ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΔΠΜΣ ΕΠΙΣΤΗΜΗ ΔΕΔΟΜΕΝΩΝ ΚΑΙ ΜΗΧΑΝΙΚΗ ΜΑΘΗΣΗ

TRANSFER LEARNING AND DOMAIN ADAPTATION IN CREDIT RISK PROBLEMS

ΜΕΤΑΠΤΥΧΙΑΚΗ ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

της

ΜΑΡΙΑΣ ΚΑΪΚΤΖΟΓΛΟΥ

ΕΠΙΒΛΕΠΩΝ: ΣΤΕΦΑΝΟΣ ΚΟΛΛΙΑΣ, ΚΑΘΗΓΗΤΗΣ Ε.Μ.Π.

ΣΥΝΕΠΙΒΛΕΠΟΥΣΑ: ΠΑΡΑΣΚΕΥΗ ΤΖΟΥΒΕΛΗ, Ε.ΔΙ.Π. Ε.Μ.Π.

Αθήνα, Μάρτιος 2022

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΔΠΜΣ ΕΠΙΣΤΗΜΗ ΔΕΔΟΜΕΝΩΝ ΚΑΙ ΜΗΧΑΝΙΚΗ ΜΑΘΗΣΗ

TRANSFER LEARNING AND DOMAIN ADAPTATION IN CREDIT RISK PROBLEMS

ΜΕΤΑΠΤΥΧΙΑΚΗ ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

της

ΜΑΡΙΑΣ ΚΑΪΚΤΖΟΓΛΟΥ

ΕΠΙΒΛΕΠΩΝ: ΣΤΕΦΑΝΟΣ ΚΟΛΛΙΑΣ, ΚΑΘΗΓΗΤΗΣ Ε.Μ.Π.

ΣΥΝΕΠΙΒΛΕΠΟΥΣΑ: ΠΑΡΑΣΚΕΥΗ ΤΖΟΥΒΕΛΗ, Ε.ΔΙ.Π. Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή στις 10 Μαρτίου 2022

Αθήνα, Μάρτιος 2022

Μαρία Καϊκτζόγλου

Πτυχιούχος Μαθηματικής σχολής

Abstract

In the contemporary market where the demand for credit is growing more and more, the necessity of Credit Risk analysis is of major importance to any institution that issues loans. The last decades machine learning models are used to profile and score the creditworthiness of the applicants. Although machine learning has provided to Credit Risk strong tools, real world situations often pose constraints or hide unseen complications that obstruct the models' learning. Two characteristic examples are the shortage of data and the change of the data's distribution over time or area or groups of people etc.

In this thesis we are concerned with the problem of the change in the distribution of the data in the Credit Risk context and we are using Transfer Learning to confront it. We are using two different datasets. In the first we use Transfer Learning to predict the class of defaulters of high amount applicants by exploiting the knowledge from lower amount applicants. In the second we use it to predict defaulters who are clients of a fintech company in a certain country based on the knowledge possessed from the company's clients in another country. After preprocessing the data and building some good base classifiers, we apply two Transfer Learning methods and thus we study the problem by experimenting with some variations regarding the distribution change between the data that is used for training and the data that is used for testing. We use the logistic model, Gradient Boosting and Random Forest as the base classifiers, as those have been used in past research on this topic.

Our results show that Transfer Learning can help in the problem of change in the distribution, although it is also possible to negatively affect the learning. Each algorithm demonstrates a distinct behavior under the Transfer Learning methods and the results also seem to strongly rely on the degree of the change in the distribution. We discuss our results based on the theoretical background of the algorithms and methods and existing research and we suggest some possible explanations for each case. We finally propose ideas for future research that can follow up this thesis that would probably shed more light to some results and help develop more adequate Transfer Learning strategies.

Key words

Transfer Learning, Domain Adaptation, Boosting, Source Domain, Target Domain, Concept Drift, Distributions Divergence, Distributions Similarity, Machine Learning, Credit Risk

Περίληψη

Στη σύγχρονη αγορά όπου η πιστωτική ζήτηση αυξάνεται ολοένα και περισσότερο, η ανάγκη για ανάλυση Πιστωτικού Κινδύνου είναι μεγάλης σημασίας για κάθε οργανισμό που εκδίδει δάνεια. Τις τελευταίες δεκαετίες χρησιμοποιούνται μοντέλα μηχανικής μάθησης για να σκιαγραφήσουν το προφίλ των αιτούντων και για να βαθμολογήσουν την πιστωτική τους φερεγγυότητα. Αν και η μηχανική μάθηση έχει προσφέρει στην Ανάλυση Κινδύνου δυνατά εργαλεία, διάφορες καταστάσεις στην πραγματική ζωή συχνά θέτουν περιορισμούς ή εμπεριέχουν μή εμφανείς επιπλοκές οι οποίες εμποδίζουν την εκμάθηση του μοντέλου. Δύο χαρακτηριστικά παραδείγματα είναι η έλλειψη δεδομένων και η μεταβολή της κατανομής των δεδομένων στο χρόνο ή ανά τόπο ή ανά ομάδες ανθρώπων και λοιπά.

Στην παρούσα εργασία μας απασχολεί το πρόβλημα της μεταβολής της κατανομής των δεδομένων στο πλαίσιο προβλημάτων Πιστωτικού Κινδύνου και χρησιμοποιούμε Μεταφερόμενη Μάθηση για να το αντιμετωπίσουμε. Χρησιμοποιούμε δύο διαφορετικά σύνολα δεδομένων. Στο πρώτο χρησιμοποιούμε Μεταφερόμενη Μάθηση για να προβλέψουμε την κλάση των εκπρόθεσμων οφειλών μεταξύ των αιτούντων υψηλών ποσών δανείου, αξιοποιώντας τη γνώση από τους αιτούντες χαμηλότερων ποσών δανείου. Στο δεύτερο τη χρησιμοποιούμε για να προβλέψουμε την κλάση των εκπρόθεσμων οφειλετών που είναι πελάτες μιας fintech εταιρείας σε μια συγκεκριμένη χώρα, με βάση τη γνώση που υπάρχει από τους πελάτες της εταιρείας σε μια άλλη χώρα. Αφότου κάνουμε μια αρχική προεπεξεργασία των δεδομένων και χτίσουμε κάποιους καλούς ταξινομητές βάσης, εφαρμόζουμε δύο μεθόδους Μεταφερόμενης Μάθησης και μελετάμε το πρόβλημα κάνοντας πειραματισμούς και με τον τρόπο που αλλάζει η κατανομή των δεδομένων, από το σύνολο εκπαίδευσης στο σύνολο επικύρωσης. Χρησιμοποιούμε το λογιστικό μοντέλο και τους ταξινομητές Gradient Boosting και Τυχαίο Δάσος, αφού έχουν χρησιμοποιηθεί και στο παρελθόν σε παρόμοιες έρευνες.

Τα αποτελέσματα δείχνουν ότι η Μεταφερόμενη Μάθηση μπορεί να βοηθήσει στο πρόβλημα της μεταβολής στην κατανομή των δεδομένων, αν και είναι επίσης δυνατό να επηρεάσει αρνητικά τη μάθηση. Κάθε αλγόριθμος έχει διαφορετική συμπεριφορά με τις μεθόδους Μεταφερόμενης Μάθησης και τα αποτελέσματα φαίνονται να εξαρτώνται σημαντικά από το βαθμό μεταβολής στην κατανομή. Συζητάμε τα αποτελέσματα με βάση το θεωρητικό υπόβαθρο των αλγορίθμων και των μεθόδων και με την υπάρχουσα έρευνα, και δίνουμε κάποιες πιθανές εξηγήσεις για κάθε περίπτωση. Τέλος, προτείνουμε ιδέες για μελλοντική έρευνα που μπορεί να ακολουθήσει αυτήν την εργασία η οποίες θα μπορούσαν να διαφωτίουν περισσότερο κάποια από τα αποτελέσματα και να συνεισφέρουν στο να αναπτυχθούν πιο κατάλληλες στρατηγικές Μεταφερόμενης Μάθησης.

Λέξεις-κλειδιά: Μεταφερόμενη Μάθηση, Προσαρμογή Τομέων, Ενίσχυση, Τομέας Πηγής, Τομέας Στόχου, Αλλαγή Έννοιας, Απόκλιση Κατανομής, Ομοιότητα Κατανομής, Μηχανική Μάθηση, Πιστωτικός Κίνδυνος

Acknowledgments

In concluding this thesis, I would like to thank Professor Stefanos Kollias for he has given me the opportunity to engage with the particularly interesting topic of Transfer Learning and learn a bunch of new things. I am also deeply thankful to Professor Paraskevi Tzouveli for all her kind and patient support and her helpful guidance during my writing of this thesis. Last, thanks for one more time to my parents and my brother for their endless support, and Sofoklis, my precious friend, who has been a glimpse of optimism and inspiration.

Table of Contents

1. Introduction

Over the past decades the field of Credit Risk has grown majorly, since it plays a crucial role in various institutions that issue loans. The increasing demand of credits has necessitated dealing with the risk of defaulting when a loan is provided to a borrower. The credit institutions must be able to estimate the trustworthiness of an applicant so that they make a correct decision about whether to provide a loan or not, and more than that, about the maximum credit value that the applicant could take. Credit risk is not a risk threatening strictly the loan provision by banks; it is a risk present in other activities as in bankers' acceptances, in trading with foreign exchanges in the global market, futures, bonds, swaps, stocks and other (Adamko, Kliestik & Birtus, 2014).

The history of credit risk is long, starting from decisions that were made under subjective evaluation of the individual applicants. This though was evidently lacking uniformity, objectivity and generalizability and was prone to underestimations. It was only in 1967 and 1968 when the first mathematical tools were developed to evaluate credit risk (Adamko, Kliestik & Birtus, 2014). Then, the advent and rise of machine and deep learning has provided new ways and perspectives to approach the credit risk problems, and lots of research has been conducted on this topic. For example, Khandani et al. (Khandani, Kim & Lo, 2010) tried a few machine learning models - radial basis functions, tree-based models and support vector machines - to predict delinquencies of credit repayments, evaluating thus the discriminative power of the models, which they found to be impressively strong. Addo et. al. (Addo, Guegan & Hassani, 2018) developed Random Forest, Gradient Boosting and four deep learning models to identify defaulters, finding among other things that the deep learning models showed to be less stable and did not exhibit particular interest, while tree-based models outperformed.

At the same time, they raised crucial questions about the transparency of machine and deep learning algorithms as well as the discrimination and bias towards groups of people that may be introduced from such models that have a very specific objective. Butaru et al. (Butaru, Chen, Clark, Das, Lo & Siddique, 2016) also tried machine learning algorithms to predict delinquencies in six different banks, and while they found that decision trees and random forests are very powerful, they highlighted that due to the particular regulations and management in each bank, there is no homogeneity in the predictability of algorithms across the banks. Although the various questions about the ethics, the generalizability or the interpretability of machine and deep learning models are legitimate, the research has showcased the power of these models and has set the ground for further research.

Whereas, thus, many institutions and companies can build and utilize machine and deep learning models in their credit risk, there are often real-life conditions that tend to pose limits on the stability and predictive power of the models. One such condition is the insufficient amount of data for a group of applicants, which renders hard or even impossible to train some models. Let's consider some specific examples where this can happen. In a bank, there are credit applicants who are already customers and others who are not. For the customers, the bank possesses a substantial amount of data that can be used to train a model, but for the non-customers there is no data at all. How can the bank evaluate the creditworthiness of the non-customers applicants? This case has been studied by Benniel et al. (Beninel, Bouaguel & Belmufti, 2012). Another example is given by the request of unfrequently high amounts of loans, something that due to its scarcity it entails too few data. This case was studied by Huang and Chen (Huang & Chen, 2018).

The fact for the data scientist in each case is the same: the data has seen a significant change and its size is too small to train the existing model. In other words, the distribution of the data has changed, hence the classifier is no longer reliable. This change in the marginal distribution of the data is commonly known as *concept drift* (Gama & Zhang, 2019). How can this problem be confronted? Collecting more data is most of the time either impossible or too time-costly, therefore, not an option. Transfer Learning is the field of machine learning that has arisen to give an answer to this question. Broadly speaking, Transfer Learning aims at utilizing the existing knowledge in a problem so as to alleviate the solution of another problem. With respect to the examples given before, in the second case for instance, Transfer Learning would take advantage of all the knowledge and information carried by the applicants of low and medium credit amounts and appropriately use it to the applicants of high credit amounts.

We found it particularly interesting thus to merge these two problems, Credit Risk and Transfer Learning, as having insufficient amount of data or changes in the distribution of the data is a frequent problem in real-life, and Transfer Learning seems to be able to provide adequate solutions.

Therefore, in this thesis we chose to use and experiment with two Transfer Learning methods, KLIEP and trAdaBoost, along with three classifiers, Logistic Regression, Random Forest and Gradient Boosting, to deal with the concept drift that occurs in credit risk problems. We are using two different datasets, one publicly available that has been used by researchers in relevant research, and one that was obtained from the airtime loan provision fintech company Channel VAS in which the author of this thesis worked for a period.

The results obtained show that the Transfer Learning methods are capable of inducing positive results if combined with other pre-processing methods, but at the same time they show that it is likely that the transfer learning methods have a negative impact. It is also noteworthy that there is no homogeneity in the effect of the Transfer Learning algorithms among the models used; on the same dataset and with the same Transfer Learning method, one model can be alleviated and the other degraded.

The structure of this thesis is: This first current chapter is introductory. In chapter 2 we give the framework of Transfer Learning and describe its various perspectives via a taxonomy that we use. In chapter 3 we provide the theoretical background of all the machine learning, transfer learning and statistical methods and algorithms that we have used. Chapter 4 contains the description of the two datasets. In chapter 5 we explain how the experiments were conducted, we present the results and we discuss them. Lastly, chapter 6 concludes the thesis.

2. Transfer Learning & Domain Adaptation – A review

Traditional machine learning engages with the problem of building a model by training it on a part of a dataset and testing it on another part, until the results obtained on the test part are satisfactory. Then, this model can be used on other, unknown datasets for classification or regression. Under this formulation, an implicit assumption is made, that is that the distribution of the dataset on which the model is built is the same with the distribution of the dataset on which the model is later used. In real world problems though, this is not always the case; often the two distributions differ, as Pan and Yang (Pan & Yang, 2009) explain, and when this happens most models need to be trained from scratch on new data that follow the distribution of the domain where application is desired. However, as they write (Pan & Yang, 2009, p.1)

“it is expensive or impossible to re-collect the needed training data and rebuild the models. It would be nice to reduce the need and effort to re-collect the training data”

In addition, in traditional machine learning the model’s task is the same in the training and in the test part. However, in real life a different task is to be solved on unknown data.

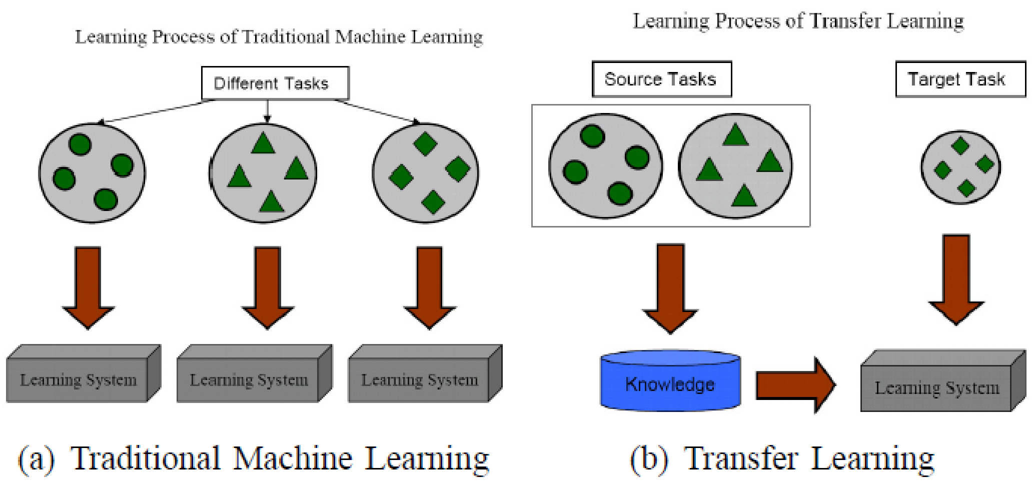


Figure 1: Traditional Machine Learning VS Transfer Learning (Pan & Yang, 2009, p.2)

It would evidently be wonderful if we could transfer the knowledge across the domains or tasks without the need to recollect data and retrain; and this is where Transfer Learning (TL) jumps in. This concept originates from educational psychology and the generalization theory of transfer, as proposed by C.H. Judd, who referred to transfer as the result of generalizing one’s experience (Zhuang, Qi, Duan, Xi, Zhu, Zhu, ... & He, 2020), which results in learning an activity more easily if knowledge from another activity is used. For example, if one knows to play the violin, it might be easier to learn to play the guitar, or, if someone speaks French, then it becomes easier to learn Spanish, as both these languages' root is Latin.

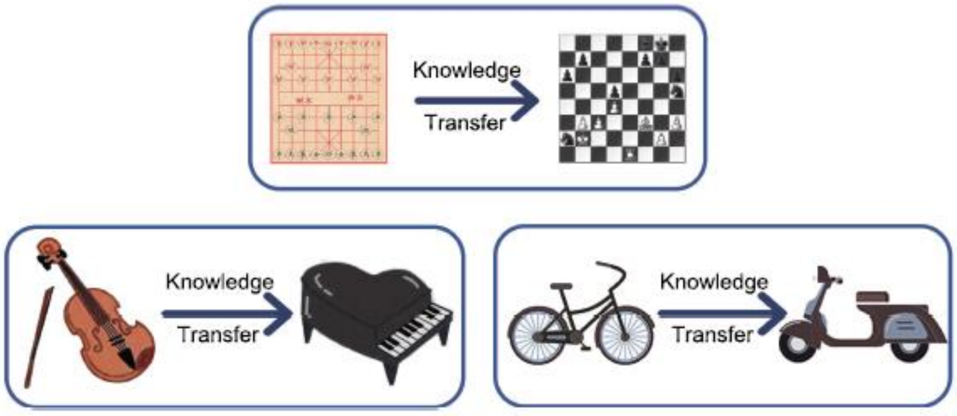


Figure 2: Intuitive examples of transferring knowledge (Zhuang, Fuzhen, et. al., 2020, p.1)

The need for Transfer Learning can arise in various cases. One typical case is what is commonly known as *concept drift,* (Gama & Zhang, 2019, p.1)

"concept drift means that the statistical properties of the target variable, which the model is trying to predict, change over time in unforeseen ways. This causes problems because the predictions become less accurate as time passes."

So, in concept drift, the data originally collected becomes outdated. This means that the marginal distribution has changed, a phenomenon commonly known as *covariate shift.* An example of this case can be seen in the behavioral usage of mobile phones; over the years, audio calls and text messages usage has shrunk, being replaced to a large extent by video calls and mobile internet usage. (Gama & Zhang, 2019).

In general, research of learning under concept drift encompasses three major components: drift detection (if concept drift actually occurs), drift understanding (when, how, where it occurs), drift adaptation (how to handle the drift).

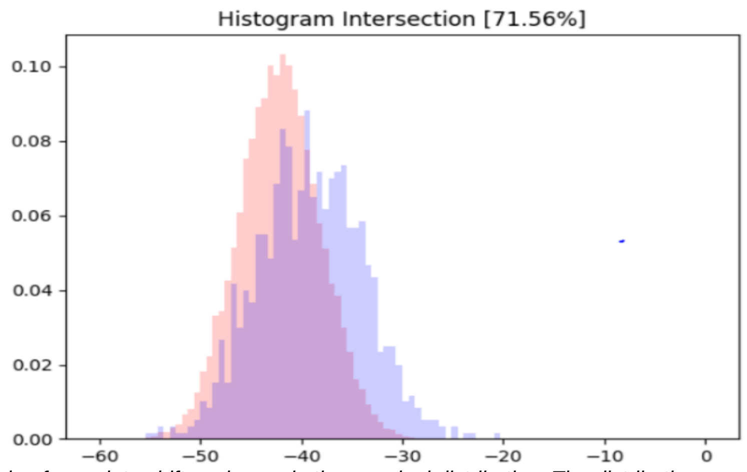


Figure 3: Example of covariate shift, a change in the marginal distribution. The distributions overlap by 71.56%

There are many cases where Transfer Learning is useful. Consider a problem of sentiment classification, which aims at classifying the reviews of tech-products to positive, negative, or neutral, based on text data. For a given product, say a smartphone, lots of data should be collected along with their labels to train a model. Nonetheless, this model may not perform well on another product, for example smartwatches, as smartwatches are evaluated under different criteria, in other words, smartwatches' reviews as a distribution may substantially differ from that of smartphones. Adapting the model to the new distribution would dispense us from re-collecting annotated data and re-training.

Another example from the field of medicine is that of a model that diagnoses lung cancer. Training a model with data collected from a town in the countryside would result in a model that would probably underperform in data from an industrial city, as the air pollution rate - a crucial factor for developing lung cancer – may significantly differ in these two areas.

A last example can be drawn from the field of Credit Risk, where lenders aim at predicting the behavior of loan applicants to appropriately determine their credit limits. A frequent problem is that of evaluating new applicants, whose consuming and behavioral distributions may differ to some extent from existing clients. Using the model built on existing clients is susceptible to underperforming, while labeled data for the new applicants can be scarce or even nonexistent. Therefore, using the knowledge from the existing model to the new dataset of new applicants can significantly improve the performance (Beninel, Bouaguel & Belmufti, 2019)

We should also make a distinction between multi-task learning, a subfield of machine learning, and Transfer Learning. In multi-task learning:

"multiple learning tasks are solved at the same time, while exploiting commonalities and differences across tasks." (Multi-task learning, 2022)

In other words, multi-task learning aims at learning simultaneously a set of related tasks. More precisely, every task is alleviated by making use of the interconnections with the other tasks, while considering the similarities as well as the differences between the tasks. This results in enhancing the generalization of each task. Although it resembles Transfer Learning in that knowledge is shared among tasks, the main difference between Transfer Learning and multi-task learning is that Transfer Learning focuses on the task of the unseen, test data, by exploiting the knowledge of the training data, while multi-task learning treats all tasks equally by learning them simultaneously (Zhang & Yang, 2020)

A fundamental condition for transfer is that there is some connection between the two activities, which in mathematics translates to having two domains that do not differ substantially. It is important to highlight what Zhuang et al. (Zhuang et al., 2020), stress as well, that transfer does not always have a positive impact, and this phenomenon of the target learner being negatively affected is called negative transfer. We elaborate more on negative transfer in section 2.3. Prior to this, we provide definitions for Transfer Learning (2.1.) and we present a categorization with respect to its approaches (2.2.).

2.1. Definitions

Based on the works of Pan and Yang (Pan & Yang, 2009) and Zhuang et al (Zhuang et al., 2020) we will give the definitions for two fundamental notions of Transfer Learning, "Domain" and "Task", with the aid of which we will then provide a definition for Transfer Learning.

Definition 1: A *domain* D consists of a feature space X and a marginal distribution $P(X)$

$$D = \{X, P(X)\}$$

$$where \;\; X = \{x_i \mid x_i \in X,\; i = 1, \dots, n\}$$

Definition 2: given a domain $D = \{X, P(X)\}$, a *task* T consists of a label space Y and a decision function $f(\cdot)$, e.g. $T = \{Y, f(\cdot)\}$. The decision function is unknown and is to be learned from the sample data, which comprises of pairs $\{x_i, y_i\}$, where $x_i \in X$ and $y_i \in Y, i = 1, \dots, n$.

We call *Source* domain the domain from which knowledge is transferred and *Target* domain the domain to which knowledge is transferred from Source. Moreover, we use the term *task* to refer to the procedure of building a predictive model. We can now provide the following definition for Transfer Learning

Definition 3: Given a source domain *DS* and learning task *TS,* a target domain *DT*, and learning task *TT*, *Transfer Learning* consists of methods that aim to assist the learning of the target predictive function *fT (・)* in *DT* by exploiting the knowledge acquired in *DS*, where *DS ≠ DT* or *TS ≠ TT.*

A special subcategory of Transfer Learning that falls in the case when the tasks in the two domains are the same but their distributions differ is known as *Domain Adaptation,* and it can tackle problems under the existence or not of target labeled data (Kouw & Loog, 2019).

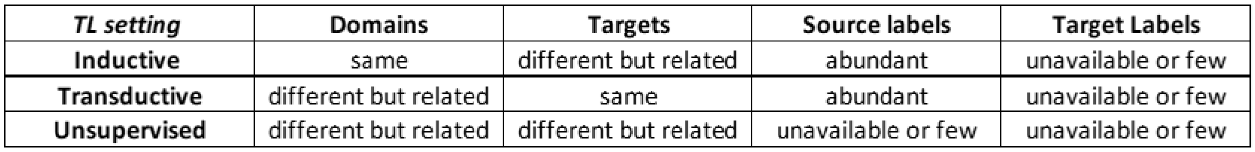
2.2. Categorization of Transfer Learning

Transfer learning can accept various categorizations that depend on the perspective from which it is approached. We are presenting a categorization of Transfer Learning based on the works of Pan and Yang (Pan & Yang, 2009, Zhuang et al (Zhuang, et al., 2020), and Kouw and Loog (Kouw & Loog, 2019).

The first level of this categorization is done between the *problem setting* of Transfer Learning and the *solution strategy*. In 2.2.1. we describe the problem setting perspective and in 2.2.2. we present the solution strategy perspective.

2.2.1. Problem setting perspective

From the problem setting perspective, we can make a categorization that takes into account the *domains' relevance*, the *tasks* to be learned and the *labels'* availability. With these three considered, we have three major categories of Transfer Learning, *inductive*, *transductive* and *unsupervised* (Pan & Yang, 2009), (Zhuang et al., 2020). In inductive learning the two distributions are the same, while the tasks, although differ, are related. There are abundant labels in Source and no or few labels in Target. In transductive learning the two tasks are the same, while the two distributions are different, yet related, and there are labels available only in Source. Unsupervised learning describes the case when both domains and tasks are different but related and there are no labels in either of them.



TL setting	Domains	Targets	Source labels	Target Labels
Inductive	same	different but related	abundant	unavailable or few
Transductive	different but related	same	abundant	unavailable or few
Unsupervised	different but related	different but related	unavailable or few	unavailable or few

Figure 4: Transfer Learning settings

From the problem-setting perspective, if we now look at the feature space of the Source and the Target domain, Transfer Learning can be characterized as *homogeneous* or *heterogeneous*. This refers to cases where the domains are of the same feature space, while the latter, those in which the feature space differs. Many homogeneous learning studies assume that domains differ only in marginal and not in conditional distributions, although this assumption does not always hold. This thesis falls into the homogeneous learning as the datasets live in the same feature space.

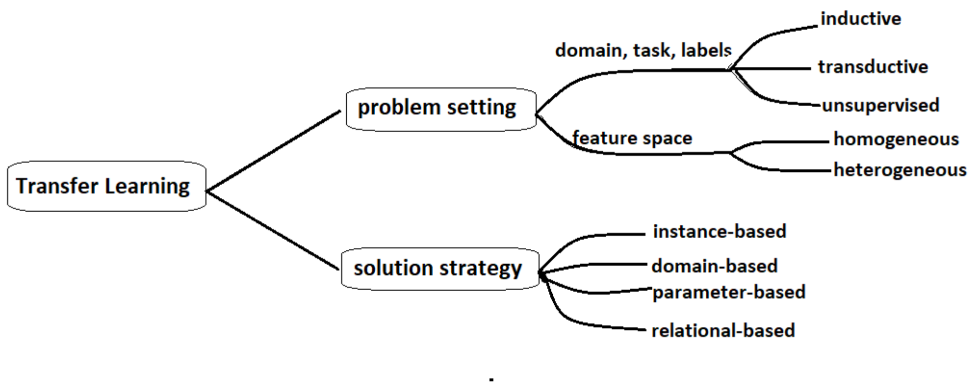


Figure 5: Categorization of Transfer Learning

2.2.2. Solution strategy perspective

Focusing thus on the solution strategies Pan and Yang (Pan & Yang, 2009) categorized Transfer Learning into four groups*: instance-based, feature-based, parameter-based and relational-based unsupervised*. Zhuang et. al. (Zhuang et al., 2020), follow a different categorization: *data-based*, which incorporates instance-based and feature-based approaches and *model-based*, which covers the *parameter based* approaches. Their survey does not cover relational-based approaches as there have been few studies on them. We are following Zhuan et al. categorization and we will expand on instance-based methods since we applied two of them in this thesis.

2.2.2.1. Data-based approach: Instance-based methods

Instance-based methods are built on the basis that some instances in the Source domain can be used to train in Target as long as some weighting is applied on them. Let's consider the case when the two marginal distributions differ and the two conditional distributions are the same, e.g., $P^S(X) \neq P^T(X)$ and $P^S(Y|X) = P^T(Y|X)$. In a real context, we can think that we want to build a model that diagnoses lung cancer in two areas, one of which is a town in the countryside and the other one is an industrial city. Building a model on data from the countryside town is reasonably susceptible to bad performance if applied on the industrial city, as air pollution rate, a crucial factor for developing lung cancer, differs significantly. Mathematically speaking, the two marginal distributions differ, and if we want to use the knowledge acquired when building the model in source to the target domain, they should first be somehow brought closer. A simple and often successful way to do this is to assign weights to source domain instances in the loss function, in a way that they will denote how this instance contributes to the proximity of the two domains. Then, instead of training the target's instances, we can train the weighted instances of the source. This strategy is expanded as follows:

$$E_{(x,y)\sim P^T}[L(x,y;f)] = \qquad (1)$$

$$E_{(x,y)\sim P^S}\left[\frac{P^T(x,y)}{P^S(x,y)}L(x,y;f)\right] = \qquad (2)$$

`

$$E_{(x,y)\sim P^S}\left[\frac{P^T(x)P^T(x)}{P^S(x)P^S(x)}L(x,y;f)\right] = \qquad (3)$$

$$E_{(x,y)\sim P^S}\left[\frac{P^T(x)}{P^S(x)}L(x,y;f)\right] \qquad (4)$$

Therefore, the objective function takes the form

$$min_f \frac{1}{n^s} \sum_{i=1}^{n^s} \beta_i L(f(x_i^s), y_i^s) + \Omega(f) \qquad (5)$$

Where $\beta_i = \frac{P^T(x_i)}{P^S(x_i)}$ (i=1,…,n^s) and $\Omega(f)$ is the structural risk - e.g. the sum of loss and the model's complexity (Zhuang et al., 2020).

A problem is that the estimation of the distributions' ratio is a difficult task with the traditional methods. There are some methods that have been developed that bypass this, among which the Kernel Mean Matching (KMM), Kullback-Leibler Importance Estimation Procedure (KLIEP) and TrAdaBoost. We are going to further describe KLIEP and TrAdaBoost in Chapter 3, as they are two domain adaptation methods that were applied in this thesis.

2.2.2.2. Data-based approach: feature-based methods

Feature-based methods involve finding a new feature space representation for the target domain, where knowledge from source can be transferred. These methods can apply to both homogeneous and heterogeneous learning; the former aims to reduce the distance between the marginal distributions, while the latter aims to reduce the distance between the feature spaces. Quoting Zhuang et al. (Zhuang et al., 2020, p.7).

"The objectives of constructing a new feature representation include minimizing the marginal and the conditional distribution difference, preserving the properties or the potential structures of the data, and finding the correspondence between features."

Strategies to find a suitable feature space representation depend on the availability of labeled data in the Source. If there are abundant labeled data, then supervised methods may be used, else, unsupervised methods are required to construct the feature representation (Pan & Yang, 2009).

Supervised feature construction methods are grounded on the idea of learning a lower-dimensional representation on a subspace of both domains. This is similar in multitask learning that seeks for a representation that is sharable by the various tasks. The new representation is aimed to reduce the classification error.

Raina et al. (Raina, Battle, Lee, Packer & Ng, 2007) have suggested an unsupervised feature construction method, which at learning a higher-dimensional representation, by first solving an optimization problem on Source that minimizes the L2-norm between the original features and their new representation. Then this takes place for Target. A disadvantage of this method is that the basis learned in the first step, e.g., from Source, might not be adequate for the Target domain.

2.2.2.3. Model-based methods

Model-based - or parameter-based – strategy is basically applied to inductive Transfer Learning problems and assumes that models of related tasks should share some parameters of prior distributions of hyperparameters (Pan & Yang, 2009).

One idea is to share the parameters of the trained model. Let’s consider the example of object classification on image data (Zhuang et al., 2020). Each class’s attribute, such as the color and the shape, has a prior from the image features that can be learned from the Source domain and then be used to foster the learning on the Target domain.

“The parameters of a model actually reflect the knowledge learned by the model. Therefore, it is possible to transfer the knowledge at the parametric level.” (Zhuang et al., 2020, p.15)

Another idea is to use the regularizers from pre-trained models on various source domains to the target domain model. Such a framework has been proposed by Duan et al. and is called Domain Adaptation Machine (DAM) (Duan, Tsang, Xu & Chua, 2009).

Finally, from the model-perspective we should not omit how neural networks have been developed for Transfer Learning. Zhuang et al. (Zhuang et al., 2015) have proposed an approach based on Autoencoders, called Transfer Learning with Deep Autoencoders (TLDA). TLDA uses two autoencoders that share the same encoding and decoding weights and it enforces the Source and Target domains to be similar using the Kullback-Leibler divergence. It uses softmax regression, a generalization of logistic regression for multiple class classification problems. Thus, the objective function contains three terms to be optimized:

- The reconstruction error for both Source and Target domains
- The KL divergence of the embedded instances between the source and target
- The loss function of the softmax regression

Another framework inspired by Generative Adversarial Networks (GAN) (Goodfellow, Pouget-Abadie, Mirza, Xu, Warde-Farley, Ozair & Bengio, 2020). Let's first say that GAN is a framework inspired by a competitive game between two models; a Generator G that is trying to learn the distribution of the data and generates imitations of it, and a Discriminator D that is trying to discriminate the true from the fake data produced by G. This way a two-fold objective is achieved: learning the distribution of the data and building a classifier. Based on this setting it has been proposed (Ganin, Ustinova, Ajakan, Germain, Larochelle, Laviolette & Lempitsky, 2016) a similar setting where it is assumed that no Target labels are available. The idea is that there are sought features that (i) assist the discriminative task in the Source domain and (ii) are indiscriminate for the Source and the Target domain. In other words, the extraction of features is the Generator's goal and the discrimination between the Source and the Target domain under the covariance shift is the Discriminator's goal. This architecture can be materialized using backpropagation and stochastic gradient descent.

2.3. Negative Transfer

As mentioned in the beginning of this chapter, Transfer Learning does not always result in enhanced performance. The negative impact to Target domain that results from transferring knowledge from Source to Target is known as *negative transfer.* Wang et al. in their paper (Wang, Dai, Póczos & Carbonell, 2019) provide a formal definition of negative transfer and analyze three important aspects of it.

Some of the crucial questions to be answered when negative transfer is discussed are the following:

i. What is its exact definition? To answer this, we should first answer some other questions like, should it be measured in a test set? What type of baseline should be used for comparison?
ii. What factors cause it and how can we exploit them?
iii. Given limited or no labeled data, how can it be detected?

Therefore, Wang et al. stress three specific points in their attempt to describe negative transfer. Before presenting these three points some necessary notations are introduced.

- $P_S(X,Y)$ and $P_T(X,Y)$ are the Source S and Target T distributions respectively, while X, Y are the input and output random variables.
- $S = \{(x_s^i, y_s^i)\}_{i=1}^{n_s}$: labeled Source set drawn from $P_S(X,Y)$
- $T_l = \{(x_l^j, y_l^j)\}_{j=1}^{n_l}$: labeled Target set drawn from $P_T(X,Y)$
- $T_u = \{(x_u^k)\}_{k=1}^{n_u}$: labeled Target set drawn from $P_T(X)$
- A is an algorithm
- $h = A(S,T)$ the hypothesis (model) output by algorithm A
- $R_{P_T}(h) \coloneqq E_{x,y\sim P_T}[l(h(x)), y)]$, the standard expected risk, where l is some specific loss function

With the above definitions we can now present the three aforementioned points discussed by Wang et al (Wang et al., 2019):

1. *"Negative transfer should be defined with respect to the algorithm"* (Wang et al., 2019, p.2). In case multiple algorithms are used it would be unsuitable and misleading to compare the results in the Target domain with only the best algorithm obtained from Source, e.g., defining the negative transfer under the objective

$$R_{P_T}(A(S,T)) > min_{A'} R_{P_T}(A'(\emptyset,T)) \quad (6)$$

as the increase in risk may not stem from the difference in domains but from the difference in the algorithms. Therefore, the study of negative transfer requires to restrict to a specific algorithm and compare its performance with and without the source-domain data. This leads to defining the *negative transfer condition*

$$R_{P_T}(A(S,T)) > R_{P_T}(A(\emptyset,T)) \quad (7)$$

2. *"Divergence between the joint distributions of Source and Target is the root of negative transfer"* (Wang et al., 2019, p.2) An extreme example is given by the authors to illustrate this; consider that $P_T(X) = P_S(X)$ and that $P_S(Y|x)$ is uniform. Then $P_S(X,Y)$ is not informative and making use of $S\sim P_S(X,Y)$ would rather damage the estimation of $P_T(Y|X)$ unless $P_S(Y|X)$ is also uniform. Practically, to apply transfer learning there must exist some similarity between the two joint distributions, and then apply an algorithm that will identify and rely only on the similar, useful part. If the divergent part is not disregarded, it will lead to negative transfer.

3. *“Negative transfer strongly depends on the amount of labeled target data”* (Wang et al., 2019, p.2)*.* This factor actually has a mixed effect. On one hand, the existence of labeled data in target does not guarantee a successful transfer. In case there are no labels in target the classification task in target becomes a zero-shot one (there are classes in test set that were not present in the training set) and using only the training samples of target would probably result in a weak classifier. Thus, the negative transfer condition is not very likely to be met. If there are a few labeled data in Target, they can be used with semi-supervised methods to build a baseline model in the Target domain. Consequently, it is relatively more likely for negative transfer to occur. Lastly, if there is an abundance of labeled target data, then transferring from an even slightly different domain while failing to identify the similar part, makes it again likely to deteriorate the performance of the model in Target compared to the baseline model built with these abundant labels. All the above come to show that negative transfer is relative with respect to the amount of available labeled target data. At the same time though, the amount of labeled data significantly impacts (a) the feasibility of discovering similar parts between the joint distributions, (b) the reliability of the similar parts detected by the transfer learning algorithm. As discussed in point 1 the similarity between $P_S(X,Y)$ and $P_T(X,Y)$ is determining. Hence, in the absence of labeled target data one should only merely rely on the marginal distributions $P_S(X)$, $P_T(X)$, which though has been shown to have theoretical limitations (Wang et al., 2019, as cited by Ben-David). If there is a considerable number of labeled target data though, this problem becomes likely to be manageable. This being said, Wang et al. conclude that (Wang et al., 2019)

“Therefore, an ideal transfer learning algorithm may be able to utilize labeled target data to mitigate the negative impact of unrelated source information”

Another study about the effects of negative transfer has been made by Jiménez-Guarneros and Gómez-Gil (Jiménez-Guarneros and Gómez-Gil, 2021), who specifically studied the effect of various distribution shifts and geometric transformations on the performance of deep learning unsupervised domain adaptation (D-UDA) models. The distributions they studied were spherical and non-spherical and they conducted tests about shifts on the marginal distributions as well as on the conditional distributions. Various interesting results and observations were derived. First, the intensity of negative transfer was dependent on all factors, e.g., the model used, the shape of the distribution, the specific geometric transformation and whether the shift regarded the marginal or the conditional distribution. Moreover, spherical distributions showed more robustness under the various transformations in comparison to non-spherical ones. Some other findings were that rotation, shearing and skewness negatively affected all models and particularly the non-spherical distributions under marginal distribution shift. Noise, overlapping clusters and subclusters degraded the models' performance under conditional distributions shifts.

Studies like these are necessary and insightful in order to get a good grasp of the causes and the effect of negative transfer under distributions shift so that it is prevented when a model is built.

3. Theoretical background of experiments

In this chapter we are describing the various algorithms and methods that we used to build base classifiers and those that we used for the Transfer Learning. At first, we explain in detail the two Domain Adaptation methods that we used, Kullback-Leibler Estimation Procedure (KLIEP) and TrAdaBoost. We then write a few things about the three classifiers deployed, Logistic Regression, Gradient Boosting and Random Forest. We describe the two feature selection methods we used, Genetic Algorithm (GA) and the Variance Inflation Factor (VIF) criterion. There follows the description of the method we tried in order to deal with the imbalance in Dataset B, Synthetic Minority Over-sampling Technique (SMOTE), and of Principal Components Analysis (PCA) that we used when we were building the base classifier in dataset B. Lastly, we write about the metrics that we chose to use to evaluate the performance of our models and algorithms.

3.1. Kullback-Leibler Importance Estimation Procedure

KLIEP is a method that aims to tackle the problems that arise when the train and the test samples do not follow the same marginal distribution, otherwise known as *covariate shift.*

"Under covariate shift, standard learning methods such as maximum likelihood estimation are no longer consistent—weighted variants according to the ratio of test and training input densities are consistent. Therefore, accurately estimating the density ratio, called the importance, is one of the key issues in covariate shift adaptation" (Sugiyama, Suzuki, Nakajima, Kashima, von Bünau & Kawanabe, 2008, p.1)

A straightforward yet naïve approach would be to estimate the two densities directly and then compute the importance. However, as already mentioned, density estimation is a hard task - especially in high dimensionality - and it could consequently result in a poorly performing model if densities were not estimated well. Sugiyama et al. propose an alternative approach that involves a direct importance estimation that skips density estimation. Let's articulate a formulation for the problem that is posed in KLIEP.

Problem Formulation: Let $p_{tr}(x)$ and $p_{te}(x)$ be the densities of the training and test distributions, where $\{x_i^{tr}\}_{i=1}^{n_{tr}}$ and $\{x_j^{te}\}_{j=1}^{n_{te}}$ are identically independent distributed samples from a domain *D* and $p_{tr}(x) > 0$ for all $x \in D$. The aim is to estimate the *importance w(x)* defined as

$$w(x) := \frac{p_{te}(x)}{p_{tr}(x)} \qquad (8)$$

Therefore, a method is required so that the importance is estimated. One thing that can be done is to linearly model the importance $w(x)$ as following:

$$\widehat{w}(x) = \sum_{l=1}^{b} a_l \varphi_l(x) \qquad (9)$$

Where $\{a_l\}_{l=1}^{b}$ are parameters to be learned from the data samples and $\{\varphi_l(x)\}_{l=1}^{b}$ are basis functions such that $\varphi_l(x) \geq 0$ for all $x \in D$ and for $l = 1, \dots, b$. Kernel models are also a possible option for modeling the importance, which means that *b* and $\{\varphi_l(x)\}_{l=1}^{b}$ could be dependent on $\{x_i^{tr}\}_{i=1}^{n_{tr}}$ and $\{x_j^{te}\}_{j=1}^{n_{te}}$

The test density can then be estimated from $\hat{p}_{te}(x)$ = $\widehat{w}(x)p_{tr}(x)$. The parameters $\{a_l\}_{l=1}^{b}$ in (4) can be determined as solutions to the problem of minimization of the Kullback-Leibler (KL) divergence from $p_{te}(x)$ to $\hat{p}_{te}(x)$.

Before we formulate the problem, we should first provide a definition for the KL divergence. This is as follows (Jiawei, 2022)

"a non-symmetric measure of the difference between two probability distributions p(x) and q(x). Specifically, the Kullback-Leibler (KL) divergence of q(x) from p(x), denoted $D_{KL}(p(x) \mid\mid q(x))$ is a measure of the information lost when q(x) is used to approximate p(x)"

The mathematical formulation is:

$$D_{KL}(p(x) \mid\mid q(x)) = \textstyle\sum_{x \in X} p(x) ln \, \frac{p(x)}{q(x)} \qquad (10)$$

Let's also note that,

"Although the KL divergence measures the "distance" between two distributions, it is not a distance measure. This is because that the KL divergence is not a metric measure. It is not symmetric: the KL from p(x) to q(x) is generally not the same as the KL from q(x) to p(x). Furthermore, it need not satisfy triangular inequality. Nevertheless, $D_{KL}(P||Q)$ is a non-negative measure. $D_{KL}(P||Q) \geq 0$ and $D_{KL}(P||Q) = 0$ if and only if P = Q". (Jiawei, 2022)

We can now return to the problem of estimating the test density from $\hat{p}_{te}(x) = \hat{w}(x)p_{tr}(x)$. As we said, the parameters $\{a_l\}_{l=1}^{b}$ in (4) can be determined as solutions to the problem of minimization of the Kullback-Leibler (KL) divergence from $p_{te}(x)$ to $\hat{p}_{te}(x)$.

$$D_{KL}\ (p_{te(x)} \,||\, \hat{p}_{te}(x)) = \qquad (11)$$

$$\int_D\ p_{te}(x)log\ (\frac{p_{te}(x)}{\hat{w}(x)\ p_{tr}(x)})dx = \qquad (12)$$

$$\int_D\ p_{te}(x)log\ (\frac{p_{te(x)}}{p_{tr}(x)})\ dx - \int_D\ p_{te}(x)log\ \hat{w}(x)\ dx \qquad (13)$$

We are interested in the second term, as the first one is independent of $\{a_l\}_{l=1}^{b}$. We denote this second term as:

$$J:=\int_D\ p_{te}(x)log\ \hat{w}(x)\ dx\ \approx \qquad (14)$$

$$\frac{1}{n_{te}}\sum_{j=1}^{n_{te}} log\ (\hat{w}(x_{j_{te}}\ )) = \qquad \mathit{(15)}$$

$$\frac{1}{n_{te}}\sum_{j=1}^{n_{te}} log\ (\sum_{l=1}^{b} a_l\varphi_l(x_{j_{te}})) \qquad (16)$$

where the empirical approximation based on the test samples is used from the first line to the second line above. Thus (16) is the objective function to be maximized with respect to the parameters $\{a_l\}_{l=1}^{b}$, and it is concave (Sugiyama et al., 2008)

A first constraint is that $\hat{w}(x) \geq 0,$ by definition of $w(x)$, which is positive. Since the basis functions φ are also positive, the constraint can be posed on a_l as

$$a_l > 0 \text{ for all } l = 1, \ldots, b$$

Moreover, since $\hat{p}_{te}(x) = \hat{w}\,(x)p_{tr}(x)$ is a probability density function, $\hat{w}\,(x)$ should also be normalized. Therefore, we obtain a second constraint via

$$1\ =\ \int_D\ \hat{p}_{te}(x)dx\ = \qquad (17)$$

$$\int_D \hat{w}(x) p_{tr}(x) dx \approx \qquad (18)$$

$$\frac{1}{n_{tr}} \sum_{l=1}^{n_{tr}} \hat{w}\left(x_{i_{tr}}\right) = \qquad (19)$$

$$\frac{1}{n_{tr}} \sum_{l=1}^{n_{tr}} \sum_{l=1}^{b} a_l \varphi_l(x_{i_{tr}}) \qquad (20)$$

where the empirical approximation based on the training samples is used from the first to the second line above. All that considered we have the convex optimization problem

$$max_{\{a_l\}_{l=1}^{b}} [\textstyle\sum_{l=1}^{n_{te}} log\ (\sum_{l=1}^{b} a_l \varphi_l(x_{i_{te}}))]$$

$$subject\ to\ \ \frac{1}{n_{tr}} \sum_{i=1}^{n_{tr}} \sum_{l=1}^{b} a_l \varphi_l(x_{i_{tr}}) = 1\ \ and\ \ a_l > 0\ \ for\ all\ l = 1,2,\dots,b$$

The model $\hat{w}(x)$, e.g. the φ_l functions are selected via likelihood cross validation (LCV) as follows: The test samples are split into *R* disjoint subsets, and each subset is used to get an estimate for $\hat{w}(x)$, so we end up with $\hat{w}_r(x)$ estimates, where $r = 1 \dots, R.$ Subsequently, every importance gives the estimated $\hat{J}$

$$\hat{J}_r := \frac{1}{|X_{r_{te}}|} \sum_{x \in X_{r_{te}}} log\ (\ \widehat{w_r}\ (x)) \qquad (21)$$

And the final estimate for *J* is derived as the mean of the *R*-estimates

$$\hat{J} := \frac{1}{R} \sum_{r=1}^{R} \hat{J}_r \qquad (22)$$

There follows pseudo-algorithm for the model selection by LCV (Sugiyama et al., 2008)

```
Input: $M = \left\{ m_k \middle| m_k = \left\{ \varphi_l^{(k)}(x) \right\}_{l=1}^{b^{(k)}} \right\}, \{x_i^{tr}\}_{i=1}^{n_{tr}},\ and\ \{x_j^{te}\}_{j=1}^{n_{te}}$
Output: $\widehat{w}(x)$
Split $\{x_j^{te}\}_{j=1}^{n_{te}}$ into R disjoint subsets $\{\mathrm{x}_\mathrm{i}^{\mathrm{te}}\}_{r=1}^{R}$;
for each model m∈ $M$:
    for each split $r = 1,2,\ldots,R$
        $\widehat{w}_r(x) \leftarrow KLIEP\left(m, \{x_i^{tr}\}_{i=1}^{n_{tr}},\ \{x_j^{te}\}_{j\neq r}\right)$;
        $\hat{J}_r(m) \leftarrow \frac{1}{|x_r^{te}|} \sum_{x \in X_r^{te}} \log \widehat{w_r}\,(x)$;
    end
    $\hat{J}(m) \leftarrow \frac{1}{R} \sum_{r=1}^{R} \widehat{J_r}(m)$;
end
$\widehat{m} \leftarrow argmax_{m \in M} \hat{J}_r(m)$;

$\widehat{w}(x) \leftarrow KLIEP\left(\widehat{m}, \{x_i^{tr}\}_{i=1}^{n_{tr}}, \{x_j^{te}\}_{j=1}^{n_{te}}\right)$
```

In our implementation we used Gaussian Kernels as the φ functions as Sugiyama et. al. did (Sugiyama et al., 2008). The choice of the hyperparameters σ and B is very important. As the authors write, when the importance $w(x) \coloneqq \frac{p_{te}(x)}{p_{tr}(x)}$ outputs large values, that is when the two distributions differ significantly, a large number of Kernels is required for a good estimation. Therefore, we chose to use many kernels, namely $B = 100$, assuming significant dissimilarity between the distributions. The authors also showed in their paper that the kernel width σ is also determinant in the performance of KLIEP. We tried KLIEP with $\sigma = 1$ and we also used 5-fold cross-validation to choose its optimal value. The best result was obtained with $\sigma = 1$.

3.2. TrAdaBoost

TrAdaBoost (Wenyuan, Qiang, Gui-rong, Yong, 2007) is a domain adaptation method that is inspired by traditional boosting-learning algorithms and extends their application to cases where covariance shift occurs. A significant difference with KLIEP is that trAdaBoost requires some labeled data from the Target domain. The concept is the same; when the available (train) data becomes outdated for some reason the existing model might fail to perform well on new (test) data. TrAdaBoost relies on the idea that some part of the old data presents similarities with the new data and so it can still be useful for the training, e.g., it can be reused. It is a key issue then to find this part of the data that is still useful, and this is the reason that some labeled test data is required: to be able to measure the similarity between the train and the test instances.

The outdated data is called *diff-distribution* data and the new data *same-distribution* data*.* The general idea is to find out by a voting mechanism how useful each instance of the diff-distribution data is and assign a corresponding weight that will help the classifier either put more emphasis on this instance or diminish its contribution to the training. More specifically, the instances will be weighted in a way so that those instances that are more relevant to the same-distribution data will contribute more to building the new classifier whereas the less relevant will play a minor role. In their paper Wenyuan et al. (Wenyuan et al. 2007) prove that boosting learning converges well to the desired model.

Before describing more trAdaBoost it would be good to give the general framework of AdaBoost, the first boosting algorithm developed by Freund and Schapire (Freund & Schapire , 1997). Broadly speaking, Boosting refers to any method that aims to create a strong classifier by using in an interactive mode many “weak” classifiers (Freund & Schapire, 1996). “Weak” is a word used to describe a classifier that does well yet not well enough, in other words, if there is space for improvement. We will describe the idea of AdaBoost considering as learners Decision Trees, although any algorithm can be used.

Consider a set of learners $L_1, \dots, L_k$ that are the simplest decision trees, that is, trees with only one node and two leaves. These are otherwise called decision stumps where each node corresponds to a predictor and the two leaves to the two classes. At first all samples are assigned the same weight, namely $w_i = \frac{1}{n}$, where n is the number of samples. Learner L_1 is trained on all training samples with the weights w_i and the total classification error ε_t is calculated for learner L_1 on the test set. The total classification error is then used to determine the strength of this learner, let us call it B_i. Each of the samples is then assigned a new weight w_i based on B_i and whether it was correctly or incorrectly classified by L_1. The rule is that the incorrectly classified samples increase their weight to let the next learner know that it should emphasize on them and improve. Moreover, the stronger the L_1, the more the weight increases. Following this logic, the correctly classified samples decrease their weight, largely if the learner L_1has a large S_1 and little otherwise. The re-weighted dataset is then passed to L_2 for training and testing, and likewise, the total classification error, the strength of the learner L_2 and the new weights of the samples are calculated. This process continues until maximum iterations are reached or until an accepted tolerance level is reached.

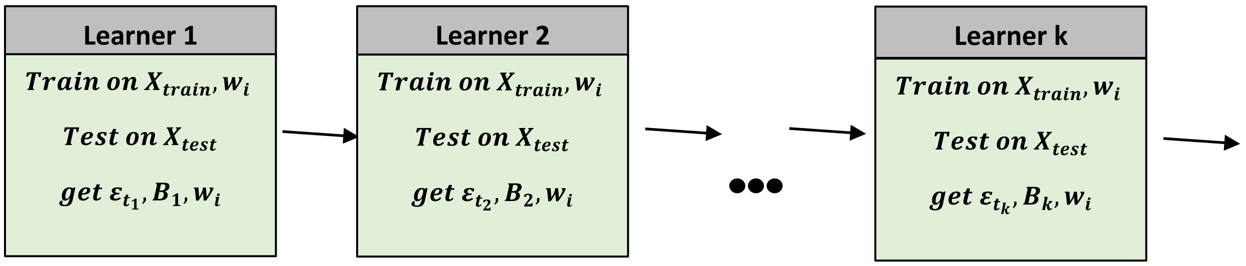

Learner 1
$Train\ on\ X_{train}, w_i$
$Test\ on\ X_{test}$
$get\ \varepsilon_{t_1}, B_1, w_i$
Learner 2
$Train\ on\ X_{train}, w_i$
$Test\ on\ X_{test}$
$get\ \varepsilon_{t_2}, B_2, w_i$
Learner k
$Train\ on\ X_{train}, w_i$
$Test\ on\ X_{test}$
$get\ \varepsilon_{t_k}, B_k, w_i$

Figure 6: The sequential way in which AdaBoost works. Each learner is helped by the previous one.

The final classifier is a weighted majority voting classifier. Specifically:

"for a given instance x, h_{fin} outputs the label y that maximizes the sum of the weights of the weak hypotheses predicting that label. The weight of hypothesis h_t is defined to be $ln\,(\frac{1}{b_t})$, so that greater weight is given to hypotheses with lower error". (Freund & Schapire, 1996, p.13)

Let's recall that trAdaBoost is an extension of AdaBoost that also makes use of the labeled samples from the same-distribution data. The training takes place by using the same-distribution samples together with the labeled samples from the diff-distribution data. During the training process TrAdaBoost assigns weights to both the same-distribution and diff-distribution samples in a way that tells the classifier how much to emphasize, how to be influenced by each sample according to some criteria. The following picture illustrates the process that this is done.

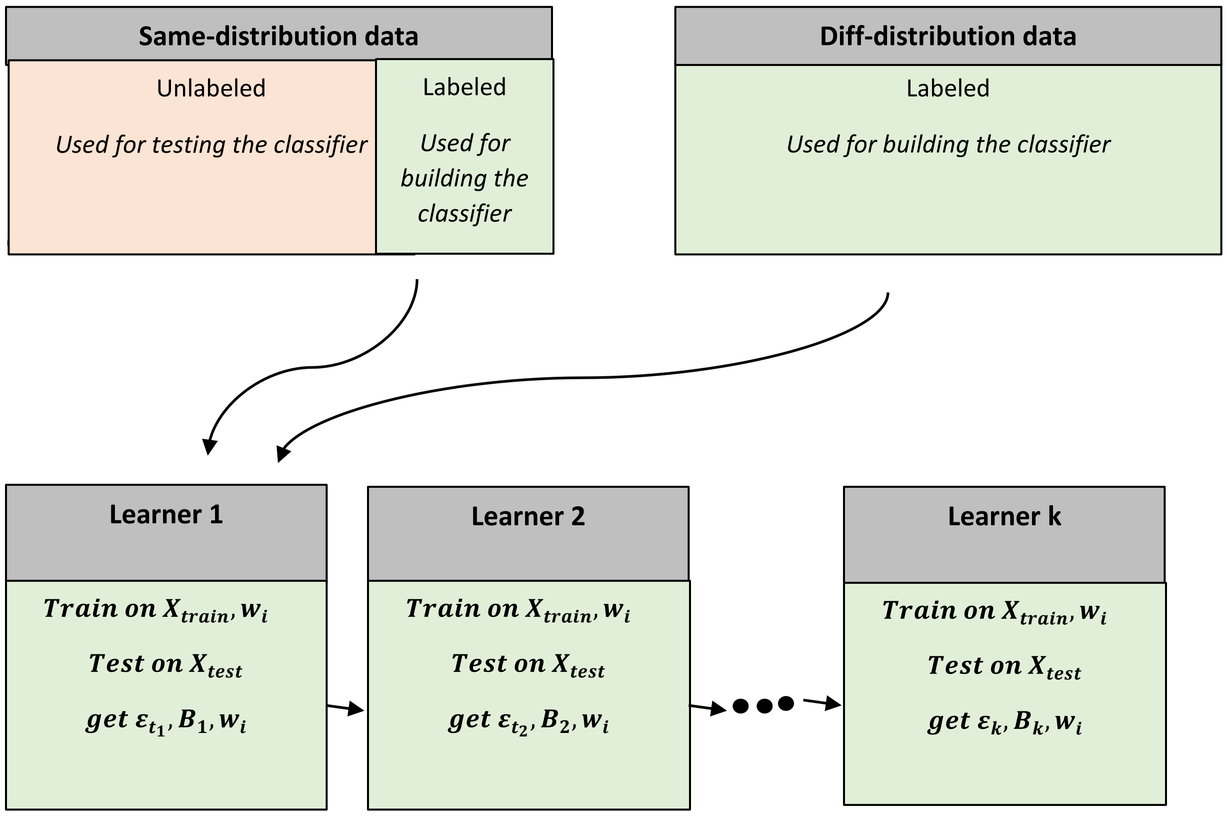

Same-distribution data
Unlabeled
Used for testing the classifier
Labeled
Used for building the classifier
Diff-distribution data
Labeled
Used for building the classifier
Learner 1
$Train\ on\ X_{train}, w_i$
$Test\ on\ X_{test}$
$get\ \varepsilon_{t_1}, B_1, w_i$
Learner 2
$Train\ on\ X_{train}, w_i$
$Test\ on\ X_{test}$
$get\ \varepsilon_{t_2}, B_2, w_i$
Learner k
$Train\ on\ X_{train}, w_i$
$Test\ on\ X_{test}$
$get\ \varepsilon_k, B_k, w_i$

Figure 7: Graphical illustration of the way trAdaBoost works. Each learner is helped by the previous one, as in AdaBoost, but here labeled Target labels are exploited too for the training.

TrAdaBoost has been applied to Credit Scoring problems. For example, Xiao et. al. (Xiao, Wang, Teng & Hu, 2014) used it along with other strategies in credit scoring data where covariance shift occurred, and trAdaBoost provided the second best roc-auc score among six methods.

Before we describe the algorithm let's introduce some notation and definitions:

- X_s: the same-distribution data
- X_d: the diff-distribution data
- $Y = \{0,1\}$: the set of class labels
- h: a Boolean function $h: X_s U X_d \to Y$ called *hypothesis*
- $T_s = \{(x_i^s, h(x_i^s))\}$: the same-distribution training data; the labeled part of the diff-distribution data $X_s = \{x_i^s \mid i = 1, \ldots, m\}$
- $T_d = \{(x_i^d, h(x_i^d))\}$: the diff-distribution training data that corresponds to $X_d = \{x_i^d \mid i = 1, \ldots, n\}$
- $T = T_s U T_d$: all training data. Precisely, $T = \{ {x_i^d, i=1,\ldots,n \atop x_i^s, i=n+1,\ldots,n+m}$

- $S = \{(x_i^t)\}$: the test data. This is the unlabeled part of X_s, e.g., $x_i^t \in X_s,\ i = 1, \dots, k$ and k is the number of unlabeled instances of X_s

So, the problem can be postulated as: "having a small portion of labeled instances from X_s, abundant labeled instances from X_d and many unlabeled instances from X_s train a classifier so that the hypothesis $h: X_s U X_d \to Y$ has the minimum prediction error on the unlabeled data" (Wenyuan et al. 2007). Here follows the pseudocode of the algorithm:

```
Input: $T_d, T_s, Learner, N = max\_iter$
Initialize: $w^1 = (w_1^1, \dots, w_{n+m}^1)$
For t = 1,…,N :
 1. Set $p^t = \frac{w^t}{\sum_{i=1}^{n+m} w_i^t}$
 2. Call Learner
    train on $T$ that follows the distribution $p^t$
    test on $S$
    get a hypothesis $h_t: X \to Y$
 3. Calculate the error of $h_t$ on $T_s$
    $\varepsilon_t = \sum_{i=n+1}^{n+m} \frac{w_i^t |h_t(x_i) - c(x_i)|}{\sum_{i=n+1}^{n+m} w_i^t}$
 4. Set $\beta_t = \frac{\varepsilon_t}{1-\varepsilon_t}$ and $\beta = \frac{1}{1+\sqrt{2ln\frac{n}{N}}}$  ; Condition $\varepsilon_t < \frac{1}{2}$ must be satisfied
 5. Update the weight vector
    $w_i^{t+1} = \begin{cases} w_i^t \beta^{|h_t(x_i) - c(x_i)|}, & 1 \le i \le n \\ w_i^t \beta_t^{-|h_t(x_i) - c(x_i)|}, & n+1 \le i \le n+m \end{cases}$
 6. Output: the hypothesis
    $h(x) = \begin{cases} 1, & \prod_{t=\lceil \frac{N}{2} \rceil}^{N} \beta_t^{-h_t(x)} \ge \prod_{t=\lceil \frac{N}{2} \rceil}^{N} \beta_t^{-\frac{1}{2}} \\ 0, & otherwise \end{cases}$
```

As one can see, in each iteration, every instance from the diff-distribution data that is incorrectly classified has its weight decreased by $\beta^{|h_t(x_i)-c(x_i)|}$ (note that $\beta \in (0,1]$). This is because, since the learner is fed with both same and diff-distribution instances, it is likely that the misclassified instances from X_d come in conflict with X_s. For example, suppose that the instances in X_d are located to a two-dimensional space as shown in figure 8, forming these two clusters. When the instances from X_s are added, the learner is influenced by their distribution's parameters and hence the formed clusters might change in a way that the previously correctly predicted instances from X_d are now misclassified.

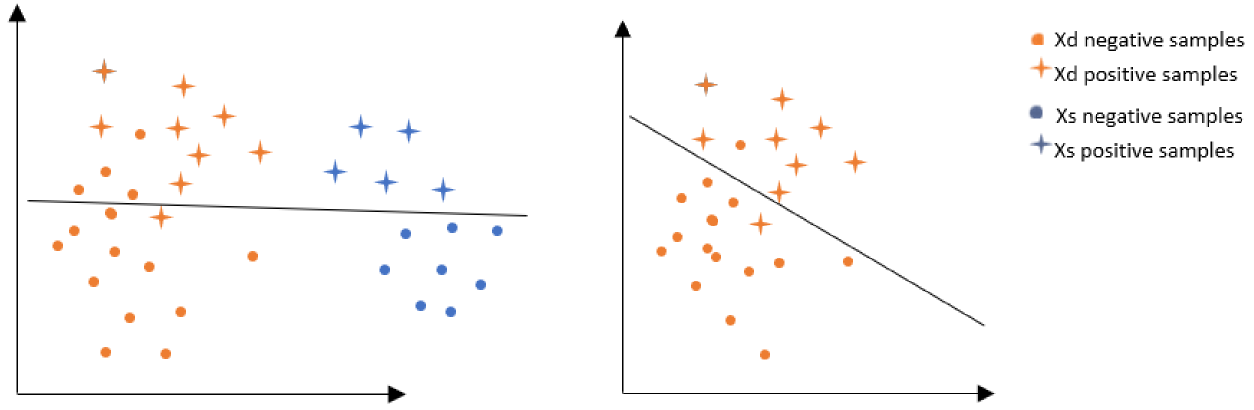


Figure 8: How the distribution of the data can change when more samples are appended in a dataset; a spatial illustration

Similarly, every instance from the same-distribution data that is misclassified increases its weight by $\beta_t^{-|h_t(x_i)-c(x_i)|}$ (where again $\beta_t < 1$). In this way, the instances from X_d that are dissimilar to same-distribution instances will not affect the classifier's learning significantly, while at the same time the instances from X_s that are misclassified will be emphasized in order to help the classifier learn them better (Freund & Schapire, 1996), (Zheng, Liu, Yan, Jiang, Zhou & Li, 2020). A drawback of AdaBoost that is also reflected in trAdaBoost is that it is not good at handling hypotheses that give out error greater than $\frac{1}{2}$ (Freund & Schapire, 1996) (Zheng et al., 2020).

Some alternations of AdaBoost have been proposed to tackle this, as for example measuring the distance of each X_d instance to X_s domain (Zheng et al., 2020); in this way, when an X_d instance is misclassified but it is close to X_s distribution, its weight is increased, whereas when it is far from X_s distribution its weight is decreased. Maximum Mean Discrepancy (MMD) in reproducing kernel Hilbert space (RKHS) is used to measure the distance. In our code we follow the original approach and when $\varepsilon_t > \frac{1}{2}$, ε_t is set to $\frac{1}{2}$ which makes β_t to retain its value and consequently w_t too.

3.3. Logistic Regression

Logistic Regression is a statistical method used for binary classification. The desired model is learned from a family of hypothesis functions $h : R^d \rightarrow [0,1]$. Although the outcome of h is some number in the continuous interval $[0,1]$, which indicates a probability, this is then turned and interpreted as a decision in a binary classification problem (Suryanto, Guan, Voumard & Beydoun, G., 2019), (Bishop, 2006). The hypothesis class in Logistic Regression is basically the composition of a sigmoid function over the class of linear functions (Suryanto et al., 2019).

In linear regression the hypothesis function to be learned is a linear function of the weights β_i for the predictors x_i, where $i \in R$, e.g., $h_L(x) = \beta_0 + \beta_1 x_1 + \cdots + \beta_k x_k$. If we pose the log of the odds $log \frac{p}{1-p}$ to be equal to $h_L(x)$, we get

$$log \frac{p}{1-p} = \beta_0 + \beta_1 x_1 + \cdots + \beta_k x_k \qquad (23)$$

$$\frac{p}{1-p} = e^{\beta_0 + \beta_1 x_1 + \cdots + \beta_k x_k} \qquad (24)$$

$$p = \frac{1}{1+e^{-z}} \quad , \quad posing\ z = \sum_{i=1}^{k} \beta_i x_i \qquad (25)$$

And then posing the probability p to be the sigmoid function of the z we get

$$\sigma(z) = \frac{1}{1+e^{-z}} \qquad (26)$$

Which explains how the model learned is the composition of the sigmoid function with a linear function of the weights to be learned. The sigmoid on z gives the probability of obtaining class 0; if this number is > 0.5 then the classifier calls class 0, else it calls class 1.

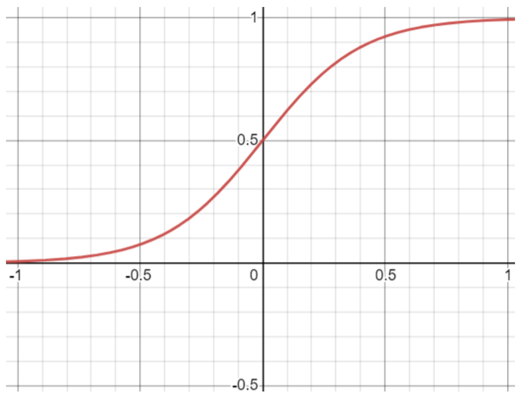


Figure 9: How a sigmoid function look like

The loss function for Logistic Regression is defined as

$$l\big(h(x,y)\big) = -ylog(h(x)) + (1-y)log\ (1-h(x)) \qquad (27)$$

where $h(x)$ denotes the class label predicted by the learned hypothesis on x, and y is the actual class label of x. This function, commonly known also as *cross-entropy,* is convex and can be proved to converge with standard methods (Suryanto et al., 2019, p. 127).

Logistic Regression has been used in past research in credit scoring and transfer learning problems (Wei, Liu & Wu, 2021), (Beninel, Bouaguel & Belmufti, 2012). This statistical model has the advantage that is simply interpretable and provides the probabilities of a sample being in a class, two characteristics that are highly appreciated in credit scoring. For instance, one reason that Channel VAS credit scoring team uses this model is that there might be the need to explain to some client why a user was denied a specific loan, so the team can use some simple math to explain such cases if requested.

We applied Logistic Regression in dataset B using sklearn's module. After experimentation with the hyperparameters, we tuned the model to have the "liblinear" solver along with "L1" penalty, which is also called Lasso, from Lasso regression. L1 regularization adds $\lambda\big||\beta_i|\big|$ as a penalty to the coefficient β_i , where λ is some real number that regulates the amount of shrinkage of the coefficient; the greater λ is the smaller the coefficient becomes. The hyperparameter C in sklearn takes the role of λ and it is called the inverse regularization parameter ($C = \frac{1}{\lambda}$); we set $C = 10$. Number of iterations was set to 2,000 or 3,000 and tolerance to 0.0001.

On the Lending Club dataset, the respective hyperparameters were set to "liblinear", "L1", 1, 5, 0.001.

3.4. Gradient Boosting Classifier

Gradient Boosting was originally developed as a regression algorithm (Friedman, 2001) but it was then modified in order to be used as a classifier. It is an algorithm grounded on AdaBoost and as a classifier it also uses the log odds, like logistic regression. It makes use of a number of weak learners that sequentially boost each other in order to eventually obtain a strong classifier. Unlike

AdaBoost, it uses trees with more than one leaf, while in every iteration it predicts residuals (actual value – predicted value) and not classes.

Let's consider a specific loss function – although many functions can be chosen for this purpose, as long as they are differentiable – and describe the algorithm in steps. Let the loss function then be

$$L(y_i, p_i) = -y_i log\, p_i - (1 - y_i) log\,(1 - p_i) \qquad (28)$$

for sample x_i where y_i is the actual label and p_i is the predicted probability. The loss function can also be written as

$$L(y_i, p_i) = -y_i\gamma_i + log\,(1 + e^{\gamma_i}) \qquad (29)$$

if we pose $\gamma_i = e^{odds} = e^{\frac{p_i}{1-p_i}}$.

Let also $X = \{x_1, \ldots, x_n\}$ be the sample, observed data and $Y = \{y_1, \ldots, y_n\}$ the labels of the samples.

The first step is to initialize the model with a constant value as the prediction for all samples. This is obtained by minimizing the loss function.

$$first\ prediction = f_0(x) = argmin_\gamma \sum_{i=1}^{n} L(y_i, \gamma)\ , \forall x \quad (30)$$

$$f_0(x) = argmin_\gamma \sum_{i=1}^{n} [-y_i \log \gamma + \log(1 + e^\gamma)] \qquad (31)$$

We can find p by setting the derivative to be zero

$$\frac{\partial \sum_{i=1}^{n} [-y_i \log \gamma + \log(1 + e^\gamma)]}{\partial \gamma} = 0 \qquad (32)$$

$$\sum_{i=1}^{n} [-y_i + \frac{e^\gamma}{1 + e^\gamma}] = 0 \qquad (33)$$

And if we replace back γ with $log\left(\frac{p}{1-p}\right)$, this will yield $p = \sum_{i=1}^{n} \frac{y_i}{n}$, e.g., the average of the observed sample labels.

So, in steps the Gradient Boosting Classifier algorithm can be written as:

```
Input: learning rate $a$, max iterations M
Step 1: Initialize the model with a constant value
    $$f_0(x) = argmin_\gamma \sum_{i=1}^{n} L(y_i, \gamma)$$
 Step 2: for k=1 to M:
        (a)  Compute $r_{ik} = -\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}, where\ F(x) = f_{k-1}(x)\ for\ i = 1, \ldots, n$
        (b)  Fit a tree model to $\{r_{ik}\}$ that will give leaf nodes
             with residuals $R_{ij}$
        (c)  For $j = 1, \ldots, J_k$, calculate $\gamma_{jk} =$
             $argmin_\gamma \sum_{x_i \in R_{ij}} L(y_i, f_{k-1}(x_i) + \gamma)$
        (d)  Update the model: $f_k(x) = f_{k-1}(x) + \alpha \sum_{j=1}^{J_k} \gamma_k I(x \in R_{jk})$
```

The algorithm thus works by reducing the residuals with every learner, which respectively then produces a better predicted probability by the subsequent learner in the subsequent iteration.

We used *GradientBoostingClassifier* from sklearn *ensemble* module and we tuned the learning rate to be 1, the maximum depth of the trees to be 1, the number of learners to be 4 and the loss function to be *"deviance",* the same as of the logistic regression's cross-entropy.

3.5. Random Forest Classifier

Random Forest in a bagging algorithm for regression and classification. It makes use of decision trees, namely, it builds a big set of de-correlated trees and averages their decisions (Hastie, Tibshirani & Friedman, 2009). It is well known that decision trees are susceptible to variance, so by using aggregately multiple noisy but quiet unbiased models with bootstrap variance is reduced. It is good to briefly overview decision trees and then explain how Random Forests are built on them.

Being an intuitive, simple to understand and tune algorithm with an inherent ability to deal with missing values, Decision Trees have been largely popular for regression and classification problems. A decision tree classifies data samples by sequentially asking questions derived from features.

“Each question is contained in a node, and every internal node points to one child node for each possible answer to its question. The questions thereby form a hierarchy, encoded as a tree” (Kingsford & Salzberg, 2008, p.1)

The first question is called *root node,* the subsequent questions are called *children nodes* and the last nodes that contain the classifications (0 or 1) are the *leaves.* For each split, the choice of the predictive variable is made upon the measurement of the *impurity* of the split. Ideally, a question should accurately separate all samples into different classes, in other words, be pure, but this is far too optimistic. One of the most popular and used impurity measures is the *Gini index,* which for a specific *node* s is defined as

$$Gini_s = 1 - \sum_{i=1}^{m} p_i^2 \qquad (34)$$

Where m is the number of items that are classified by node s, and p_i is the fraction of samples classified by s that belong to class i. This quantity becomes zero if and only if all samples classified by the node belong to the same class. Highlight though that this is the Gini index of a node; a variable, when used as a question splits the samples into $k \geq 1$ nodes, and the Gini index of the predictive variable is the weighted average of the Gini indexes of all the k nodes.

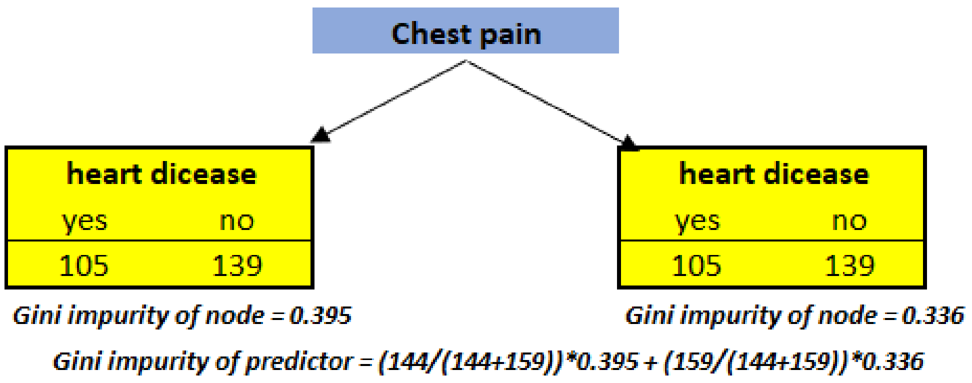

Chest pain
heart dicease
yes
no
105
139
heart dicease
yes
no
105
139
Gini impurity of node = 0.395
Gini impurity of node = 0.336
Gini impurity of predictor = (144/(144+159))*0.395 + (159/(144+159))*0.336

Figure 10: Example of the calculation of Gini impurity of a predictive variable

The variable that takes a node every time is the one with the smallest Gini impurity. Note that for a further split to take place to a node, the new Gini impurity must be smaller than the current one.

Random Forests build on this process using multiple decision trees in an ensemble mechanism. Given a dataset with N samples, setting the number of trees to be B, we have:

```
Input: N: Dataset of size N, B: number of trees for the ensemble, M: max depth of the trees

For b=1, …, B:

    1. Draw a bootstrap sample Z* of size N; the samples left out from the
       bootstrap are called Out Of Bag (OOB) samples.
    2. For various values of $k$, create trees with $k$ predictors each of max
       depth M by using the bootstrapped sample.
    3. Classify the OOB samples with majority voting. Choose the $k$ that
       gives the lowest OOB error (misclassified samples).
```

3.6. Genetic Algorithm

Feature selection describes the process of selecting a subset of predictive features to train a model. The purpose of feature selection is twofold; it can help reduce the time and the resources required, while at the same time improving the performance of the model by removing the non-useful features.

«Many models, especially those based on regression slopes and intercepts, will estimate parameters for every term in the model. Because of this, the presence of non-informative variables can add uncertainty to the predictions and reduce the overall effectiveness of the model» (Kuhn & Johnson, 2013, p.488)

A first categorization of feature selection algorithms is that of supervised and unsupervised ones. Supervised algorithms take into consideration the target variable, while unsupervised algorithms do not; the latter use other criteria such as the features' variance, entropy, or their ability to preserve local similarity etc. A second categorization falls between wrapper and filter methods. Wrapper methods evaluate a set of models according to a specific metric, on different subsets of the initial feature space. Otherwise stated, they repetitively search among the predictors to determine which ones improve the current model when they enter it.

Stepwise elimination, simulated annealing and genetic algorithms fall into this category (Kuhn & Johnson, 2013). Filter methods select the predictors without involving the model. These methods usually evaluate each predictor separately, something that makes them prone to multicollinearity and multiplicity problems, that can be addressed with using simultaneously various statistical tests. Lastly, let's mention that there are machine learning algorithms for which feature selection is an intrinsic part of them. Such algorithms are tree-based algorithms and those that use penalization, like Lasso and Ridge regression.

In this thesis Genetic Algorithm (GA) was used as a primary feature selection technique for Dataset B, which was then followed by some intrinsic algorithms that were tried while developing a model. The way GAs function is inspired by concepts of population evolution in biology. Babatunde et. al. write,

"The operations in a GA are iterative procedures manipulating one population of chromosomes (solution candidates) to produce a new population through genetic functionals such as crossover and mutation (in a similar way to Charles Darwin evolution principle of reproduction, genetic recombination, and the survival of the fittest)." (Babatunde, Armstrong, Leng & Diepeveen, 2014, p.3)

In biology, chromosomes consist of genes and occasionally perform crossover and mutation. Under these processes the strongest or fittest new chromosomes (children) take over the old chromosomes (parents). The set of possible chromosomes is referred to as population. In machine learning language, chromosomes are binary vectors and genes are features. Therefore, the presence or absence of a feature in the dataset is indicated by the respective binary value in the vector. In other words, if the value of the n_{th} element of the vector is 1 it means that the n_{th} feature will participate in the dataset, and if the value it is 0 it will not. Fitness or strength is the outcome of the evaluation of a binary vector on an objective (fitness) function. The values of this evaluation are used in ranking the chromosomes at a certain iteration of the algorithm (Babatunde, Armstrong, Leng & Diepeveen, 2014). The postulation of the fitness functions turns thus, the problem of feature selection into a convex optimization problem.

Assuming that the feature space is of dimension n, the size of the population, e.g. the total number of possible chromosomes, is 2^n. Exploring the whole space would thus be a very costly procedure. To avoid this GA usually starts with a random subset of the population, which forms the first generation. All the chromosomes' fitness is then calculated, and the two strongest chromosomes are selected in order to "reproduce". Reproduction consists of two processes, mutation and crossover. In crossover, the two chromosomes are split at a specific position - randomly chosen - and they exchange one part of the split. Picture 1, illustrating crossover, shows the selected chromosomes of a certain generation that consist of genes C,H,A,G,E and B,D,H,F,A,C respectively. The split takes place at the fifth position and so the yellow part of chr_i exchanges place with the green part of chr_j. Note here that because gene A already participates in chr_i, after the crossover only gene F is added to it.

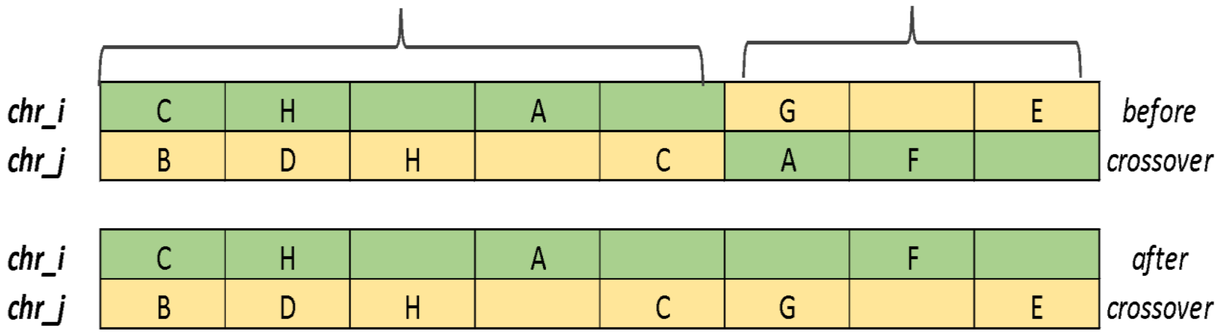


Figure 11: Crossover operation

In mutation, the binary value of every feature is changed into the other one, so the selected chromosome drops all the existing genes and takes all the remaining ones. Picture 2 illustrates mutation.

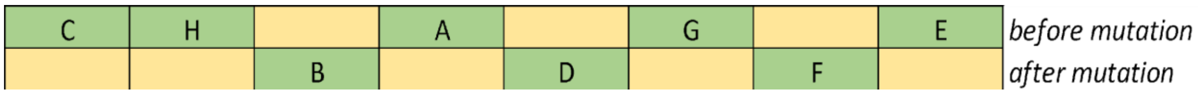


Figure 12: Mutation operation

The resulting chromosomes after selection, crossover and mutation are called children and they replace their parents (initially selected chromosomes). This way, the new generation is formed. Both crossover and mutation occur with a probability, which turns them into hyperparameters of a GA along with the population size and the number of generations. Hassanat et al. explain:

“Determining the interactions that occur among different GA parameters has a direct impact on the quality of the solution, and keeping parameters values "balanced" improves the solution of the GA.” (Hassanat, Almohammadi, Alkafaween, Abunawas, Hammouri & Prasath, 2019, p.5)

Crossover operation allows selected parents to exchange genetic material, resulting in chromosomes that are likely to be stronger, as they have genes from both parents. In other words, it contributes to the balancing between exploration and exploitation (Hassanat & Alkafaween, 2017). However, using only the crossover operation runs the risk of making the GA stacking to local optima areas, as the algorithm would basically produce copies of the initial population without enriching with new members the new generations *.”* (Hassanat et al., 2019). Mutation is the operation that plays this very role of providing new children distinct from their parents and boosts the diversity in the population.

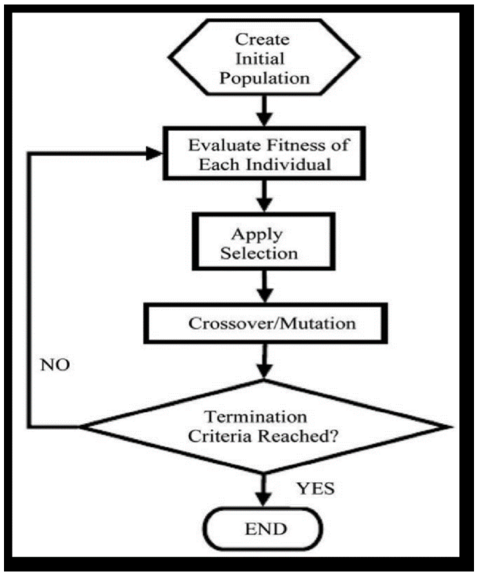

Create Initial Population
Evaluate Fitness of Each Individual
Apply Selection
Crossover/Mutation
NO
Termination Criteria Reached?
YES
END

Figure 13: visualization of GA phases (Hassanat et al., 2019)

We experimented with combinations for (a) population size, (b) number of generations, (c) crossover probability, (d) mutation probability, guided by the work of Hassanat et al. (Hassanat et al., 2019) that includes a literature review over this problem. We chose the roc-auc scoring function for the fitness function to evaluate the individuals, and the final parameters we concluded to are shown in table below.

Population size	100
Number of generations	10
Mutation probability	0.05
Crossover probability	0.8

Table 1: tuning of Genetic Algorithm

3.7. Variance Inflation Factor

When building a model one must select the set of variables to use for the regression or classification. If the predicted variable is uncorrelated with some of the predictors, then one should pose the question of the adequacy of the selected variables in the model. One thing that can cause this low performance in multiple regression analysis is collinearity and multicollinearity.

"Collinearity is a linear association between two explanatory (predictor) variables. Two regressor variables are perfectly collinear if there is an exact linear relationship between the two. Multicollinearity: Multicollinearity refers to a situation in which two or more explanatory (predictor) variables in a multiple regression model are related with each other and likewise related with the response variable." (Akinwande, Dikko & Samson, 2015, p.2)

The reason why multicollinearity negatively impacts the model is that it inflates the standard errors of the coefficients without any reason, and this in turn tells the model that these variables are statistically insignificant for the model, whereas without the multicollinearity they would be significant (Akinwande et al., 2015).

One method that estimates the degree of multicollinearity is the Variance Inflation Factor (VIF), which, for a particular predictor x_j, can be described as the ratio of the variance of its coefficient β_j when all variables fit the model over the variance of β_j if only x_j first the model (James, Witten, Hastie & Tibshirani, 2013). In other words, it assesses for a specific predictor how much its variance is inflated under the presence of multicollinearity; if it is highly inflated its contribution to the model cannot be evaluated properly.

The mathematical formulation of VIF is given by the formula:

$$VIF(\hat{\beta}_j) = \frac{1}{1-R^2_{X_j|X_{-j}}} \qquad (35)$$

Where $R^2_{X_j|X_{-j}}$ is the Pearson correlation coefficient R^2 from a regression of X_j onto all of the other predictive variables (James, Witten, Hastie & Tibshirani, 2013). If $R^2_{X_j|X_{-j}}$ has value close to 1 it signifies high multicollinearity which it turn increases the value of $VIF(\hat{\beta}_j)$, while when $R^2_{X_j|X_{-j}}$ is close to 0 then VIF's value goes towards 1. A general rule states that the smallest value of VIF is 1, when there is no multicollinearity, values between 1 and 5 indicate the existence of some multicollinearity but not of a degree of concern, and values greater than 5 require to handle in some way these predictors.

We applied the VIF criterion to Lending Club Dataset variables and we removed all the variables whose VIF exceeded the threshold 5. We conducted the experiments then both with the full dataset and the one after the removal of these predictors. The effect and full results are presented in chapter 5.

3.8. Dimensionality reduction – Principal Components Analysis

Among the methods we tried in our effort to enhance the performance of the classifier for Dataset B was dimensionality reduction. Dataset B originally lived in R^{236} and all tis variables were numeric. Many of the variables were correlated, something that we initially spotted by reading their interpretation and then we confirmed by plotting a correlation heatmap. For example, *anstd13to18ma* and *anstd13to24ma* denoted the advances number standard deviation 13 to 18 and 13 to 24 months ago respectively, two values that are correlated.

To address this issue, we applied a well-known technique known as Principal Component Analysis (PCA). PCA is a quite old technique, as it was first invented by Karl Pearson in 1901 and later developed by Harold Hotelling during the 1930s. Let us first provide a definition originally given by Hotelling in 1933

"PCA can be defined as the orthogonal projection of the data onto a lower dimensional linear space, known as the principal subspace, such that the variance of the projected data is maximized" (Bishop, 2006, p.561).

or, equivalently

"as the linear projection that minimized the average projection cost, defined as the mean squared distance between the data points and their projections". (Bishop, 2006, p.561).

Let's assume we are in R^2 and we have a set of samples as shown in picture 8. We want to reduce the features' dimension onto *R.* We can do this by projecting every sample into a vector onto R^2 and consider the projections of all the samples instead of the original ones. There are infinitely many choices to choose such a vector, however, the optimal would be the one that maximizes the projected points' variance, so that we lose the minimum information possible.

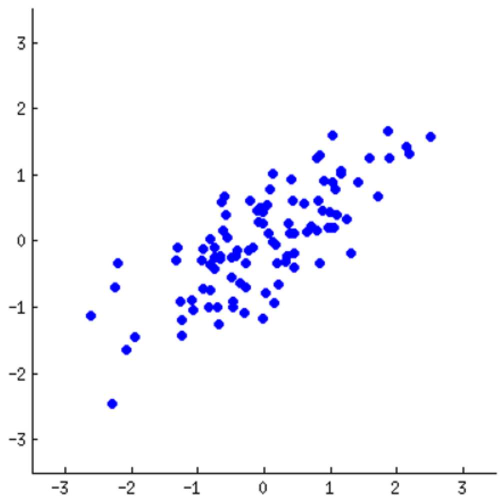


Figure 14: Samples in R^2

For example, comparing the projection onto the two vectors as shown in pictures 9 and 10, we can see that in the former case the projected points look more compact, hence retain less variance, hence losing more information.

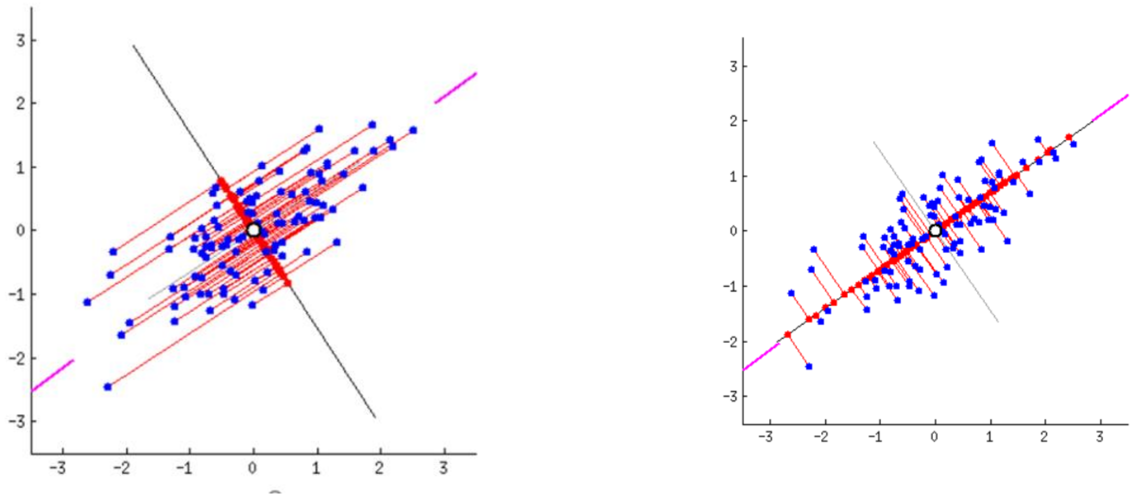


Figure 15: not optimal projection (left), optimal projection (right)

Mathematically speaking, let $X = \{x_i,\ \dots,\ x_d\}$ be the samples set and u_1 he vector onto which they are projected. We want to maximize the quantity *Var(*$u_1 x$) for u_i, or equivalently, the quantity

$u_1^T \mathrm{S} u_1$, where S is the covariance matrix of X, which translates into the following convex optimization problem:

$$max_{u_1} u_1^T \mathrm{S} u_1$$
$$\text{under the constraint } u_1^T \mathrm{S} u_1 = 1$$

The constraint is posed because the quantity $u_1^T \mathrm{S} u_1$ is quadratic and monotonically increasing when u_1 is free.

This is a convex optimization problem that can be resolved with Lagrangian multipliers and it translates into

$$max_{u_1} L(u_1, \lambda) \ = u_1^T \mathrm{S} u_1 - \lambda(u_1^T \mathrm{S} u_1 - 1) \qquad (36)$$

So,

$$\frac{\partial \mathrm{L}}{\partial u_1} = 2\mathrm{S} u_1 - 2\lambda u_1 = 0 \rightarrow S u_1 = \lambda u_1 \qquad (37)$$

Which results in obtaining the solution by determining the max eigenvalue and its corresponding eigenvector.

This can be generalized as moving from R^d to R^p where *p<d,* where u_1,… u_d are the principal components that span the p-dimensional space onto which the samples are projected. Principal components are orthonormal linear transformations of the original spanning vectors of R^d.

PCA is an uncostly technique that can speed up the training of a model and increase the predictive ability of the model as it reduces or even eliminates multicollinearity that can lead to overfit.

3.9. Stratified train-test split

The next step after choosing the methods and algorithms to use and tuning the hyperparameters is to train the model on the training set and evaluate its performance on some data points that it has never seen. Therefore, one part of the dataset is used for the training process, and another one, usually smaller, to the validation process. This is commonly known as train-test split. The model's parameters are tuned with the guidance of the loss function used.

One essential parameter to consider when splitting the dataset is whether the dataset's classes are balanced or not. In Dataset B the degree of imbalance was very high, with the minority class (positive class) to constitute merely 3.79% for Source and for Target 2.23%. Thus, when splitting the dataset one should ensure that both train and test part contain the same ratio of 0 and 1 labeled samples. Assume this condition is not satisfied, and that the train set contains none of the 1-labeled samples. Then the model will be trained to identify the 0-labeled but will have no clue about the 1-labeled. Hence, when it is fed with 1-labeled samples in the validation process, its decision will not have grounds on some training process via which it learned those samples' distribution, but it will simply apply the parameters that were obtained with respect to the training of the 0-labeled samples. In other words, the model will have learned nothing for the one of the two classes, which is evidently problematic.

Stratified split offers a solution to this problem. It is an alteration of the traditional train-test split that during the process of dividing the dataset into train and test, it takes into consideration the class ratio. So, if a dataset consists of 980 positive and 20 negative samples, and we decide to allocate 50% of it for training and 50% for validation, then 490 positives and 10 negatives will be allotted to each subset.

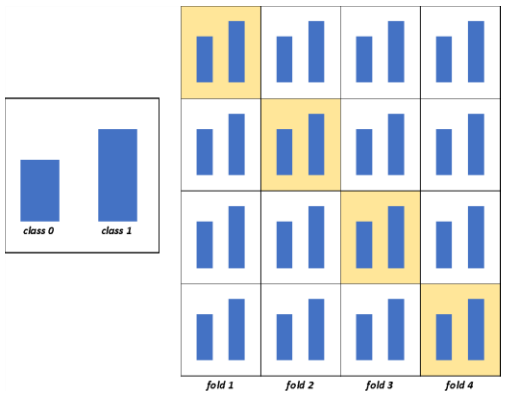


Figure 16: Stratified train-test split in a 4-fold cross validation

Train-test split is a validation technique that requires a sufficiently large dataset to work well. What "sufficiently large" means, is of course subject to each particular problem, but with too few data points the model will probably not be able to learn sufficiently. The necessity of a large dataset is more evident in the case of an imbalanced dataset where the minority class is harder to learn. We used Sklearn's implementation of stratified train-test split and applied it to Dataset B to ensure that the two splits contain the same ratio of majority to minority class samples.

3.10. Synthetic Minority Over-Sampling Technique

The Dataset B was highly imbalanced with the minority class to hold merely 3.79% in Source and 2.23% in Target. This is reasonable as the Credit Scoring aims, among other things, at minimizing the number of users who do not pay their loan back. The dataset is split into segments with respect to the recharge amount, and in general, the users who spend less on recharging are less trustworthy and more likely to default on their loan. The table below depicts the default rates per segment in Source and Target.

	segment 1	**segment 2**	**segment 3**	**segment 4**	**segment 5**	**segment 6**	**segment 7**	**segment 8**
Source	4.53%	5.27%	4.48%	4.15%	4.56%	3.99%	2.77%	1.16%
Target	2.75%	2.87%	3.21%	2.77%	1.43%	1.72%	1.00%	1.97%

Table 2: Default rates of Source and Target in Dataset B

The small number of positive samples in comparison to the negative samples makes it hard for a classifier to learn the positive class. The problem has been observed in applications of diverse nature, such as the detection of oil spills in satellite radar images, of fraudulent phone calls, in information retrieval and filtering, in in-flight helicopter gearbox fault monitoring and also in diagnosing infrequent medical cases like thyroid diseases (Japkowicz & Stephen, 2002). The sample size, the level of imbalance and the complexity of the problem to be solved are three determinant factors for the impact of the imbalance in the model's performance (Japkowicz & Stephen, 2002). The second one is definitely present to a large extent in our case.

A number of methods have been proposed to deal with class imbalance, which could be categorized into three big categories: oversampling, undersampling and cost-sensitive training. Cost-sensitive training is based on the idea that every misclassified sample is assigned a cost and these costs take part in the training process. The goal is to minimize this cost of misclassification while not all misclassifications are treated equivalently; instead, they are assigned weights (Ling & Sheng, 2008). This approach has been tried also in neural networks with positive results (Zhou & Liu, 2005). The undersampling method occurs in the majority class. It will reduce the number of negative samples (without loss of generality, the negative class can be considered as the minority) and bring the dataset to a more balanced state. An important drawback of this method is that the sample size might be significantly reduced. Oversampling is done to the minority class and it involves increasing the number of positive samples by bootstrapping or by creating synthetic positive samples. One of the most popular techniques for the latter is Synthetic Minority Oversampling Technique (SMOTE) (Chawla, Bowyer, Hall, & Kegelmeyer, 2002).

SMOTE creates synthetic minority samples among the real ones. It chooses a positive sample, it finds its k nearest neighbors - where k is a hyperparameter – and it selects one or more of them. Then it connects with a line the first sample chosen with its selected neighbor(s) and it places in the middle of the line a new negative sample.

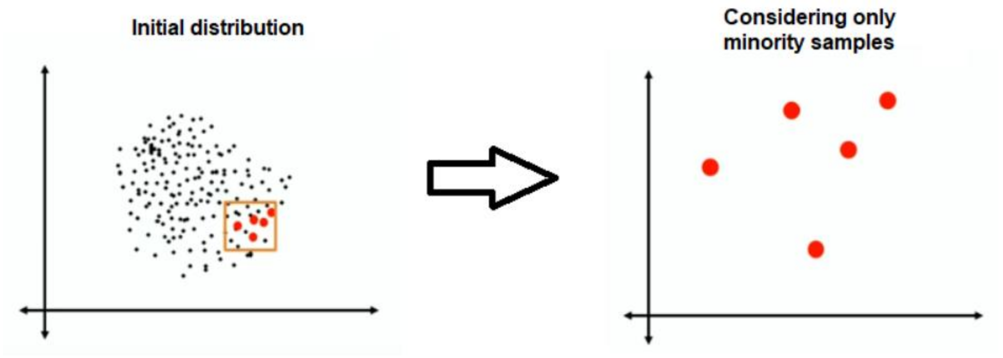


Figure 17: Illustration of SMOTE; the 2 nearest neighbors of A are found and the synthetic samples are placed in between the lines drawn (https://iq.opengenus.org/smote-for-imbalanced-dataset/)

SMOTE comes with drawbacks of course; it is susceptible to creating noise and it might make the decision surface more complicated for some classifiers. We used SMOTE from *imblearn* 's *over_sampling* module and we experimented with the number of neighbors as well as with the final ratio (minority/majority) after the oversampling, eventually setting $k = 5$ and $ratio = 0.3$.

3.11. Evaluation metrics

Part of the process when building a model is its performance evaluation. It is crucial when choosing the best model among various candidates. Numerous metrics have been introduced, since in various problems and applications what needs to be measured differs. Ferri et al. (Ferri, Hernández-Orallo & Modroiu, 2009) have proposed a taxonomy that classifies measures into three families:

- Threshold metrics: these evaluate the ability of a classifier to minimize the number of wrong predictions and are informative with respect to the qualitative understanding of the error. Such measures are accuracy, F-score and Kappa statistics.

- Probabilistic metrics: these metrics are interested in evaluating not only whether the correct class was predicted, but also whether the wrong class is predicted with high or low probability. Such measures are Brier Score, Log Loss (cross-entropy), some probability-rate and some calibration metrics.
- Ranking metrics: these emphasize on the ability of the classifier to distinguish between the classes. Area Under the Curve (AUC) is such a metric.

The existence of imbalance between the classes in a dataset - as is the case in Dataset B - affects the choice of the metric. Branco et. al. (Branco, Torgo & Ribeiro, 2015) stress that choosing a common metric in imbalanced datasets can result in sub-optimal and misleading classification models as these metrics are insensitive to the existence of skewness in domains. Moreover, they highlight that metrics play a role when the model learns

“Adequate metrics should not only provide means to compare the models according to the user preferences, but can also be used to drive the learning of these models.” (Branco et al., 2015, p.4)

An example that can illustrate the need of a correct metric choice is the case of building a model to predict whether a tumor is benign or malignant over a dataset that consists of 99,000 benign and 1,000 malicious samples. Evaluating the model with Accuracy could possibly result in high scores, as the model would learn very well the majority class while remain ignorant in the minority class. However, this would be a useless, if not dangerous, model, as all tumors would be classified as benign.

As far as Dataset B is concerned, the Credit Scoring team of Channel VAS is mainly interested in distinguishing between the two classes. However, it is also useful to be able to predict the probability with which the class was predicted. This is because Credit Scoring aims at determining the Temporary Credit Limit (TCL) that defines exactly the limit of the amount someone can borrow. Loan providers are not only interested in deciding whether to provide a loan or not, but also in determining the exact loan amount someone can borrow. If TCL is too low then loan providers run the risk of falling subject to many defaults. On the other hand, if TCL is too high, few loans are provided and the ability to increase profits is confined. This is the reason why probabilistic and ranking metrics are preferred over threshold metrics.

Specifically, we have chosen to use the Receiving Operator Characteristic – Area Under the Curve (roc-auc score) to evaluate the performance of the classifiers when using Transfer Learning methods, and Log Loss and Brier score also when building a base classifier for dataset B. Each of these are described below.

3.11.1. Area Under the Curve

Area Under the Curve (AUC) of a binary classifier reflects the probability that a randomly chosen positive sample will be ranked higher than a randomly chosen negative sample. To understand this on a high level it is helpful to introduce the notions of True Positive Rate (TPR) and False Positive Rate (FPS).

TPR is defined as the fraction of the instances that are correctly classified as positive out of all the actually positive ones. Intuitively, it expresses the ability of the classifier to capture the positive instances. A 100% TPR means that the classifier does not miss a single positive instance. It can be thought of as the number of instances that are positive and are correctly classified as such

$$TPP = \frac{TP}{TP + FN}$$

FPS is defined as the fraction of the instances that have been wrongly classified as positive over the total number of negative instances. Intuitively, it expresses how prone the classifier is to missing negative instances. A 100% FPR means that the classifier is unable to capture negative instances. It can be thought of as the number of instances that are negative but were classified as positive

$$FPN = \frac{FP}{FP+TN}$$

Receiving Operator Characteristic (ROC) curve plots the TPR versus the FPR at different classification thresholds. Lower thresholds benefit positive classification and consequently TP and FP, while higher thresholds benefit negative classification and this TN and FN. Therefore, the curve is a measurement of how well the classifier distinguishes the two classes. In the extreme scenario where a classifier classifies completely at random the curve becomes a straight line and the respective area under the curve, 0.5. On the other extreme and ideal scenario where a classifier classifies all instances correctly the curve takes the shape of a Gamma, and the respective area under the curve is 1. Figure 18 illustrates the two extreme cases as well as another case when the area under the curve is at 0.65 (its average performance) in comparison to a random classification.

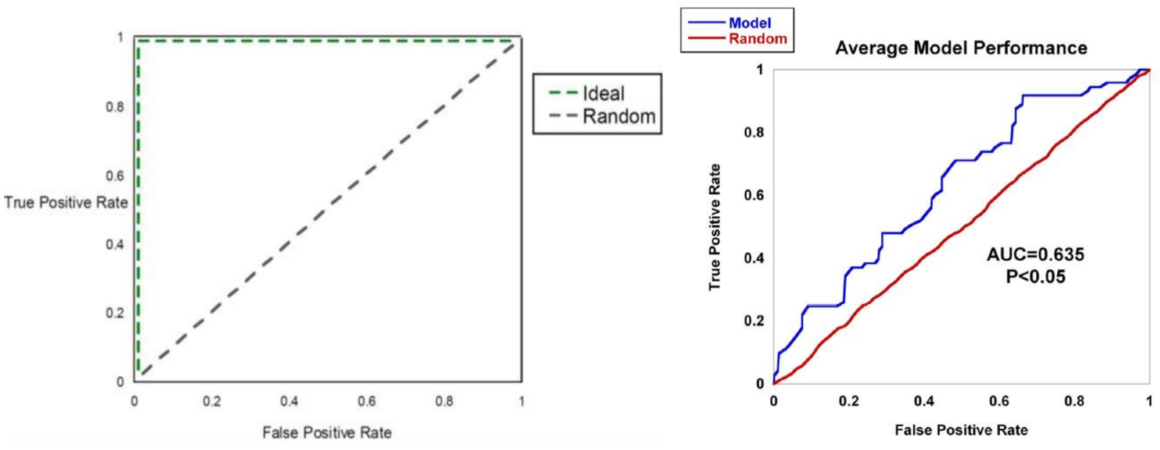


Figure 18: The Receiving Operator Characteristic. Left: the extreme cases, right: a ROC whose AUC is 0.635

3.11.2. Brier score and log loss

Brier score is also known as Mean Squared Error and for binary classification is defined as

$$Brier = \frac{\sum_{i=1}^{m}(f(i,j)-p(i,j))^2}{m} \quad (38)$$

where $f(i,j)$ represents the actual probability of the sample i to be in class j, $p(i,j)$ represents the estimated probability of sample i to be in class j, and m is the total number of samples (Ferri et al., 2009). Since we are dealing here with binary classification, (1) can be written as

$$Brier = \frac{\sum_{i=1}^{m}(y_i-\hat{y}_i)^2}{m} \quad (39)$$

Where y_i is the class label of sample i and $\hat{y}_i$ the estimated probability that the sample is positive.

Log Los also measures how good probability estimates are, however, it may be misleading in imbalanced datasets. Log Loss, otherwise known as cross-entropy, is defined as

$$Log\ Loss = \frac{\sum_{i=1}^{m}(y_i + (1-y_i)log\ (1-y_i))^2}{m} \quad (40)$$

In a highly imbalanced dataset few elements of the sum are non-zero which results to low log loss score.

We used roc-auc metric to evaluate the performance of the classifiers before and after the Domain Adaptation techniques, in both datasets, and we used Brier score and Log Loss when we were building base classifiers for Dataset B.

4. Datasets and preparation of data

For our experiments we used two different datasets, one is the Lending Club Dataset, which is the same as in papers (Huang & Chen, 2018), (Suryanto et al, 2019) and a dataset obtained from Channel VAS, a FinTech company based in Greece, which we will call Dataset B. In the following two sections we will describe the two datasets.

4.1. Lending Club Dataset data

The Lending Club Dataset which is publicly available (https://www.kaggle.com/adarshsng/lending-club-loan-data-csv?select=LCDataDictionary.xlsx) in a csv form. There are various versions depending on the period that the data was collected. The one we chose contains loan data for loans issued through Lending Club institution for the period 2007-2015 that involve among others credit scores, number of finance inquiries, address including zip codes and state and more. One of the columns regards the loan status, which, after the preprocessing we did, contained the values:

- Fully Paid
- Late (31-120 days)
- Charged Off
- Default

"Fully Paid" was the value for non-defaulters, and all the rest for defaulters. Initially, the dataset consisted of 2,260,668 and 145 columns. A full description of the columns can be found in the Appendix.

We followed a similar preprocessing as the one followed by Huang & Chen that is described in their paper (Huang & Chen, 2018) who also used Lending Club Dataset among other for their experiments in their domain adaptation approach in a credit risk problem.

Firstly, we cleaned the dataset from missing values by removing columns that had more than 50% of their missing values removed. Moreover, the rows that had at least one missing value, were dropped as well. After this cleaning, we ended up with 113,534 rows and 81 columns. In their paper, Huan & Chen (Huang & Chen, 2018) removed features with very high collinearity, as it is commonly known that collinearity when developing a model may obscure the contribution of each variable in prediction. We attempted to remove collinear features by applying the Variance

Inflation Factor (VIF) criterion to see how it affects the learning with and without domain adaptation. When VIF was applied (with threshold = 5) 40 variables were filtered out of the 81. Eventually we used all variables as feature selection with VIF negatively affected the results.

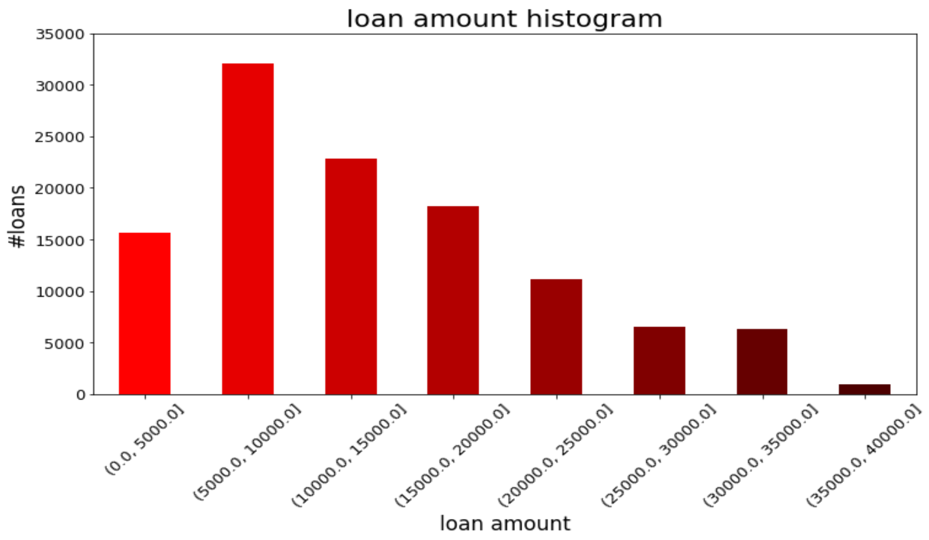


Figure 19: distribution of the variable loan_amnt

The column *loan_amnt* that describes the amount of the loan that has been requested by the borrower, was used to split the dataset into Source and Target. We grouped the *loan_amnt* into groups of range 5,000 and the distribution can be seen in the graph above. The exact splits that were used are described in a subsequent chapter where the experiments and the results are presented in detail.

The default rate in the whole dataset is 26.9% but it differs for every category as it can be seen on the table below. It gradually decreases until the last category, the one with the highest *loan_amnt*, where it increases again.

Interval (loan_amnt in $)	**Defaulters**	**Non-defaulters**
(0.0, 5000.0]	21.19%	78.81%
(5000.0, 10000.0]	24.67%	76.33%
(10,000.0, 15,000.0]	27.53%	72.47%
(15000.0, 20000.0]	30.16%	69.84%
(20,000.0, 25,000.0]	30.72%	69.28%
(25,000.0, 30,000.0]	31.33%	68.67%
(30,000.0, 35,000.0]	34.81%	65.19%
(35,000.0, 40,000.0]	23.47%	76.52%

Table 3: Default rates per bin constructed by loan_amnt variable

There is some degree of imbalance, with the minority class holding approximately 25%, but this was expected, as a lending company would not be profitable and well-operating if there was a significant number of defaults on the loans.

4.2. Dataset B data

This dataset was provided by the company Channel VAS, in which the person writing this thesis worked for one year and four months. The company collaborates with mobile telephone providers and provides to the end users airtime and data loans when they do not have money in their account. Channel VAS operates in more than 25 countries and works along with more than 35 operators. A *project* refers to a specific provider together with the country of operation. The company receives from its collaborators transactional, behavioral data which it then processes and uses to profile the users and assign to them a Temporary Credit Limit (TCL). Thus, every user who requests a loan can receive an amount up to their TCL.

We got two datasets from two different countries, Nigeria and Ghana, one used as the Source and the other as the Target. These countries were chosen because they are located in the same geographical area and it has been noted by the people working in the Credit Risk department that they present behavioral similarities, and this should offer the ground for the existence of the domain similarity that is required for Transfer Learning and Domain Adaptation.

The variables of the dataset describe the expenditures of the users on airtime recharges, the loans that they have taken and the repayments of the loans, as well as some other information like how long they have been to the operator. A user is characterized as a defaulter when they have not repaid their loan after six months. The users are split into segments resulting from a segmentation that is based on the number and amount of recharge expenditures made. The segments are usually four, but we expanded them into eight. The overall default rate for Source was 3.79% and for Target 2.23%. For each segment the corresponding rates are shown in the table below. The existence of high imbalance between the two classes is evident for both datasets. This issue was attempted to be addressed via methods that will be described in a subsequent chapter.

	segment 1	segment 2	segment 3	segment 4	segment 5	segment 6	segment 7	segment 8
Source	4.53%	5.27%	4.48%	4.15%	4.56%	3.99%	2.77%	1.16%
Target	2.75%	2.87%	3.21%	2.77%	1.43%	1.72%	1.00%	1.97%

Table 4: default rates of Source and Target in Dataset B

There was a variable that denoted the telephone number of the user, which is used in Channel VAS as a key to provide a TCL. Since this information is sensitive and in order to comply with the General Data Protection Regulation (GDPR) we encrypted these numbers.

The two datasets did not have any missing values or any sort of inconsistency as all data is subject to validation when received from the operators. Source's dataset originally consisted of 314 features and Target's of 277. Out of all these features some were common. So, since we followed approaches developed for homogeneous cases, we selected a subset of 236 common features, plus the target variable. All of them were numerical. There were 850,000 rows in Source's dataset and 750,000 in Target's. The data was retrieved with queries using matlab and SQL from the company's cluster.

5. Experiments and results

In this chapter we are presenting the experiments that were ran on the datasets and their results. The results from each dataset are presented separately.

5.1. Lending Club Dataset

The two domain adaptation methods applied were KLIEP and trAdaBoost and the machine learning algorithms used were Gradient Boosting, Random Forest and Logistic Regression. The Lending Club dataset consisted - after the cleaning – of 113,534 samples. Binning the loaned amount in bins of range=5,000 we can see that approximately 42% of the borrowers loaned up to $10,000 and approximately 62% loaned up to $15,000. Only 6.3% loaned more than $30,000.

Interval (loan_amnt in $)	**percentage**
(0.0, 5000.0]	13.74%
(5000.0, 10000.0]	28.2%
(10,000.0, 15,000.0]	20.14%
(15000.0, 20000.0]	16.01%
(20,000.0, 25,000.0]	9.81%
(25,000.0, 30,000.0]	5.78%
(30,000.0, 35,000.0]	5.54%
(35,000.0, 40,000.0]	0.78%

Table 5: Distribution of loan_amnt variable

We ran the experiments with three different splits for the Source and the Target domain:

a. Source: loans < $10,000, Target loans > $10,000
b. Source: loans < $15,000, Target loans > $25,000
c. Source: loans < $10,000, Target loans > $30,000

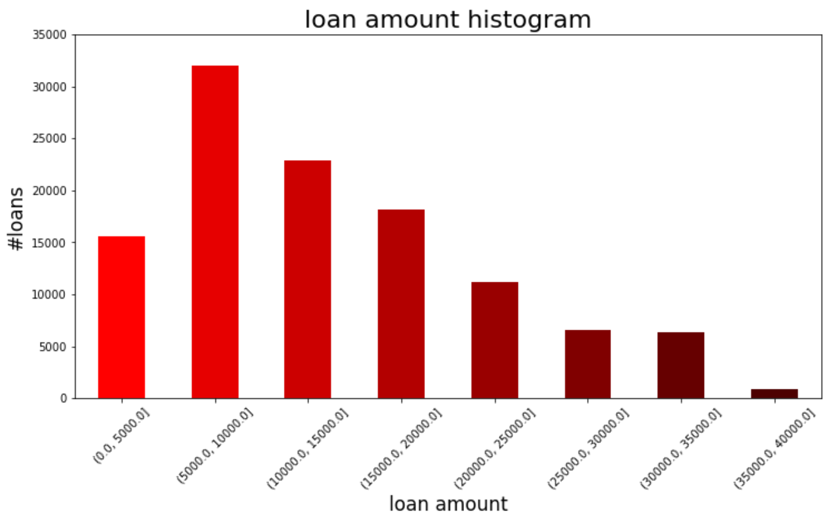


Figure 20: Histogram of the distribution of loan_amnt variable

This is because we wanted to get some insight on how the size of the domains and their similarity influence transfer learning. To explain what is meant by this, let us first notice that every split allocates different amounts of data to Source and Target. The amount of data allocated to Source is determinant because it is the Source where the training takes place. The amount and the variance of the Target's (where testing takes place) samples is also very important, in the sense that if the test samples are close to each other on the feature space (less variance), then it is likely easier for a classifier to learn well and less likely to be confused. This case has been studied by Jiménez-Guarneros and Pilar (Jiménez-Guarneros & Pilar, 2021) as the case of subclusters in the Target domain's classes, which has been shown to induce negative transfer. The exemplary pictures below illustrate this.

Therefore, if Target's samples include those with *loan_amnt* > 10,000 - the maximum being 40,000 - the samples' features could have more variance and greater range of values, thus be more expanded in the feature space, while if the Target's samples include those with *loan_amnt* > 30,000, the features are likely to have less variance and higher proximity in the feature space. Of course, this is not certain, but taking some insight through graphs, it is reasonably probable.

Moreover, the amount of loan requested by an applicant usually comes in accordance with various indexes, like the annual income for example. The higher loan amounts are more likely to be requested by applicants with higher income. In other words, there is correlation between the variable *loan_amnt* and other variables. Consequently, there are many features that follow a different distribution when samples are restricted with respect to *loan_amnt* beyond or above some threshold, and this respectively affects the similarity between the Source and the Target domains. The following graphs show two characteristic features that are distributed evidently differently when the threshold of *loan_amnt* changes.

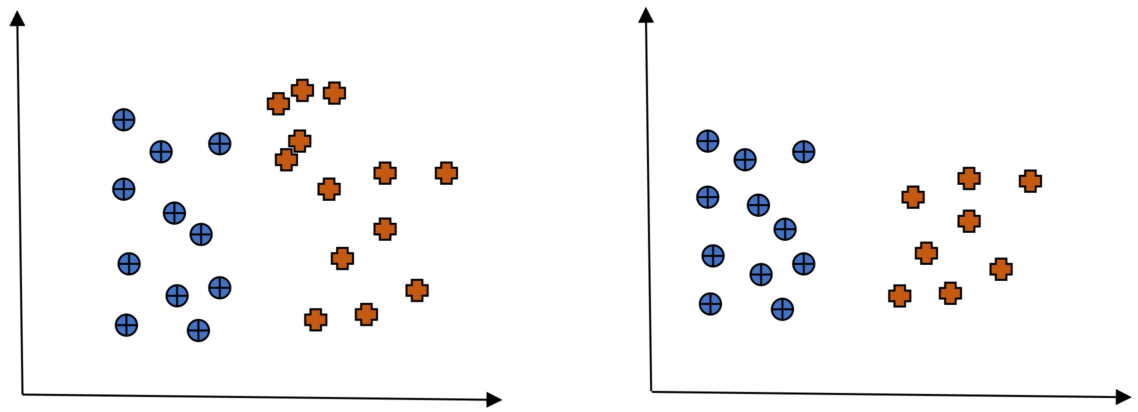
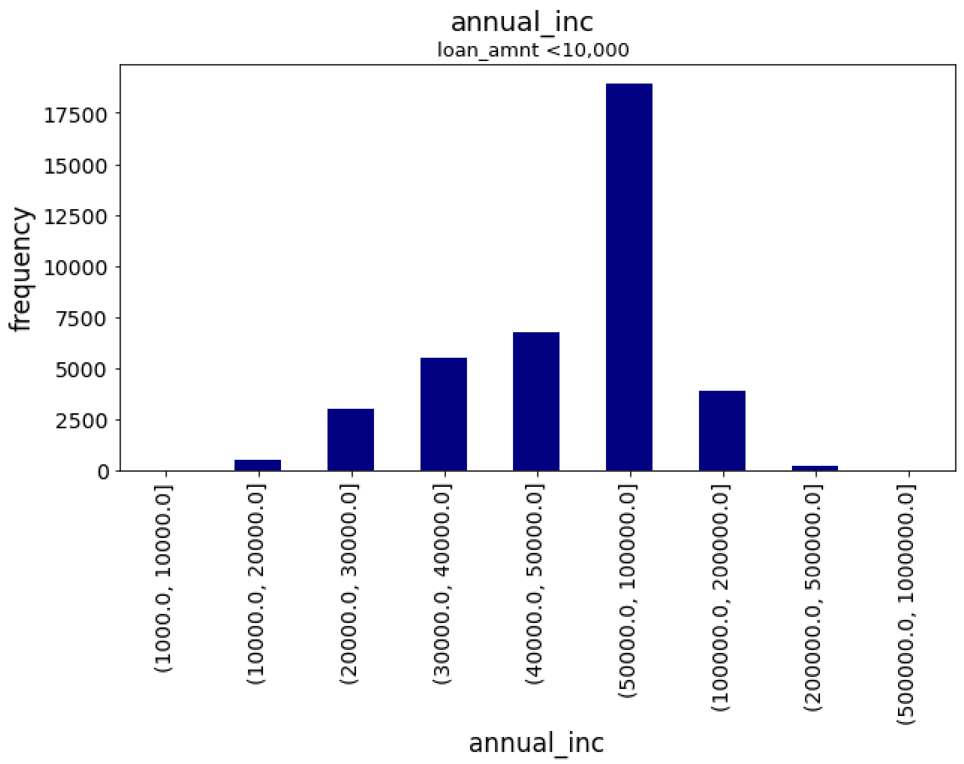


Figure 21: on a two-dimensional space, the positive test samples on the right form a single cluster and it is easier for a classifier to separate them from the negatives. On the left, we see a case where the positive samples are located in two areas on the space, and this could make some classifiers more error prone.



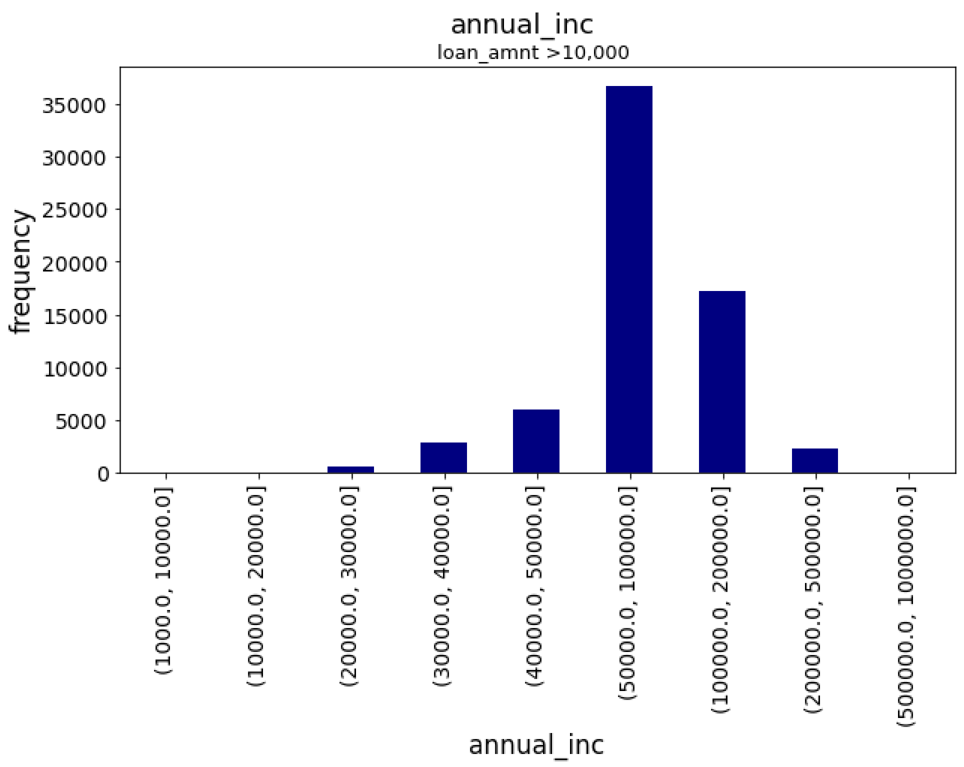
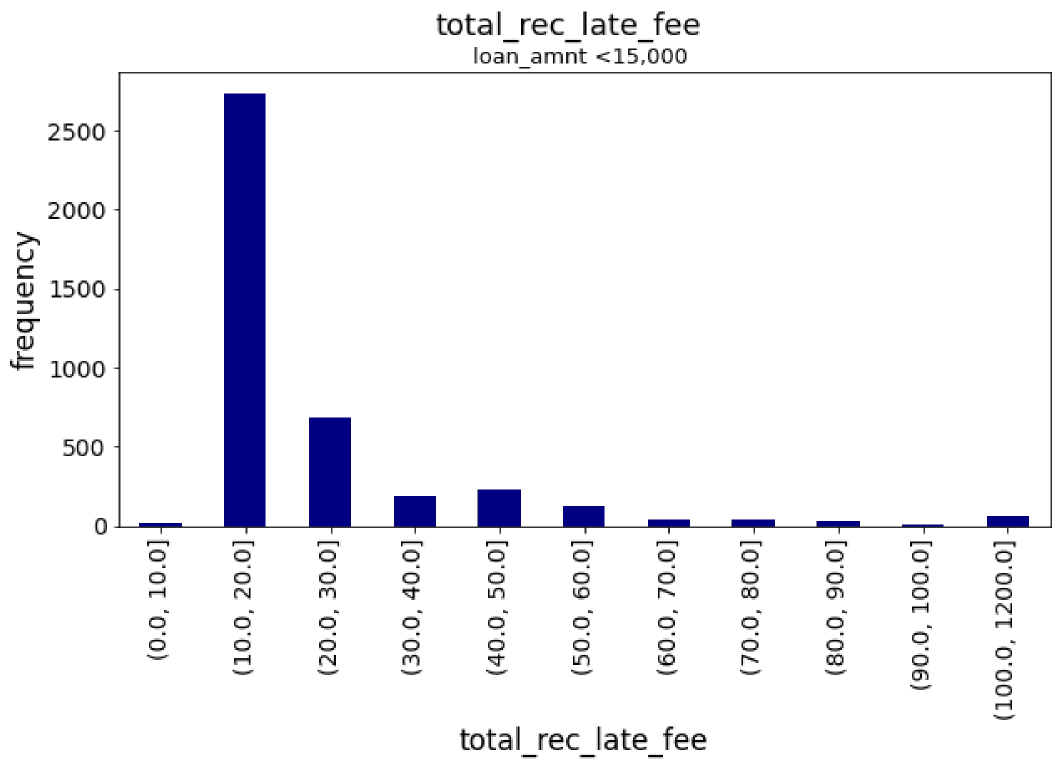


Figure 22: Top: annual income for applicants of amounts<10,000; the distribution is left skewed. Bottom: annual income for applicants of amount>10,000; the distribution is right skewed



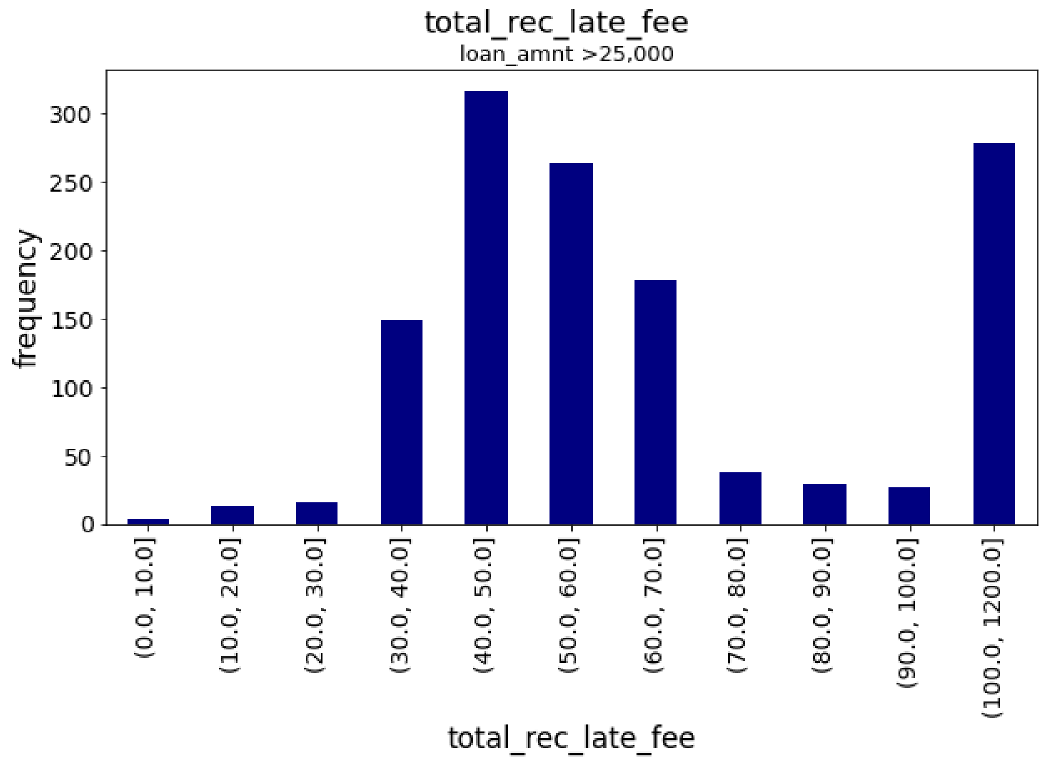


Figure 23: Top: late fees received to date: intensely right skewed with most values on range 10-20. Bottom: late fees received to date: non symmetric bimodal

Such geometrical transformations in the Target's distribution has been studied as well by Jiménez-Guarneros and Pilar (Jiménez-Guarneros & Pilar, 2021), who observed that some transformations like skewness for example, negatively affected the learning. Considering the above, in split (a) there are several samples that are more related, those closer to *loan_amnt*=10,000, hence some degree of similarity between the domains, and there is more variance in the Target domain. In split (b) there are fewer similar samples and less variance in Target as well. Split (c) could be considered the most extreme case where there is little similarity between the two domains and little variance on Target. To get some insight into the similarity of the domains, we applied PCA with 2 principal components to do scatter plots and see how the clusters formed resemble or differ.

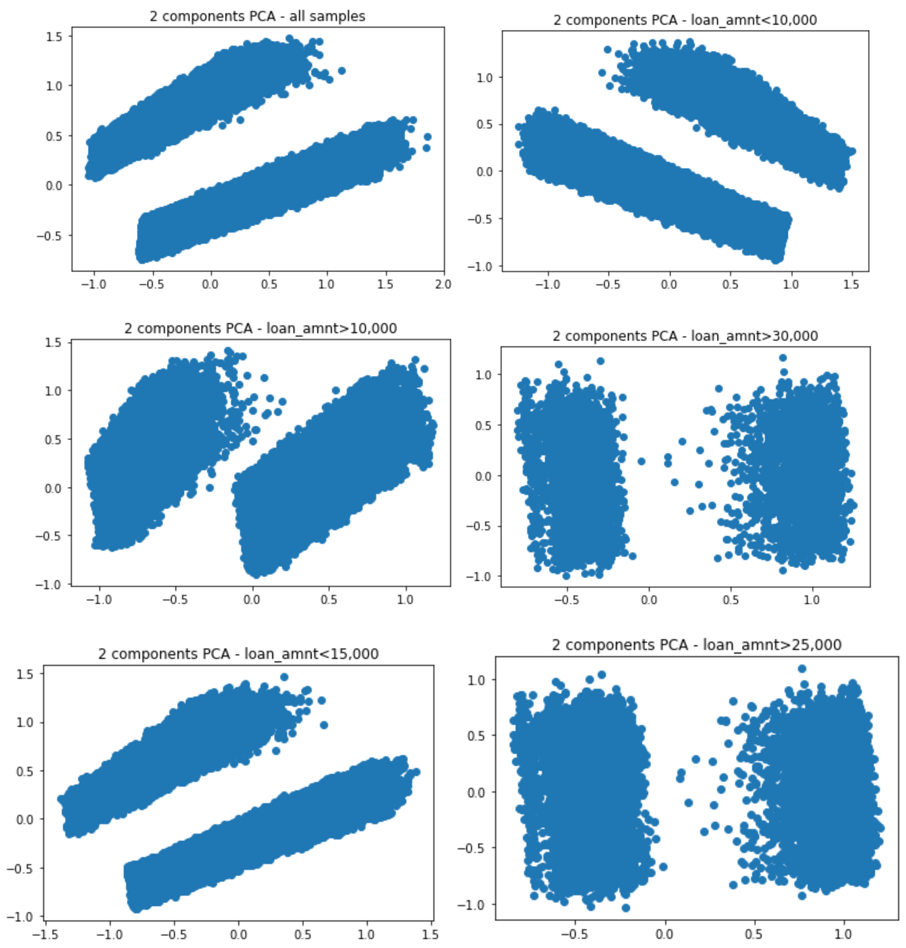


Figure 24: Top left depicts the two clusters formed when using the whole dataset. Top middle picture can be viewed in comparison with top right and bottom left (for splits (a) and (c) respectively) to observe that the top right seems to have been reflected along the vertical axis and widened, and the bottom left seems to have been rotated by 45° and widened. Lastly, the bottom right image looks also rotated by - 45° and widened.

5.1.1. Lending Club Dataset - KLIEP

We applied the KLIEP method to Gradient Boosting, Random Forest and Logistic Regression, with and without feature selection with VIF criterion. We tested the method in various sample sizes, namely, 25,000, 50,000, 75,000 and 100,000 that were drawn randomly. In general, the feature selection using VIF impeded the performance of the classifiers with and without transfer learning. This can have happened because the features that were removed might have been the features over which the two domains were more similar, as well as more predictive. Thus, removing them may trigger the root cause of negative transfer, the divergence between the two domains (Wang et al., 2019). The table below shows a characteristic example on how the roc-auc scores were with/without feature selection and KLIEP, in this case with Gradient Boosting classifier.

Gradient Boosting - Lending Club Dataset - without VIF					
sample size	lower-upper bound	roc-auc w/o KLIEP w/o f.s.	roc-auc with KLIEP w/o f.s.	roc-auc w/o KLIEP with f.s.	roc-auc with KLIEP with f.s.
25000	10000-10000	0.836	0.923	0.789	0.793
50000	10000-10000	0.832	0.922	0.763	0.763
75000	10000-10000	0.837	0.922	0.763	0.756
100000	10000-10000	0.836	0.925	0.763	0.763

Table 6: roc-auc scores with Gradient boosting for various settings for loan_amnt thresholds 10,000 and 10,000, per sample size

Gradient Boosting improved its performance on splits (a) and (b) on all sample sizes by approximately 9%, while decreased it in split (c) on all sample sizes by approximately 8% except for sample size=50,000. This could hint that this classifier had more difficulty in predicting Target's class when there was not so much similarity between the domains. The sample size did not play an important role on the predictions. This may be explained by the fact that, as explained in previous chapter, KLIEP targets to solve the following convex optimization problem:

$$max_{\{a_l\}_{l=1}^{b}} [\sum_{l=1}^{n_{te}} log\, (\sum_{l=1}^{b} a_l \varphi_l (x_{i_{te}}))]$$

$$subject\ to\ \frac{1}{n_{tr}} \sum_{i=1}^{n_{tr}} \sum_{l=1}^{b} a_l \varphi_l (x_{i_{tr}}) = 1 \quad and\ \ a_l > 0 \quad for\ all\ l = 1,2, \dots, b$$

Where a_ι are the parameters to be learned and φ_i kernel functions, where the number of training samples is fixed. Thus, an adequate choice for the number of kernels should provide a global optima solution to the above problem regardless of the number of training samples.

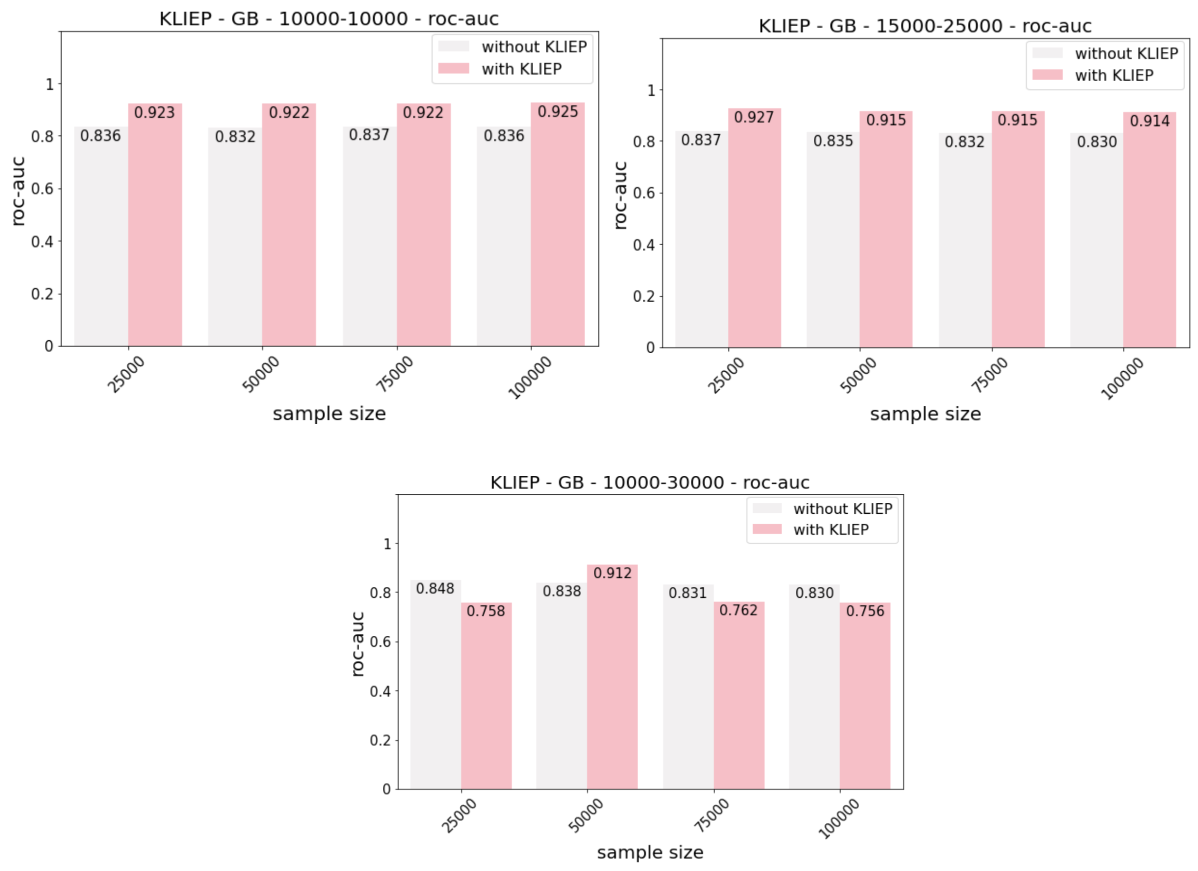
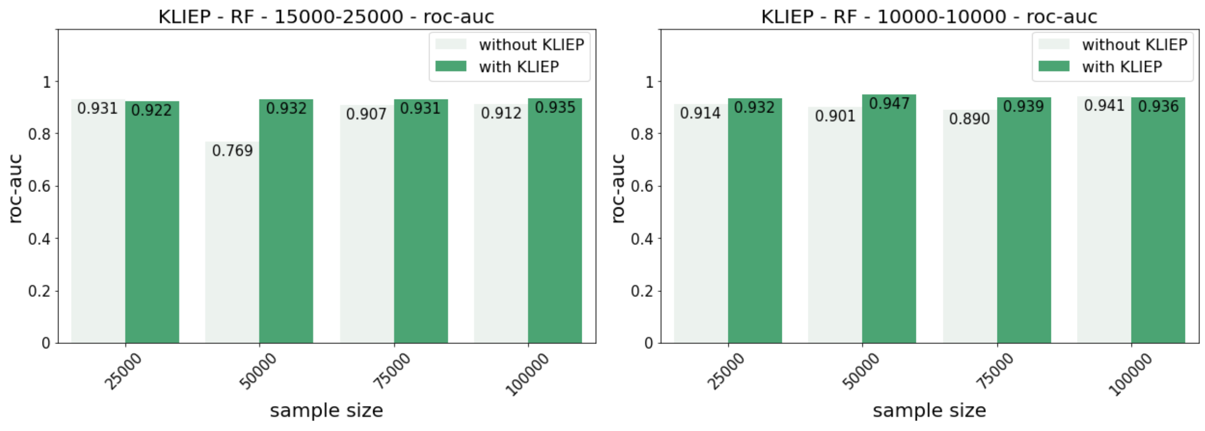


Figure 25: Gradient Boosting roc-auc results with and without KLIEP on different Source-Target splits and sample sizes with all features.

Random Forest improved in most cases its roc-auc score but certainly to a smaller degree than Gradient Boosting. There were two cases when negative and neutral (no significant change) transfer occurred. Sample size also did not seem to play an important role here. Only in one case there was an astonishing increase of 16% in roc-auc score, that is when sample size was 50,000 in split (b). Given that this happened only once we do not have some hint for its cause; we could assume that it happened due to the particular sample drawn, as it was randomly selected. Re-running the experiments multiple times or using cross-validation could be insightful for this.



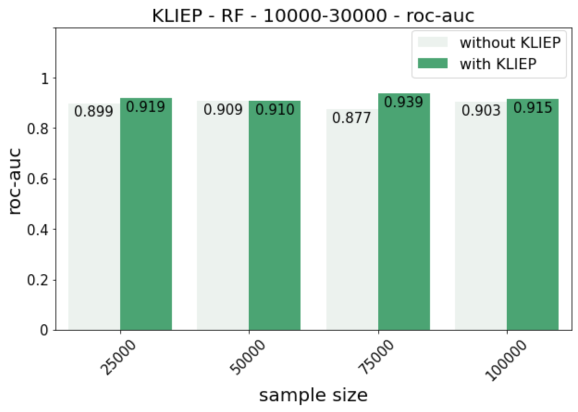


Figure 26: Random Forest roc-auc results with and without KLIEP on different Source-Target splits and sample sizes and without feature selection.

Logistic regression did not seem to be affected at all by KLIEP; the roc-auc scores were already, without the use of KLIEP, very high (approximately 99%), therefore KLIEP had minor impact in all cases (less than 0.4%).

5.1.2. Lending Club Dataset - TrAdaBoost

With trAdaBoost, we experimented with various test sizes, namely, 0.7, 0.8, 0.9, 0.95, 0.995. The smaller the test size the more samples from the Target domain are available in the training, something that may boost a classifier’s learning, but at the same time it contains the risk of overfitting.

Gradient Boosting Classifier in split (a) had a slight increase in roc-auc in test sizes 0.7 and 0.8, significant negative transfer in test sizes 0.9 and 0.95 and a good increase in test size 0.995

(approximately 8%), that is, only when few samples from the Target were allocated to training as labeled there was a pretty high boost. Split (a) allows for more similarity between the Source and the Target domain samples. An adequate number of labeled Target samples is required to prevent both overfitting and underfitting. For example, providing only a few labeled Target samples (0.5%) might have been an adequate amount of data to help the classifier adjust the instances' weights without confusing it, as for example it could happen in figure 8. The decreased performance (negative transfer) in test sizes 5% and 10% looks strange at first. However, for negative transfer to be avoided it is essential that appropriate size of Target labeled data be used. As they write,

"When labeled target data is available … a better target-only baseline can be obtained using semi-supervised learning methods and so negative transfer is relatively more likely to occur. At the other end of the spectrum, if there is an abundance of labeled target data, then transferring from an even slightly different source domain could hurt the generalization. Thus, this shows that negative transfer is relative." (Wang et al., 2019, p.3)

This could be an explanation of the negative transfer induced with test sizes 5% and 10%. Another explanation could be that the algorithm did not run in sufficiently many epochs (iterations) in order to adjust the weights appropriately. There could be Source instances that should further decrease their weights in order that they have a minor role in the classifier's learning, or, respectively, some Target's instances should have had their weights further increased so that the classifier puts more emphasis on them. Let us recall that the Source's and Target's weights are updated in every iteration as following:

$$w_i^{t+1} = \{w_i^t \beta^{|h_t(x_i)-c(x_i)|}, \quad 1 \le i \le n \; w_i^t \beta_t^{-|h_t(x_i)-c(x_i)|}, \quad n+1 \le i \le n+m\} \qquad (41)$$

where betas belong to (-0,1], $x_i, i = 1, \dots, n$ are the Source's instances and $x_i, i = n+1, \dots, m$ the Target's instances.

Test sizes 20% and 30% means that there were abundant labeled samples from Target that were used in training, and these samples themselves could be sufficiently many to help the classifier learn the Target's classes.

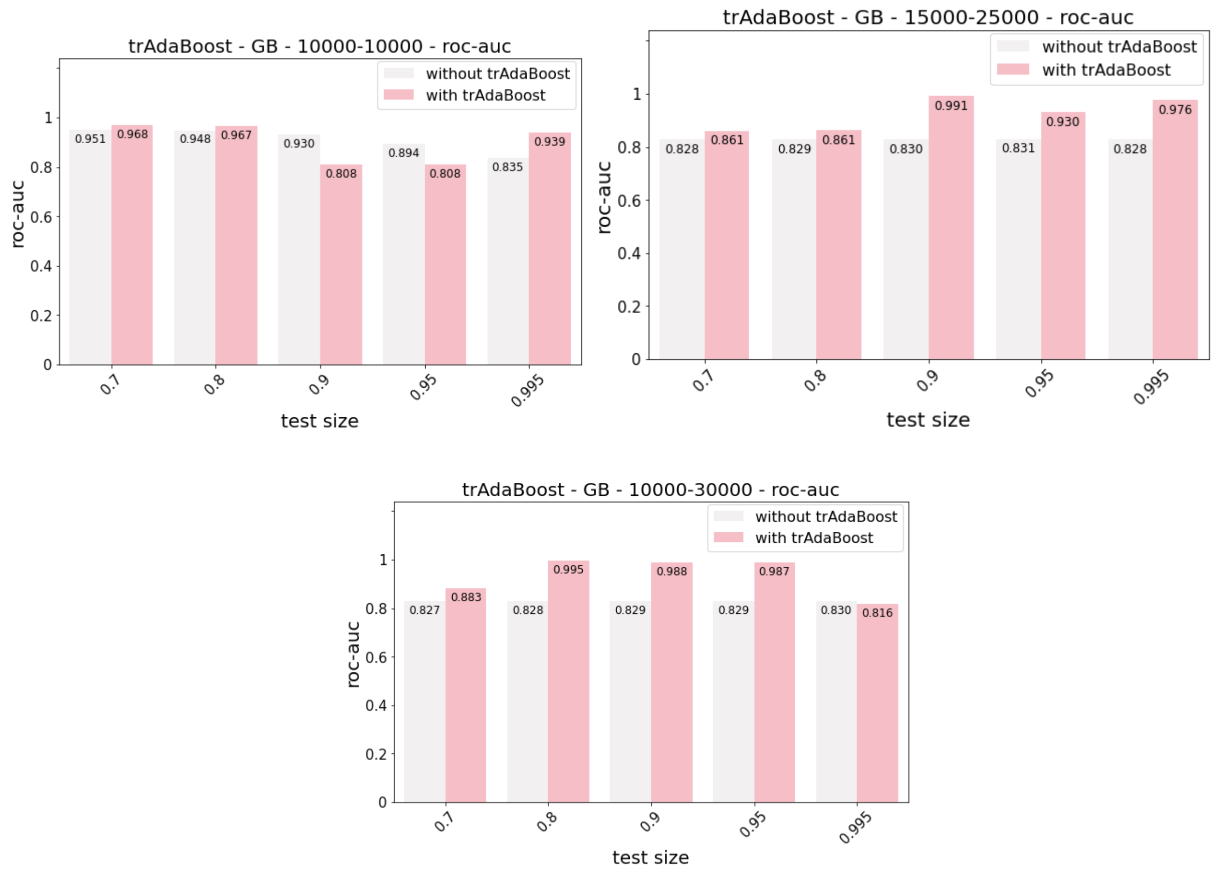


Figure 27: Gradient Boosting roc-auc results with and without trAdaBoost on different Source-Target splits and sample sizes and without feature selection.

In split (b), only positive transfer occurred, which ranged from 4% up to the impressive 16%, which led to almost a perfect classifier with a 99.1% roc-auc score when test size was 90%. The distance between the domains here is bigger and there are fewer samples allocated to Target.

In split (c), slight negative transfer occurred only when test size was 99.5%, i.e., when merely 0.5% of the Target's samples participated in training as labeled. The similarity between Source and Target was even smaller here as was the number of samples of the Target. Just 0.5% of labeled Target samples might have been too little to help the classifier learn the similarities between the two domains.

The table below summarizes when there was negative (-), positive (+) and highly positive (++) transfer.

Split / test size	0.7	0.8	0.9	0.95	0.995
10,000-10,000	+	+	-	-	++
15,000-25,000	+	+	++	++	++
10,000-30,000	+	++	++	++	-

Table 7: Impact of trAdaBoost (positive or negative) with Gradient Boosting Classifier with respect to test size

Random Forest presented a slightly negative or neutral (no significant change) transfer in split (a) for all test sizes. With split (b), there was positive transfer of 2%-6% in test sizes 0.7, 0.8 and 0.9 but negative transfer of 2%-9% in test sizes 0.95% and 0.995%. Probably there were not sufficiently many labeled Target samples to boost the training. In split (c) negative transfer was spotted only when test size was 0.95% and in the other cases positive or neutral transfer of up to 9%.

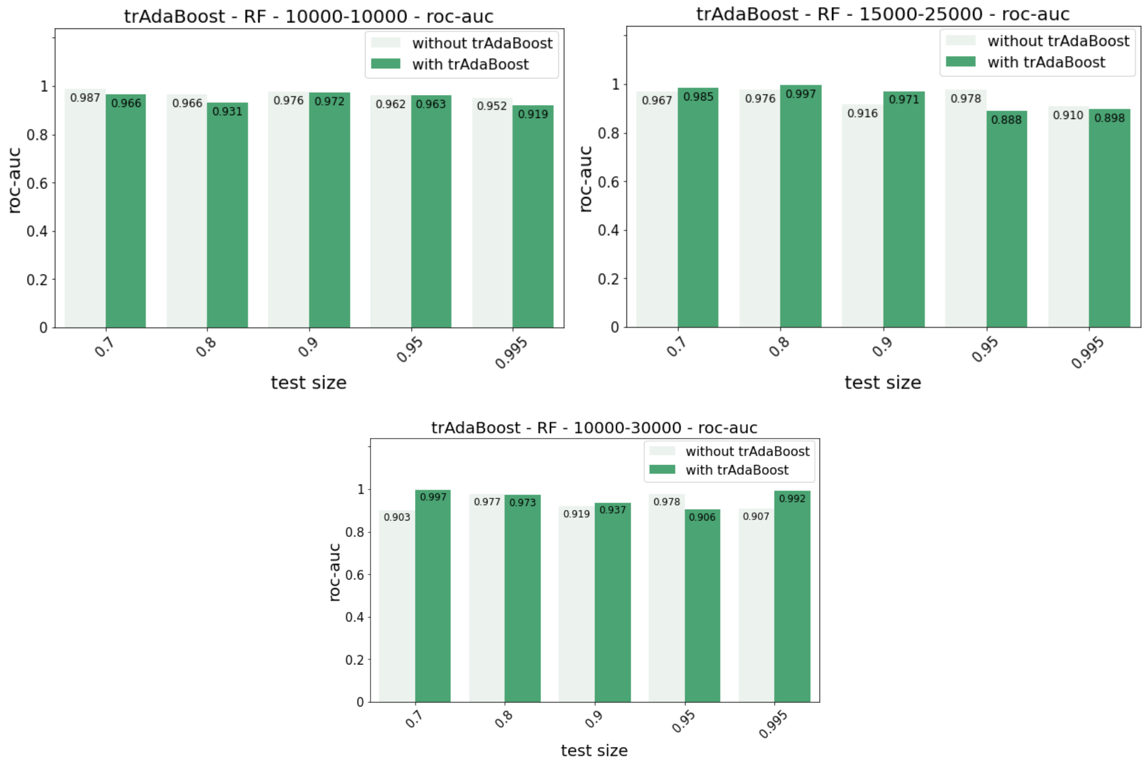


Figure 28: Random Forest roc-auc results with and without trAdaBoost on different Source-Target splits and sample sizes and without feature selection.

It is evidently more complicated to understand the behavior of Random Forest. It seems to do better in smaller test sizes, e.g., when more labeled Target data are provided, however, more experiments should be conducted in order that we make some meaningful observations. The table below summarizes the situation (+ stands for positive transfer, - for negative and N for neutral)

Split / test size	0.7	0.8	0.9	0.95	0.995
10000-10000	-	-	N	N	-
15000-25000	+	+	+	-	-
10000-30000	+	N	+	-	+

Table 8: Impact of trAdaBoost (positive or negative) with Random Forest Classifier with respect to test size

Logistic Regression did not show any noticeable changes with trAdaBoost either; as also mentioned before its performance was already approximately 99% in auc-roc score, so the minor changes induced by using trAdaBoost are not of interest or informative.

5.2. Dataset B

In this section we will present the results we obtained applying KLIEP and TrAdaBoost on Dataset B, using Logistic Regression. Before this though, we will show the results we obtained when we were trying to build a good base classifier to later use with transfer learning.

Without any process the roc-auc score obtained with Logistic Regression on Source was 0.728. We then applied PCA, feature selection with Genetic Algorithm and oversampling with SMOTE, under-sampling with sklearn's RandomUnderSampler, as well as combinations of these. As one can see from the table below, nothing managed to contribute to a significant increase; roc-auc increased from 0.3% up to 1%. PCA was applied to maintain a 99% explained variance. We experimented with the sampling strategy (ratio between classes after oversampling) and the number of neighbors in SMOTE, eventually choosing the values 0.3 and 5 respectively. Let's also note that SMOTE and the combinations increased the log-loss and the Brier scores.

We did the same experiments with Random Forest and Gradient Boosting to see if this difficulty in improving the learning was due to the specific classifier, but the results we obtained were not better; in some cases actually, some of the things we tried deteriorated the scores.

Source	**no process**	**PCA**	**f.s.**	**Under sample**	**SMOTE**	**f.s. - PCA**	**f.s. – SMOTE**	**PCA – SMOTE**	**f.s-PCA-SMOTE**
roc-auc	0.728	0.738	0.736	0.741	0.736	0.736	0.735	0.737	0.731
Log loss	0.150	0.150	0.148	0.147	0.150	0.148	0.282	0.150	0.282
Brier	0.035	0.035	0.035	0.035	0.076	0.035	0.076	0.076	0.078

Table 9: evaluation metrics on Source with various training settings (PCA, feature selection with GA, undersampling, SMOTE and combinations of these)

We did the same with Target to see what can be a maximum (from our trials) score that can be reached using the whole dataset. The results are shown in the table below and at a first glance one can see that no significant changes occurred here either. The roc-auc scores varied from 0.76 up to 0.77.

Target	**no process**	**PCA**	**f.s.**	**Under sample**	**SMOTE**	**f.s. - PCA**	**f.s. – SMOTE**	**PCA – SMOTE**	**f.s-PCA-SMOTE**
roc-auc	0.768	0.765	0.764	0.741	0.770	0.761	0.766	0.765	0.760
log loss	0.096	0.097	0.097	0.147	0.258	0.097	0.260	0.260	0.260
Brier	0.021	0.021	0.021	0.035	0.070	0.021	0.070	0.070	0.071

Table 10: evaluation metrics on Target with various training settings (PCA, feature selection with GA, undersampling, SMOTE and combinations of these)

	trAdaBoost	Feature Sel.
Setting 1	X	X
Setting 2	X	V
Setting 3	V	X
Setting 4	V	V

Table 11: the four settings used with TrAdaBoost

Since there was not any substantial improvement on the classifier's performance, we chose to use only feature selection, something that would contribute to decreasing the time required for running the algorithms on our subsequent experiments. Therefore, the experiments were performed under four different settings as shown in table 11.

	KLIEP	Feature Sel.
Setting 1	X	X
Setting 2	X	V
Setting 3	V	X
Setting 4	V	V

Table 12: the four settings used with KLIEP

We also wanted to take some insight into the similarity between the two datasets. Since the distributions are unknown, we could not calculate directly the KL-divergence, as we would have to first estimate the distributions, something very expensive operationally wise.

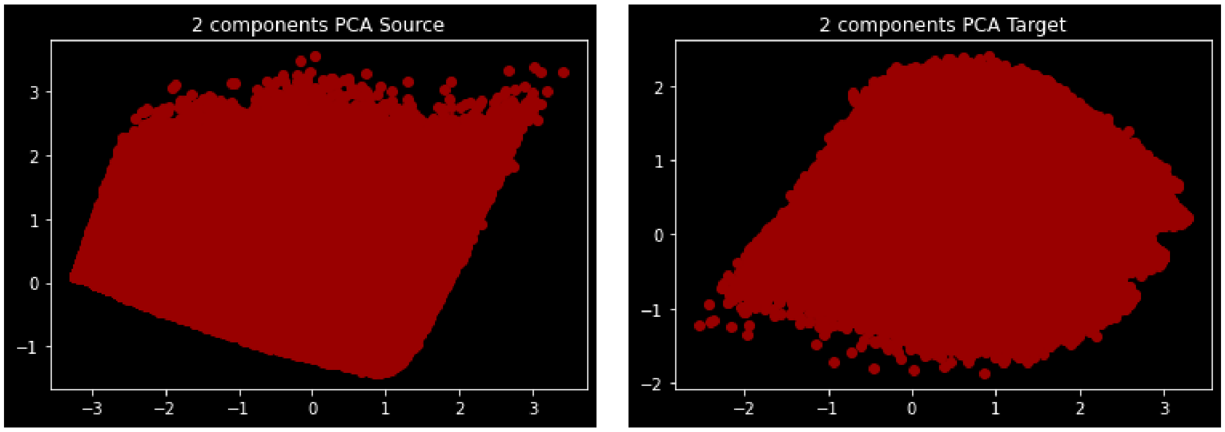
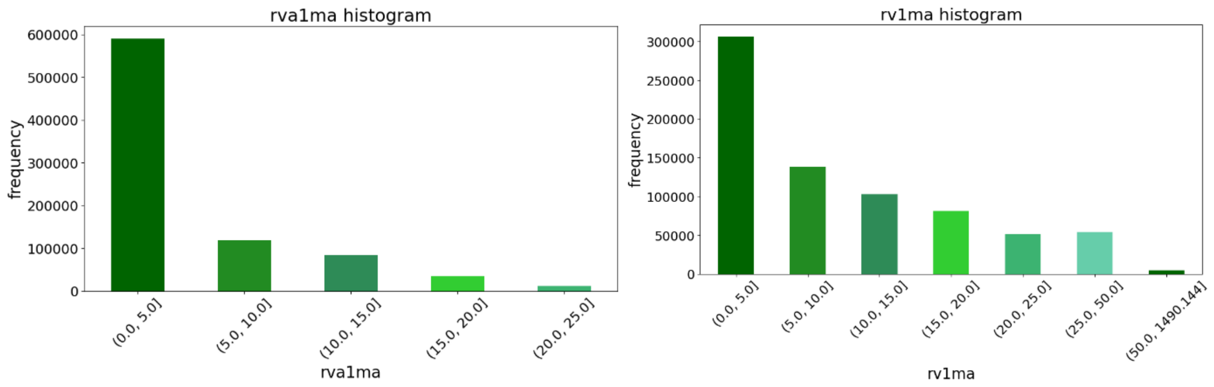


Figure 29: 2 principal components PCA on Source and Target. The clusters formed have a different shape and they look like they are reflected along the horizontal axis.

We resorted to simpler and more naïve methods again, like with the other dataset. We did PCA on two principal components to compare the emerging graphs and some histograms on some variables after binning. We chose two very characteristic and important variables to show, as shown in the graphs below, namely: recharge value (approximation) 1 month ago (rv(a)1ma) and recharges number 1 month ago (rn1ma).



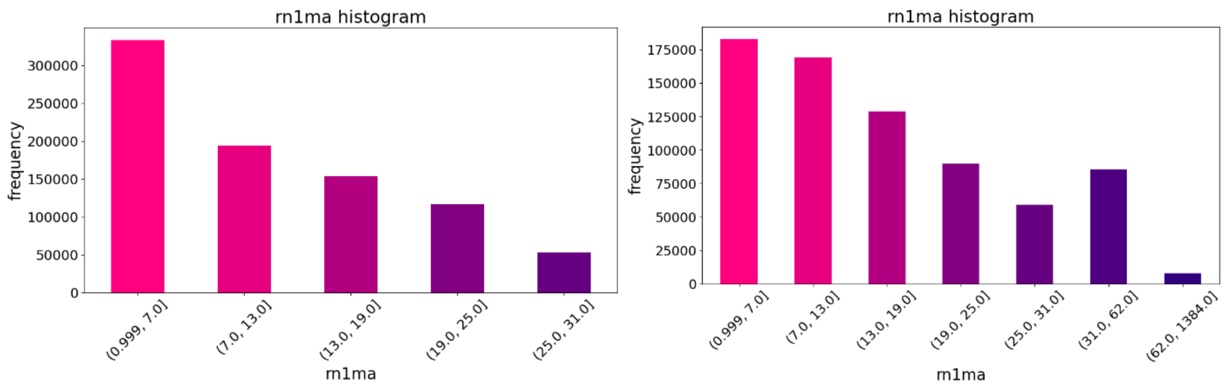


Figure 30: distribution of rv1ma and rn1ma for Source and Target; graphs on left are the Source's, graphs on right are the Target's

5.2.1. Dataset B – KLIEP

As it has been said before, dataset B is split into segments based on the users' behavior. Segment 1 includes the users who are thriftier and spend less money on airtime recharges and segment 8 includes those who spend more. The default rate is then expectedly different in every segment, and this is particularly significant as it signifies the magnitude of the imbalance. Namely the default rates per segment are:

	segment 1	**segment 2**	**segment 3**	**segment 4**	**segment 5**	**segment 6**	**segment 7**	**segment 8**
Source	4.53%	5.27%	4.48%	4.15%	4.56%	3.99%	2.77%	1.16%
Target	2.75%	2.87%	3.21%	2.77%	1.43%	1.72%	1.00%	1.97%

Table 13: default rates per segment on Source and Target

Noticeably, the default rate is higher in Source in all segments except for the last one.

The graph below shows the results obtained on each segment and each line corresponds to each setting described above.

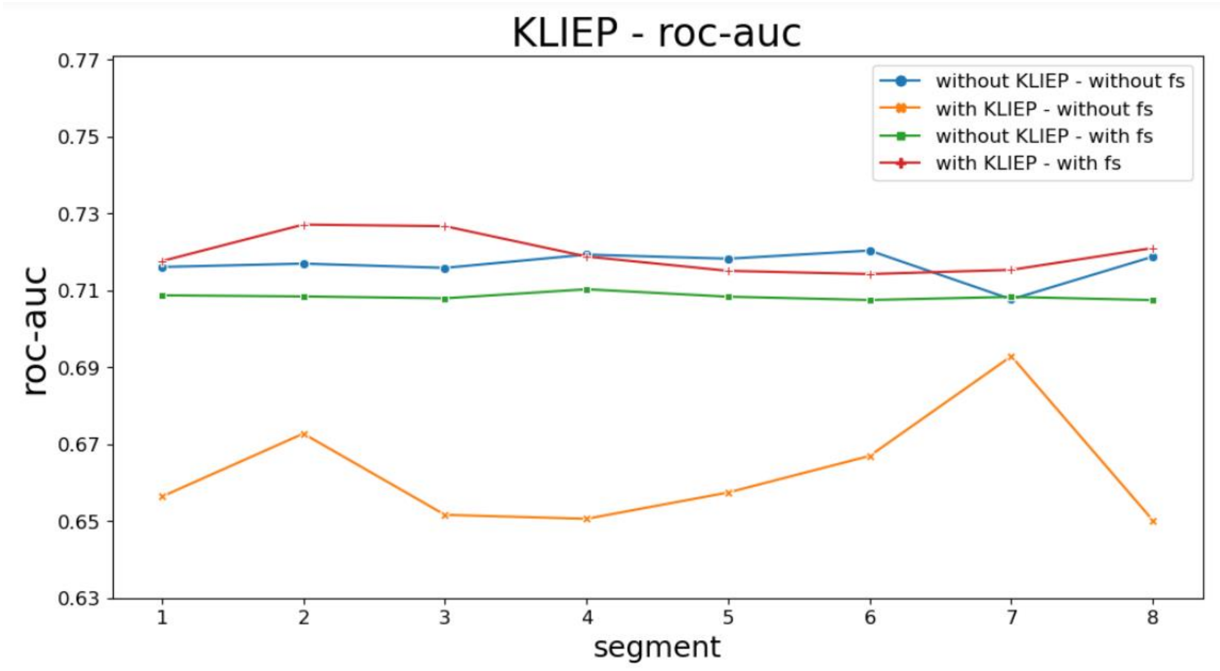


Figure 31: line graphs with the roc-auc scores on the four settings (with/without KLIEP, with/without feature selection)

The feature selection was performed on Source's dataset and the set of selected features was also applied to Target's dataset. Of course, in this way we cannot know how good this subset is for Target; we merely rely on the similarity between the domains. That is, the Genetic Algorithm ran on Source's data and it selected an optimal subset of the features' set for Source. This does not entail that these specific features that were selected are good and predictive for Target. There is a huge field of research that studies transfer learning based on feature space (Dai, Xue, Yang & Yu. 2009), (Johnson & Zhang, 2005), (Blitzer, John, Ryan McDonald, & Fernando Pereira 2006), (Evgeniou, Evgeniou & Pontil, 2007), (Jebara, 2004), (Lee, Chatalbashev, Vickrey & Koller, 2007) . However, since we chose to follow two instance-based methods (Sugiyama et al., 2008), (Wenyuan et. al., 2007) we did not search an adequate shared representation space any further.

	without KLIEP - without FS	**without KLIEP - with FS**	**with KLIEP - without FS**	**with KLIEP - with FS**
segment 1	0.716	0.709	0.656	0.718
segment 2	0.717	0.708	0.673	0.727
segment 3	0.716	0.708	0.652	0.727
segment 4	0.719	0.710	0.651	0.719
segment 5	0.718	0.708	0.657	0.715
segment 6	0.720	0.707	0.667	0.714
segment 7	0.708	0.708	0.693	0.715
segment 8	0.719	0.707	0.650	0.721

Table 14: roc-auc scores in Dataset B when using KLIEP along with feature selection

The worst results were obtained when KLIEP was applied without feature selection which is depicted in the orange line. The best scores were obtained in segments 2,3,7 by setting (b) KLIEP and feature selection with a slightly increased roc-auc score over setting (a) no KLIEP, no feature selection, by 1%. In segments 1, 4 and 8 the scores were almost equal, and in segments 5 and 6 setting (a) exceeded setting (b) in roc-auc but so little (no more than 0.6%) that can be considered insignificant. Therefore, KLIEP did not seem to be able to systematically adapt the domains and boost the performance of the classifier on Target. Let's note that there are limitations on the success of domain adaptation when there are no labeled Target data and the adaptation relies merely on the similarity between the two distributions (Wang et al., 2019, as cited by Ben-David).

When using feature selection and KLIEP, this specific subset of features could be one that allowed for little similarity between the domains, something that inevitably induces transfer learning (Wang et al., 2019). Blitzer et. al. (Blitzer, McDonald & Pereira, 2006) for example have suggested a method that identifies pivot features, e.g., those that exhibit in the same way across two distributions, and uses them to create a new representational, augmented space for the two domains. We could apply this method to see if the negative transfer is eliminated, at least to some extent. Another thing to consider as a reason for KLIEP's failing is that assumption 1 as stated by Sugiyama et. al. (Sugiyama et al., 2008) is violated. Specifically, it is assumed that the two distributions are continuous, something that does not hold true in Dataset B, as discrete features are involved. Another subset, only with continuous variables could be used to see if better results are obtained, however, restricting only to these variables could leave out some that are very informative and predictive.

5.2.2. Dataset B – TrAdaBoost

We tried various test sizes when we were experimenting with trAdaBoost. We managed to obtain best results when we allocated 0.1% and 0.05% of the Target's samples to training as labeled. When testing size was 99.95%, in all segments but segment 1 the use of trAdaBoost boosted the classifier's learning and performance, increasing roc-auc score from 1% up to 10%. Feature selection improved the learning in general but did not make any difference to trAdaBoost particularly. The most impressive boost is noted in segment 8, in which Target's default rate is slightly higher than Source's, while both are very low.

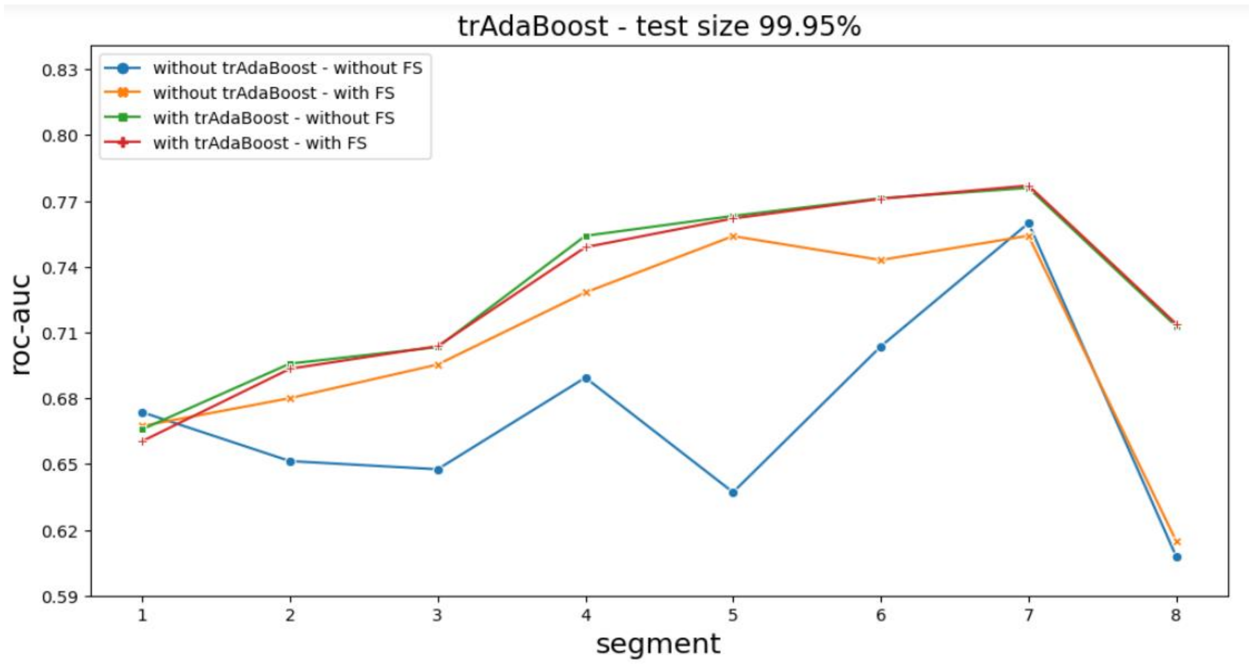


Figure 32: line graphs with the roc-auc scores with test size 99.95% on the four settings (with/without trAdaBoost, with/without feature selection)

When test size was set to 99.9%, e.g., when the allocated labeled samples from Target for training were 0.1%, the use of trAdaBoost along with feature selection gave the best results. Using trAdaBoost without feature selection alleviated the learning in segments 3 to 8, while on segments 1 and 2 it induced negative transfer. The former segments include the thriftier users, whose behavior is reflected on various variables. This could be an explanation for the drop in the performance of trAdaBoost with feature selection in segments 1 and 2. Moreover, when using trAdaBoost, feature selection influenced the results in segments 1-3, as it can be seen from the graph below, but on the other segments there was no difference between trAdaBoost with feature selection and trAdaBoost without feature selection. A similar explanation as the one before could be given. Not using either trAdaBoost or feature selection proved to be once again the worst case.

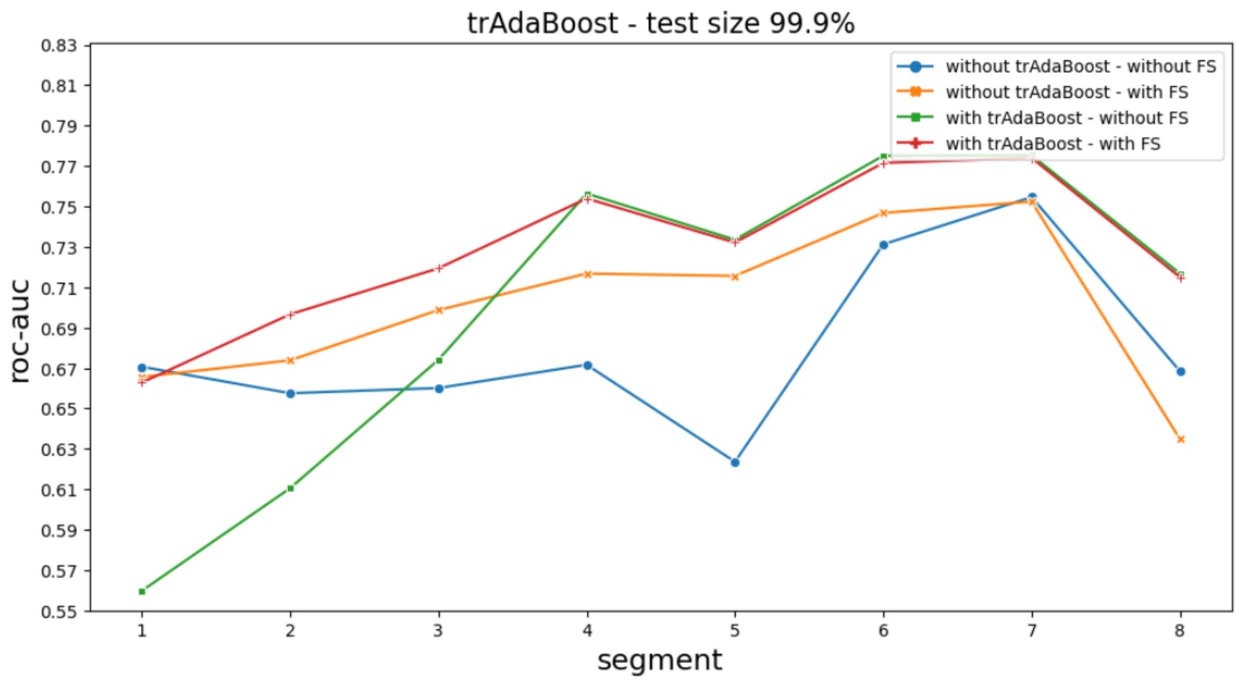


Figure 33: line graphs with the roc-auc scores with test size 99.9% on the four settings (with/without trAdaBoost, with/without feature selection)

Tables 15 and 16 summarize the results from the 4 different settings with test sizes 99.9%, 99.5%.

test size = 0.999	**without trAdaBoost - without fs**	**without trAdaBoost - with fs**	**with trAdaBoost - without fs**	**with trAdaBoost - with fs**
segment 1	0.671	0.666	0.560	0.663
segment 2	0.658	0.674	0.611	0.697
segment 3	0.660	0.699	0.674	0.719
segment 4	0.672	0.717	0.756	0.754
segment 5	0.624	0.716	0.734	0.732
segment 6	0.731	0.747	0.775	0.772
segment 7	0.755	0.752	0.775	0.774
segment 8	0.668	0.635	0.717	0.715

Table 15: roc-auc scores on test size 99.9% on four different settings (with/without trAdaBoost,, with/without feature selection)

test size = 0.9995	**without trAdaBoost - without FS**	**without trAdaBoost - with FS**	**with trAdaBoost - without FS**	**with trAdaBoost - with FS**
segment 1	0.674	0.667	0.666	0.660
segment 2	0.651	0.680	0.696	0.693
segment 3	0.648	0.695	0.703	0.704
segment 4	0.689	0.728	0.754	0.749
segment 5	0.637	0.754	0.763	0.762
segment 6	0.704	0.743	0.771	0.771
segment 7	0.760	0.754	0.776	0.777
segment 8	0.608	0.615	0.713	0.714

Table 16: roc-auc scores on test size 99.9% on four different settings (with/without trAdaBoost, with/without feature selection)

6. Conclusions and further research

In this thesis we used Transfer Learning, and specifically two Domain Adaptation techniques, to confront the situation of concept drift occurring in two Credit Risk problems. We used two different loan provision datasets split into a Source and a Target domain and we tried to predict those who default on their loans. One of the datasets (Dataset B) was highly imbalanced, with minority class holding less than 3.8%, while the other dataset was imbalanced yet to a more moderate extent, with minority class holding approximately 25%. The basic metric used to evaluate our methods was the roc-auc score, because this metric focuses on distinguishing the two classes, that is to say to recognize the defaulters from the non-defaulters in our case.

Before applying the Transfer Learning methods, we experimented with various techniques to preprocess the data and build good base classifiers. We eventually chose to try feature selection with Genetic Algorithm in Dataset B and some data cleaning in Lending Club Dataset. Dataset B was moreover divided into eighth segments that were created with respect to the amount that the users spend on recharging. We then applied the two Domain Adaptation techniques, (a) KLIEP, and (b) TrAdaBoost. In Dataset B we used four different settings as shown in table 11 and we experimented with the sample size in KLIEP and the test size in trAdaBoost. Likewise, in Lending Club Dataset we experimented with the sample size in KLIEP and the test size in trAdaBoost after splitting the dataset with respect to some thresholds of the loaned amount, as specifically described in section 4.1. The results after using the Domain Adaptation methods were compared to the results of training on Source and testing on Target domain but without the use of any such method. One thing to begin with as far as Lending Club Dataset is concerned, in which three different classifiers were used, is that the three classifiers exhibited a different behavior under the Domain Adaptation methods.

In Lending Club Dataset, KLIEP proved to significantly boost Gradient Boosting Classifier when the similarity between the domains was small or medium - splits (a) and (b) - while it induced negative transfer in most cases where the similarity was high - split (c). With Random Forest the results of KLIEP were more moderate, with a smaller increase (or neutral/negative two times) in the roc-auc score. Logistic Regression performed already too well without the use of a Transfer Learning method, as well as with them, therefore the results were of no interest. In Dataset B, KLIEP slightly boosted Logistic Regession’s learning in three out of eight segments, and had minor negative or no impact at all in the remaining five segments. In other words, it did not manage to substantially assist the learning. It is known that when there are no labeled Target data, the success of the domain adaptation exclusively depends on the similarity between the two distributions (Wang et al., 2019, as cited by Ben-David).

Therefore, it is most probable that due to the high divergence between Source and Target in Dataset B, KLIEP did not manage to assist the learning. However, observing that the results vary with respect to the algorithm used for classification, it is highly probable that there be other machine or deep learning algorithms that would perform better under KLIEP, and this could be one extension of this current work. Another interesting observation is that the sample size did not influence the learning with KLIEP; this can be supported by looking at the mathematical postulation of the convex optimization problem, where we can see that the sample size is a parameter that is not involved in it.

TrAdaBoost proved to have varied behavior under changes in the similarity between the domains, the test size, and of course the dataset and the algorithm used. With Lending Club dataset and Gradient Boosting it was successful in boosting the learning when there was more divergence between the domains - splits (b) and (c) - but when the domains were more similar - split (a) - the performance differed with the various test sizes. It is particularly interesting in split (a) the fact that with very small - 0.5% - and larger - 20% and 30% - test sizes transfer learning was positive, whereas on medium test sizes - 5% and 10% - it was negative. The causes may lie in the exact similarity between the domains, the number of samples in Source and Target, and the epochs that trAdaBoost ran. Using some metrics like the Kullback-Leibler or Jensen-Shannon divergence could shed some light into this, as well as experimenting more with more test sizes, epochs and splits. Random Forest on the other hand showed quite unstable behavior with the various splits and test sizes, while all the changes, either positive or negative, in the performance were in general small.

Let’s recall that Random Forest is a bagging classifier and Gradient Boosting is a boosting one. Thus, it would be possible that Gradient Boosting blends together better with TrAdaBoost which is also in its nature a boosting algorithm. More experiments with some other bagging and boosting models could also help us test this hypothesis. With Dataset B and Logistic Regression the provision of Target labeled samples in the training proved to be determinant, as it boosted the learning in all but one segments when test size was set to 0.5% and all features were used, with increase in roc-auc to be up to 9%. It is also interesting that under a slightly different test-size, that of 1%, some segments were influenced by the feature selection applied, resulting in significantly lower performance. We can deduce that the common feature space where the data are brought is essential to be chosen adequately.

An exploration of some feature-based methods that aim to find such a space representation would probably be insightful for this question. We manually restricted the feature space to be that of the set of common features of Source and Target and then we applied feature selection to Source domain, but a feature-based method would probably learn a more adequate representation in a strategic way.

Bibliography

Adamko, P., Kliestik, T., & Birtus, M. (2014). History of credit risk models. In 2nd international conference on economics and social science, Information Engineering Research Institute (pp. 148-153).

Addo, P. M., Guegan, D., & Hassani, B. (2018). Credit risk analysis using machine and deep learning models. Risks, 6(2), 38.

Akinwande, M. O., Dikko, H. G., & Samson, A. (2015). Variance inflation factor: as a condition for the inclusion of suppressor variable (s) in regression analysis. Open Journal of Statistics, 5(07), 754.

Argyriou, A., Evgeniou, T., & Pontil, M. (2006). Multi-task feature learning. Advances in neural information processing systems, 19.

Babatunde, O. H., Armstrong, L., Leng, J., & Diepeveen, D. (2014). A genetic algorithm-based feature selection.

Beninel, F., Bouaguel, W., & Belmufti, G. (2012). Transfer learning using logistic regression in credit scoring. arXiv preprint arXiv:1212.6167.

Bishop, C. M., & Nasrabadi, N. M. (2006). Pattern recognition and machine learning (Vol. 4, No. 4, p. 738). New York: springer.

Blitzer, J., McDonald, R., & Pereira, F. (2006, July). Domain adaptation with structural correspondence learning. In Proceedings of the 2006 conference on empirical methods in natural language processing (pp. 120-128).

Blitzer, J., McDonald, R., & Pereira, F. (2006, July). Domain adaptation with structural correspondence learning. In Proceedings of the 2006 conference on empirical methods in natural language processing (pp. 120-128).

Branco, P., Torgo, L., & Ribeiro, R. P. (2016). A survey of predictive modeling on imbalanced domains. ACM Computing Surveys (CSUR), 49(2), 1-50.

Butaru, F., Chen, Q., Clark, B., Das, S., Lo, A. W., & Siddique, A. (2016). Risk and risk management in the credit card industry. Journal of Banking & Finance, 72, 218-239.

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: synthetic minority over-sampling technique. Journal of artificial intelligence research, 16, 321-357.

Creswell, A., White, T., Dumoulin, V., Arulkumaran, K., Sengupta, B., & Bharath, A. A. (2018). Generative adversarial networks: An overview. IEEE Signal Processing Magazine, 35(1), 53-65.

Dai, W., Xue, G. R., Yang, Q., & Yu, Y. (2007, August). Co-clustering based classification for out-of-domain documents. In Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 210-219).

Duan, L., Tsang, I. W., Xu, D., & Chua, T. S. (2009, June). Domain adaptation from multiple sources via auxiliary classifiers. In Proceedings of the 26th annual international conference on machine learning (pp. 289-296).

Ferri, C., Hernández-Orallo, J., & Modroiu, R. (2009). An experimental comparison of performance measures for classification. Pattern recognition letters, 30(1), 27-38.

Freund, Y., & Schapire, R. E. (1996, July). Experiments with a new boosting algorithm. In icml (Vol. 96, pp. 148-156).

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

Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. Annals of statistics, 1189-1232.

Gama, J., & Zhang, G. (2019). Learning under Concept Drift: A Review. IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, 31(12).

Ganin, Y., Ustinova, E., Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., ... & Lempitsky, V. (2016). Domain-adversarial training of neural networks. The journal of machine learning research, 17(1), 2096-2030.

Hassanat, A. B., & Alkafaween, E. A. (2017). On enhancing genetic algorithms using new crossovers. International Journal of Computer Applications in Technology, 55(3), 202-212.

Hassanat, A., Almohammadi, K., Alkafaween, E., Abunawas, E., Hammouri, A., & Prasath, V. B. (2019). Choosing mutation and crossover ratios for genetic algorithms—a review with a new dynamic approach. Information, 10(12), 390.

Hastie, T., Tibshirani, R., & Friedman, J. (2009). Random forests. In The elements of statistical learning (pp. 587-604). Springer, New York, NY.

Jiawei, H. (2022). Lecture 2: Know Your Data [PDF]. CS412: An Introduction to Data Warehousing and Data Mining. Illinois. Retrieved from http://hanj.cs.illinois.edu/cs412/bk3/KL-divergence.pdf

Multi-task Learning. (2022, January 30). In https://en.wikipedia.org/wiki/Multi-task_learning

Huang, J., & Chen, M. (2018, January). Domain adaptation approach for credit risk analysis. In Proceedings of the 2018 International Conference on Software Engineering and Information Management (pp. 104-107).

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). An introduction to statistical learning (Vol. 112, p. 18). New York: springer.

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). An introduction to statistical learning (Vol. 112, p. 18). New York: springer.

Japkowicz, N., & Stephen, S. (2002). The class imbalance problem: A systematic study. Intelligent data analysis, 6(5), 429-449.

Jebara, T. (2004, July). Multi-task feature and kernel selection for SVMs. In Proceedings of the twenty-first international conference on Machine learning (p. 55).

Jiménez-Guarneros, M., & Gomez-Gil, P. (2021). A study of the effects of negative transfer on deep unsupervised domain adaptation methods. Expert Systems with Applications, 167, 114088.

Johnson, R., & Zhang, T. (2005, June). A high-performance semi-supervised learning method for text chunking. In Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05) (pp. 1-9).

Khandani, A. E., Kim, A. J., & Lo, A. W. (2010). Consumer credit-risk models via machine-learning algorithms. Journal of Banking & Finance, 34(11), 2767-2787.

Kingsford, C., & Salzberg, S. L. (2008). What are decision trees?. Nature biotechnology, 26(9), 1011-1013.

Kouw, W. M., & Loog, M. (2019). A review of domain adaptation without target labels. IEEE transactions on pattern analysis and machine intelligence, 43(3), 766-785.

Kuhn, M., & Johnson, K. (2013). Applied predictive modeling (Vol. 26, p. 13). New York: Springer.

Lee, S. I., Chatalbashev, V., Vickrey, D., & Koller, D. (2007, June). Learning a meta-level prior for feature relevance from multiple related tasks. In Proceedings of the 24th international conference on Machine learning (pp. 489-496).

Ling, C. X., & Sheng, V. S. (2008). Cost-sensitive learning and the class imbalance problem. Encyclopedia of machine learning, 2011, 231-235.

Pan, S. J., & Yang, Q. (2009). A survey on transfer learning. IEEE Transactions on knowledge and data engineering, 22(10), 1345-1359.

Raina, R., Battle, A., Lee, H., Packer, B., & Ng, A. Y. (2007, June). Self-taught learning: transfer learning from unlabeled data. In Proceedings of the 24th international conference on Machine learning (pp. 759-766).

Sugiyama, M., Suzuki, T., Nakajima, S., Kashima, H., von Bünau, P., & Kawanabe, M. (2008). Direct importance estimation for covariate shift adaptation. Annals of the Institute of Statistical Mathematics, 60(4), 699-746.

Suryanto, H., Guan, C., Voumard, A., & Beydoun, G. (2019, September). Transfer learning in credit risk. In Joint European Conference on Machine Learning and Knowledge Discovery in Databases (pp. 483-498). Springer, Cham.

Wang, Z., Dai, Z., Póczos, B., & Carbonell, J. (2019). Characterizing and avoiding negative transfer. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 11293-11302).

Wei, Q., Liu, Y., & Wu, K. (2021, May). Transfer Learning Based Credit Scoring. In 2021 IEEE 24th International Conference on Computer Supported Cooperative Work in Design (CSCWD) (pp. 1251-1255). IEEE.

Wenyuan Dai , Qiang Yang , Gui-rong Xue , Yong Yu, (2007). Boosting for transfer learning. In Proceedings of the 24th International Conference on Machine Learning (pp. 193–200).

Xiao, J., Wang, R., Teng, G., & Hu, Y. (2014, July). A transfer learning based classifier ensemble model for customer credit scoring. In 2014 Seventh International Joint Conference on Computational Sciences and Optimization (pp. 64-68). IEEE.

Zhang, Y., & Yang, Q. (2018). An overview of multi-task learning. National Science Review, 5(1), 30-43.

Zheng, L., Liu, G., Yan, C., Jiang, C., Zhou, M., & Li, M. (2020). Improved TrAdaBoost and its application to transaction fraud detection. IEEE Transactions on Computational Social Systems, 7(5), 1304-1316.

Zhou, Z. H., & Liu, X. Y. (2005). Training cost-sensitive neural networks with methods addressing the class imbalance problem. IEEE Transactions on knowledge and data engineering, 18(1), 63-77.

Zhuang, F., Cheng, X., Luo, P., Pan, S. J., & He, Q. (2015, June). Supervised representation learning: Transfer learning with deep autoencoders. In Twenty-Fourth International Joint Conference on Artificial Intelligence.

Zhuang, F., Qi, Z., Duan, K., Xi, D., Zhu, Y., Zhu, H., ... & He, Q. (2020). A comprehensive survey on transfer learning. Proceedings of the IEEE, 109(1), 43-76.

Appendix

A.1. Dictionary for Lending Club Dataset variables

LoanStatNew	Description
acc_now_delinq	The number of accounts on which the borrower is now delinquent.
acc_open_past_24mths	Number of trades opened in past 24 months.
addr_state	The state provided by the borrower in the loan application
all_util	Balance to credit limit on all trades
annual_inc	The self-reported annual income provided by the borrower during registration.
annual_inc_joint	The combined self-reported annual income provided by the co-borrowers during registration
application_type	Indicates whether the loan is an individual application or a joint application with two co-borrowers
avg_cur_bal	Average current balance of all accounts
bc_open_to_buy	Total open to buy on revolving bankcards.
bc_util	Ratio of total current balance to high credit/credit limit for all bankcard accounts.
chargeoff_within_12_mths	Number of charge-offs within 12 months
collection_recovery_fee	post charge off collection fee
collections_12_mths_ex_med	Number of collections in 12 months excluding medical collections
delinq_2yrs	The number of 30+ days past-due incidences of delinquency in the borrower's credit file for the past 2 years

delinq_amnt	The past-due amount owed for the accounts on which the borrower is now delinquent.
desc	Loan description provided by the borrower
dti	A ratio calculated using the borrower’s total monthly debt payments on the total debt obligations, excluding mortgage and the requested LC loan, divided by the borrower’s self-reported monthly income.
dti_joint	A ratio calculated using the co-borrowers' total monthly payments on the total debt obligations, excluding mortgages and the requested LC loan, divided by the co-borrowers' combined self-reported monthly income
earliest_cr_line	The month the borrower's earliest reported credit line was opened
emp_length	Employment length in years. Possible values are between 0 and 10 where 0 means less than one year and 10 means ten or more years.
emp_title	The job title supplied by the Borrower when applying for the loan.*
fico_range_high	The upper boundary range the borrower’s FICO at loan origination belongs to.
fico_range_low	The lower boundary range the borrower’s FICO at loan origination belongs to.
funded_amnt	The total amount committed to that loan at that point in time.
funded_amnt_inv	The total amount committed by investors for that loan at that point in time.
grade	LC assigned loan grade
home_ownership	The home ownership status provided by the borrower during registration or obtained from the credit report. Our values are: RENT, OWN, MORTGAGE, OTHER

id	A unique LC assigned ID for the loan listing.
il_util	Ratio of total current balance to high credit/credit limit on all install acct
initial_list_status	The initial listing status of the loan. Possible values are – W, F
inq_fi	Number of personal finance inquiries
inq_last_12m	Number of credit inquiries in past 12 months
inq_last_6mths	The number of inquiries in past 6 months (excluding auto and mortgage inquiries)
installment	The monthly payment owed by the borrower if the loan originates.
int_rate	Interest Rate on the loan
issue_d	The month which the loan was funded
last_credit_pull_d	The most recent month LC pulled credit for this loan
last_fico_range_high	The upper boundary range the borrower's last FICO pulled belongs to.
last_fico_range_low	The lower boundary range the borrower's last FICO pulled belongs to.
last_pymnt_amnt	Last total payment amount received
last_pymnt_d	Last month payment was received
loan_amnt	The listed amount of the loan applied for by the borrower. If at some point in time, the credit department reduces the loan amount, then it will be reflected in this value.
loan_status	Current status of the loan
max_bal_bc	Maximum current balance owed on all revolving accounts

member_id	A unique LC assigned Id for the borrower member.
mo_sin_old_il_acct	Months since oldest bank installment account opened
mo_sin_old_rev_tl_op	Months since oldest revolving account opened
mo_sin_rcnt_rev_tl_op	Months since most recent revolving account opened
mo_sin_rcnt_tl	Months since most recent account opened
mort_acc	Number of mortgage accounts.
mths_since_last_delinq	The number of months since the borrower's last delinquency.
mths_since_last_major_derog	Months since most recent 90-day or worse rating
mths_since_last_record	The number of months since the last public record.
mths_since_rcnt_il	Months since most recent installment accounts opened
mths_since_recent_bc	Months since most recent bankcard account opened.
mths_since_recent_bc_dlq	Months since most recent bankcard delinquency
mths_since_recent_inq	Months since most recent inquiry.
mths_since_recent_revol_delinq	Months since most recent revolving delinquency.
next_pymnt_d	Next scheduled payment date
num_accts_ever_120_pd	Number of accounts ever 120 or more days past due
num_actv_bc_tl	Number of currently active bankcard accounts
num_actv_rev_tl	Number of currently active revolving trades
num_bc_sats	Number of satisfactory bankcard accounts

num_bc_tl	Number of bankcard accounts
num_il_tl	Number of installment accounts
num_op_rev_tl	Number of open revolving accounts
num_rev_accts	Number of revolving accounts
num_rev_tl_bal_gt_0	Number of revolving trades with balance >0
num_sats	Number of satisfactory accounts
num_tl_120dpd_2m	Number of accounts currently 120 days past due (updated in past 2 months)
num_tl_30dpd	Number of accounts currently 30 days past due (updated in past 2 months)
num_tl_90g_dpd_24m	Number of accounts 90 or more days past due in last 24 months
num_tl_op_past_12m	Number of accounts opened in past 12 months
open_acc	The number of open credit lines in the borrower's credit file.
open_acc_6m	Number of open trades in last 6 months
open_il_12m	Number of installment accounts opened in past 12 months
open_il_24m	Number of installment accounts opened in past 24 months
open_act_il	Number of currently active installment trades
open_rv_12m	Number of revolving trades opened in past 12 months
open_rv_24m	Number of revolving trades opened in past 24 months
out_prncp	Remaining outstanding principal for total amount funded
out_prncp_inv	Remaining outstanding principal for portion of total amount funded by investors

pct_tl_nvr_dlq	Percent of trades never delinquent
percent_bc_gt_75	Percentage of all bankcard accounts > 75% of limit.
policy_code	publicly available policy_code=1 new products not publicly available policy_code=2
pub_rec	Number of derogatory public records
pub_rec_bankruptcies	Number of public record bankruptcies
purpose	A category provided by the borrower for the loan request.
pymnt_plan	Indicates if a payment plan has been put in place for the loan
recoveries	post charge off gross recovery
revol_bal	Total credit revolving balance
revol_util	Revolving line utilization rate, or the amount of credit the borrower is using relative to all available revolving credit.
sub_grade	LC assigned loan subgrade
tax_liens	Number of tax liens
term	The number of payments on the loan. Values are in months and can be either 36 or 60.
title	The loan title provided by the borrower
tot_coll_amt	Total collection amounts ever owed
tot_cur_bal	Total current balance of all accounts
tot_hi_cred_lim	Total high credit/credit limit
total_acc	The total number of credit lines currently in the borrower's credit file

total_bal_ex_mort	Total credit balance excluding mortgage
total_bal_il	Total current balance of all installment accounts
total_bc_limit	Total bankcard high credit/credit limit
total_cu_tl	Number of finance trades
total_il_high_credit_limit	Total installment high credit/credit limit
total_pymnt	Payments received to date for total amount funded
total_pymnt_inv	Payments received to date for portion of total amount funded by investors
total_rec_int	Interest received to date
total_rec_late_fee	Late fees received to date
total_rec_prncp	Principal received to date
total_rev_hi_lim	Total revolving high credit/credit limit
url	URL for the LC page with listing data.
verification_status	Indicates if income was verified by LC, not verified, or if the income source was verified
verified_status_joint	Indicates if the co-borrowers' joint income was verified by LC, not verified, or if the income source was verified
zip_code	The first 3 numbers of the zip code provided by the borrower in the loan application.
revol_bal_joint	Sum of revolving credit balance of the co-borrowers, net of duplicate balances
sec_app_fico_range_low	FICO range (high) for the secondary applicant
sec_app_fico_range_high	FICO range (low) for the secondary applicant
sec_app_earliest_cr_line	Earliest credit line at time of application for the secondary applicant

sec_app_inq_last_6mths	Credit inquiries in the last 6 months at time of application for the secondary applicant
sec_app_mort_acc	Number of mortgage accounts at time of application for the secondary applicant
sec_app_open_acc	Number of open trades at time of application for the secondary applicant
sec_app_revol_util	Ratio of total current balance to high credit/credit limit for all revolving accounts
sec_app_open_act_il	Number of currently active installment trades at time of application for the secondary applicant
sec_app_num_rev_accts	Number of revolving accounts at time of application for the secondary applicant
sec_app_chargeoff_within_12_mths	Number of charge-offs within last 12 months at time of application for the secondary applicant
sec_app_collections_12_mths_ex_med	Number of collections within last 12 months excluding medical collections at time of application for the secondary applicant
sec_app_mths_since_last_major_derog	Months since most recent 90-day or worse rating at time of application for the secondary applicant
hardship_flag	Flags whether or not the borrower is on a hardship plan
hardship_type	Describes the hardship plan offering
hardship_reason	Describes the reason the hardship plan was offered
hardship_status	Describes if the hardship plan is active, pending, canceled, completed, or broken
deferral_term	Amount of months that the borrower is expected to pay less than the contractual monthly payment amount due to a hardship plan
hardship_amount	The interest payment that the borrower has committed to make each month while they are on a hardship plan

hardship_start_date	The start date of the hardship plan period
hardship_end_date	The end date of the hardship plan period
payment_plan_start_date	The day the first hardship plan payment is due. For example, if a borrower has a hardship plan period of 3 months, the start date is the start of the three-month period in which the borrower is allowed to make interest-only payments.
hardship_length	The number of months the borrower will make smaller payments than normally obligated due to a hardship plan
hardship_dpd	Account days past due as of the hardship plan start date
hardship_loan_status	Loan Status as of the hardship plan start date
orig_projected_additional_accrued_interest	The original projected additional interest amount that will accrue for the given hardship payment plan as of the Hardship Start Date. This field will be null if the borrower has broken their hardship payment plan.
hardship_payoff_balance_amount	The payoff balance amount as of the hardship plan start date
hardship_last_payment_amount	The last payment amount as of the hardship plan start date
disbursement_method	The method by which the borrower receives their loan. Possible values are: CASH, DIRECT_PAY
debt_settlement_flag	Flags whether or not the borrower, who has charged-off, is working with a debt-settlement company.
debt_settlement_flag_date	The most recent date that the Debt_Settlement_Flag has been set
settlement_status	The status of the borrower's settlement plan. Possible values are: COMPLETE, ACTIVE, BROKEN, CANCELLED, DENIED, DRAFT

settlement_date	The date that the borrower agrees to the settlement plan
settlement_amount	The loan amount that the borrower has agreed to settle for
settlement_percentage	The settlement amount as a percentage of the payoff balance amount on the loan
settlement_term	The number of months that the borrower will be on the settlement plan

A.2. Dictionary for Dataset B variables

Recharge Features - Calendar

Short Name	Description
lrdt	Last Recharge Date (before RD, or last day of RM if RD is missing)
lrdsTnr	Days passed from last recharge
rvXma	Recharge value X calendar months ago
rvXtoYma	Total recharge value in X to Y calendar months ago
mrvXtoYma	Mean of monthly (calendar) recharge values of X to Y months ago
rvstdXtoYma	Standard deviation of monthly (calendar) recharge values of X to Y months ago
rnXma	Recharges number X calendar month ago
mrnXtoYma	Mean of monthly (calendar) recharge number of X to Y months ago
rnstdXtoYma	Standard deviation of monthly (calendar) recharge number of X to Y months ago
mxrXma	Maximum Recharge Amount in X calendar month ago
mxrXtoYma	Maximum Recharge Amount in X to Y calendar months ago
arvXma	Average Recharge Value (the average value of top-ups) X months ago
arvXtoYma	Average Recharge Value (the average value of top-ups) in X to Y months ago
mwrXma	Months with Recharges in 1 to X months ago (depreciated)
mwrXtoYma	Months with Recharges in X to Y months ago
rfgXma	Recharge flag in X months ago

STV Recharge Features - Calendar

Short Name	Description
stvrvXma	STV recharge value X calendar months ago
mstvrvXtoYma	Mean of monthly (calendar) STV recharge values of X to Y months ago
stvrvstdXtoYma	Standard deviation of monthly (calendar) STVrecharge values of X to Y months ago
stvrnXma	STV recharges number X calendar month ago
mstvrnXtoYma	Mean of monthly (calendar) STV recharge number of X to Y months ago
stvrnstdXtoYma	Standard deviation of monthly (calendar) STV recharge number of X to Y months ago
mxstvrXma	Maximum STV Recharge Amount in X calendar month ago

mxstvrXtoYma	Maximum STV Recharge Amount in X to Y calendar months ago
stvrvpXma	STV recharge value percentage of total in X calendar months
stvrvpXtoYma	STV recharge value percentage of total in X to Y months ago
stvrnpXma	STV recharges number percentage of total in X calendar months ago
stvrnpXtoYma	STV recharges number percentage of total in X to Y months ago
mwstvrXtoYma	Months with STV Recharges in X to Y months ago

Monetary Recharge Features - Calendar

Short Name	Description
mntrvXma	MNT recharge value X calendar months ago
mmntrvXtoYma	Mean of monthly (calendar) monetary recharge values of X to Y months ago
mntrvstdXtoYma	Standard deviation of monthly (calendar) monetary recharge values of X to Y months ago
mntrnXma	MNT recharges number X calendar month ago
mmntrnXtoYma	Mean of monthly (calendar) monetary recharge number of X to Y months ago
mntrnstdXtoYma	Standard deviation of monthly (calendar) monetary recharge number of X to Y months ago
mxmntrXma	Maximum MNT Recharge Amount in X calendar month ago
mxmntrXtoYma	Maximum Monetary Recharge Amount in X to Y calendar months ago
mntrvpXma	Monetary recharge value percentage of total in X calendar months
mntrvpXtoYma	Monetary recharge value percentage of total in X to Y months ago
mntrnpXma	Monetary recharges number percentage of total in X calendar months ago
mntrnpXtoYma	Monetary recharges number percentage of total in X to Y months ago
mwmntrXtoYma	Months with Monetary Recharges in X to Y months ago

Recharge Value Approximation - Calendar

Short Name	Description
rvaXma	Recharge value approximation X calendar months ago
mrvaXtoYma	Mean of monthly (calendar) recharge value approximations of X to Y months ago
rvastdXtoYma	Standard deviation of monthly (calendar) recharge value approximations of X to Y months ago
mwrvaXtoYma	Months with Recharges Value Approximation higher than 0 in X to Y months ago

Business Rules Features - Calendar

Short Name	Description
brbXma	Business Rules eligible Band X months ago
mbrbXtoYma	Mean Business Rules Band X to Y months ago
stdbrbXtoYma	Standard Deviation of Business Rules eligible Band X to Y months ago

p2p Features - Calendar

Short Name	Description
ip2pnXma	Incoming p2p transfers number X months ago
ip2pnXtoYma	Total Incoming p2p transfers number in X to Y months ago
ip2pvXma	Incoming p2p transfers value X months ago
ip2pvXtoYma	Total Incoming p2p transfers value in X to Y months ago

Recharge - Calendar

Short Name	Description
rmthd	Recharge method

Recharge Features - Rolling

Short Name	Description
rvXtoYd	Recharge value in X to Y days before (inclusive)
mrvXtoYd	Mean of 30-days rolling recharge values of X to Y days before
rvstdXtoYd	Standard deviation of 30-days rolling recharge values of X to Y days before
rnXtoYd	Recharge Counts in X to Y days before (inclusive)
mrnXtoYd	Mean of 30-days rolling recharge number of X to Y days before
rnstdXtoYd	Standard deviation of 30-days rolling recharge number of X to Y days before

STV Recharge Features - Rolling

Short Name	Description
stvrvXtoYd	STV recharge value in X to Y 30-days rolling before (inclusive)
stvrnXtoYd	STV recharges number in X to Y 30-days rolling before (inclusive)
mxstvrXtoYd	Maximum STV Recharge Amount in X to Y 30-days rolling before (inclusive)
stvrvpXtoYd	STV recharge value percentage in X to Y days before (inclusive)
stvrnpXtoYd	STV recharges number percentage in X to Y days before (inclusive)

Monetary Recharge Features - Rolling

Short Name	Description
mntrvXtoYd	MNT recharge value in X to Y days before (inclusive)
mntrvpXtoYd	MNT recharge value percentage in X to Y days before (inclusive)
mntrnXtoYd	MNT recharges number in X to Y days before (inclusive)
mntrnpXtoYd	MNT recharges number percentage in X to Y days before (inclusive)

Temporal

Short Name	Description
fadt	First Advance Date (format: yyyMMdd)
smTnr	Service usage tenure in months
ladt	Last Advance Date
ladsTnr	Days passed from last advance
oldUsr	Flag indicating whether the subscriber has made and fully repaid an advance within the last 6 calendar months (deprecated)
usrwlXm	Flag indicating whether the subscriber has made and fully repaid an advance within the last X calendar months

Advance Features - Calendar

Short Name	Description
anXma	Advances number X calendar month ago
manXtoYma	Monthly Mean advances number of X to Ycalendar months ago
anstdXtoYma	Standard Deviation of monthly Advances number of X to Y calendar months ago
avXma	Advances gross amount (principal & fee) X calendar month ago
mavXtoYma	Monthly Mean advances amount (principal and fees) of X to Ycalendar months ago
avstdXtoYma	Standard Deviation of monthly Advances amount (principal and fees) of X to Y calendar months ago
mwaXma	Months with advances in 1 to X months ago (depreciated)
mwaXtoYma	Months with advances in X to Y months ago
afgXma	Advance flag in X months ago

Paid Advance Features - Calendar

Short Name	Description
panXma	Number of repaid advances X calendar month ago, based on full repayment date
mpanXtoYma	Average number of repaid advances X to Y calendar months ago, based on full repayment date
panstdXtoYma	Standard deviation of number of repaid advances X to Y calendar months ago, based on full repayment date
paaXma	Total amount (principal & fees) of fully repaid advances X calendar months ago, based on full repayment date

mpaaXtoYma	Average total amount (principal & fees) of repaid advances in X to Y calendar months ago, based on full repayment date
paastdXtoYma	Standard deviation of total amount (principal & fees) of repaid advances in X to Y calendar months ago, based on full repayment date
taaraXtoYma	Total Advanced and Paid Amount (principal & fees) of advances from X to Y calendar months (advances should have been made and repaid from X to Y calendar months ago; new taapawXlm)
bXaapawXlm	Total Advanced and Paid Amount (principal & fees) of advances of Band X within the last X calendar months (advances should have been made and repaid within the last X calendar months).
mrcXma	Mean recovery cycle of advances repaid in X calendar months ago, based on full payment date (an advance paid on the same day has recovery cycle 1, the next day 2, etc).
mrcXtoYma	Mean recovery cycle of advances repaid in X to Y calendar months ago, based on full payment date (an advance paid on the same day has recovery cycle 1, the next day 2, etc).
mrcstdXtoYma	Standard deviation of recovery cycle of advances repaid in X to Y calendar months ago, based on full payment date (an advance paid on the same day has recovery cycle 1, the next day 2, etc).

Credit Limit Usage Features - Calendar

Short Name	Description
atclpuXma	Average TCL percent usage of advances made in X months ago (based on debt)
atclpuXtoYma	Average TCL percent usage of advances made in X to Y months ago
tZptclrafrwXlm	Times Z percent of TCL reached and fully repaid within X last months

Advance Features - Rolling

Short Name	Description
anXtoYd	Advances number in X to Y days before (inclusive)
manXtoYd	Mean of 30-days rolling advances number of X to Y days before
anstdXtoYd	Standard deviation of 30-days rolling advances number of X to Y days before
avXtoYd	Advances gross amount in X to Y days before (inclusive)
mavXtoYd	Mean of 30-days rolling advances gross amount of X to Y days before
avstdXtoYd	Standard deviation of 30-days rolling advances gross amount of X to Y days before

Paid Advance Features - Rolling

Short Name	Description
panXtoYd	Number of repaid advances in X to Y days before (inclusive), based on full repayment date
mpanXtoYd	Mean of 30-days rolling number of fully repaid advances of X to Y days before (inclusive), based on full repayment date
panstdXtoYd	Standard deviation of 30-days rolling number of fully repaid advances of X to Y days before (inclusive), based on full repayment date
paaXtoYd	Gross amount (principal & fees) of fully repaid advances n X to Y days before (inclusive), based on full repayment date

mpaaXtoYd	Mean of 30-days rolling gross amount of fully repaid advances of X to Y days before (inclusive), based on full repayment date
paastdXtoYd	Standard deviation of 30-days rolling amount of fully repaid advances of X to Y days before (inclusive), based on full repayment date
mrcXtoYd	Mean recovery cycle of advances repaid in X to Y days before (inclusive)
taaraXtoYd	Gross Advanced and Paid Amount (principal & fees) of advances in X to Y days before (inclusive) (advances should have been made and repaid in X to Y days before)

Credit Limit Usage Features - Rolling

Short Name	**Description**
atclpuXtoYd	Average TCL percent usage of advances made in X to Y days before (inclusive)
tZptclrafrwXld	Times Z percent of TCL reached and the advance was fully repaid within X last days