_{GAT}$	83.2	83.0	83.1	84.0	83.0	83.5	91.6
RCS-XGB1$_{LR}$	86.1	86.2	86.1	82.1	85.9	84.0	92.0
RCS-XGB2$_{LR}$	85.6	84.0	84.8	84.0	83.9	83.9	92.0
RCS-XGB3$_{LR}$	85.8	84.3	85.0	84.2	84.1	84.2	92.3
RCS-XGB1$_{GNN}$	86.5	84.0	85.3	83.9	83.8	83.8	92.1
RCS-XGB2$_{GNN}$	85.1	84.0	84.5	83.9	83.9	83.9	92.1
RCS-XGB3$_{GNN}$	85.4	84.8	85.1	83.7	84.6	84.2	92.3
RCS-XGB1$_{GAT}$	86.7	84.1	85.4	83.9	83.9	83.9	92.1
RCS-XGB2$_{GAT}$	85.2	84.2	84.7	84.1	84.1	84.1	92.2
RCS-XGB3$_{GAT}$	86.6	84.4	85.5	83.9	84.2	84.1	92.1

The credit data set under consideration does not include the credit amount disbursed to the borrower as one of its features. As a result, sub-datasets could not be generated using rejection sampling based on credit amounts and were instead formed using random sampling. Furthermore, *LGD* analyses could not be performed for model comparisons. Table 4.42 demonstrates that the instance-specific cost-sensitive algorithms resulted in balanced classification when compared to XGBoost$_{scaled}$ and CatBoost$_{scaled}$; however, the threshold analyses in Figure 4.22 indicate that the proposed algorithms only marginally improved the *Sensitivity-Specificity* values in test data. Furthermore, k-50 fold cross-validations are applied for these algorithms and the medians of thresholds that maximize the G_{mean} values for the validation data sets are applied for the test data. Table 4.43 shows that threshold adjustments also indicate a very small improvement in the test data predictions.

In all credit data sets, except for the one under consideration, credit amount has been identified as a significant explanatory variable for LR, GNN, GAT models, and all other algorithms. Therefore, the absence of this feature may have impacted the predictive capabilities of the proposed algorithms.

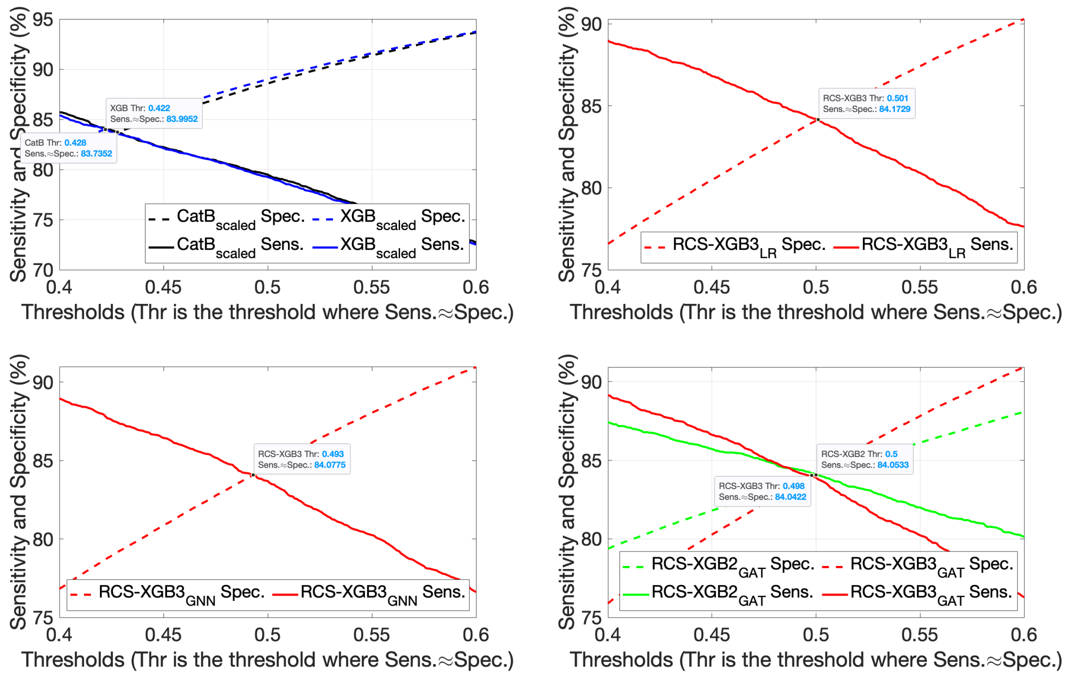


Figure 4.22: *Sensitivity-Specificity* and *LGD* with Different Thresholds in Test Data

Table 4.43: Freddie Mac Single Family Loan Data
Competing Performances of Algorithms After a Threshold Adjustment

Algorithms$_{Weights^1, thr^2}$	TRAIN DATA Sens.	Spec.	G$_{mean}$	TEST DATA Sens.	Spec.	G$_{mean}$	AUC
XGBoost$_{ClassRatio, thr=0.422}$	85.5	84.2	84.8	84.0	84.0	84.0	92.3
CatBoost$_{ClassRatio, thr=0.428}$	84.0	83.9	83.9	83.7	83.7	83.7	92.2
RCS-XGB1$_{LR, thr=0.472}$	87.6	84.0	85.8	83.8	83.8	83.8	92.0
RCS-XGB2$_{LR, thr=0.510}$	85.5	84.0	84.8	83.9	83.9	83.9	92.0
RCS-XGB3$_{LR, thr=0.501}$	85.8	84.3	85.1	84.2	84.2	84.2	92.3
RCS-XGB1$_{GNN, thr=0.498}$	86.6	83.9	85.2	84.1	83.6	83.8	92.1
RCS-XGB2$_{GNN, thr=0.499}$	85.1	84.0	84.5	83.9	83.8	83.8	92.1
RCS-XGB3$_{GNN, thr=0.494}$	85.8	84.3	85.1	84.1	84.2	84.1	92.3
RCS-XGB1$_{GAT, thr=0.498}$	86.9	83.9	85.4	84.0	83.6	83.8	92.1
RCS-XGB2$_{GAT, thr=0.510}$	85.1	84.3	84.7	84.1	84.1	84.1	92.2
RCS-XGB3$_{GAT, thr=0.497}$	86.8	84.2	85.5	84.0	84.0	84.0	92.1

[1] $P(D_{LR})$, $P(D_{GNN})$ and $P(D_{GAT})$ in weights are the default probabilities of the instances estimated with LR, GNNs, and GATs, respectively.
[2] Optimal Threshold: It is the median of G_{mean} maximizing thresholds of validation data sets predicted in k-50 fold cross-validations.

4.7 DATA SET 7: SBA National Loan Data$_{2004-2006}$

SBA National Loan Data has information on 899,164 credit applicants with 26 reported attributes. Credit issue dates start from 1962 to 2014. The source of the data set is kaggle.com [5]. In the empirical studies, increasing the credits issued in different years is observed to result in unsuccessful estimations in the LR models, therefore, the credit issue years are selected as close to each year to obtain more homogeneous data. The credits issued in 2004, 2005, and 2006, and 2002 and 2003 generate the first and the second data set, respectively.

SBA National Loan Data$_{2004-2006}$ has a class ratio of 2.79. The default class instances are deleted with random selection and a data set with non-defaults/defaults of 40 is generated for severely imbalanced data. Furthermore, the reason for the significant decrease in the feature size is that some of the features are repeated in different formats in the data set. For this reason, 15 features are deleted (The deleted features are Name, City, Zip, NAICS, FranchiseCode, MIS Status, ChgOffDate, ApprovalFY, LoanNrChkDgt, ApprvDate, DisburseDate, ApprovalDate, Disbursement-Date, ChgOffPrinGr, and BalanceGross). Eight features are found explanatory for the models of which two categorical features are defined with dummy variables and six numerical features generate sixty numerical variables for the classification. Table 4.44 summarizes SBA National Loan Data$_{2004-2006}$ used in the classification algorithms.

Table 4.44: SBA National Loan Data$_{2004-2006}$

Data Information				
Original Data File Name: SBAnational				
Original Data Size	Data Size After the Cleansing	Number of Categorical Variables	Number of Numerical Variables	Class Ratio
221,855	164,021	11	15	40
8 features found explanatory for the models, they are:				
Term - Credit Amount (Disbursement Gross) - Guarantee for the Loan by SBA (US Small Business Administration) - Days between the Approval and the Disbursement of the Loan - State - Number of Business - Employees - The Number of Retained Jobs - Urban/Rural				

SBA National Loan Data$_{2004-2006}$ reports the credit amounts disbursed to the applicants but does not give its monetary unit. Figure 4.23 demonstrates the histograms of the credit amounts of non-defaults and defaults and their distributions are found to be expressed best by Gamma($\alpha = 0.834$, $\beta = 0.0236$) with $M = 1$ for the rejection sampling.

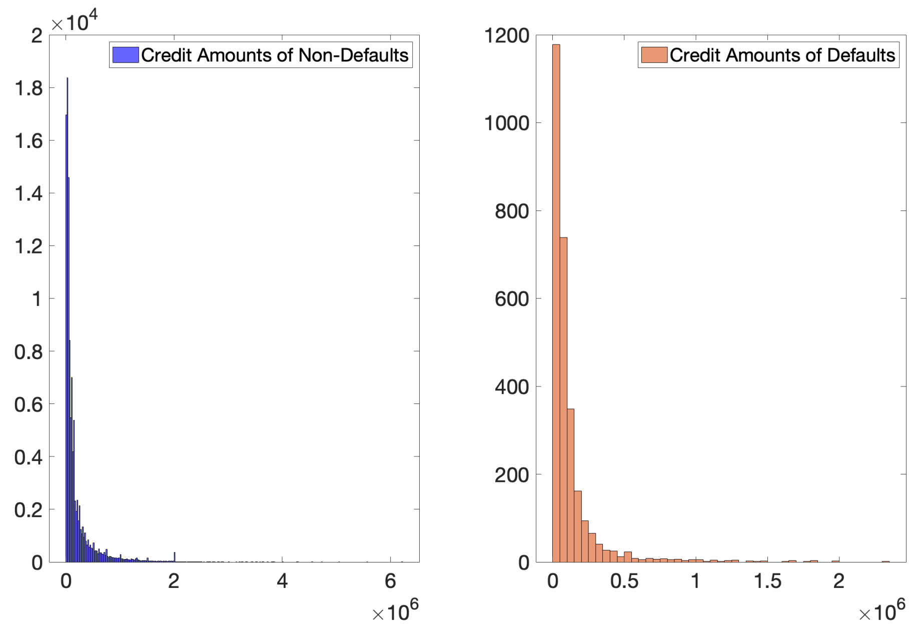


Figure 4.23: SBA National Loan Data$_{2004-2006}$
Histograms of Credit Amounts

The hyperparameters of ANN, AdaBoost and XGBoost are optimized with the target of maximum G_{mean} values in the test data and they are reported in tables 4.45-4.47. The hyperparameters vary in all models but all XGBoost algorithms have a sub-sample ratio of columns less than 1 and a maximum delta higher than 0. The optimal value of m is observed as $m > 1$ for C_1 and C_3, while $m < 1$ for C_2 for all cost-sensitive algorithms.

The algorithm classification results for the train and test data are listed in tables 4.48-4.50. Table 4.48 informs about the train and test data sizes, class ratio, and simulations with LR, GNN, and GAT models and also it presents the cost-insensitive algorithm results and cost-sensitive ANN model results. The cost-insensitive algorithms have higher *Specificity* values but lower *Specificity* values, therefore, *LGD* values are not reported for these ordinary algorithms. Table 4.49 shows the classification performances of boosting algorithms with the threshold of 0.5 and they are observed to be superior to the cost-sensitive ANN algorithms.

Table 4.45: SBA National Loan Data$_{2004-2006}$
ANN Optimized Hyperparameters

Activation Functions:	Sigmoid
Learning Rate:	0.001 and 0.01 for ANN with Focal Loss and Cross Entropy Loss, respectively.
Number of Hidden Layers:	64 32 8
Training Function:	Resilient Back Propagation
Increment to weight change:	1.2
Decrement to weight change:	0.5
Initial weight change:	0.07
Maximum weight change:	50.0
Classification Threshold	0.5

	Focal Loss Function					**Cross Entropy Loss Function**		
	Iterations	m	α	γ_1	γ_2	Iterations	m	$thrs$
ANN	500	-	1	1	1	2000	-	-
RCS-ANN1$_{LR}$	248	1.17	C_1	0.1	2	1000	1	-
RCS-ANN2$_{LR}$	-	-	-	-	-	2000	0.125	0.53
RCS-ANN3$_{LR}$	211	1.18	C_3	0.1	2	500	1	-
RCS-ANN1$_{GNN}$	149	1.09	C_1	0.1	2	1000	1	-
RCS-ANN2$_{GNN}$	-	-	-	-	-	2000	0.125	0.555
RCS-ANN3$_{GNN}$	174	1.16	C_3	0.1	2	1000	1	-
RCS-ANN1$_{GAT}$	177	1.11	C_1	0.1	2	1000	1	-
RCS-ANN2$_{GAT}$	-	-	-	-	-	2000	0.125	0.5
RCS-ANN3$_{GAT}$	166	1	$.165C_3$	0.1	2	1000	1	-

Table 4.46: SBA National Loan Data$_{2004-2006}$
AdaBoost Optimized Hyperparameters

	Iterations	m	$thrs$
RCS-AdaB1$_{LR}$	1000	1.0	-
RCS-AdaB2$_{LR}$	500	0.008	0.53
RCS-AdaB1$_{GNN}$	1000	1	-
RCS-AdaB2$_{GNN}$	1000	0.09	0.555
RCS-AdaB1$_{GAT}$	1000	1	-
RCS-AdaB2$_{GAT}$	1500	0.009	0.5

Table 4.47: SBA National Loan Data$_{2004-2006}$
XGBOOST Optimized Hyperparameters

Evaluation Metric: Log Loss/Cross Entropy Loss Function.

	Learning Rate:	Depth from a Root to Leaf	Sub-sample Ratio of Columns	Maximum Delta	Number of Estimators	Scale Positive Jobs	m	$thrs$
XGBoost	0.1	6	0.6	1	2000	-	-	
CatBoost	0.05	8	-	-	2000	-	-	-
LightGBM	0.1	-	-	-1000	-	-	-	
XGBoost$_{Scaled}$	0.05	5	0.4	1	400	40	-	-
CatBoost$_{Scaled}$	0.025	8	-	-	598	40	-	-
RCS-XGB1$_{LR}$	0.025	5	0.55	5	625	C_1	1.1	-
RCS-XGB2$_{LR}$	0.05	6	0.4	3	310	C_2	0.155	0.53
RCS-XGB3$_{LR}$	0.015	6	0.6	3	566	C_3	1.15	-
RCS-XGB1$_{GNN}$	0.025	5	0.5	4	620	C_1	1.1	-
RCS-XGB2$_{GNN}$	0.02	6	0.5	3	412	C_2	0.128	0.555
RCS-XGB3$_{GNN}$	0.015	6	0.6	2	572	C_3	1.2	-
RCS-XGB1$_{GAT}$	0.025	5	0.5	3	610	C_1	1.1	-
RCS-XGB2$_{GAT}$	0.05	6	0.4	2	358	C_2	0.16	0.5
RCS-XGB3$_{GAT}$	0.015	6	0.6	3	566	C_3	1.2	-

	Sampling Strategy	Random State	k Neighbors	Number of Estimators
SMOTHEBoost	Only Minority	$np.random$ generator of Python	5	2000
RUSBoost	Only Majority Class	$np.random$ generator of Python	-	2000

Table 4.48: SBA National Loan Data$_{2004-2006}$
Performances of Algorithms

Train Data Size=$114,815$, Test Data Size=$49,206$, Train Data/Test Data = $7/3$
Non-defaults/Defaults=40
Numbers of LR/GNN/GAT Models with Sub-Data Sets=$10,000/1,000/1,000$
Sub-data Sizes for LR/GNN/GAT Models with Sub-Data Sets=$5602/3,600/3,600$
LR: Models with ordinary $R^2_{ordinary} >= 0.45$, $R^2_{adjusted} >= 0.45$, $R^2_{AdjustedGeneralized} >= 0.49$
LR:$thrs = 0.53$ for C_2
Levels of Node Embeddings in GNN/GAT: $3/1$
β_{RAW} used in C_2 estimated with LR/GNN/GAT: $22.145/18.811/19.718$
GNN: Iterations= 100, $knn = 20$, $lR = 0.05$ $thrs = 0.555$ for C_2
GAT: Iterations=150, $knn = 20$, $K = 1$, $lR = 0.02$ $thrs = 0.5$ for C_2
Threshold for ANNs: 0.5

Cost-Insensitive Algorithms

	TRAIN DATA			TEST DATA		
Algorithms	Sens.	Spec.	G$_{mean}$	Sens.	Spec.	G$_{mean}$
LR	81.0	81.0	81.0	81.6	80.2	80.9
GNN	83.8	83.8	83.8	82.7	82.7	82.7
GAT	83.3	81.7	82.5	83.3	80.6	82.0
ANN$CrossEntropyLoss$	4.1	99.9	20.2	3.1	99.9	17.6
ANN$FocalLoss$	0.0	100	0	0.0	100	0
AdaBoost	16.4	99.7	40.4	16.9	99.7	41.1
XGBoost	91.0	100	95.40	36.6	99.3	60.3
CatBoost	60.2	99.8	77.5	35.8	99.4	59.7
LightGBM	97.2	99.9	98.6	34.2	99.4	58.3

Cost-Sensitive ANNs with Focal Loss Function

	TRAIN DATA			TEST DATA					
Algorithms	Sens.	Spec.	G$_{mean}$	Sens.	Spec.	G$_{mean}$	AUC	MC$^{x10^6}_{Y=1}$	Loss Given Default$^{x10^6}$
RCS-ANN1$_{LR}$	82.5	83.2	82.9	83.2	83.2	83.2	88.2	2.090	54.976
RCS-ANN2$_{LR}$	83.3	82.1	82.7	83.7	82.0	82.8	88.0	2.067	49.203
RCS-ANN1$_{GNN}$	83.2	82.6	82.9	82.9	82.6	82.9	87.9	2.516	55.355
RCS-ANN2$_{GNN}$	83.1	82.0	82.6	83.6	82.0	82.8	87.9	2.162	50.236
RCS-ANN1$_{GAT}$	81.8	83.5	82.6	82.4	83.5	83.0	88.0	2.463	51.640
RCS-ANN2$_{GAT}$	81.9	82.9	82.4	82.8	82.9	82.9	87.8	2.044	46.617

Cost-Sensitive ANNs with Cross Entropy Loss Function

	TRAIN DATA			TEST DATA					
	Sens.	Spec.	G$_{mean}$	Sens.	Spec.	G$_{mean}$	AUC	MC$^{x10^6}_{Y=1}$	Loss Given Default$^{x10^6}$
RCS-ANN1$_{LR}$	85.0	87.5	86.2	80.3	87.3	83.8	92.5	2.165	59.483
RCS-ANN2$_{LR}$	90.6	88.3	89.4	85.0	88.2	87.6	92.8	0.466	53.442
RCS-ANN3$_{LR}$	90.1	86.3	88.2	85.7	86.0	85.8	93.7	0.935	45.439
RCS-ANN1$_{GNN}$	90.3	83.4	84.8	86.5	83.2	84.8	93.0	0.824	44.570
RCS-ANN2$_{GNN}$	95.4	84.0	89.5	88.3	83.8	86.0	93.5	0.299	49.231
RCS-ANN3$_{GNN}$	87.3	88.0	87.6	83.3	87.7	85.5	93.8	1.400	51.826
RCS-ANN1$_{GAT}$	90.4	85.2	87.7	86.7	85.0	85.8	93.6	1.186	49.086
RCS-ANN2$_{GAT}$	89.9	89.3	89.6	84.1	89.1	86.5	94.6	0.955	56.490
RCS-ANN3$_{GAT}$	89.1	88.8	89.0	83.7	88.5	89.1	94.6	0.873	39.852

Table 4.49: SBA National Loan Data$_{2004-2006}$
Performances of Algorithms

Boosting Algorithms for Imbalanced Data

	TRAIN DATA			TEST DATA					
	Sens.	Spec.	G$_{mean}$	Sens.	Spec.	G$_{mean}$	AUC	MC$^{x10^6}_{Y=1}$	Loss Given Default$^{x10^6}$
SMOTEBoost	75.4	96.7	85.4	76.1	96.7	85.8	95.3	2.124	65.841
RUSBoost	93.9	89.5	54.6	92.8	89.4	91.1	95.7	0.984	27.490
XGBoost$_{scaled}$	96.4	92.7	94.6	92.8	92.6	92.7	97.4	0.532	26.252
CatBoost$_{scaled}$	97.2	92.6	94.9	92.7	92.5	92.6	97.3	0.542	31.162
RCS-Ada1$_{LR}$	89.7	90.1	89.9	90.8	90.3	90.5	96.8	1.018	29.969
RCS-Ada2$_{LR}$	90.4	90.4	90.4	91.8	90.4	91.1	96.8	1.107	34.063
RCS-Ada1$_{GNN}$	90.4	91.9	91.2	91.3	92.0	91.7	96.8	0.767	27.788
RCS-Ada2$_{GNN}$	91.8	90.2	91.0	93.1	90.3	91.7	96.7	0.405	24.753
RCS-Ada1$_{GAT}$	90.4	92.2	91.3	91.1	92.3	91.7	96.8	0.731	26.869
RCS-Ada2$_{GAT}$	91.9	90.7	91.3	92.9	90.8	91.8	96.8	0.404	22.336
RCS-XGB1$_{LR}$	95.6	93.1	94.3	92.7	92.9	92.8	97.3	0.461	22.592
RCS-XGB2$_{LR}$	94.8	92.0	93.4	91.8	91.8	91.8	96.5	0.386	39.445
RCS-XGB3$_{LR}$	96.5	93.0	94.7	92.9	93.0	92.9	97.4	0.499	24.699
RCS-XGB1$_{GNN}$	96.7	92.6	94.6	93.5	92.4	93.0	97.4	0.362	20.404
RCS-XGB2$_{GNN}$	95.3	92.3	93.8	92.1	92.2	92.2	97.0	0.542	38.109
RCS-XGB3$_{GNN}$	97.3	92.5	92.9	93.3	92.4	92.9	97.4	0.448	23.232
RCS-XGB1$_{GAT}$	96.7	92.4	94.5	93.3	92.2	92.8	97.4	0.455	23.268
RCS-XGB2$_{GAT}$	96.4	92.3	94.3	92.6	92.0	92.3	97.1	0.569	32.186
RCS-XGB3$_{GAT}$	97.1	92.7	94.9	93.1	92.6	92.8	97.4	0.484	23.992

The results of SMOTEBoost, RUSBoost, XGBoost$_{scaled}$, CatBoost$_{scaled}$ and cost-sensitive boosting algorithms presented in 4.49 are the first results of the algorithms that the probability predictions are classified with the threshold of 0.5. The prediction results of XGBoost$_{scaled}$ are superior to other existing ordinary algorithms with the highest G_{mean}. The prediction results for *Sensitivity-Specificity* and *LGD* values are analyzed with the thresholds of 0-4-0.6 since the maximums of G_{mean} values in k-50 fold cross-validations are observed in the test data.

Figures 4.25 and 4.26 display the maximum G$_{mean}$ values of the successful algorithms in the first graphs and the third graph of each figure shows how *LGD* values change as the threshold changes. The last graphs in these figures clearly demonstrate the instance-specific cost-sensitive modifications of XGBoost have relatively lower *LGD* values given *Specificity* values when compared with the successful ordinary algorithms of XGBoost$_{scaled}$, CatBoost$_{scaled}$. The monetary unit is not reported in the data set, therefore, *LGD* are reported in unit values. Considering the maximum G_{mean} values in the test data, XGBoost$_{scaled}$ can reach the highest G_{mean} value

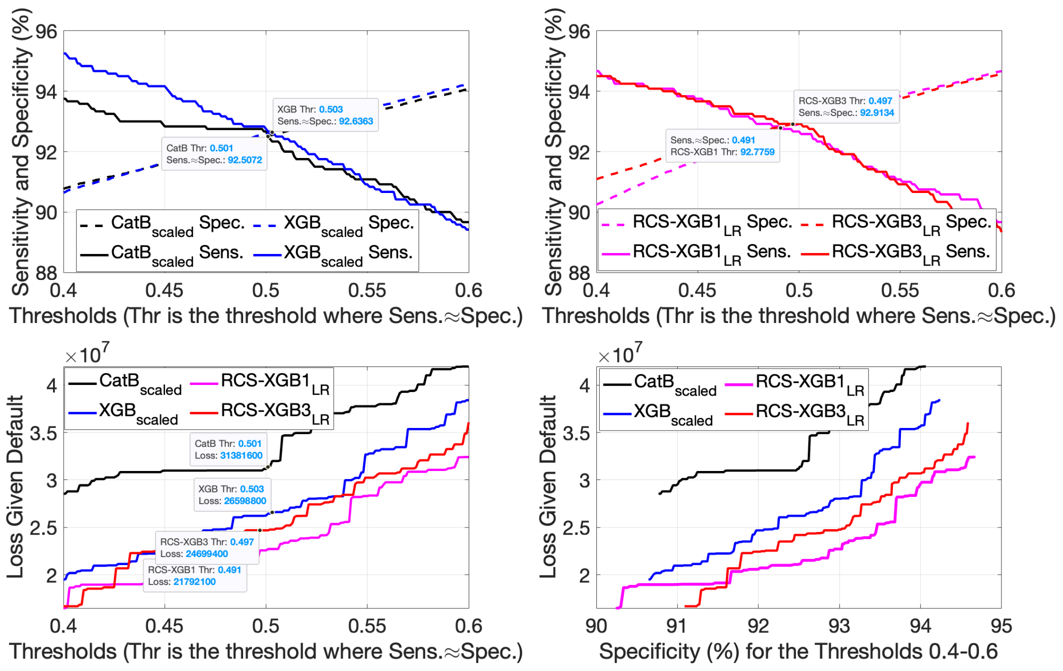


Figure 4.24: Cost Parameters are Estimated with LRs *Sensitivity-Specificity* and *LGD* with Different Thresholds in Test Data

as 92.6 where *LGD* is 26.599 million and comparing the proposed algorithms with XGBoost$_{scaled}$, XGB1$_{LR}$ reduces the loss by 4.807 million unit with G_{mean}/LGD: 92.8/21.792 million loss. This cost-saving corresponds to 18.1 % of the *LGD* value of XGBoost$_{scaled}$. Moreover, XGB3$_{LR}$ decreases the loss by 1.899 millions with G_{mean}/LGD: 92.9/24.699 million loss, XGB1$_{GNN}$ saves 3.313 millions unit with G_{mean}/LGD: 92.7/23.286 million loss, XGB3$_{GNN}$ drops the loss by 406 thousand unit with G_{mean}/LGD: 92.7/26.193 million loss, XGB1$_{GAT}$ leads a decrease in the loss by 620 thousand unit with G_{mean}/LGD: 92.8/25.979 million loss and XGB3$_{GAT}$ saves 1.450 millions unit with G_{mean}/LGD: 92.9/25.149 million loss.

k-50 fold validations are conducted and G_{mean} maximizing optimal thresholds for the validation data are computed. The medians of the thresholds are applied for the test data classification and Table 4.50 presents how the optimal thresholds of validation data result in *Sensitivity*-$Specifcity$ and *LGD* for the test data. XGB1$_{LR}$, XGB3$_{LR}$, XGB1$_{GNN}$, XGB3$_{GNN}$, XGB1$_{GAT}$ and XGB3$_{GAT}$ result in lower *LGD* values with the given $Specicificty$ values when compared to XGBoost$_{scaled}$ and CatBoost$_{scaled}$.

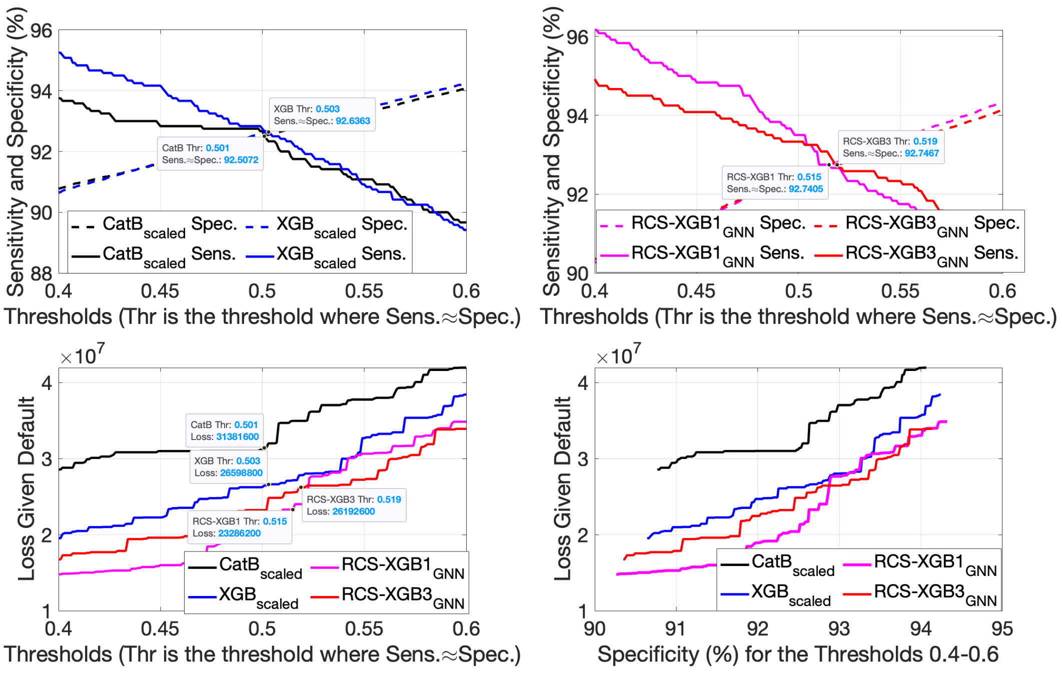


Figure 4.25: Cost Parameters are Estimated with GNNs
Sensitivity-Specificity and *LGD* with Different Thresholds in Test Data

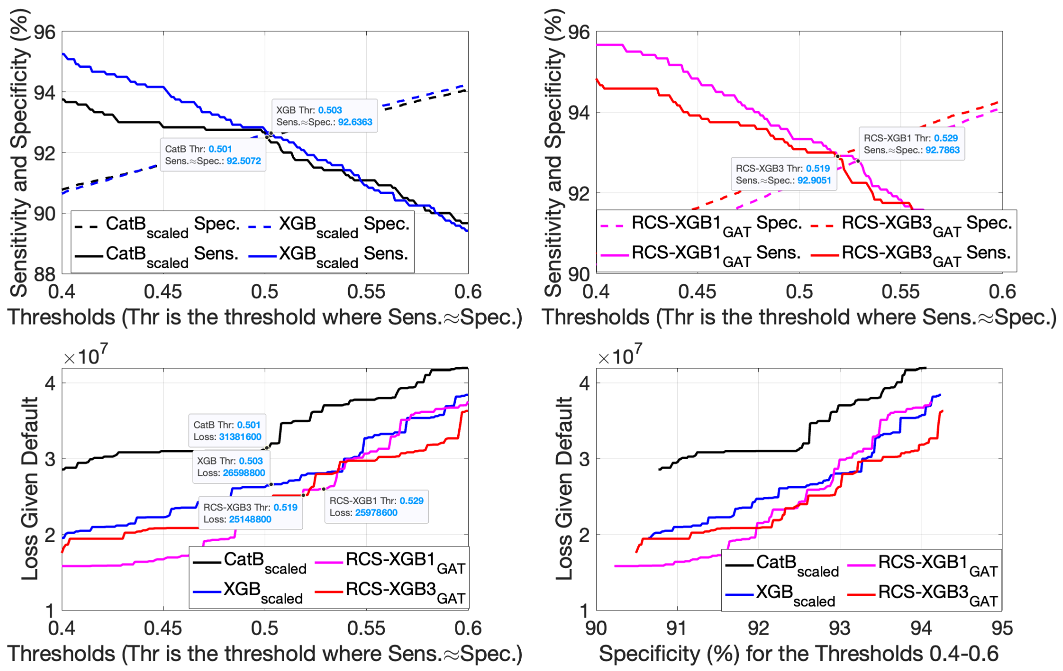


Figure 4.26: Cost Parameters are Estimated with GATs
Sensitivity-Specificity and *LGD* with Different Thresholds in Test Data

Table 4.50: SBA National Loan Data$_{2004-2006}$
Competing Performances of Algorithms After a Threshold Adjustment

	TRAIN DATA			TEST DATA					
Algorithms$_{Weights^1, thr^2}$	Sens.	Spec.	G$_{mean}$	Sens.	Spec.	G$_{mean}$	AUC	MC$^{x10^6}_{Y=1}$	Loss Given Default$^{x10^6}$
XGBoost$_{ClassRatio, thr=0.488}$	96.6	92.5	92.6	92.9	92.4	92.6	97.4	0.514	26.076
CatBoost$_{ClassRatio, thr=0.430}$	97.8	91.4	94.5	93.0	91.3	92.1	97.3	0.506	30.829
RCS-XGB1$_{LR, thr=0.453}$	96.5	91.9	94.2	93.7	91.8	92.7	97.3	0.331	20.373
RCS-XGB2$_{LR, thr=0.504}$	94.8	92.1	93.4	91.8	91.9	91.9	96.5	0.386	39.445
RCS-XGB3$_{LR, thr=0.460}$	97.2	92.4	94.8	93.5	92.3	92.9	97.4	0.421	23.524
RCS-XGB1$_{GNN, thr=0.470}$	97.2	91.9	94.5	94.8	91.8	93.3	97.4	0.187	16.233
RCS-XGB2$_{GNN, thr=0.45}$	95.8	91.6	93.7	92.8	91.5	92.1	97.0	0.430	36.162
RCS-XGB3$_{GNN, thr=0.483}$	97.4	92.2	94.8	93.6	92.1	92.8	97.4	0.416	22.454
RCS-XGB1$_{GAT, thr=0.483}$	97.0	92.0	94.5	93.8	91.9	92.9	97.4	0.310	19.590
RCS-XGB2$_{GAT, thr=0.459}$	98.8	91.6	94.2	92.8	91.4	92.0	97.1	0.531	31.482
RCS-XGB3$_{GAT, thr=0.486}$	97.4	92.5	94.9	93.6	92.3	92.9	97.4	0.415	22.593

[1] $P(D_{LR})$, $P(D_{GNN})$ and $P(D_{GAT})$ in weights are the default probabilities of the instances estimated with LR, GNNs, and GATs, respectively.
[2] Optimal Threshold: It is the median of G_{mean} maximizing thresholds of validation data sets predicted in k-50 fold cross-validations.

4.8 DATA SET 8: SBA National Loan Data$_{2002-2003}$

SBA National Loan Data$_{2002-2003}$ generated with the credits issued in 2002 and 2003 has a class ratio of 6.52. The default class instances are deleted with random selection and a data set with non-defaults/defaults of 50 is generated for severely imbalanced data. Furthermore, the reason for the significant decrease in the feature size is that some of the features are repeated in different formats in the data set. The deleted features are the same as in Data Set 7 as well as the feature properties. Table 4.51 presents the details of SBA National Loan Data$_{2002-2003}$ with the features found explanatory for the classification algorithms.

Table 4.51: SBA National Loan Data$_{2002-2003}$

Data Information

Original Data File Name: SBAnational				
Original Data Size	Data Size After the Cleansing	Number of Categorical Variables	Number of Numerical Variables	Class Ratio
102,584	95,932	11	15	50
8 features are found explanatory for the models, they are:				
Term - Credit Amount (Disbursement Gross) - Guarantee for the Loan by SBA (US Small Business Administration)-Days between the Approval and the Disbursement of the Loan - State - Number of Business Employees - The Number of Retained Jobs - Urban/Rural				

Figure 4.23 demonstrates the histograms of the credit amounts of non-defaults and defaults and their distributions are found to be expressed best by Gamma ($\alpha = 0.737$,

$\beta = 0.0507$) with $M = 1$ for the rejection sampling.

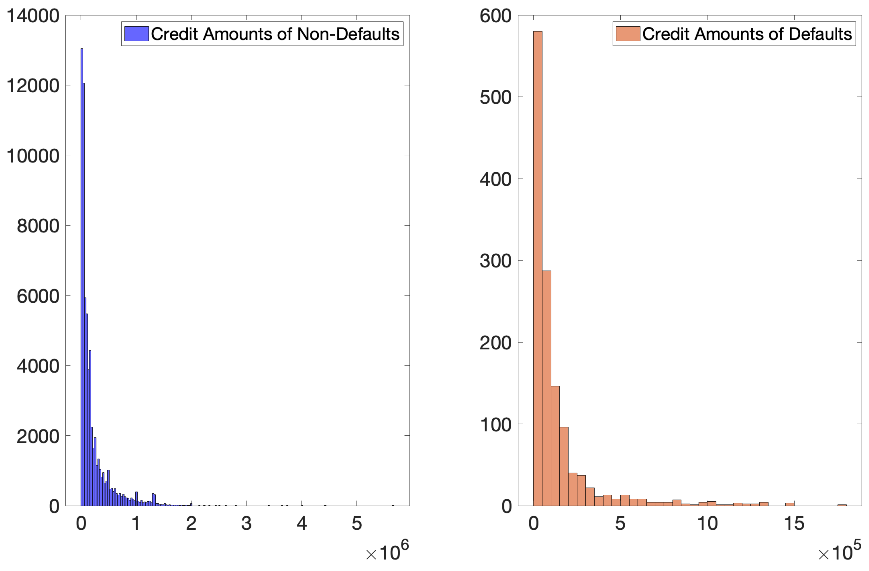


Figure 4.27: SBA National Loan Data$_{2002-2003}$
Histograms of Credit Amounts

Tables 4.52-4.54 present the optimized hyperparameters of ANN, AdaBoost, and XGBoost with the target of maximum G_{mean} values in the test data. The characteristics of the hyperparameters cannot be fully generalized but iteration numbers of cost-sensitive ANNs estimated with focal loss function are the lowest in algorithms. All XGBoost algorithms have a maximum delta higher than 0. The optimal value of m is observed to be as $m > 1$ for C_1 and C_3, while $m < 1$ for C_2 for all cost-sensitive algorithms.

Table 4.55 summarizes data-specific information, LR, GNN, and GAT models details and the cost-insensitive algorithm results as well as cost-sensitive ANN model results. The cost-insensitive algorithms have higher *Specificity* values but lower *Specificity* values, therefore, *LGD* values are not reported for these ordinary algorithms. Table 4.56 shows the classification performances of outperforming boosting algorithms when compared with the cost-sensitive ANN algorithms. The predictions of SMOTEBoost, RUSBoost, XGBoost$_{scaled}$, CatBoost$_{scaled}$, and cost-sensitive boosting algorithms presented in Table 4.56 are classified with a threshold of 0.5.

Table 4.52: SBA National Loan Data$_{2002-2003}$: ANN Optimized Hyperparameters

Activation Functions:	Sigmoid
Learning Rate:	0.001 and 0.01 for ANN with Focal Loss and Cross Entropy Loss, respectively.
Number of Hidden Layers:	64 32 8
Training Function:	Resilient Back Propagation
Increment to weight change:	1.2
Decrement to weight change:	0.5
Initial weight change:	0.07
Maximum weight change:	50.0
Classification Threshold	0.5

	Focal Loss Function					**Cross Entropy Loss Function**		
	Iterations	m	α	γ_1	γ_2	Iterations	m	thr
ANN	200	-	1	1	1	1000	-	-
RCS-ANN1$_{LR}$	150	1.11	C_1	0.1	2	500	1.1	-
RCS-ANN2$_{LR}$	-	-	-	-	-	500	0.1	0.51
RCS-ANN3$_{LR}$	131	1.12	C_3	0.1	2	500	1.2	-
RCS-ANN1$_{GNN}$	151	1.12	C_1	0.1	2	500	1.15	-
RCS-ANN2$_{GNN}$	-	-	-	-	-	500	0.118	0.55
RCS-ANN3$_{GNN}$	130	1.14	C_3	0.1	2	500	1.12	-
RCS-ANN1$_{GAT}$	144	1.114	C_1	0.1	2	500	1.15	-
RCS-ANN2$_{GAT}$	-	-	-	-	-	500	0.126	0.52
RCS-ANN3$_{GAT}$	135	1.154	C_3	0.1	2	500	1.27	-

Table 4.53: SBA National Loan Data$_{2002-2003}$
AdaBoost Optimized Hyperparameters

	Iterations	m	thr
RCS-AdaB1$_{LR}$	1500	1.0	-
RCS-AdaB2$_{LR}$	1500	0.05	0.5
RCS-AdaB1$_{GNN}$	1000	1.125	-
RCS-AdaB2$_{GNN}$	600	0.1	0.55
RCS-AdaB1$_{GAT}$	1000	1.115	-
RCS-AdaB2$_{GAT}$	1200	0.012	0.52

Table 4.54: SBA National Loan Data$_{2002-2003}$
XGBOOST Optimized Hyperparameters

XGBoost
Evaluation Metric: Log Loss/Cross Entropy Loss Function.

	Learning Rate:	Depth from a Root to Leaf	Sub-sample Ratio of Columns	Maximum Delta	Number of Estimators	Scale Positive Weight	m	$thrs$
XGBoost	0.1	6	1	1	2000	-	-	-
CatBoost	0.05	6	-	-	2000	-	-	-
LightGBM	0.05	-	-	-2000	-	-	-	
XGBoost$_{Scaled}$	0.01	5	1	1	598	50	-	-
CatBoost$_{Scaled}$	0.01	5	-	-	600	50	-	-
RCS-XGB1$_{LR}$	0.1	5	0.65	5	129	C_1	1.2	-
RCS-XGB2$_{LR}$	0.05	5	0.6	3	90	C_2	0.1	0.51
RCS-XGB3$_{LR}$	0.05	5	0.6	3	102	C_3	1.16	-
RCS-XGB1$_{GNN}$	0.05	5	1	1	246	C_1	1.22	-
RCS-XGB2$_{GNN}$	0.01	5	1	5	312	C_2	0.12	0.55
RCS-XGB3$_{GNN}$	0.02	5	1	4	165	C_3	1.15	-
RCS-XGB1$_{GAT}$	0.05	5	0.6	1	257	C_1	1.2	-
RCS-XGB2$_{GAT}$	0.01	5	1	2	482	C_2	0.152	
RCS-XGB3$_{GAT}$	0.02	5	1	6	173	C_3	1.2	-

	Sampling Strategy	Random State	k Neighbors	Number of Estimators
SMOTHEBoost	Only Minority	$np.random$ generator of Python	5	2000
RUSBoost	Only Majority Class	$np.random$ generator of Python	-	2000

Table 4.55: SBA National Loan Data$_{2002-2003}$
Performances of Algorithms

Train Data Size=$67,153$, Test Data Size=$28,779$, Train Data/Test Data = $7/3$
Non-defaults/Defaults=50
Numbers of LR/GNN/GAT Models with Sub-Data Sets=$10,000/1,000/1,000$
Sub-data Sizes for LR/GNN/GAT Models with Sub-Data Sets=$2,634/2,634/2,634$
LR: Models with R$^2_{ordinary}$ $>= 0.477$, R$^2_{adjusted}$ $>= 0.468$, R$^2_{adjustedGeneralized}$ $>= 0.540$
LR: Models with $thrs = 0.51$ for C_2
Levels of Node Embeddings in GNN/GAT: 3/1
β_{RAW} used in C_2 estimated with LR/GNN/GAT: $28.223/26.170/21.404$
GNN: Iterations= 80, $knn = 5$, $lR = 0.02$ $thrs = 0.55$ for C_2
GAT: Iterations=80, $knn = 10$, $K = 1$, $lR = 0.03$ $thrs = 0.52$ for C_2
Threshold for ANNs: 0.5

Cost-Insensitive Algorithms

	TRAIN DATA			TEST DATA		
Algorithms	Sens.	Spec.	G$_{mean}$	Sens.	Spec.	G$_{mean}$
LR	80.9	80.8	80.9	82.3	78.9	80.6
GNN	79.8	79.3	79.6	76.1	82.0	79.0
GAT	80.3	81.7	81.0	81.8	78.4	80.1
ANN$CrossEntropyLoss$	8.9	99.9	29.8	2.8	99ç9	16.8
ANN$FocalLoss$	0.00	100	0	0.00	100	0
AdaBoost	15.3	99.6	39.0	15.2	99.6	39.0
XGBoost	99.2	100	99.6	34.0	99.5	58.2
CatBoost	71.8	100	84.7	34.2	99.5	58.4
LightGBM	99.9	100	100	31.4	99.6	55.9

Cost-Sensitive ANNs with Focal Loss Function

	TRAIN DATA			TEST DATA					
Algorithms	Sens.	Spec.	G$_{mean}$	Sens.	Spec.	G$_{mean}$	AUC	MC$^{x10^6}_{Y=1}$	Loss Given Default$^{x10^6}$
RCS-ANN1$_{LR}$	81.9	79.3	80.6	81.4	79.6	80.5	87.5	1.153	22.507
RCS-ANN2$_{LR}$	81.9	78.6	80.3	81.2	78.9	80.0	87.2	1.225	22.358
RCS-ANN1$_{GNN}$	82.2	80.0	81.1	82.4	80.2	81.3	87.6	1.218	22.220
RCS-ANN2$_{GNN}$	82.5	78.7	80.6	81.7	79.1	80.4	87.3	1.116	22.150
RCS-ANN1$_{GAT}$	81.9	79.8	80.8	81.4	80.2	80.8	87.4	1.261	22.791
RCS-ANN2$_{GAT}$	80.0	80.7	80.3	80.1	81.0	80.5	87.2	1.283	22.811

Cost-Sensitive ANNs with Cross Entropy Loss Function

	TRAIN DATA			TEST DATA					
	Sens.	Spec.	G$_{mean}$	Sens.	Spec.	G$_{mean}$	AUC	MC$^{x10^6}_{Y=1}$	Loss Given Default$^{x10^6}$
RCS-ANN1$_{LR}$	88.6	80.8	84.6	81.0	80.6	80.5	85.8	0.640	22.815
RCS-ANN2$_{LR}$	90.4	80.2	85.1	81.4	80.2	80.8	87.1	0.416	25.885
RCS-ANN3$_{LR}$	89.0	79.1	83.9	82.8	79.2	81.0	87.0	0.556	19.136
RCS-ANN1$_{GNN}$	92.1	81.7	86.7	83.0	81.4	82.2	87.6	0.467	21.247
RCS-ANN2$_{GNN}$	89.9	81.7	85.7	82.4	82.0	82.2	86.4	0.359	29.021
RCS-ANN3$_{GNN}$	87.5	81.9	84.7	82.3	81.9	82.1	88.4	0.626	22.154
RCS-ANN1$_{GAT}$	90.8	82.6	86.6	81.2	82.6	81.9	87.7	0.715	23.821
RCS-ANN2$_{GAT}$	88.5	83.0	85.7	81.4	83.3	82.3	87.3	0.546	30.915
RCS-ANN3$_{GAT}$	91.6	82.0	86.7	83.0	82.1	82.5	87.8	0.617	25.647

Table 4.56: SBA National Loan Data$_{2002-2003}$
Performances of Algorithms

Boosting Algorithms for Imbalanced Data

	TRAIN DATA			TEST DATA					
	Sens.	Spec.	G$_{mean}$	Sens.	Spec.	G$_{mean}$	AUC	MC$_{Y=1}^{x10^6}$	Loss Given Default$^{x10^6}$
SMOTEBoost	68.6	97.4	81.8	68.6	97.3	81.7	96.2	5.567	35.506
RUSBoost	95.7	87.8	91.6	89.4	87.8	88.6	95.1	0.641	12.054
XGBoost$_{scaled}$	95.6	90.9	93.2	90.8	90.7	90.8	96.4	0.388	15.362
CatBoost$_{scaled}$	91.4	87.3	89.4	89.4	87.4	88.4	95.1	0.663	19.275
RCS-Ada1$_{LR}$	90.0	90.0	90.0	89.4	90.0	89.7	95.7	0.465	15.796
RCS-Ada2$_{LR}$	91.6	87.2	89.4	90.2	87.2	88.7	95.7	0.445	14.832
RCS-Ada1$_{GNN}$	91.4	89.4	90.4	90.4	89.3	89.9	96.2	0.329	15.125
RCS-Ada2$_{GNN}$	90.8	89.9	90.3	91.0	89.8	90.4	96.2	0.325	16.520
RCS-Ada1$_{GAT}$	91.7	89.7	90.7	91.3	89.8	90.5	96.1	0.319	14.311
RCS-Ada2$_{GAT}$	91.4	90.2	90.8	90.6	90.2	90.4	96.2	0.389	15.438
RCS-XGB1$_{LR}$	98.5	90.9	94.6	90.8	90.6	90.7	96.2	0.217	10.179
RCS-XGB2$_{LR}$	95.7	87.3	91.4	91.5	87.3	89.4	95.9	0.430	17.319
RCS-XGB3$_{LR}$	95.9	90.8	93.3	91.5	90.5	91.0	96.6	0.294	12.290
RCS-XGB1$_{GNN}$	97.2	90.4	93.7	91.5	90.2	90.8	96.7	0.293	12.898
RCS-XGB2$_{GNN}$	92.9	86.8	89.8	90.2	86.9	88.6	94.8	0.383	20.178
RCS-XGB3$_{GNN}$	93.6	90.6	92.1	91.0	90.3	90.6	96.4	0.364	14.360
RCS-XGB1$_{GAT}$	97.5	90.4	93.9	91.3	90.3	90.8	96.8	0.266	11.384
RCS-XGB2$_{GAT}$	93.5	87.6	90.5	90.8	87.6	89.2	95.7	0.474	20.539
RCS-XGB3$_{GAT}$	93.6	91.1	92.3	91.1	90.8	91.0	96.4	0.373	14.354

The success of cost-sensitive algorithms arises with lower *LGD* values given *Specificity* values which are clearly observed in the fourth graphs of figures 4.28, 4.28 and 4.29 when compared with higher *LGD* values give *Specificity* values of XGBoost$_{scaled}$ and CatBoost$_{scaled}$. The monetary unit is not reported in the data set, therefore, *LGD* are reported in unit values. XGBoost$_{scaled}$ has a higher G_{mean} value than CatBoost$_{scaled}$ but they are not superior to all cost-sensitive algorithms. The predictions reported with the maximum G_{mean} values in the test data indicate that the proposed algorithms decrease the costs when they are compared with G_{mean}/LGD: $90.8/15.362$ million reported by XGBoost$_{scaled}$. XGB1$_{LR}$ has G_{mean}/LGD: $90.7/10.179$ million and this result is evaluated as the best result. Since XGB1$_{LR}$ leads a reduction in the financial loss by 5.183 million which amounts to 33.7% of 15.362 million loss of XGBoost$_{scaled}$. Furthermore, XGB3$_{LR}$ decreases the loss by 1.321 millions with G_{mean}/LGD: $90.7/14.040$ million, XGB1$_{GNN}$ reduces the loss by 2.331 million with G_{mean}/LGD: $91.0/13.031$ million, XGB3$_{GNN}$ drops the loss by 817 thousands unit with G_{mean}/LGD: $90.6/14.545$ million, XGB1$_{GAT}$ saves 3.602 million with G_{mean}/LGD: $91.0/11.759$ million and XGB3$_{GAT}$ decreases the loss by 1.007 million with G_{mean}/LGD: $90.9/14.354$ million.

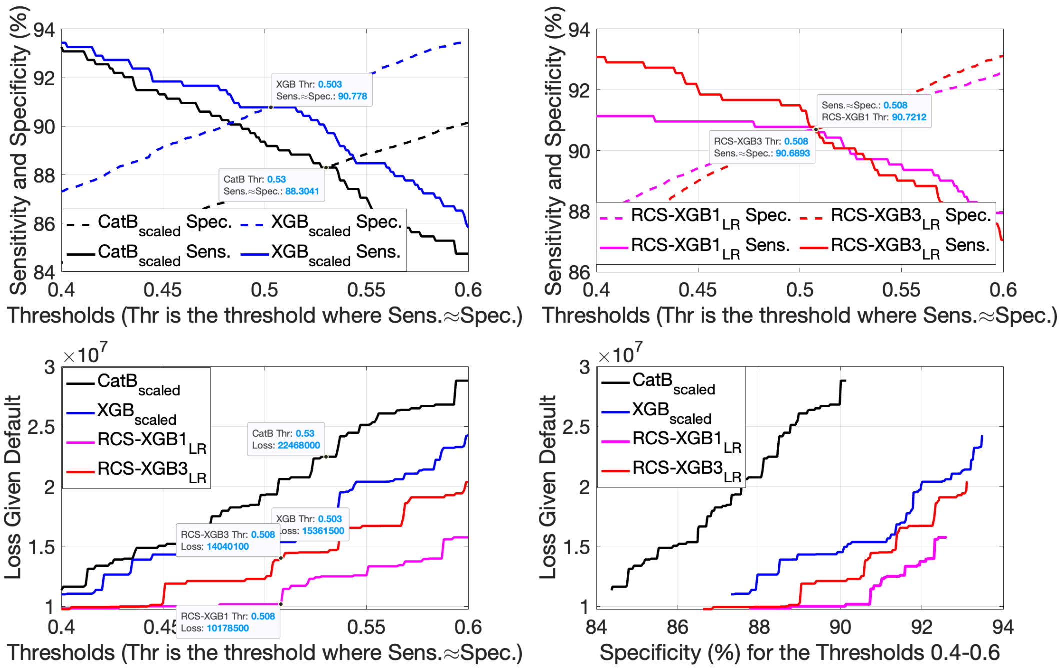


Figure 4.28: Cost Parameters are Estimated with LRs
Sensitivity-Specificity and *LGD* with Different Thresholds in Test Data

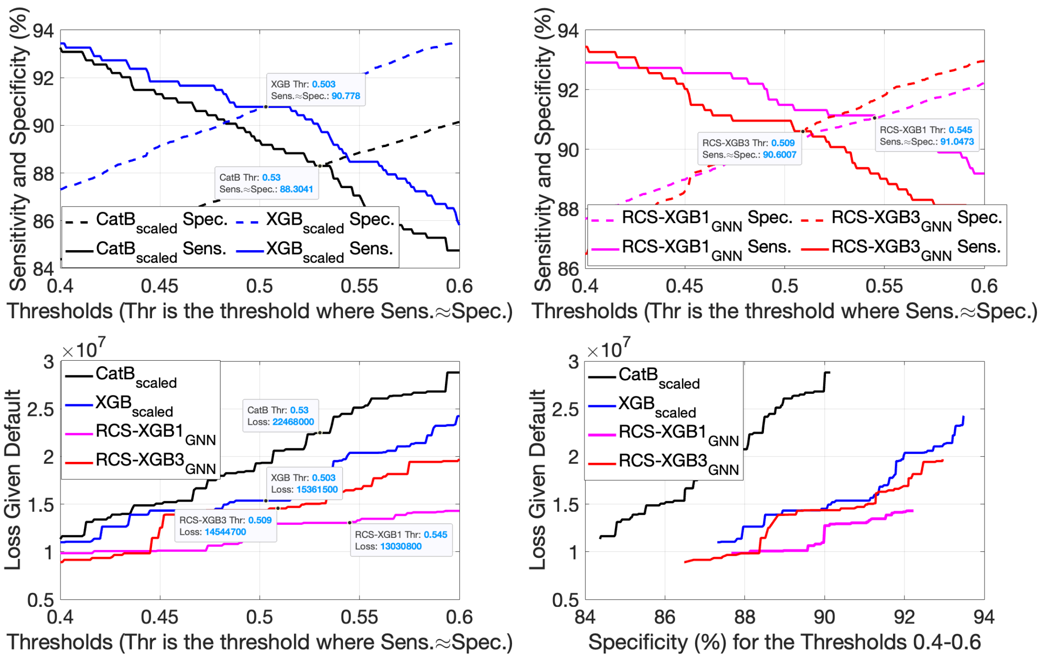


Figure 4.29: Cost Parameters are Estimated with GNNs
Sensitivity-Specificity and *LGD* with Different Thresholds in Test Data

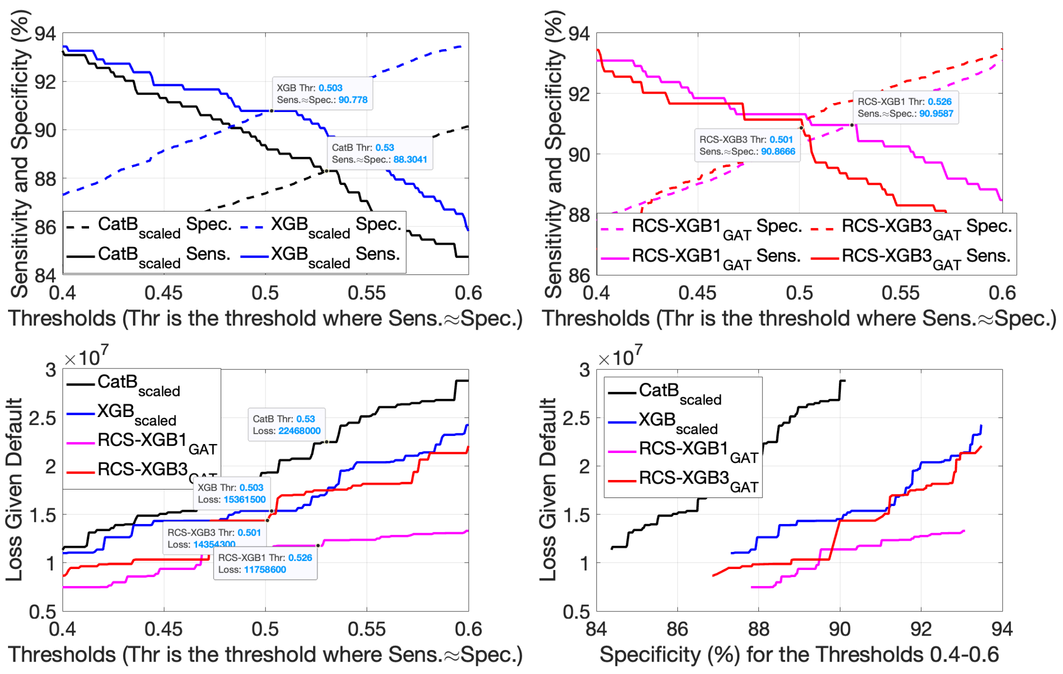


Figure 4.30: Cost Parameters are Estimated with GATs
Sensitivity-Specificity and *LGD* with Different Thresholds in Test Data

The thresholds maximizing G_{mean} values in the validation data in k-50 fold validations are analyzed and the medians of the thresholds are applied for the test data and Table 4.57 presents how the balanced classification thresholds of validation data result in *Sensitivity-Specifcity* and *LGD* for the test data. XGB1$_{LR}$, XGB3$_{LR}$, XGB1$_{GNN}$, XGB3$_{GNN}$, XGB1$_{GAT}$ and XGB3$_{GAT}$ result in lower *LGD* values with given *Specificty* values when compared to XGBoost$_{scaled}$ and CatBoost$_{scaled}$.

Table 4.57: SBA National Loan Data$_{2002-2003}$
Competing Performances of Algorithms After a Threshold Adjustment

	TRAIN DATA			TEST DATA					
Algorithms$_{Weights^1, thr^2}$	Sens.	Spec.	G$_{mean}$	Sens.	Spec.	G$_{mean}$	AUC	MC$^{x10^6}_{Y=1}$	Loss Given Default$^{x10^6}$
XGBoost$_{ClassRatio,thr=0.508}$	95.4	91.2	93.2	90.8	90.9	90.8	96.4	0.388	15.362
CatBoost$_{ClassRatio,thr=0.535}$	89.4	88.4	88.9	88.3	88.4	88.4	95.1	0.922	22.468
RCS-XGB1$_{LR,thr=0.508}$	98.3	91.0	94.6	90.8	90.7	90.8	96.2	0.217	10.179
RCS-XGB2$_{LR,thr=0.508}$	95.4	87.7	91.5	91.0	87.6	89.3	95.9	0.489	18.044
RCS-XGB3$_{LR,thr=0.507}$	95.9	91.0	93.4	90.8	90.7	90.7	96.6	0.407	14.040
RCS-XGB1$_{GNN,thr=0.530}$	97.0	91.0	94.0	91.1	90.8	90.7	96.7	0.312	13.031
RCS-XGB2$_{GNN,thr=0.56}$	91.8	87.8	89.8	88.7	87.9	88.3	94.8	0.617	23.052
RCS-XGB3$_{GNN,thr=0.511}$	93.5	90.9	92.2	90.6	90.7	90.7	96.4	0.387	14.548
RCS-XGB1$_{GAT,thr=0.488}$	97.5	90.1	93.7	91.3	90.0	90.7	96.8	0.266	11.383
RCS-XGB2$_{GAT,thr=0.463}$	94.0	87.0	90.4	91.3	87.0	89.1	95.7	0.352	18.301
RCS-XGB3$_{GAT,thr=0.498}$	93.5	90.8	92.1	91.1	90.8	91.0	96.4	0.373	14.354

[1] $P(D_{LR})$, $P(D_{GNN})$ and $P(D_{GAT})$ in weights are the default probabilities of the instances estimated with LR, GNNs, and GATs, respectively.
[2] Optimal Threshold: It is the median of G_{mean} maximizing thresholds of validation data sets predicted in k-50 fold cross-validations.

4.9 Discussion

Common performance metrics computed for the algorithm results are generally expressed as the percentage of correct identification of borrowers (instances) and they do not fully represent the true success of machine learning. The success of the algorithms should be evaluated by considering the financial losses in the credit dataset classifications. Therefore, the comparative analysis is predominantly done with the *LGD* values given the *Specificity* values. The proposed instance-specific cost-sensitive algorithms surpass XGBoost$_{scaled}$ and CatBoost$_{scaled}$ which are found to be the most successful ones as the ordinary algorithms.

Table 4.58 reports the financial savings of the proposed algorithms in test data prediction results. The cost savings or the decrease in the monetary losses are computed based on the monetary losses of the most successful ordinary algorithms, XGBoost$_{scaled}$ and CatBoost$_{scaled}$. In addition, the comparisons are based on the maximum G$_{mean}$ values observed in test data predictions, and *LGD* values are reported as units and percentages. The most significant findings are that RCS-XGB1$_{LR}$, RCS-XGB1$_{GNN}$ and XGB1$_{GNN}$ decrease the monetary loss by 33.7 $\%$, 23.4 $\%$, and 15.2 $\%$, respectively when compared with the loss of XGBoost$_{scaled}$ in SBA Loans$_{Y=2002-2003}$ data set which has the highest data imbalance among all data sets. The second best success of the proposed algorithms appears in the data set with the second highest class imbalance, SBA Loans$_{Y=2004-2006}$, in which RCS-XGB1$_{LR}$, RCS-XGB1$_{GNN}$ and RCS-XGB3$_{LR}$ lead a decrease in the monetary loss by 18.1 $\%$, 12.5 $\%$ and 7.13 $\%$, respectively. Furthermore, RCS-XGB2$_{LR}$, RCS-XGB3$_{GNN}$, XGB3$_{GNN}$ and XGB2$_{GNN}$ reduce the monetary loss by 16.3 $\%$, 15.9 $\%$, 14 $\%$ and 8.1 $\%$ in Irish Loan Data$_{2013}$.

The success of the instance-specific cost-sensitive models strictly depends on the success of the LRs, GNN, and GAT models in sub-datasets. If the LRs have higher R^2 values with higher *Sensitivity* and *Specificity* values in test data predictions, the cost parameters estimated with LRs are observed to lead to higher performance, and lower *LGD* given *Specificity* in the final predictions for the test data. Moreover, if the probability estimations of the sub-datasets are well-dispersed around their target labels and the classification is successful with *Sensitivity* and *Specificity* values of more than 85% in GNNs and GATs, the proposed algorithms can outperform the class ratio ad-

Table 4.58: Cost Savings of Instance-Specific Cost Sensitive Algorithms in Test Data Predictions Where the G$_{mean}$ Values are Maximized

Data Sets[1]	Data Size	Number of Features	Class Ratio	Successful Algorithms	Decrease in the Loss
Home Credit Risk D. Data	307,217	30	11.4:1	RCS-XGB3$_{LR}$	5.9 million-2.2 (%)
Risky Loans Data	231,285	13	8.8:1	RCS-XGB2$_{GNN}$	160.0 thousand-4.2 (%)
				RCS-XGB2$_{GNN}$	156.7 thousand-4.2 (%)
				RCS-XGB3$_{LR}$	119.1 thousand-3.2 (%)
				RCS-XGB3$_{GNN}$	101.1 thousand-2.7 (%)
Irish Loan Data$_{2013}$	124,381	12	15.0:1	RCS-XGB3$_{LR}$	374.4-thousand-16.3 (%)
				RCS-XGB3$_{GNN}$	364.3 thousand-15.9 (%)
				RCS-XGB3$_{GNN}$	322.3 thousand-14.0 (%)
				RCS-XGB2$_{GNN}$	185.4 thousand-8.1 (%)
				RCS-XGB2$_{GNN}$	181.2 thousand-7.9 (%)
				RCS-XGB1$_{GNN}$	96.0 thousand-4.2 (%)
Irish Loan Data$_{2014}$	235,626	12	9.5:1	RCS-XGB3$_{GNN}$	594.0 thousand-6.7 (%)
				RCS-XGB3$_{LR}$	590.7 thousand-6.7 (%)
				RCS-XGB1$_{GNN}$	440.1 thousand-5.0 (%)
				RCS-XGB3$_{GNN}$	306.1 thousand-3.5 (%)
				RCS-XGB2$_{GNN}$	290.3 thousand-3.3 (%)
				RCS-XGB2$_{GNN}$	190.3 thousand-2.1 (%)
Irish Loan Data$_{2015}$	421,092	11	33.8:1	RCS-XGB3$_{GNN}$	233 thousand-3.0 (%)
				RCS-XGB2$_{GNN}$	129.6 thousand-0.3 (%)
SBA N. Loans$_{2004-2006}$	164,021	8	40.0:1	RCS-XGB1$_{LR}$	4.807 million-18.1 (%)
				RCS-XGB1$_{GNN}$	3.313 million-12.5 (%)
				RCS-XGB3$_{LR}$	1.899 million-7.1 (%)
				RCS-XGB3$_{GNN}$	1.450 million-5.5 (%)
				RCS-XGB1$_{GNN}$	620 thousand-2.3 (%)
				RCS-XGB3$_{GNN}$	406 thousand-1.5 (%)
SBA N. Loans$_{2002-2003}$	95,932	8	50.0:1	RCS-XGB1$_{LR}$	5.183 million-33.7 (%)
				RCS-XGB1$_{GNN}$	3.602 million-23.4 (%)
				RCS-XGB1$_{GNN}$	2.331 million-15.2 (%)
				RCS-XGB3$_{LR}$	1.321 million-8.6 (%)
				RCS-XGB3$_{GNN}$	1.007 million-6.6 (%)
				RCS-XGB3$_{GNN}$	817 thousand-5.3 (%)

[1]Source: Kaggle.com.

justed XGBoost and CatBoost. The aggregated results of LRs, GNNs, and GATs of sub-datasets in Irish Loan Data$_{Y=2015}$ have *Sensitivity* and *Specificity* values lower than 85 % and the proposed cost-sensitive parameters are not successful as they are in other data sets.

It is also important to underline the fact that if LR, GNN, and GAT models can not exceed $80 - 82\%$ for *Sensitivity-Specificity* in both train and test data, this does not indicate these models are not well-designed at all. Some data sets have a high number of instances that are hard to separate for both classes such as default instances with the features signaling he/she is not to default with default probability close to 0, or vice versa. These inseparable instances for both classes can also reduce the success of the sub-data models in simulations. Furthermore, the credit amount disbursed to the borrower is one of the important explanatory features indicating the financial cost for the default instances and it is included in seven data sets. Data Set 6: Freddie Mac Single Family Loan does not report the credit amount in its feature list. The proposed borrower-specific cost-sensitive algorithms do not considerably improve the test data predictions in this data set and the lack of this attribute might prevent good estimations for the default probabilities and finally the cost-sensitive algorithms.

The sub-datasets of LR, GNN, and GAT model simulations differ and the success of LRs can also be attributed to these dimensional choices of the sub-datasets. The reason for the higher sub-data sizes for LR simulations is that all non-default instances are included in all sub-datasets and there is no time problem for test data predictions with the estimated LR models. However, the sizes of the sub-datasets in GNN and GAT models are lower due to the longer time required for the computation of the test data adjacency matrix with the search of the neighbors in the trained sub-dataset. The lower the dimension of the sub-datasets, the quicker to estimate the test data predictions which is essential for controlling the over-fitting problem of the GNN and GAT models. Since GNN and GAT models can easily overfit in the train data, these models are very sensitive to neighbors in the clustering, the node-embedding levels, learning parameter sizes, learning rate, number of epochs, and the sub-data sizes. An increase in any of these parameters causes the model to memorize the details in the train data and the optimized parameters through the loss minimization process in the train data estimations can not be generalized for the new data instances.

To generate well-functioning instance-specific cost parameters, it is significant to avoid the discrepancy between the train and test data predictions since the default probabilities in cost parameters are extracted from the train data estimates. The over-fitted train data default estimates can not generate cost parameters identifying the test data. To illustrate this crucial point, one example from the experimental stage can be given. GNN models of Risky Loans Data are estimated with two different designs. In the first one, k-NN=20, iteration=60, node embeddings=2, and learning parameter size=60×60 are selected. The aggregated prediction results with the median values of predictions are observed to be *Sensitivity*$_{Train}$-*Specificity*$_{Train}$: 94.7-94.7, *Sensitivity*$_{Test}$-*Specificity*$_{Test}$: 93.9-90.2. In the second design, the parameters are adjusted as k-NN=5, iteration=25, node embeddings=1, learning parameter size=30×30 and the aggregated prediction results with the median values of predictions are *Sensitivity*$_{Train}$-*Specificity*$_{Train}$: 89.9-90.0, *Sensitivity*$_{Test}$-*Specificity*$_{Test}$-*Specificity*$_{Test}$: 89.8-88.1.

Figure 4.31 demonstrates the sub-dataset results for both GNN designs on four graphs. The blue dots display the first model results and the red dots display the second model results. The first and the second graphs display the *Sensitivity* and the *Specificity* values for the train data, and the third and the fourth graphs show the same metrics for the test data. *Sensitivity* and the *Specificity* values in the first two graphs are relatively higher in the first model design whereas *Sensitivity* values for the test data set are also higher in the first model design but *Specificity* values of the first model are not high as they are in the train data. They drop to lower levels compared to the second model results. This issue remarks that the GNN models with the first design learn the train data very well but they cannot make generalizations for the test data instances. Moreover, the aggregated median value of prediction data results of the first model has a higher discrepancy between the train and test data results. It is crucial to notice that the proposed cost-sensitive parameters are computed with the $P(D)$ of the train data. Over-estimations of the $P(D)$ in train data do not reflect the real values for the borrowers' default risk levels. Over-fitting in the train data results, in other words, the higher discrepancy between the train and test data predictions in LRs, GNNs, and GATs prevents the generation of a generalized weight adjustment mechanism for the test data instances.

Figure 4.32 displays *LGD* given *Specificity* values for both GNN model designs. *LGD* given *Specificity* graph given on the left displays that the instance-specific cost parameters are not well-operating. Even though the prediction performances in test data are superior in the previous estimations, the decrease in the discrepancy in the train and

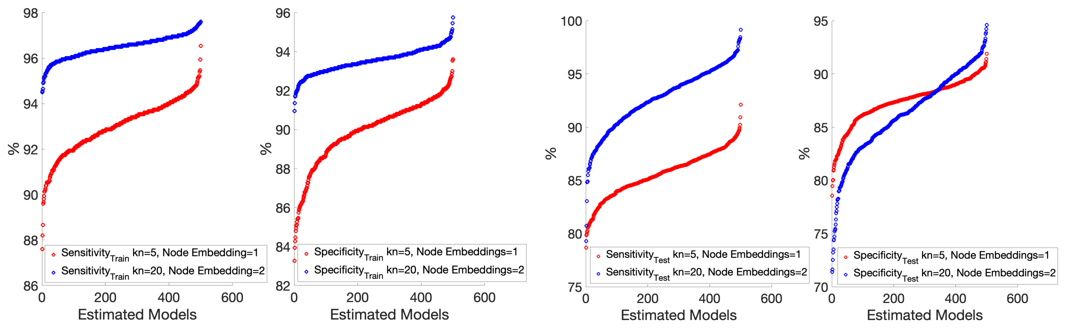


Figure 4.31: *Sensitivity-Specificity*Values Over-fitted and Not Over-Fitted GNN Models

test data predictions given in the second model design has been reflected in the cost parameters. The graph in the right column of Figure 4.32 shows that *LGD Specificity* values lie behind the ones of XGBoost$_{scaled}$ and CatBoost$_{scaled}$ when the cost parameters are estimated with the GNN's second model design.

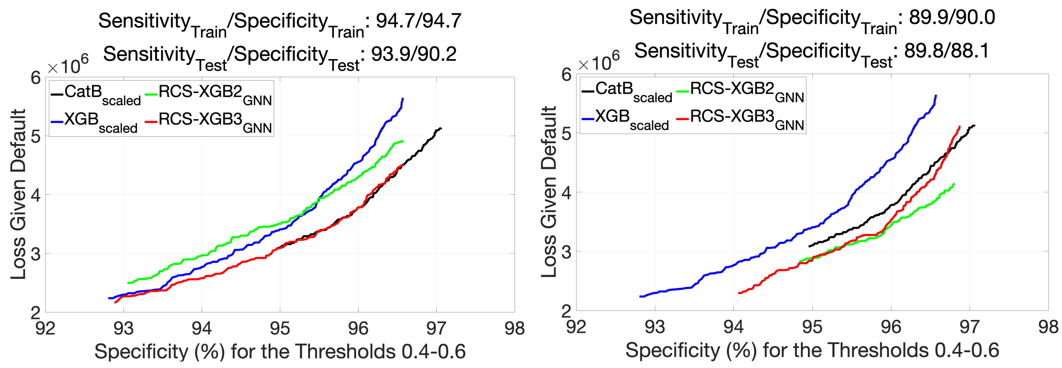


Figure 4.32: The Effects of the Cost Parameters Estimated with Over-fitted and Not Over-Fitted GNN Models

The experimental studies of eight data sets are conducted with MacBook Pro(M1, 2020)-Version 12.0.1. MATLAB 2021b is used for AdaBoost and ANN and their cost-sensitive modifications and PYTHON 3.7-Spider is used for XGBoost and its cost-sensitive modifications. The cost parameters are estimated faster with LRs when compared with ones estimated by GNNs and GATs. The monetary cost saving without decreasing the number of correctly identified non-defaults is the most principal target of the lender. If LR models can not be estimated with higher *Sensitivity* and *Specificty* values in the test data, the generation of GNN and GAT models will certainly increase the computation time but the lenders are more interested in the minimization of the monetary loss.

CHAPTER 5

CONCLUSION

Consumer credit risk in the financial sector appears as a binary classification problem in which risk level changes for every instance. The amount of the credit line and the interest rate charged change for each individual, therefore, the expected earnings and possible defaults differ as well. Credit risk identification differs from other binary classifications with its imbalance data problem and requires a cost-sensitive approach for the minority class of default instances.

The related literature suggests different pre-algorithm levels and various distinct hybrid models of combining sampling and classical algorithms for imbalance data problems, but there are no sophisticated, cost-sensitive algorithm recommendations attaching a specific valuation to each instance remarking its default risk level. The use of the expected nominal credit amount of each instance might be misleading in instance-specific cost-sensitive algorithms since the high amount of credit with low default probability can cause the applicant to be assessed in the risky group. The monetary values should be better evaluated with relative indicators, and the high credit amount might not indicate a high-risk level if the applicant already has a high income and enough assets to repay the loan.

This study presents unique instance-specific risk identifiers to be used in classical algorithms. The method proposed with its distinctive approach to the classification problem of imbalanced data has not been encountered in the literature. The methodology does not depend on trying different constant positive scalars for the out-weighting of the minority class with grid-search applications. The proposed methodology focuses on the extraction of an original weight parameter that is specific to each data instance. Each data uses the relevant information of its features and takes the new weight following its relative risk in the whole data set. The use of the 'cost/risk parameter' refers to the relative default risk of the instances/borrowers and implies that the misclassification errors do not incur the same level of cost but one is significantly more costly or fatal. Therefore, the main goal is to shift the priority of the algorithm toward this more important class without distorting the identification of other classes severely.

Equalizing the classes not on a class size basis but a relative default risk basis is the crucial part distinguishing this study from other studies dealing with imbalanced data. The default probabilities of each borrower are used for the default risk level indicators. LRs and deep learning methods of GNN and GAT models are used for the estimations of default probabilities. The default probabilities are insufficient to boost the minority class alone and therefore, relative total class-based default risk levels are incorporated in the new cost/risk parameter to strengthen the minority class instance weights and equalize the loss function for both classes. The second and third cost parameters are generated in the settings of AdaBoost cost-sensitive modifications offered in the literature.

The generated instance/borrower-specific cost/risk parameters are embedded in the classical algorithms through the initial instance weight initialization and instance-specific error re-weighting in cross entropy and focal loss functions. The weights of the minority class have to be boosted and high default probabilities of the default instances are compatible with this weight adjustment process. Therefore, outweighing the instances according to their risk levels is very rational; the real effect is that the share of the majority class in the total loss decreases while the share of the minority one increases.

The instance/borrower-specific cost-sensitive modifications of AdaBoost, ANN, and XGBoost are analyzed with their cost-insensitive designs. The empirical results indicate that class weight-adjusted XGBoost and CatBoost outperform the ordinary algorithms in all data sets and the proposed models are compared with these algorithms. The algorithm results are analyzed with thresholds ranging between 0.4-0.6 with *LGD* values and correct predictions in the non-default class. The success of the proposed borrower-specific cost-sensitive algorithms is directly related to sub-train data probability estimates which are not over-fitted and also have *Sensitivity-Specificity* values higher than 85 $\%$ in test data predictions.

The performance of the measure of the new models is not only the *Sensitivity-Specificity* values but the real financial burden caused by *LGD* and the corresponding *Specificity* value. The actual success of the instance-specific cost-sensitive algorithms in identifying the defaults becomes more remarkable as the class ratios in the data set increase. In other words, the proposed instance-specific cost-sensitive algorithms can even classify the minority and the majority class more successfully as the data sets become more unbalanced with lower *LGD* values given the *Specificity* values in test data. The empirical findings prove the success of the proposed methodology by decreasing the monetary loss by 18$\%$ and 33.7 $\%$ when the non-defaults to default ratio is 40 and 50, respectively.

REFERENCES

[1] Freddie Mac Single Family Loan, `https://www.kaggle.com/ankursg8/loan-delinquency-prediction`, [accessed: 15.05.2023].

[2] Home Credit Default Risk Data, `https://www.kaggle.com/competitions/home-credit-default-risk/data`, [accessed: 15.05.2023].

[3] Irish Loan Data, `https://www.kaggle.com/ikpeleambrose/irish-loan-data`, [accessed: 15.05.2023].

[4] Risky Loans Data, `https://www.kaggle.com/code/harshitlakhani/credit-risk-eda-on-risky-loans/input?select=LCDataDictionary.xlsx`, [accessed: 15.05.2023].

[5] SBA National Loan Data, `https://www.kaggle.com/datasets/mirbektoktogaraev/should-this-loan-be-approved-or-denied?select=SBAnational.csv`, [accessed: 15.05.2023].

[6] M. Ala'raj, M. F. Abbod, and M. Majdalawieh, Modelling customers credit card behaviour using bidirectional lstm neural networks, Journal of Big Data, 8(1), pp. 1–27, 2021.

[7] S. Ayouche, R. Aboulaich, and R. Ellaia, Partnership credit scoring classification probem: A neural network approach, International Journal of Applied Engineering Research, 12(5), pp. 693–704, 2017.

[8] A. C. Bahnsen, D. Aouada, and B. Ottersten, Example-dependent cost-sensitive logistic regression for credit scoring, in *2014 13th International conference on machine learning and applications*, pp. 263–269, IEEE, 2014.

[9] A. C. Bahnsen, A. Stojanovic, D. Aouada, and B. Ottersten, Cost sensitive credit card fraud detection using bayes minimum risk, in *2013 12th international conference on machine learning and applications*, volume 1, pp. 333–338, IEEE, 2013.

[10] A. Bansal, M. Goldblum, V. Cherepanova, A. Schwarzschild, C. B. Bruss, and T. Goldstein, Metabalance: High-performance neural networks for class-imbalanced data, arXiv preprint arXiv:2106.09643, 2021.

[11] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, Algorithms for hyper-parameter optimization, Advances in neural information processing systems, 24, 2011.

[12] U. Brefeld, P. Geibel, and F. Wysotzki, Support vector machines with example dependent costs, in *European Conference on Machine Learning*, pp. 23–34, Springer, 2003.

[13] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, Smote: synthetic minority over-sampling technique, Journal of artificial intelligence research, 16, pp. 321–357, 2002.

[14] T. Chen and C. Guestrin, Xgboost: A scalable tree boosting system, in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pp. 785–794, 2016.

[15] Y. Chen, L. Wu, and M. Zaki, Iterative deep graph learning for graph neural networks: Better and robust node embeddings, Advances in neural information processing systems, 33, pp. 19314–19326, 2020.

[16] W.-L. Chiang, X. Liu, S. Si, Y. Li, S. Bengio, and C.-J. Hsieh, Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks, in *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 257–266, 2019.

[17] A. S. Desuky and S. Hussain, An improved hybrid approach for handling class imbalance problem, Arabian Journal for Science and Engineering, 46(4), pp. 3853–3864, 2021.

[18] P. Domingos, Metacost: A general method for making classifiers cost-sensitive, in *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 155–164, 1999.

[19] Y. Dou, Z. Liu, L. Sun, Y. Deng, H. Peng, and P. S. Yu, Enhancing graph neural network-based fraud detectors against camouflaged fraudsters, in *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pp. 315–324, 2020.

[20] W. Fan, S. J. Stolfo, J. Zhang, and P. K. Chan, Adacost: misclassification cost-sensitive boosting, in *Icml*, volume 99, pp. 97–105, Citeseer, 1999.

[21] Z. Feng, Solutions to binary imbalanced classification.

[22] J. Frery, A. Habrard, M. Sebban, and L. He-Guelton, Non-linear gradient boosting for class-imbalance learning, in *Second International Workshop on Learning with Imbalanced Domains: Theory and Applications*, pp. 38–51, PMLR, 2018.

[23] Y. Freund and R. E. Schapire, A decision-theoretic generalization of on-line learning and an application to boosting, Journal of computer and system sciences, 55(1), pp. 119–139, 1997.

[24] J. Friedman, T. Hastie, and R. Tibshirani, Special invited paper. additive logistic regression: A statistical view of boosting, Annals of statistics, pp. 337–374, 2000.

[25] A. Ghosh, P. Mal, and A. Majumdar, *Advanced optimization and decision-making techniques in textile manufacturing*, Crc Press, 2019.

[26] L. Gong and Q. Cheng, Exploiting edge features for graph neural networks, in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 9211–9219, 2019.

[27] P. Goyal and E. Ferrara, Graph embedding techniques, applications, and performance: A survey, Knowledge-Based Systems, 151, pp. 78–94, 2018.

[28] K. Halteh, K. Kumar, and A. Gepp, Using cutting-edge tree-based stochastic models to predict credit risk, Risks, 6(2), p. 55, 2018.

[29] W. Hamilton, Z. Ying, and J. Leskovec, Inductive representation learning on large graphs, Advances in neural information processing systems, 30, 2017.

[30] W. L. Hamilton, Graph representation learning, Synthesis Lectures on Artifical Intelligence and Machine Learning, 14(3), pp. 1–159, 2020.

[31] S. Hamori, M. Kawai, T. Kume, Y. Murakami, and C. Watanabe, Ensemble learning or deep learning? application to default risk analysis, Journal of Risk and Financial Management, 11(1), p. 12, 2018.

[32] H. Han, W.-Y. Wang, and B.-H. Mao, Borderline-smote: a new over-sampling method in imbalanced data sets learning, in *International conference on intelligent computing*, pp. 878–887, Springer, 2005.

[33] H. He, W. Zhang, and S. Zhang, A novel ensemble method for credit scoring: Adaption of different imbalance ratios, Expert Systems with Applications, 98, pp. 105–117, 2018.

[34] Q. He and C. Qin, Adaptive optimization swarm algorithm ensemble model applied to the classification of unbalanced data, Intelligent Information Management, 13(5), pp. 251–267, 2021.

[35] Y. Ho and S. Wookey, The real-world-weight cross-entropy loss function: Modeling the costs of mislabeling, IEEE access, 8, pp. 4806–4813, 2019.

[36] S. Höppner, B. Baesens, W. Verbeke, and T. Verdonck, Instance-dependent cost-sensitive learning for detecting transfer fraud, European Journal of Operational Research, 2021.

[37] M. S. Hossain, J. M. Betts, and A. P. Paplinski, Dual focal loss to address class imbalance in semantic segmentation, Neurocomputing, 462, pp. 69–87, 2021.

[38] S. Hu, Y. Liang, L. Ma, and Y. He, Msmote: Improving classification performance when training data is imbalanced, in *2009 second international workshop on computer science and engineering*, volume 2, pp. 13–17, IEEE, 2009.

[39] Z. Huang, Y. Tang, and Y. Chen, A graph neural network-based node classification model on class-imbalanced graph data, Knowledge-Based Systems, 244, p. 108538, 2022.

[40] D. ILTER and O. KOCADAGLI, Credit scoring by artificial neural networks based crossentropy and fuzzy relations., Sigma: Journal of Engineering & Natural Sciences/Mühendislik ve Fen Bilimleri Dergisi, 2019.

[41] A. Iranmehr, H. Masnadi-Shirazi, and N. Vasconcelos, Cost-sensitive support vector machines, Neurocomputing, 343, pp. 50–64, 2019.

[42] S. Kang, K-nearest neighbor learning with graph neural networks, Mathematics, 9(8), p. 830, 2021.

[43] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, Lightgbm: A highly efficient gradient boosting decision tree, Advances in neural information processing systems, 30, 2017.

[44] I. Landesa-Vázquez and J. L. Alba-Castro, Shedding light on the asymmetric learning capability of adaboost, Pattern Recognition Letters, 33(3), pp. 247–255, 2012.

[45] J. W. Lee, W. K. Lee, and S. Y. Sohn, Graph convolutional network-based credit default prediction utilizing three types of virtual distances among borrowers, Expert Systems with Applications, 168, p. 114411, 2021.

[46] M. Leo, S. Sharma, and K. Maddulety, Machine learning in banking risk management: A literature review, Risks, 7(1), p. 29, 2019.

[47] H. Li, Y. Cao, S. Li, J. Zhao, and Y. Sun, Xgboost model and its application to personal credit evaluation, IEEE Intelligent Systems, 35(3), pp. 52–61, 2020.

[48] Q. Li, C. Zhao, X. He, K. Chen, and R. Wang, The impact of partial balance of imbalanced dataset on classification performance, Electronics, 11(9), p. 1322, 2022.

[49] Y. Li and W. Chen, A comparative performance assessment of ensemble learning for credit scoring, Mathematics, 8(10), p. 1756, 2020.

[50] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, Focal loss for dense object detection, in *Proceedings of the IEEE international conference on computer vision*, pp. 2980–2988, 2017.

[51] W. Lin, P. Wu, and X. Xiao, Boundary focal loss for class imbalanced learning, in *2021 14th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)*, pp. 1–5, IEEE, 2021.

[52] L. Liu and H. Qi, Learning effective binary descriptors via cross entropy, in *2017 IEEE winter conference on applications of computer vision (WACV)*, pp. 1251–1258, IEEE, 2017.

[53] W. Liu, H. Fan, M. Xia, and M. Xia, A focal-aware cost-sensitive boosted tree for imbalanced credit scoring, Expert Systems with Applications, 208, p. 118158, 2022.

[54] X.-Y. Liu, J. Wu, and Z.-H. Zhou, Exploratory undersampling for class-imbalance learning, IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 39(2), pp. 539–550, 2008.

[55] Y. Liu, X. Ao, Z. Qin, J. Chi, J. Feng, H. Yang, and Q. He, Pick and choose: a gnn-based imbalanced learning approach for fraud detection, in *Proceedings of the Web Conference 2021*, pp. 3168–3177, 2021.

[56] Y. Ma, S. Wang, C. C. Aggarwal, and J. Tang, Graph convolutional networks with eigenpooling, in *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 723–731, 2019.

[57] M. Mahbobi, S. Kimiagari, and M. Vasudevan, Credit risk classification: an integrated predictive accuracy algorithm using artificial and deep neural networks, Annals of Operations Research, pp. 1–29, 2021.

[58] F. Manessi, A. Rozza, and M. Manzo, Dynamic graph convolutional networks, Pattern Recognition, 97, p. 107000, 2020.

[59] H. Mardiansyah, R. W. Sembiring, and S. Efendi, Handling problems of credit data for imbalanced classes using smotexgboost, in *Journal of Physics: Conference Series*, volume 1830, p. 012011, IOP Publishing, 2021.

[60] H. Masnadi-Shirazi, N. Vasconcelos, and A. Iranmehr, Cost-sensitive support vector machines, arXiv preprint arXiv:1212.0975, 2012.

[61] A. Merćep, L. Mrčela, M. Birov, and Z. Kostanjčar, Deep neural networks for behavioral credit rating, Entropy, 23(1), p. 27, 2021.

[62] L. Munkhdalai, T. Munkhdalai, O.-E. Namsrai, J. Y. Lee, and K. H. Ryu, An empirical comparison of machine-learning methods on bank client credit assessments, Sustainability, 11(3), p. 699, 2019.

[63] J. Mushava and M. Murray, A novel xgboost extension for credit scoring class-imbalanced data combining a generalized extreme value link and a modified focal loss function, Expert Systems with Applications, 202, p. 117233, 2022.

[64] N. N. Nguyen and A. T. Duong, Comparison of two main approaches for handling imbalanced data in churn prediction problem, J. Adv. Inf. Technol. Vol, 12, pp. 1–7, 2021.

[65] A. G. Nikolaev and S. H. Jacobson, Simulated annealing, Handbook of metaheuristics, pp. 1–39, 2010.

[66] K. Pasupa, S. Vatathanavaro, and S. Tungjitnob, Convolutional neural networks based focal loss for class imbalance problem: a case study of canine red blood cells morphology classification, Journal of Ambient Intelligence and Humanized Computing, pp. 1–17, 2020.

[67] V. Prasath, H. A. A. Alfeilat, A. Hassanat, O. Lasassmeh, A. S. Tarawneh, M. B. Alhasanat, and H. S. E. Salman, Distance and similarity measures effect on the performance of k-nearest neighbor classifier–a review, arXiv preprint arXiv:1708.04321, 2017.

[68] C. Qin, Y. Zhang, F. Bao, C. Zhang, P. Liu, and P. Liu, Xgboost optimized by adaptive particle swarm optimization for credit scoring, Mathematical Problems in Engineering, 2021, 2021.

[69] D. Ramos, J. Franco-Pedroso, A. Lozano-Diez, and J. Gonzalez-Rodriguez, Deconstructing cross-entropy for probabilistic binary classifiers, Entropy, 20(3), p. 208, 2018.

[70] D. Sahoo, S. Hoi, and P. Zhao, Cost sensitive online multiple kernel classification, in *Asian Conference on Machine Learning*, pp. 65–80, PMLR, 2016.

[71] V. G. Satorras, E. Hoogeboom, and M. Welling, E (n) equivariant graph neural networks, in *International conference on machine learning*, pp. 9323–9332, PMLR, 2021.

[72] R. E. Schapire and Y. Freund, Boosting: Foundations and algorithms. adaptive computation and machine learning, 2012.

[73] P. C. Schuur, Classification of acceptance criteria for the simulated annealing algorithm, Mathematics of Operations Research, 22(2), pp. 266–275, 1997.

[74] C. Seiffert, T. M. Khoshgoftaar, J. Van Hulse, and A. Napolitano, Rusboost: A hybrid approach to alleviating class imbalance, IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans, 40(1), pp. 185–197, 2009.

[75] N. Shahri, S. B. S. Lai, M. B. Mohamad, H. Rahman, and A. B. Rambli, Comparing the performance of adaboost, xgboost, and logistic regression for imbalanced data, Math Stat, 9, pp. 379–85, 2021.

[76] V. S. Sheng and C. X. Ling, Thresholding for making classifiers cost-sensitive, in *AAAI*, volume 6, pp. 476–481, 2006.

[77] K. Sigman, Acceptance-rejection method, Columbia University, 2007.

[78] I. Sukharev, V. Shumovskaia, K. Fedyanin, M. Panov, and D. Berestnev, Ews-gcn: Edge weight-shared graph convolutional network for transactional banking data, in *2020 IEEE International Conference on Data Mining (ICDM)*, pp. 1268–1273, IEEE, 2020.

[79] Y. Sun, M. S. Kamel, A. K. Wong, and Y. Wang, Cost-sensitive boosting for classification of imbalanced data, Pattern recognition, 40(12), pp. 3358–3378, 2007.

[80] Y. Sun, A. K. Wong, and Y. Wang, Parameter inference of cost-sensitive boosting algorithms, in *International Workshop on Machine Learning and Data Mining in Pattern Recognition*, pp. 21–30, Springer, 2005.

[81] Z. Sun, J. Su, D. Jeon, A. Velasquez, H. Song, and S. Niu, Reinforced contrastive graph neural networks (rcgnn) for anomaly detection, in *2022 IEEE International Performance, Computing, and Communications Conference (IPCCC)*, pp. 65–72, IEEE, 2022.

[82] T. Sypherd, M. Diaz, L. Sankar, and P. Kairouz, A tunable loss function for binary classification, in *2019 IEEE international symposium on information theory (ISIT)*, pp. 2479–2483, IEEE, 2019.

[83] J. Tanha, Y. Abdi, N. Samadi, N. Razzaghi, and M. Asadpour, Boosting methods for multi-class imbalanced data classification: an experimental review, Journal of Big Data, 7(1), pp. 1–47, 2020.

[84] K. M. Ting, A comparative study of cost-sensitive boosting algorithms, in *In Proceedings of the 17th International Conference on Machine Learning*, Citeseer, 2000.

[85] D. Trisanto, N. Rismawati, M. F. Mulya, and F. I. Kurniadi, Modified focal loss in imbalanced xgboost for credit card fraud detection.

[86] D. Trisanto, N. Rismawati, M. F. Mulya, and F. I. Kurniadi, Modified focal loss in imbalanced xgboost for credit card fraud detection, Int J Intell Eng Syst, 14, pp. 350–8, 2021.

[87] P. J. Van Laarhoven, E. H. Aarts, P. J. van Laarhoven, and E. H. Aarts, *Simulated annealing*, Springer, 1987.

[88] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, Graph attention networks, stat, 1050, p. 20, 2017.

[89] C. Wang, C. Deng, and S. Wang, Imbalance-xgboost: leveraging weighted and focal losses for binary label-imbalanced classification with xgboost, Pattern Recognition Letters, 136, pp. 190–197, 2020.

[90] H. Wang, Q. Xu, and L. Zhou, Large unbalanced credit scoring using lasso-logistic regression ensemble, PloS one, 10(2), p. e0117844, 2015.

[91] K. Wang, J. An, M. Zhou, Z. Shi, X. Shi, and Q. Kang, Minority-weighted graph neural network for imbalanced node classification in social networks of internet of people, IEEE Internet of Things Journal, 2022.

[92] Y. Wang, Y. Zhao, N. Shah, and T. Derr, Imbalanced graph classification via graph-of-graph neural networks, in *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, pp. 2067–2076, 2022.

[93] M. Welling and T. N. Kipf, Semi-supervised classification with graph convolutional networks, in *J. International Conference on Learning Representations (ICLR 2017)*, 2016.

[94] P. Werbos, Beyond regression:" new tools for prediction and analysis in the behavioral sciences, Ph. D. dissertation, Harvard University, 1974.

[95] D. West, Neural network credit scoring models, Computers & operations research, 27(11-12), pp. 1131–1152, 2000.

[96] Z. Wu, W. Lin, and Y. Ji, An integrated ensemble learning model for imbalanced fault diagnostics and prognostics, IEEE Access, 6, pp. 8394–8402, 2018.

[97] Y. Xia, C. Liu, Y. Li, and N. Liu, A boosted decision tree approach using bayesian hyper-parameter optimization for credit scoring, Expert Systems with Applications, 78, pp. 225–241, 2017.

[98] Y. Xie, Y. Zhang, M. Gong, Z. Tang, and C. Han, Mgat: Multi-view graph attention networks, Neural Networks, 132, pp. 180–189, 2020.

[99] M. Yeung, E. Sala, C.-B. Schönlieb, and L. Rundo, Unified focal loss: Generalising dice and cross entropy-based losses to handle class imbalanced medical image segmentation, Computerized Medical Imaging and Graphics, 95, p. 102026, 2022.

[100] Z. Ying, D. Bourgeois, J. You, M. Zitnik, and J. Leskovec, Gnnexplainer: Generating explanations for graph neural networks, Advances in neural information processing systems, 32, 2019.

[101] B. Zadrozny and C. Elkan, Learning and making decisions when costs and probabilities are both unknown, in *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 204–213, 2001.

[102] D. Zhao, X. Wang, Y. Mu, and L. Wang, Experimental study and comparison of imbalance ensemble classifiers with dynamic selection strategy, Entropy, 23(7), p. 822, 2021.

[103] Z. Zhao, S. Xu, B. H. Kang, M. M. J. Kabir, Y. Liu, and R. Wasinger, Investigation and improvement of multi-layer perceptron neural networks for credit scoring, Expert Systems with Applications, 42(7), pp. 3508–3516, 2015.

[104] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, Graph neural networks: A review of methods and applications, AI Open, 1, pp. 57–81, 2020.

[105] L. Zhou and H. Wang, Loan default prediction on large imbalanced data using random forests, TELKOMNIKA Indonesian Journal of Electrical Engineering, 10(6), pp. 1519–1525, 2012.

[106] X. Zhou, W. Liang, W. Li, K. Yan, S. Shimizu, I. Kevin, and K. Wang, Hierarchical adversarial attacks against graph neural network based iot network intrusion detection system, IEEE Internet of Things Journal, 2021.

APPENDIX A

CORRELATION COEFFICIENTS

Table A.1: DATA SET 1: Home Credit Default Risk Data

	AMT INCOME TOTAL	AMT CREDIT	DAYS_ BIRTH	DAYS_ID_ PUBLISH	REGION RATING CLIENT	EXT SOURCE_2	DEF60 CNT SOCIAL CIRCLE	DAYS LAST PHONE CHANGE	IntR	Term	Annuity _to_Inc	GoodsTo Credit	Crcard_ Tol_m	DAYS EMPLOYED BIRTH	IntR2_ TERM	TARGET
AMT_INCOME_TOTAL	1	0,302	-0,014	-0,002	-0,168	0,137	-0,029	0,056	-0,048	0,054	-0,349	0,048	-0,053	-0,089	0,002	-0,089
AMT_CREDIT	0,30	1	0,13	0,05	-0,11	0,19	-0,04	0,10	-0,07	0,65	0,33	0,01	-0,05	0,02	0,33	-0,23
DAYS_BIRTH	-0,01	0,13	1	0,26	-0,04	0,14	-0,01	0,10	-0,10	0,16	0,07	0,00	0,00	0,69	0,02	-0,16
DAYS_ID_PUBLISH	0,00	0,05	0,26	1	0,00	0,07	-0,01	0,10	-0,07	0,06	0,03	0,02	-0,01	0,28	-0,02	-0,09
REGION_RATING_CLIENT	-0,17	-0,11	-0,04	0,00	1	-0,29	0,02	-0,03	0,07	-0,02	0,08	-0,05	0,01	0,01	0,04	0,12
EXT_SOURCE_2	0,14	0,19	0,14	0,07	-0,29	1	-0,04	0,22	-0,16	0,09	-0,01	0,09	-0,02	0,04	-0,07	-0,29
DEF_60_CNT_SOCIAL_CIRCLE	-0,03	-0,04	-0,01	-0,01	0,02	-0,04	1	-0,01	0,03	-0,02	0,00	-0,01	0,00	0,00	0,01	0,05
DAYS_LAST_PHONE_CHANGE	0,06	0,10	0,10	0,10	-0,03	0,22	-0,01	1	-0,08	0,05	0,01	0,05	0,06	0,04	-0,03	-0,12
IntR	-0,05	-0,07	-0,10	-0,07	0,07	-0,16	0,03	-0,08	1	0,13	-0,07	-0,06	0,04	-0,09	0,76	0,56
Term	0,05	0,65	0,16	0,06	-0,02	0,09	-0,02	0,05	0,13	1	0,02	-0,09	0,02	0,09	0,68	-0,09
Annuity_to_Inc	-0,35	0,33	0,07	0,03	0,08	-0,01	0,00	0,01	-0,07	0,02	1	0,01	0,00	0,09	-0,03	-0,06
GoodsToCredit	0,05	0,01	0,00	0,02	-0,05	0,09	-0,01	0,05	-0,06	-0,09	0,01	1	-0,07	0,02	-0,09	-0,10
Crcard_Tol_m	-0,05	-0,05	0,00	-0,01	0,01	-0,02	0,00	0,06	0,04	0,02	0,00	-0,07	1	-0,04	0,05	0,08
DAYS_EMPLOYED_BIRTH	-0,09	0,02	0,69	0,28	0,01	0,04	0,00	0,04	-0,09	0,09	0,09	0,02	-0,04	1	-0,01	-0,12
IntR2_TERM	0,00	0,33	0,02	-0,02	0,04	-0,07	0,01	-0,03	0,76	0,68	-0,03	-0,09	0,05	-0,01	1	0,38
TARGET	-0,09	-0,23	-0,16	-0,09	0,12	-0,29	0,05	-0,12	0,56	-0,09	-0,06	-0,10	0,08	-0,12	0,38	1

Table A.2: DATA SET 2: Risky Loans Data

	Credit Amount	Term	Interest Rate	Ratio of Monthly Debt to All Debt Obligations	Number of Inquiries in Past 6 Months (except auto and mortgage)	Number of derogatory public records	Interest Received to Date	Late Fees Received to Date	Last Payment Amount of Credit	Days Between the Last Credit Use and This Credit	Ratio of Annuity to Income	TARGET
Credit Amount	1	0.401	0.211	0.053	0.004	-0.075	0.638	0.047	0.578	-0.115	0.501	0.055
Term	0.401	1	0.446	0.074	0.036	-0.014	0.398	0.015	0.316	-0.101	0.066	0.151
Interest Rate	0.211	0.446	1	0.173	0.212	0.083	0.403	0.056	0.12	-0.123	0.231	0.24
Ratio of Monthly Debt to All Debt Obligations	0.053	0.074	0.173	1	-0.001	-0.033	0.057	-0.003	0.021	-0.096	0.241	0.112
Number of Inquiries in Past 6 Months (except auto and mortgage)	0.004	0.036	0.212	-0.001	1	0.048	0.052	0.024	-0.008	0.028	-0.049	0.076
Number of derogatory public records	-0.075	-0.014	0.083	-0.033	0.048	1	-0.066	-0.014	0.003	-0.111	-0.049	0.003
Interest Received to Date	0.638	0.398	0.403	0.057	0.052	-0.066	1	0.093	0.194	0.232	0.349	0.045
Late Fees Received to Date	0.047	0.015	0.056	-0.003	0.024	-0.014	0.093	1	-0.077	0.049	0.034	0.152
Last Payment Amount of Credit	0.578	0.316	0.12	0.021	-0.008	0.003	0.194	-0.077	1	-0.343	0.199	-0.387
Days Between the Last Credit Use and This Credit	-0.115	-0.101	-0.123	-0.096	0.028	-0.111	0.232	0.049	-0.343	1	-0.057	-0.041
Ratio of Annuity to Income	0.501	0.066	0.231	0.241	-0.049	-0.049	0.349	0.034	0.199	-0.057	1	0.134
TARGET	0.055	0.151	0.24	0.112	0.076	0.003	0.045	0.152	-0.387	-0.041	0.134	1

Table A.3: DATA SET 3: Irish Loan Data$_{2013}$

	Employment Length	Loan Amount	Term	Interest Rate	Total Principal Received to Date	Days Between the Issue Date and Final Date	Annuity to Monthly Income	TARGET
Employment Length	1	0.114	0.088	-0.009	0.053	0.058	0.002	-0.042
Loan Amount	0.114	1	0.394	0.132	0.586	0.142	0.384	-0.228
Term	0.088	0.394	1	0.482	-0.081	0.043	0.009	0.004
Interest Rate	-0.009	0.132	0.482	1	-0.217	-0.142	0.166	0.258
Total Principal Received to Date	0.053	0.586	-0.081	-0.217	1	0.233	0.204	-0.64
Days Between the Issue Date and Final Date	0.058	0.142	0.043	-0.142	0.233	1	0.01	-0.431
Annuity to Monthly Income	0.002	0.384	0.009	0.166	0.204	0.01	1	0.005
TARGET	-0.042	-0.228	0.004	0.258	-0.64	-0.431	0.005	1

Table A.4: DATA SET 4: Irish Loan Data$_{2014}$

	Employment Length	Loan Amount	Term	Interest Rate	Total Principal Received to Date	Days Between the Issue Date and Final Date	Annuity to Monthly Income	TARGET
Employment Length	1	0.096	0.069	0.027	0.047	0.005	-0.02	-0.022
Loan Amount	0.096	1	0.404	0.118	0.549	0.038	0.433	0.013
Term	0.069	0.404	1	0.45	-0.03	0.032	0.049	0.053
Interest Rate	0.027	0.118	0.45	1	-0.073	-0.091	0.205	0.181
Total Principal Received to Date	0.047	0.549	-0.03	-0.073	1	-0.293	0.245	-0.208
Days Between the Issue Date and Final Date	0.005	0.038	0.032	-0.091	-0.293	1	0.015	-0.31
Annuity to Monthly Income	-0.02	0.433	0.049	0.205	0.245	0.015	1	0.08
TARGET	-0.022	0.013	0.053	0.181	-0.208	-0.31	0.08	1

Table A.5: DATA SET 5: Irish Loan Data$_{2015}$

	Employment Length	Loan Amount	Term	Interest Rate	Total Payment	Total Principal Received to Date	Days Between the Issue Date and Final Date	Annuity to Monthly Income	TARGET
Employment Length	1	0.089	0.055	-0.012	0.035	0.026	-0.001	-0.034	-0.016
Loan Amount	0.089	1	0.407	0.141	0.388	0.288	0.005	0.408	0.006
Term	0.055	0.407	1	0.439	0.074	-0.011	-0.003	0.044	0.023
Interest Rate	-0.012	0.141	0.439	1	0.105	0.014	-0.056	0.28	0.126
Total Payment	0.035	0.388	0.074	0.105	1	0.979	-0.245	0.192	-0.023
Total Principal Received to Date	0.026	0.288	-0.011	0.014	0.979	1	-0.265	0.137	-0.038
Days Between the Issue Date and Final Date	-0.001	0.005	-0.003	-0.056	-0.245	-0.265	1	-0.011	-0.235
Annuity to Monthly Income	-0.034	0.408	0.044	0.28	0.192	0.137	-0.011	1	0.038
TARGET	-0.016	0.006	0.023	0.126	-0.023	-0.038	-0.235	0.038	1

Table A.6: DATA SET 6: Freddie Mac Single Family Loan Data

	CREDIT_SCORE	MORTGAGE_ INSURANCE_ PERCENTAGE	ORIGINAL_DEBT _TO_INCOME_R ATIO	ORIGINAL_ LOAN_TO_ VALUE	ORIGINAL _INTEREST _RATE	NUMBER_OF_ BORROWERS	TARGET
CREDIT_SCORE	1	-0.184	-0.124	-0.213	-0.122	-0.03	-0.178
MORTGAGE_INSURANCE_PERCENTAGE	-0.184	1	0.105	0.644	0.146	-0.047	0.091
ORIGINAL_DEBT_TO_INCOME_RATIO	-0.124	0.105	1	0.127	0.084	-0.107	0.038
ORIGINAL_LOAN_TO_VALUE	-0.213	0.644	0.127	1	0.124	-0.033	0.083
ORIGINAL_INTEREST_RATE	-0.122	0.146	0.084	0.124	1	-0.067	0.072
NUMBER_OF_BORROWERS	-0.03	-0.047	-0.107	-0.033	-0.067	1	-0.074
TARGET	-0.178	0.091	0.038	0.083	0.072	-0.074	1

Table A.7: DATA SET 7: SBA National Loan Data$_{2004-2006}$

	Term	Number of Business Employees	Number of Retained Jobs	Loan Amount	Days Between the Approval and the Disbursement of the Loan	Guararantee given by SBA	TARGET
Term	1	0.054	0.02	0.527	0.156	0.377	-0.386
Number of Business Employees	0.054	1	0.143	0.127	0.016	0.041	-0.028
Number of Retained Jobs	0.02	0.143	1	0.076	-0.005	-0.01	-0.012
Loan Amount	0.527	0.127	0.076	1	0.087	0.201	-0.12
Days Between the Approval and the Disbursement of the Loan	0.156	0.016	-0.005	0.087	1	0.086	-0.137
Guararantee given by SBA	0.377	0.041	-0.01	0.201	0.086	1	-0.1
TARGET	-0.386	-0.028	-0.012	-0.12	-0.137	-0.1	1

Table A.8: DATA SET 8: SBA National Loan Data$_{2002-2003}$

	Term	Number of Business Employees	Number of Retained Jobs	Loan Amount	Days Between the Approval and the Disbursement of the Loan	Guararantee given by SBA	TARGET
Term	1	0.041	0.049	0.575	0.059	0.323	-0.315
Number of Business Employees	0.041	1	0.17	0.095	0.007	0.035	-0.018
Number of Retained Jobs	0.049	0.17	1	0.121	-0.003	-0.029	-0.02
Loan Amount	0.575	0.095	0.121	1	0.026	0.19	-0.104
Days Between the Approval and the Disbursement of the Loan	0.059	0.007	-0.003	0.026	1	0.017	-0.079
Guararantee given by SBA	0.323	0.035	-0.029	0.19	0.017	1	-0.046
TARGET	-0.315	-0.018	-0.02	-0.104	-0.079	-0.046	1

APPENDIX B

CROSS VALIDATIONS: K-10 FOLD

Table B.1: Home Credit Default Risk Data: Sensitivity/Specificity (%)

	RCS-XGB1$_{\acute{C}_1(P\acute{D}_{LR})}$			**RCS-XGB3**$_{\acute{C}_3(P\acute{D}_{LR})}$		
	Train	Validation	Test	Train	Validation	Test
	Sens/Spec	Sens/Spec	Sens/Spec	Sens/Spec	Sens/Spec	Sens/Spec
1	95.0/93.1	92.7/92.6	91.8/92.9	95.1/92.9	91.3/92.7	92.0/92.9
2	94.9/93.1	92.2/92.7	91.8/92.9	95.1/92.8	91.7/92.5	92.2/92.7
3	95.1/93.0	91.5/92.8	91.9/92.9	95.1/92.9	91.6/92.7	91.8/92.9
4	94.9/93.0	91.4/93.0	91.8/92.8	94.9/93.0	92.0/92.7	91.8/92.9
5	94.8/93.1	91.6/93.1	91.8/92.9	95.1/92.9	93.4/92.4	92.1/92.8
6	95.1/93.0	91.9/92.8	91.8/92.8	95.1/93.1	92.3/92.8	92.0/93.0
7	95.0/92.9	92.9/92.8	91.8/92.7	95.2/93.1	91.3/92.8	91.8/93.0
8	95.0/93.1	92.3/92.9	91.7/92.9	95.1/93.1	91.1/93.1	91.5/93.1
9	95.1/92.9	92.0/92.3	92.2/92.8	94.9/93.0	93.0/92.7	92.0/92.9
10	95.0/93.2	91.4/92.7	91.7/93.0	95.0/93.0	92.0/93.1	92.0/93.0
	RCS-XGB2$_{\acute{C}_2(P\acute{D}_{GNN})}$					
	Train	Validation	Test			
	Sens/Spec	Sens/Spec	Sens/Spec			
1	96.1/94.0	91.0/93.4	90.8/93.5			
2	96.2/94.2	89.1/93.7	90.9/93.7			
3	96.1/94.0	91.3/93.3	90.9/93.5			
4	96.1/94.1	91.1/93.4	91.0/93.6			
5	96.0/94.1	90.7/93.6	91.0/93.6			
6	96.1/94.2	91.2/93.5	90.8/93.6			
7	96.1/94.1	90.9/93.1	90.9/93.5			
8	96.0/94.2	91.3/93.6	90.9/93.7			
9	96.1/93.9	91.1/93.4	90.9/93.4			
10	96.0/94.2	91.1/93.8	90.8/93.7			
	RCS-XGB2$_{\acute{C}_2(P\acute{D}_{GAT})}$			**RCS-XGB3**$_{\acute{C}_3(P\acute{D}_{GAT})}$		
	Train	Validation	Test	Train	Validation	Test
	Sens/Spec	Sens/Spec	Sens/Spec	Sens/Spec	Sens/Spec	Sens/Spec
1	95.5/94.6	90.1/94.0	90.4/94.0	94.4/94.1	91.0/93.6	90.7/93.9
2	95.7/94.5	90.1/94.0	90.4/93.9	94.3/94.1	92.0/93.6	90.9/93.9
3	95.6/94.5	90.3/93.9	90.4/93.9	94.6/94.0	91.4/93.5	90.8/93.8
4	95.5/94.5	90.6/93.8	90.4/93.9	94.3/94.0	91.8/94.2	90.8/93.9
5	95.7/94.5	89.7/93.9	90.4/94.0	94.3/94.1	91.6/94.0	90.8/94.0
6	95.7/94.4	90.6/93.5	90.6/93.8	94.3/94.1	90.4/93.7	90.9/93.9
7	95.7/94.3	90.2/93.5	90.6/93.8	94.4/94.1	90.4/93.8	90.8/94.0
8	95.6/94.4	90.9/93.6	90.4/93.8	94.4/94.1	90.6/93.7	90.7/93.9
9	95.7/94.4	89.8/94.1	90.4/93.9	94.4/94.1	91.0/94.1	90.8/94.0
10	95.6/94.5	91.3/94.0	90.7/94.0	94.5/94.1	89.9/94.2	90.9/94.0

Table B.2: Risky Loans Data: Sensitivity/Specificity (%)

	RCS-XGB2$_{\acute{C}_2(\acute{PD}_{LR})}$			**RCS-XGB3**$_{\acute{C}_3(\acute{PD}_{LR})}$		
	Train Sens/Spec	Validation Sens/Spec	Test Sens/Spec	Train Sens/Spec	Validation Sens/Spec	Test Sens/Spec
1	98.7/96.5	96.3/95.8	96.2/95.9	98.4/96.1	95.9/96.0	96.1/95.9
2	98.7/96.5	96.9/95.6	96.6/95.8	98.5/96.1	96.2/95.5	96.4/95.8
3	98.7/96.3	96.3/95.3	96.5/95.6	98.5/96.1	96.5/95.7	96.2/95.9
4	98.7/96.5	96.9/95.6	96.4/95.8	98.4/96.0	96.5/95.7	96.2/95.9
5	98.8/96.2	96.1/96.0	96.4/95.6	98.5/96.0	96.9/95.6	96.4/95.8
6	98.7/96.3	96.8/95.9	96.6/95.6	98.5/96.1	96.3/95.7	96.1/95.9
7	98.8/96.5	95.9/95.6	96.3/95.9	98.5/96.1	96.0/95.5	96.3/95.8
8	98.6/96.4	96.6/95.8	96.4/95.8	98.5/96.0	95.4/95.8	96.1/95.8
9	98.7/96.4	96.9/95.7	96.6/95.8	98.4/96.1	95.0/95.6	96.3/95.9
10	98.6/96.1	96.9/95.2	96.6/95.5	98.4/96.1	96.8/95.7	96.2/95.9
	RCS-XGB2$_{\acute{C}_2(\acute{PD}_{GNN})}$			**RCS-XGB3**$_{\acute{C}_3(\acute{PD}_{GNN})}$		
	Train Sens/Spec	Validation Sens/Spec	Test Sens/Spec	Train Sens/Spec	Validation Sens/Spec	Test Sens/Spec
1	98.9/96.8	96.7/95.6	96.1/95.9	99.0/96.3	95.3/95.8	96.0/96.0
2	99.0/96.9	95.7/96.4	96.0/96.2	98.9/96.1	95.7/95.8	96.1/95.9
3	99.0/96.8	96.8/96.0	96.2/96.0	98.9/96.2	96.3/95.6	96.0/95.9
4	98.9/96.8	96.4/96.2	96.0/96.1	98.8/96.3	95.3/95.8	96.1/96.0
5	98.9/96.8	96.6/96.0	96.1/95.9	98.9/96.2	96.1/95.9	95.9/96.0
6	99.0/96.9	95.8/96.0	95.9/96.1	98.8/96.2	95.9/95.5	96.0/95.9
7	99.0/96.8	96.6/95.9	96.1/96.0	99.0/96.2	96.4/95.9	95.9/95.9
8	99.0/96.8	96.6/96.0	96.1/96.0	98.9/96.2	95.5/95.4	96.1/95.9
9	99.0/96.9	96.0/96.2	96.1/96.1	98.9/96.1	96.4/95.9	96.1/95.9
10	98.9/96.9	96.6/96.0	96.1/96.1	98.9/96.2	95.9/96.0	96.1/95.9
	RCS-XGB2$_{\acute{C}_2(\acute{PD}_{GAT})}$			**RCS-XGB3**$_{\acute{C}_3(\acute{PD}_{GAT})}$		
	Train Sens/Spec	Validation Sens/Spec	Test Sens/Spec	Train Sens/Spec	Validation Sens/Spec	Test Sens/Spec
1	98.6/96.1	96.3/95.5	96.5/95.8	98.1/95.6	96.6/95.4	96.3/95.5
2	98.5/96.2	97.1/95.8	96.5/95.9	97.9/95.6	95.6/95.4	96.3/95.5
3	98.6/96.2	96.2/95.7	96.3/95.8	98.0/95.5	96.2/95.3	96.5/95.3
4	98.6/96.0	96.7/95.5	96.3/95.6	98.0/95.7	96.5/95.1	96.3/95.5
5	98.6/96.3	96.5/95.9	96.3/95.9	98.0/95.6	96.4/95.6	96.4/95.5
6	98.5/96.2	96.4/95.4	96.4/95.8	98.0/95.5	96.6/95.6	96.3/95.5
7	98.6/96.1	97.6/95.5	96.5/95.7	98.1/95.5	96.7/95.1	96.4/95.4
8	98.6/96.3	96.4/95.7	96.5/95.9	98.1/95.6	96.1/95.3	96.3/95.4
9	98.6/96.1	97.0/95.5	96.5/95.6	98.0/95.5	97.3/95.3	96.5/95.3
10	98.6/96.1	96.3/95.9	96.4/95.7	98.1/95.5	96.2/95.1	96.3/95.4

Table B.3: Irish Loan Data$_{2013}$: Sensitivity/Specificity (%)

	RCS-XGB3$_{\acute{C}_3(\acute{PD}_{LR})}$			**RCS-XGB2**$_{\acute{C}_2(\acute{PD}_{GNN})}$		
	Train Sens/Spec	Validation Sens/Spec	Test Sens/Spec	Train Sens/Spec	Validation Sens/Spec	Test Sens/Spec
1	97.3/95.8	91.4/95.7	94.0/95.6	97.7/96.1	91.5/95.7	93.4/95.7
2	97.4/95.6	93.5/95.1	94.1/95.4	97.6/96.0	93.9/95.4	93.6/95.4
3	97.4/95.4	93.9/94.9	94.0/95.2	97.5/95.9	92.8/95.1	93.3/95.4
4	97.5/95.6	92.4/95.6	93.9/95.4	97.8/96.1	92.7/95.8	93.4/95.5
5	97.3/95.8	93.0/95.8	93.9/95.5	97.7/96.0	93.0/95.4	93.3/95.5
6	97.3/95.7	92.5/94.9	94.2/95.4	97.8/96.3	92.1/95.7	93.4/95.8
7	97.2/95.6	94.1/95.0	93.9/95.4	97.8/96.2	91.3/95.8	93.2/95.7
8	97.0/95.7	94.0/95.5	94.0/95.5	97.8/96.0	91.9/95.5	93.6/95.7
9	97.3/95.6	92.4/95.7	93.8/95.5	97.8/96.1	92.0/95.4	93.5/95.5
10	97.2/95.6	93.5/95.2	94.0/95.4	97.7/96.0	92.5/95.6	93.4/95.5

	RCS-XGB3$_{\acute{C}_3(\acute{PD}_{GNN})}$			**RCS-XGB1**$_{\acute{C}_1(\acute{PD}_{GAT})}$		
	Train Sens/Spec	Validation Sens/Spec	Test Sens/Spec	Train Sens/Spec	Validation Sens/Spec	Test Sens/Spec
1	96.6/95.7	93.4/95.5	94.2/95.5	97.3/95.1	90.7/94.9	92.9/94.8
2	96.8/95.7	92.4/95.3	93.6/95.5	97.6/96.4	88.8/95.8	92.9/96.0
3	96.5/95.7	94.2/95.9	94.1/95.5	97.3/95.4	94.7/95.2	93.2/95.1
4	96.6/96.0	92.5/95.5	93.9/95.8	97.4/94.9	93.0/94.7	93.1/94.7
5	96.7/95.7	93.8/95.2	94.0/95.6	97.3/96.0	90.1/95.7	92.7/95.6
6	96.5/95.7	92.2/95.5	94.1/95.6	97.6/95.5	89.9/95.4	93.0/95.1
7	96.6/95.7	91.2/95.4	94.1/95.6	97.4/95.1	93.8/94.7	93.1/94.9
8	96.5/95.9	92.1/95.9	93.5/95.8	97.4/95.4	90.9/94.7	92.9/95.2
9	96.4/95.9	93.0/95.5	94.0/95.7	97.4/95.0	93.6/94.6	93.4/94.8
10	96.7/95.8	92.6/95.7	94.1/95.7	97.2/95.6	94.1/94.8	93.1/95.2

	RCS-XGB2$_{\acute{C}_2(\acute{PD}_{GAT})}$			**RCS-XGB3**$_{\acute{C}_3(\acute{PD}_{GAT})}$		
	Train Sens/Spec	Validation Sens/Spec	Test Sens/Spec	Train Sens/Spec	Validation Sens/Spec	Test Sens/Spec
1	97.5/96.0	93.2/95.5	93.3/95.6	96.9/95.9	92.4/95.5	94.0/95.7
2	97.6/96.2	89.3/95.4	93.5/95.7	97.1/95.9	89.8/95.8	94.1/95.7
3	97.3/95.9	93.6/95.5	93.4/95.5	96.9/95.8	93.9/95.4	93.9/95.7
4	97.4/96.0	92.1/95.7	93.1/95.7	97.2/95.5	93.8/95.4	94.2/95.3
5	97.5/95.9	91.5/95.5	93.5/95.5	96.8/95.8	93.4/95.4	93.9/95.6
6	97.5/96.0	90.3/95.4	93.5/95.6	97.0/95.7	90.6/95.3	94.3/95.5
7	97.2/95.9	93.5/95.2	93.3/95.4	96.8/95.8	93.3/95.6	93.9/95.7
8	97.2/96.1	91.3/95.7	92.9/95.5	96.6/95.7	95.5/95.6	94.3/95.5
9	97.4/96.0	92.6/95.4	93.4/95.5	97.1/95.8	93.0/95.5	93.9/95.6
10	97.3/95.8	95.0/95.5	93.7/95.5	96.8/95.6	93.4/95.7	93.7/95.4

Table B.4: Irish Loan Data$_{2014}$: Sensitivity/Specificity (%)

	RCS-XGB2$_{\acute{C}_2(\acute{PD}_{LR})}$			**RCS-XGB3**$_{\acute{C}_3(\acute{PD}_{LR})}$		
	Train	Validation	Test	Train	Validation	Test
	Sens/Spec	Sens/Spec	Sens/Spec	Sens/Spec	Sens/Spec	Sens/Spec
1	94.1/92.7	90.2/92.6	91.8/92.2	95.9/92.8	91.2/91.9	92.0/92.0
2	93.8/92.7	91.5/92.3	91.9/92.1	96.1/92.8	92.5/92.3	91.9/92.1
3	94.0/92.7	90.5/92.8	91.8/92.2	95.9/93.2	91.5/92.7	92.0/92.5
4	93.9/92.7	91.4/91.8	91.8/92.1	95.8/92.8	91.7/92.3	92.2/92.2
5	94.0/92.9	90.1/92.5	91.7/92.3	95.9/93.0	91.7/91.9	91.9/92.3
6	93.9/92.8	92.0/92.1	91.7/92.3	95.9/93.0	91.3/92.6	91.9/92.3
7	93.9/92.9	91.2/91.9	91.9/92.3	95.8/93.1	91.1/92.3	91.9/92.4
8	93.8/92.7	92.0/91.9	91.9/92.2	96.1/93.0	92.0/92.6	91.8/92.3
9	94.0/92.7	89.5/92.5	91.9/92.3	96.0/92.6	91.3/92.5	91.9/92.0
10	93.9/92.8	91.7/92.6	91.9/92.2	96.0/92.9	89.6/92.4	91.8/92.3

	RCS-XGB2$_{\acute{C}_2(\acute{PD}_{GNN})}$			**RCS-XGB3**$_{\acute{C}_3(\acute{PD}_{GNN})}$		
	Train	Validation	Test	Train	Validation	Test
	Sens/Spec	Sens/Spec	Sens/Spec	Sens/Spec	Sens/Spec	Sens/Spec
1	93.3/92.4	92.2/91.8	91.9/92.0	93.9/92.6	91.3/92.4	91.9/92.1
2	93.4/92.7	90.0/92.6	91.8/92.3	93.9/92.6	91.8/92.5	92.0/92.1
3	93.3/92.7	90.2/91.9	91.7/92.3	93.7/92.8	91.8/92.3	92.1/92.3
4	93.4/92.5	91.1/92.4	91.8/92.1	93.7/92.7	92.3/92.4	91.8/92.3
5	93.5/92.4	91.0/92.4	91.9/92.1	93.9/92.8	90.6/92.4	91.9/92.3
6	93.2/92.6	91.8/92.2	91.8/92.3	93.7/92.9	90.1/92.9	91.8/92.5
7	93.3/92.7	90.5/92.7	91.7/92.4	93.7/92.6	91.1/92.0	91.8/92.2
8	93.2/92.7	90.8/92.5	91.7/92.3	93.6/92.9	91.5/92.4	91.9/92.4
9	93.4/92.5	91.4/92.0	91.9/92.1	93.7/92.5	91.6/92.2	91.9/92.1
10	93.3/92.4	91.5/92.2	91.7/92.1	93.8/92.8	89.3/92.3	91.6/92.3

	RCS-XGB1$_{\acute{C}_1(\acute{PD}_{GAT})}$			**RCS-XGB2**$_{\acute{C}_2(\acute{PD}_{GAT})}$		
	Train	Validation	Test	Train	Validation	Test
	Sens/Spec	Sens/Spec	Sens/Spec	Sens/Spec	Sens/Spec	Sens/Spec
1	94.6/92.9	92.2/92.5	91.9/92.4	95.3/93.2	91.6/92.5	91.8/92.4
2	94.8/92.8	89.5/92.1	91.8/92.1	95.2/93.2	91.2/92.9	92.2/92.6
3	94.8/92.6	90.1/92.6	92.0/92.1	95.3/93.2	91.4/92.7	92.2/92.5
4	94.7/92.7	91.3/91.9	91.7/92.2	95.4/93.2	90.4/92.5	92.0/92.5
5	94.9/92.6	91.2/92.1	92.1/92.0	95.2/93.2	91.1/92.4	92.0/92.4
6	94.7/92.9	92.3/92.2	92.0/92.3	95.3/93.3	92.0/92.6	92.1/92.5
7	94.7/92.9	91.6/92.3	91.8/92.5	95.2/93.2	91.3/92.3	92.0/92.5
8	94.8/92.7	91.6/92.1	91.9/92.1	95.4/93.0	90.9/92.1	92.2/92.2
9	94.8/92.9	91.2/92.4	91.6/92.3	95.3/93.2	90.8/92.4	92.0/92.4
10	94.8/92.6	91.9/91.9	92.0/92.0	95.3/93.0	92.1/92.5	92.1/92.3

	RCS-XGB3$_{\acute{C}_3(\acute{PD}_{GAT})}$		
	Train	Validation	Test
	Sens/Spec	Sens/Spec	Sens/Spec
1	97.3/92.9	92.0/92.2	92.4/92.1
2	97.2/93.1	92.5/92.0	92.2/92.2
3	97.2/93.1	92.3/92.1	92.5/92.2
4	97.2/93.2	91.6/92.5	92.1/92.3
5	97.2/93.2	90.7/92.7	92.3/92.4
6	97.2/92.9	91.7/92.3	92.3/92.0
7	97.2/93.1	91.8/92.1	92.2/92.3
8	97.2/93.1	89.9/92.8	92.4/92.4
9	97.3/93.1	90.9/92.7	92.0/92.2
10	97.0/93.2	91.1/92.4	92.1/92.3

Table B.5: Irish Loan Data$_{2015}$: Sensitivity/Specificity (%)

	RCS-XGB3$_{\acute{C}_3(\acute{PD}_{GNN})}$			**RCS-XGB2**$_{\acute{C}_2(\acute{PD}_{GAT})}$		
	Train	Validation	Test	Train	Validation	Test
	Sens/Spec	Sens/Spec	Sens/Spec	Sens/Spec	Sens/Spec	Sens/Spec
1	90.7/87.6	85.8/87.8	86.9/87.5	89.6/86.7	86.2/86.5	87.0/86.6
2	90.6/87.1	85.6/87.1	87.2/87.1	89.6/86.6	86.5/86.4	87.0/86.5
3	90.7/87.2	87.3/86.9	87.2/87.0	89.6/86.7	86.1/86.7	86.9/86.6
4	90.9/87.3	87.1/86.6	87.0/87.1	89.6/86.6	86.9/86.6	87.2/86.5
5	90.6/87.4	87.5/87.5	87.0/87.3	89.6/86.6	87.0/86.4	87.1/86.5
6	90.8/87.4	84.7/87.4	87.0/87.3	89.7/86.8	85.3/86.8	86.6/86.7
7	90.7/87.2	87.5/87.2	87.3/87.1	89.6/86.5	87.5/87.0	87.0/86.5
8	90.7/87.3	85.8/87.1	87.0/87.3	89.7/86.7	85.3/86.6	86.7/86.6
9	90.5/87.4	85.3/87.3	87.0/87.3	89.6/86.8	86.4/86.3	86.9/86.7
10	90.9/87.3	85.1/87.3	87.1/87.2	89.6/86.7	87.2/86.3	87.1/86.5

	RCS-XGB3$_{\acute{C}_3(\acute{PD}_{GAT})}$		
	Train	Validation	Test
	Sens/Spec	Sens/Spec	Sens/Spec
1	91.1/87.4	85.2/87.3	86.9/87.3
2	90.9/87.3	87.2/86.9	87.3/87.2
3	91.2/87.4	85.6/87.3	87.2/87.3
4	91.2/87.3	87.2/87.2	86.9/87.2
5	91.1/87.4	86.4/87.2	87.1/87.3
6	91.0/87.4	86.4/87.3	87.1/87.2
7	91.1/87.1	87.8/87.3	87.4/87.0
8	91.4/87.2	85.3/87.1	87.4/87.1
9	90.9/87.5	85.4/87.4	87.3/87.3
10	91.0/87.3	85.7/87.4	87.1/87.2

Table B.6: Freddie Mac Single Family Loan Data: Sensitivity/Specificity (%)

	RCS-XGB3$_{\acute{C}_3(\acute{PD}_{LR})}$			**RCS-XGB3**$_{\acute{C}_3(\acute{PD}_{GNN})}$		
	Train	Validation	Test	Train	Validation	Test
	Sens/Spec	Sens/Spec	Sens/Spec	Sens/Spec	Sens/Spec	Sens/Spec
1	86.2/84.3	84.8/84.1	83.5/84.1	85.6/84.9	81.9/84.7	83.4/84.7
2	85.9/84.3	83.6/84.3	83.5/84.2	85.8/84.7	84.9/84.9	83.6/84.5
3	86.2/84.3	81.9/84.6	83.9/84.2	85.6/84.9	83.4/84.5	83.5/84.7
4	86.1/84.5	82.4/84.4	83.6/84.3	85.8/84.8	82.8/84.5	83.3/84.5
5	85.9/84.3	84.1/84.4	83.8/84.2	85.7/84.8	82.9/84.5	83.5/84.6
6	86.0/84.5	82.5/84.2	83.8/84.2	85.7/85.1	81.0/85.0	83.2/84.9
7	86.2/84.4	82.2/84.2	84.0/84.1	85.8/84.7	83.3/84.7	83.4/84.5
8	85.8/84.5	83.2/84.4	83.8/84.3	85.6/84.8	84.7/84.5	83.5/84.6
9	85.9/84.3	83.1/84.2	83.9/84.1	85.8/85.0	81.8/85.2	83.4/84.8
10	86.1/84.2	85.2/84.1	83.9/84.0	85.6/85.0	82.1/84.9	83.2/84.8

	RCS-XGB2$_{\acute{C}_2(\acute{PD}_{GAT})}$			**RCS-XGB3**$_{\acute{C}_3(\acute{PD}_{GAT})}$		
	Train	Validation	Test	Train	Validation	Test
	Sens/Spec	Sens/Spec	Sens/Spec	Sens/Spec	Sens/Spec	Sens/Spec
1	85.2/84.3	84.3/84.4	83.7/84.1	87.1/84.6	82.7/84.7	83.4/84.5
2	85.2/84.4	83.3/84.0	83.8/84.2	86.9/84.8	82.3/84.8	83.4/84.5
3	85.1/84.3	84.5/84.2	83.7/84.2	86.6/84.7	83.2/84.3	83.4/84.5
4	85.3/84.3	83.4/84.0	83.8/84.1	86.8/84.7	82.6/84.4	83.3/84.5
5	85.2/84.3	83.9/84.3	83.8/84.1	86.8/84.6	82.7/84.3	83.4/84.4
6	85.3/84.3	82.1/84.2	83.8/84.1	86.9/84.5	82.5/84.7	83.7/84.3
7	85.2/84.3	83.0/84.1	83.8/84.1	86.9/84.5	82.4/84.4	83.3/84.3
8	85.5/84.2	81.7/84.1	83.8/84.0	86.7/84.7	83.4/84.7	83.3/84.4
9	85.4/84.2	83.6/84.3	84.0/84.1	86.8/84.7	82.8/84.7	83.2/84.5
10	85.2/84.4	81.5/84.7	83.6/84.3	86.8/84.6	84.7/84.3	83.7/84.3

Table B.7: SBA National Loan Data$_{2004-2006}$: Sensitivity/Specificity (%)

	RCS-XGB1$_{\acute{C}_1(\acute{PD}_{LR})}$			**RCS-XGB3**$_{\acute{C}_3(\acute{PD}_{LR})}$		
	Train Sens/Spec	Validation Sens/Spec	Test Sens/Spec	Train Sens/Spec	Validation Sens/Spec	Test Sens/Spec
1	95.7/92.3	91.8/92.0	92.0/92.2	96.7/93.2	92.1/92.6	91.7/93.0
2	95.4/93.1	87.8/93.1	91.6/93.0	96.8/93.4	89.6/92.9	92.0/93.2
3	95.8/93.3	90.7/93.2	92.1/93.2	96.8/93.1	92.7/92.9	92.4/92.9
4	96.1/92.9	92.0/92.7	92.5/92.7	97.0/93.2	87.3/93.2	92.0/93.1
5	95.7/93.1	90.3/92.8	91.8/92.9	96.7/93.2	88.5/93.3	92.1/93.1
6	95.8/92.7	91.0/92.5	92.1/92.6	96.6/93.0	94.0/93.3	92.6/92.9
7	95.8/92.6	92.5/92.8	92.2/92.5	6.9/93.1	90.6/92.6	92.5/93.0
8	95.9/93.2	89.9/92.9	91.8/93.0	6.7/93.1	90.6/93.2	92.2/93.0
9	95.8/93.1	89.1/92.9	91.7/92.9	96.8/93.1	87.9/93.1	92.3/93.0
10	95.9/93.1	90.0/92.5	92.3/92.9	96.8/93.3	90.7/93.0	92.5/93.1
	RCS-XGB1$_{\acute{C}_1(\acute{PD}_{GNN})}$			**RCS-XGB3**$_{\acute{C}_3(\acute{PD}_{GNN})}$		
	Train Sens/Spec	Validation Sens/Spec	Test Sens/Spec	Train Sens/Spec	Validation Sens/Spec	Test Sens/Spec
1	97.1/92.6	91.3/92.6	93.1/92.4	97.3/92.6	91.6/92.4	92.6/92.5
2	97.1/92.8	90.1/92.5	92.8/92.7	97.2/92.8	89.3/92.3	92.3/92.6
3	97.0/92.8	88.6/92.7	92.8/92.6	97.5/92.7	91.5/91.9	93.0/92.6
4	96.7/92.8	90.5/92.0	93.2/92.5	97.0/92.5	90.2/92.7	92.4/92.4
5	96.8/92.6	94.4/92.4	93.2/92.4	97.3/92.8	93.1/92.4	92.9/92.6
6	96.6/92.7	92.6/92.6	92.9/92.6	97.3/92.7	89.2/92.1	92.7/92.5
7	96.5/92.7	93.5/92.5	92.7/92.5	97.1/92.7	91.8/92.9	92.8/92.6
8	97.1/92.5	89.9/92.3	93.3/92.4	97.5/92.6	92.7/92.9	93.2/92.5
9	97.0/92.9	84.2/92.8	92.6/92.8	97.2/92.6	90.7/93.0	93.1/92.5
10	97.0/92.7	89.9/92.8	92.7/92.6	97.7/92.6	91.0/92.4	93.1/92.5
	RCS-XGB1$_{\acute{C}_1(\acute{PD}_{GAT})}$			**RCS-XGB3**$_{\acute{C}_3(\acute{PD}_{GAT})}$		
	Train Sens/Spec	Validation Sens/Spec	Test Sens/Spec	Train Sens/Spec	Validation Sens/Spec	Test Sens/Spec
1	97.0/92.4	92.0/92.6	93.5/92.3	97.4/92.8	92.2/92.2	93.0/92.5
2	96.8/92.5	90.7/91.7	92.9/92.3	97.7/92.8	91.0/92.8	92.6/92.6
3	96.9/92.4	89.6/92.4	93.3/92.3	97.2/92.8	89.5/92.9	92.8/92.7
4	97.2/92.3	91.6/92.3	93.5/92.2	97.2/92.7	93.2/92.6	92.5/92.6
5	96.7/92.5	93.0/92.3	93.4/92.4	97.3/92.8	93.4/92.7	92.6/92.7
6	96.8/92.5	90.6/92.4	92.6/92.3	97.4/92.9	89.3/92.5	92.2/92.7
7	96.8/92.6	89.3/92.7	93.1/92.5	97.4/92.8	90.7/92.4	92.8/92.6
8	96.7/92.4	92.9/92.3	93.1/92.2	97.4/92.7	88.5/92.6	93.1/92.5
9	96.8/92.6	91.0/92.4	93.2/92.5	97.3/92.9	90.2/92.6	92.6/92.8
10	96.9/92.5	87.1/92.4	93.4/92.4	97.4/92.7	91.7/92.5	93.2/92.6

Table B.8: SBA National Loan Data$_{2002-2003}$: Sensitivity/Specificity (%)

	RCS-XGB1$_{\acute{C}_1(\acute{PD}_{LR})}$			**RCS-XGB3**$_{\acute{C}_3(\acute{PD}_{LR})}$		
	Train Sens/Spec	Validation Sens/Spec	Test Sens/Spec	Train Sens/Spec	Validation Sens/Spec	Test Sens/Spec
1	98.4/91.0	89.6/91.1	90.4/90.7	95.6/90.6	82.1/91.4	89.7/90.4
2	98.3/91.2	89.2/90.4	90.2/90.9	96.4/90.5	93.8/90.0	91.3/90.1
3	98.4/90.8	86.9/91.1	89.7/90.7	96.4/90.5	93.8/90.0	91.3/90.1
4	98.1/90.9	96.6/90.4	90.1/90.5	95.9/90.8	89.8/90.9	89.7/90.7
5	98.3/91.8	86.5/91.4	88.8/91.4	94.9/91.1	86.9/90.9	90.1/90.9
6	98.3/90.6	92.2/90.2	89.9/90.1	96.3/90.5	90.7/90.3	90.6/90.3
7	98.3/91.0	88.6/90.9	88.8/90.8	96.0/90.7	87.1/90.3	90.6/90.5
8	98.5/90.9	88.3/90.7	89.9/90.7	95.8/90.8	94.4/90.6	90.2/90.4
9	98.4/91.4	88.1/91.1	89.4/91.2	95.8/90.9	92.8/90.9	90.6/90.6
10	98.3/90.8	89.3/90.4	90.4/90.4	95.6/91.1	91.6/90.6	90.4/90.7

	RCS-XGB1$_{\acute{C}_1(\acute{PD}_{GNN})}$			**RCS-XGB3**$_{\acute{C}_3(\acute{PD}_{GNN})}$		
	Train Sens/Spec	Validation Sens/Spec	Test Sens/Spec	Train Sens/Spec	Validation Sens/Spec	Test Sens/Spec
1	97.4/90.5	90.6/90.9	91.1/90.3	93.8/91.0	87.6/90.5	89.0/90.8
2	97.5/90.8	87.8/90.4	91.8/90.6	94.0/90.4	93.6/90.9	90.1/90.2
3	97.5/90.7	90.0/90.2	91.3/90.5	93.6/90.7	91.2/91.1	90.1/90.5
4	97.4/90.7	92.1/90.5	91.0/90.6	94.5/90.9	84.6/90.2	90.1/90.6
5	97.2/90.7	92.0/91.2	91.0/90.6	93.6/91.2	90.4/91.0	90.1/91.1
6	97.0/90.8	94.6/90.2	90.6/90.4	93.8/90.8	87.0/90.9	90.6/90.6
7	97.3/90.6	91.5/90.4	91.3/90.4	93.6/90.6	92.0/90.9	90.6/90.5
8	97.7/90.5	86.7/90.4	90.6/90.3	93.6/90.9	91.0/90.3	89.9/90.6
9	97.4/90.6	89.2/90.5	90.1/90.4	4.4/90.5	87.9/90.9	91.0/90.4
10	97.2/90.7	93.5/90.1	91.0/90.5	93.9/90.7	88.3/90.5	89.5/90.5

	RCS-XGB1$_{\acute{C}_1(\acute{PD}_{GAT})}$			**RCS-XGB3**$_{\acute{C}_3(\acute{PD}_{GAT})}$		
	Train Sens/Spec	Validation Sens/Spec	Test Sens/Spec	Train Sens/Spec	Validation Sens/Spec	Test Sens/Spec
1	97.6/91.6	86.0/91.2	90.2/91.2	96.7/88.8	88.3/88.0	91.8/88.5
2	97.7/90.6	94.4/90.4	90.6/90.3	96.2/88.4	90.8/88.6	91.7/88.3
3	97.6/90.9	90.2/90.7	91.0/90.8	96.8/88.3	94.9/88.2	92.7/88.1
4	97.8/90.6	91.3/90.4	89.7/90.4	93.9/90.2	91.4/89.6	90.8/90.0
5	97.9/91.6	86.5/91.7	89.4/91.3	94.5/90.3	87.7/89.9	90.6/90.1
6	98.0/90.7	87.8/90.5	89.9/90.4	96.5/88.2	89.3/88.7	91.5/88.0
7	97.6/90.9	90.5/90.5	90.6/90.7	94.5/90.1	90.2/89.5	91.1/89.8
8	97.5/90.8	96.1/90.2	90.6/90.5	96.5/89.5	91.7/89.5	90.8/89.4
9	97.8/90.9	90.6/90.7	89.5/90.7	96.9/85.5	94.7/86.0	93.3/85.4
10	97.6/90.7	93.9/90.8	90.2/90.5	94.4/90.1	90.0/90.8	90.8/90.1

CURRICULUM VITAE

PERSONAL INFORMATION

Surname, Name: Yaman Kanmaz, Yasemin
Nationality: TC

EDUCATION

Degree	Institution	Year of Graduation
Ph.D.	METU, Financial Mathematics	2023
M.S.	METU, Economics	2012
B.S.	METU, Economics	2008
High School	Bursa Şükrü Şankaya Anadolian High School	2003

PROFESSIONAL EXPERIENCE

Year	Place	Enrollment
2009	Central Bank of Republic of Turkey	Specialist

REFEREED CONGRESS

Imbalances in Foreign Trade and Characteristics of Net Capital Inflows
Gaygısız E., Yaman Kanmaz Y.
83rd International Atlantic Economic Conference March 22-25, 2017 Berlin, Germany
https://iaes.confex.com/iaes/83am/webprogram/Paper13365.html