Transfer Learning for Tabular Data with XGBoost

Master's Thesis submitted by

Ulrike Wöll

(621289)

in partial fulfillment of the requirements

for the degree of

Master of Science

School of Business and Economics

Humboldt-Universität zu Berlin

Berlin, September 22, 2023

Abstract

This master's thesis explores the efficacy of various transfer learning strategies in the context of tabular data, utilizing the XGBoost algorithm. Transfer learning, a machine learning paradigm that allows models to transfer knowledge gained from one domain to another, has traditionally been more prevalent in non-tabular data than in tabular data. Through an extensive literature review, the thesis traces the development of transfer learning for tabular data from classic machine learning algorithms to its applications in deep neural networks, and further to XGBoost. It designs a simulation study using synthetic tabular data. Four transfer learning strategies — Combination, Freezing, Progressive Learning, and Fine-tuning — are proposed and thoroughly compared. The results indicate that the effectiveness of these strategies depends on the type of domain dissimilarity. The findings contribute to a better understanding of transfer learning applications in tabular data and offer valuable insights for leveraging the combined potential of transfer learning and the XGBoost algorithm.

The repository with all scripts for replicating the results of this thesis is available under `https://github.com/UlrikeWoell/transfer_learning_with_xgb`.

Contents

List of Abbreviations

CNN	Convolutional Neural Network
DNN	Deep Neural Network
ML	Machine Learning
MLP	Multilayer Perceptron
NLP	Natural Language Processing
NN	Neural Network
NGL	normalized gain/loss in the ROC-AUC compared to the Target Only baseline
ROC-AUC	Receiver Operating Characteristic Area Under the Curve
TL	Transfer Learning
XGB	Extreme Gradient Boosting

List of Figures

List of Tables

1 Introduction

Insufficient training data presents a significant challenge in machine learning projects and is due to several factors. Primarily, the collection and annotation of data are resource-intensive, especially in specialized domains where relevant data is scarce. For example, medical diagnoses often lack diverse data sets, natural language processing projects may have inadequate annotated texts, and image recognition models typically necessitate thousands of labeled images for effective training. In cases of data scarcity, models are prone to over-fitting, leading to subpar generalization performance.

Given these constraints, training new models from scratch may be impractical or prohibitively expensive. Two principal strategies exist to address this issue. The first involves re-purposing pre-existing models. This rather naive approach may yield poor results if the target domain diverges significantly from the original domain. On the other, the second strategy, known as transfer learning, involves adapting pre-existing models according to problem-specific assumptions. Transfer learning enables algorithms to leverage abundant data from a source domain, and to apply this knowledge to a target domain. This approach fosters more accurate and efficient machine learning models by capitalizing on models pre-trained with ample general data, thereby reducing the data requirements in the target domain and expediting the training process.

Yang (2020) identify three main issues in the field of transfer learning: (1) when to transfer, (2) what to transfer, and (3) how to transfer. These questions are covered in this thesis as follows.

When to transfer asks in which situations transferring skills could be beneficial. If the target and source domains are not related, brute-force transfer may be unsuccessful or even harmful to the performance (so-called negative transfer). Often, it is implicitly assumed that the domains are related or similar. A number of metrics have been proposed to measure the similarity or distance between two domains, such as the Kulback-Leibler divergence, Earth Mover's Distance, and the Maximum Mean Discrepancy. However, these metrics are not suitable for deciding whether transfer learning will be successful because there is no limit beyond which transfer learning is discouraged. Therefor, we focus on the categories of difference and less on the size of differences between domains. An assumption of whether a relevant difference is present or not can be based on either data analysis or on justified assumptions.

The various categories of differences between domains are presented in Section 2.4.1. We investigate the capabilities of different approaches to transfer learning under the assumption that a specific category of domain difference is present. To this end, we will generate synthetic data which gives us complete control over the difference between the domains. In Section 5.2, we present a data-generating process that allows us to generate synthetic data from domain pairs that are identical in all but one aspects. This allows us to investigate different categories of domain differences in an isolated way.

What to transfer asks which parts of the knowledge can be transferred. The literature on transfer learning differentiates between knowledge that is specific to the domains or tasks and knowledge that may be common between domains. What is meant by ”knowledge” remains to be clarified in the given context. Examples of transferable knowledge include, but are not limited to the complete trained source model or parts of it, the hyper-parameters found when tuning the source model, and borrowing instances of data across domains. In Section 4.1, we present three strategies that transfer learned hyperparameters and (parts of) trained models, as well as one strategy that transfers instances.

How to transfer specifies the method for transferring knowledge. This method depends not only on the available data and the assumed underlying data-generating process, but also on the used algorithms. Some transfer learning methods for classic machine learning algorithms are presented in Section 3.1. While those are not widely used, three approaches have crystallized in the realm of neural networks: Freezing, finetuning, and progressive learning are the most commonly used methods. On the other hand, with XGBoost, there is currently no standard procedure. In this thesis, we investigate several methods for transfer learning with XGBoost that draw upon experience with the established procedures for neural networks and classic machine learning. These methods are explained in more detail in Section 4.1.

The scope of this thesis is centered on exploring transfer learning strategies using XGBoost for tabular data. We aim to provide a comprehensive understanding of how transfer learning, specifically with the XGBoost algorithm, can be effectively applied to tabular data scenarios. Our focus encompasses a detailed examination of four transfer learning strategies, namely Combination, Freezing, Progressive Learning, and Fine-tuning. We conduct a simulation study using synthetic tabular data to evaluate and compare the performance of the proposed strategies. The data is carefully designed to represent seven different scenarios of distributional differences. Although our findings provide valuable insights, it is important to note that the scope is limited to the synthetic data sets created for the simulation study, and the generalizability of the results to real-world data may vary. Additionally, while we delve into the nuances of XGBoost and its application to transfer learning, other machine learning algorithms and deep learning approaches will not be included in this thesis.

The remainder of this thesis is organized as follows. Chapter 2 lays the foundation by defining key terms and notations and situating both tabular data and XGBoost within the realm of transfer learning. Chapter 3 provides an in-depth literature review, capturing the evolution and current state of transfer learning. In Chapter 4, we outline the methodology, detailing the strategies we suggest for implementing transfer learning with XGBoost. Chapter 5 describes the design of our simulation study, elaborating on how we generated synthetic data and selected relevant parameters. Finally, Chapter 6 presents the results of our study, offering an analysis of each strategy's performance, summarizing key findings, and discussing the limitations of the simulation study.

The repository with all scripts for replicating the results of this thesis is available under `https://github.com/UlrikeWoell/transfer_learning_with_xgb`.

2 Background

This section provides definitions of important terms and notations that will be used throughout this thesis. We will recapitulate the concepts of tabular data and XGBoost in the context of transfer learning and point out some conceptual parallels between XGBoost and Neural Networks. In addition, we will dedicate a larger section to the different subcategories of transfer learning, which will help to understand the structure of the simulation study.

2.1 Definition and Notation of Transfer Learning Tasks

Throughout this document, we will use the notation introduced by Yang (2020).

A *domain* $\mathbb{D}$ consists of two components: the feature space $\mathcal{X}$, the label space $\mathcal{Y}$ with the marginal probability distributions $\mathcal{P}(X)$ and $\mathcal{P}(Y)$. Two domains are considered different if they have different feature spaces or different marginal probability distributions. We will focus on the case of a single source domain, denoted by $\mathbb{D}_S$, and a target domain, denoted by $\mathbb{D}_T$.

A *task* $\mathbb{T}$ consists of two components: the label space $\mathcal{Y}$ and a function $f : \mathcal{X} \to \mathcal{Y}$. The function f is used to make predictions on unseen data drawn from the given domain $\mathbb{D} = \{\mathcal{X}, \mathcal{P}(X)\}$. For $x \in \mathcal{X}, y \in \mathcal{Y}$, we can write $f(x) = P(y|x)$. In a classification task, the feature space can be binary, that is, $\mathcal{Y} = \{0, 1\}$, or have any number of discrete values, which signifies a multi-class classification task.

Definition 1 (Yang (2020)). *(Transfer Learning) Given a source domain $\mathbb{D}_S$ and a learning task $\mathbb{T}_S$, a target domain $\mathbb{D}_T$ and a learning task $\mathbb{T}_T$, where $\mathbb{D}_S \neq \mathbb{D}_T$ or $\mathbb{T}_\mathbb{S} \neq \mathbb{T}_\mathbb{T}$, transfer learning aims to help improve the learning of the target predictive function f_T, using the knowledge of $\mathbb{D}_S$ and $\mathbb{T}_S$.*

This preceding definition is to be contrasted to the traditional machine learning problem, where both $\mathbb{D}_S = \mathbb{D}_T$ and $\mathbb{T}_\mathbb{S} = \mathbb{T}_\mathbb{T}$.

The source domain training data of size n_S is drawn from $\mathbb{D}_S$, and the target domain training data of size n_T is drawn from $\mathbb{D}_T$. Transfer learning is particularly promising when n_T is too small to train a model with satisfactory performance. The data are considered identically and independently distributed (i.i.d) within one domain, but not identically distributed across the two domains.

2.2 Tabular Data in the Context of Transfer Learning

For the purpose of this thesis, the term *tabular data* needs to be distinguished from *textual data* and *image data.* Neural networks achieve impressive results in text and image problems, and transfer learning techniques are applied routinely with neural networks for image and text problems. Hence, images and texts are the prevalent data types in the context of transfer learning. However, as of today, neural networks have not been able to replicate the performance of XGBoost for tasks that involve tabular data. More details on this performance gap can be found in Section 3.3.

The term *tabular data* refers to data that are arranged in rows and columns. Observations or entities are represented by rows. Every row contains the same number of cells, which provide values of properties of the object described by the row. Some of these cells may be empty, i.e. data may be missing. Tabular data is prevalent in many domains such as business, biology, and social sciences, forming the basis of descriptive, predictive, and prescriptive statistical analysis and, likewise, the corresponding machine learning tasks.

Textual data consists of corpora of text, documents, words, etc, and metadata about these corpora. Text data are relevant for machine learning tasks such as text classification, speech recognition, and other natural language processing tasks (NLP). Image data are a pixel-based representation of a photographed, filmed, scanned, or artificially generated object. Image data are relevant for computer vision tasks such as object detection, image classification, motion detection, or handwriting recognition. Image and textual data can sometimes be presented in a tabular format, but are not considered tabular data.

2.3 XGBoost in the Context of Transfer Learning

XGBoost is a machine learning algorithm which iteratively builds an ensemble of weak learners (typically decision trees) to create a strong final model. In this section, we will recapitulate the XGBoost algorithm and draw conceptual parallels with neural networks. Understanding these parallels will help us to translate transfer learning methods from neural networks to XGBoost.

2.3.1 The XGBoost Algorithm

The XGBoost algorithm was introduced by Chen and Guestrin (2016). The algorithm is capable of constructing a model with high predictive power, while effectively managing overfitting. This is achieved by iteratively adding new trees that correct the errors of the ensemble of existing trees, and applying regularization to control the complexity of individual trees.

Throughout this thesis, we use the term XGBoost interchangeably with the algorithm and its publicly available implementation by the DMLC. The algorithm is available in several programming languages including C++, Python, R, Java, Scala and Julia. For the simlation study, we have mostly used the *scikit learn* API. If an objective could not be reached via *scikit learn*, we fell back to the native/functional API.

On the website [1] of the DMLC, the algorithm is introduced as follows:

"XGBoost is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable. It implements machine learning algorithms under the Gradient Boosting framework. XGBoost provides a parallel tree boosting (also known as GBDT, GBM) that solve many data science problems in a fast and accurate way. The same code runs on major distributed environment (Hadoop, SGE, MPI) and can solve problems beyond billions of examples." (sic).

The steps of the algorithm are as follows:

1. **Initialization:** Initially, XGBoost assigns equal predictions for all instances in the data set. Often the average of the target for regression tasks or the log odds for classification tasks is used. This initial model is usually represented as a single leaf.

2. **Construction of Trees:** In each iteration, a new decision tree is constructed. This tree is built in a greedy manner, where binary splits are made on features to partition the data. At each step, the algorithm searches over all features and possible splits to find the one that produces the maximum reduction in the chosen loss function. The specific criteria used for the splits often involve a combination of the improvement of the loss function and a regularization term.

3. **Calculation of Leaf Scores:** Once a tree is constructed, XGBoost calculates the output score for each of its leaves. This score is derived from the objective function that includes both the loss term and a regularization term. The regularization term penalizes the complexity of the individual trees in the ensemble.

4. **Weighted Update of Predictions:** After the leaf scores were calculated, the predictions for each instance in the data set are updated. Rather than fully applying the raw leaf scores, a learning rate (also called shrinkage or step size) is applied. This parameter

[1]https://xgboost.readthedocs.io/en/stable/, 29 August 2023

scales down the contribution of each tree, allowing the model to learn slowly and avoid over-fitting.

5. **Calculation of Residuals:** After updating the predictions, residuals (or pseudo residuals) are calculated as the difference between the updated predictions and the actual target values. These residuals serve as the “new target” for the subsequent tree to be built in the next iteration.

6. **Iterative Building of Trees:** Steps 2 to 5 are repeated for a pre-defined number of iterations, or until the addition of new trees fails to bring a significant reduction in the loss function.

7. **Final Prediction:** The final prediction for a given instance is computed as the sum of the weighted outputs from all of the individual trees constructed during the iterations.

8. **Cross-Validation and Hyperparameter Tuning:** XGBoost includes functionality for k-fold cross-validation to assess the predictive performance of the model. The hyperparameters of the algorithm, including tree depth, learning rate, regularization terms, and number of trees, are tuned to find the most predictive model using this cross-validation approach. This is usually achieved via *GridSearch* or *RandomizedSearch*

2.3.2 Analogies between XGBoost and Neural Networks

While XGboost and the Neural Network algorithms are entirely different from each other, it is still possible to draw some analogies between them. The analogy between XGBoost’s chain of trees and the layers of a neural network is rooted in their structural and functional similarities. XGBoost builds an ensemble of decision trees sequentially, with each tree correcting the errors of its predecessors, forming a chain of trees. This resembles the way how layers of nodes in a neural network sequentially processes input from the previous one and pass it on to the next.

In XGBoost, each tree takes the residual errors from previous trees as inputs and aims to correct them. The final prediction is made by combining all the trees’ outputs. Similarly, information in a neural network passes through the layers, being transformed at each step, with the final layer’s output being the network’s cumulative prediction. This process of sequential refinement is common to both.

Learning from errors is another shared concept. XGBoost’s chain of trees learns by sequentially adding trees that correct the previous ones’ mistakes, an iterative process known as boosting. In neural networks, the back-propagation algorithm adjusts weights based on

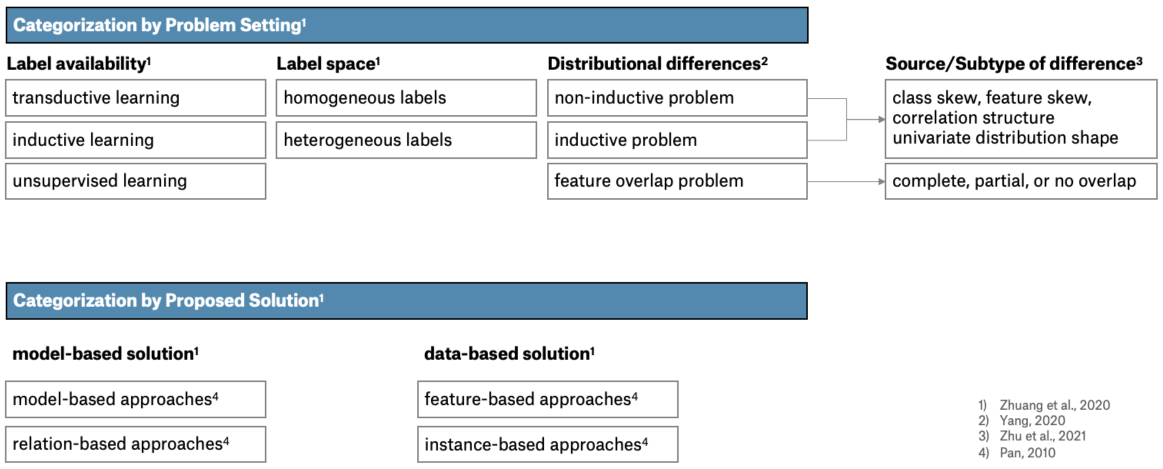
Categorization by Problem Setting[1]
Label availability[1]
transductive learning
inductive learning
unsupervised learning
Label space[1]
homogeneous labels
heterogeneous labels
Distributional differences[2]
non-inductive problem
inductive problem
feature overlap problem
Source/Subtype of difference[3]
class skew, feature skew, correlation structure univariate distribution shape
complete, partial, or no overlap
Categorization by Proposed Solution[1]
model-based solution[1]
model-based approaches[4]
relation-based approaches[4]
data-based solution[1]
feature-based approaches[4]
instance-based approaches[4]
1) Zhuang et al., 2020
2) Yang, 2020
3) Zhu et al., 2021
4) Pan, 2010

Figure 1: Hierarchy of categorization criteria

output errors, learning from mistakes in a way conceptually akin to boosting. Both algorithms have a learning rate parameter to regulate the size of update steps.

Finally, the analogy extends to the ability to fine-tune and control complexity in both systems. In XGBoost, model performance can be tuned by adjusting parameters like the number of trees or their depth. Similarly, a neural network's complexity can be controlled by changing the number of layers, and the number of nodes in each layer.

2.4 Categorization and Analysis of Transfer Learning Problems

The first taxonomy of transfer learning can be found in Pan and Yang (2010). Since its publication, transfer learning has been categorized into feature- , model-, instance- and relation-based approaches. This categorization has proven to be useful for structuring the literature on the topic. However, it does not provide any insights for researchers or practitioners on how to structure a transfer learning project. The survey by Zhuang et al. (2020) provides a useful categorization of transfer learning that does not simply reiterate the categorization by Pan et al. Namely, Zhuang et al. identify different criteria that can be used for categorization on the one hand, but also for a structured problem analysis on the other hand. For this reason, we will follow the *categorization by problem setting* and *by suggested solution*, as proposed by Zhuang et al. Other compatible categorization criteria have been provided in Zhu et al. (2021) and Yang (2020). Figure 1 illustrates the hierarchy of the various categorization criteria.

2.4.1 Categorization by Problem Setting

The first criterion proposed by Zhuang et al. (2020) is the **problem setting**, which has two distinct aspects: label availability and label space.

Label availability distinguishes between inductive, transductive, and unsupervised transfer learning. We speak of *inductive* transfer learning if the labels of the target and source domain are available. *Transductive* transfer learning refers to situations where the labels are available only for the source domain. Thirdly, if both domains are without labels, we are dealing with *unsupervised* learning.

It is noteworthy that this definition of "inductive" in the context of label availability is not consistent with the context of distributional differences between domains (see below).

The *label space* setting can be categorized as either *homogenous* or *heterogenous*, depending on whether values of source and target labels stem from the same space. Zhuang et al. (2020) provide the following two definitions:

Definition 2 (Zhuang et al. (2020)). *(Homogeneous Transfer Learning) Homogeneous transfer learning aims to help improve the learning of the target predictive function $f_T()$ for $\mathbb{D}_T$ using the knowledge of $\mathbb{D}_S$ and $\mathbb{T}_S$, where $\mathcal{X}_S \cap \mathcal{X}_T \neq \emptyset$ and $\mathcal{Y}_S = \mathcal{Y}_T$, but $\mathcal{P}_S(X) \neq \mathcal{P}_T(X)$ or $\mathcal{P}_S(Y|X) \neq \mathcal{P}_T(Y|X)$.*

The counterpart to homogeneous transfer learning is heterogeneous transfer learning:

Definition 3 (Zhuang et al. (2020)). *(Heterogeneous Transfer Learning) Heterogeneous transfer learning aims to help improve the learning of the target predictive function f_T for $\mathbb{D}_T$ using the knowledge of $\mathbb{D}_S$ and $\mathbb{T}_S$, where $\mathcal{X}_S \cap \mathcal{X}_T \neq \emptyset$ and $\mathcal{Y}_S \neq \mathcal{Y}_T$*

Heterogeneous transfer problems are beyond the scope of this thesis. If not stated otherwise, this thesis refers to the homogeneous case of transfer learning with binary labels. The labels are available for source and target data. The distributional differences can be posed as inductive, non-inductive or related to feature overlap. These three problem settings will be represented in the simulation study (see Section 5.2).

On a lower level than Zhuang et al, Yang (2020) further subdivide the homogeneous setting into thee subcategories of distributional differences. However, we prefer to treat **distributional differences** as another main axis of describing the problem setting.

1. In **non-inductive** problems, we have $\mathcal{P}_S(X) \neq \mathcal{P}_T(X)$ and $\mathcal{P}_S(Y|X) = \mathcal{P}_T(Y|X)$.

2. In **inductive** problems, we have $\mathcal{P}_S(X) \neq \mathcal{P}_T(X)$ and $\mathcal{P}_S(Y|X) \neq \mathcal{P}_T(Y|X)$.

3. In a **feature overlap** problem, we have $\mathcal{X}_S \neq \mathcal{X}_T$.

The distributional differences can be further subdivided by they type or source. The literature on data drift provides many options for sub-classifying the source of distributional differences. As an example, we follow the classification of provided in Zhu et al. (2021), because it also provides descriptions of real-world environments that can cause the drift.

In their extensive survey on federated learning on non-i.i.d. data, Zhu et al. provide a fine-grained classification for non-i.i.d. data. Non-i.i.d. data is a shared challenge between federated and transfer learning. The authors apply their categorization scheme to data residing on different clients in a federated learning environment. The goal of federated learning is to train a global model with data that resides on different devices, while protecting data privacy.

The so called horizontal federated learning setting is very similar to the transfer learning setting: In horizontal FL, we consider the problem of data residing on different clients. The clients share the same feature space but have a different sample space. For example, Client 1 stores the age and height of persons A and B, and Client 2 stores the age and height of persons C and D. In transfer learning, we speak of data from different domains. In both contexts, we are concerned about diverging distributions, or non-i.i.d. data. The authors explain that the training data on each client heavily depend on the usage of particular local devices and, therefore, the data distribution of connected clients may be very different from each other.

Zhu et al.'s **sub-types or sources of distributional differences** beautifully translate to the differences between source and target domain in a transfer learning setting. It provides additional insights as it is more fine-grained than the distinction between homogeneous and heterogeneous, or between inductive, non-inductive and feature overlap problems.

1. **Feature distribution skews** indicate the scenarios in which the feature distributions across attributes in each domain/client are different, i.e., $\mathcal{P}_S(X) \neq \mathcal{P}_T(X)$: Three sub-scenarios are possible:
   (a) non-overlapped attribute skew: $\mathcal{X}_S \cap \mathcal{X}_T = \emptyset$
   (b) partially overlapped attribute skew: $\mathcal{X}_S \subset \mathcal{X}_T$ or $\mathcal{X}_T \subset \mathcal{X}_S$
   (c) fully overlapped attribute skew: $\mathcal{X}_S = \mathcal{X}_T$
2. **Label distribution skews** are scenarios where label distributions are different, i.e.,

$\mathcal{P}_S(Y) \neq \mathcal{P}_T(Y)$, and a conditional feature distribution is shared across the domains: $\mathcal{P}_S(Y|X) = \mathcal{P}_T(Y|X)$. The authors distinguish three types of label distribution skew:

(a) Label size imbalance, where each client owns data samples with a fixed number c of label classes

(b) Label distribution imbalance, in which a portion of the instances of label class c is assigned to the domain S or T with probability $p_S(c) \neq p_T(c)$.

(c) Label preference skew, where $\mathcal{P}_S(X) = \mathcal{P}_T(X)$ while $\mathcal{P}_S(Y|X) \neq \mathcal{P}_T(Y|X)$

3. **Attribute and label skew**, where the domains have different labels *and* different features

4. **Temporal skew** occurs in temporal data, spatio-temporal data, and time series data. Unlike attribute skew and label skew, temporal skew refers to the inner correlation of data observations in the time domain. In particular, for time series data, the data distributions $\mathcal{P}_S(X, Y|t)$ and $\mathcal{P}_T(X, Y|t)$ keep changing over time t.

5. **Quantity skew** refers to the number of training data in the domains. It can occur in combination with all of the above situations discussed and is a typical motivator for transfer learning approaches.

After discussing the possible sources of distributional differences, we would like comment on the usefulness of **distance metrics** (Kulback-Leibler divergence, Minimum Mean Discrepancy, etc) for describing a problem setting.

Many researchers have pondered the intuition that transfer learning will be more successful when the domains are "more similar". This led to the assumption that a distance metric could capture the similarity between the domains and help to predict the success of transfer learning approaches. However, even if we found the perfect distance metric, we would still lack a cut-off value to decide whether transfer learning should be applied or not. On the other hand, we have a range of proven statistical methods at our disposal to pinpoint the source of the distributional differences, and we can use any knowledge that we have about the data generating and data collection process to inform our assumptions. This makes the analysis of the source of the distributional differences a much more expedient than calculating distance metrics. The reader is referred to Section 5.2 for examples of how distributional differences can be identified from a simple pair plot.

The results of our simulation study (see Section 6) support our comments on distance metrics. They indicate that the fine-grained distinction of distributional differences is actually decisive for choosing the best transfer learning strategy, while neither distance measures nor the coarse distinction between inductive, non-inductive and feature overlap problems provide a basis for decision making.

2.4.2 Categorization by Proposed Solution

The second criterion suggested by Zhuang et al. (2020) for the categorization of transfer learning is based on the **proposed solution**: This criterion employs the categorization by Pan and Yang (2010) as listed below. Yang (2020) provide details on the assumptions about the relationships between the source and target domains which underlying each category.

1. *Instance-based* methods transfer knowledge by assigning weights to the source instances. These methods implicitly assume that the source and target domains have overlapping features and share the same or similar support. However, instance-based methods may fail if the overlap is insufficient.

2. *Parameter-based* or *model-based* approaches transfer knowledge embedded in a part of the source domain model. These approaches assume that the source and target domains share parameters or hyperparameters of the learning models. The intuition is that a well-trained source model has captured many useful structures that can be transferred to the target domain. This approach is prevalent in deep learning. The models are pre-trained on source data that can be quite different from the target domain. In the second step, the model is adjusted or extended using labeled data from the target domain. The prevalent approaches in deep transfer learning are freezing, progressive learning, and finetuning (see Section 4.1).

3. *Feature-based* approaches transform the original features to obtain a new feature representation. By projecting the data onto a common space, the source data can be reused to train the model for the target domain.

4. Lastly, *relational* approaches aim to transfer logical relationships or rules that define the relations between the entities in the source and target domains.

Zhuang et al. (2020) propose to restructure this four-fold solution-centered categorization into a data-based and a model-based category.

1. The *data-based* category focuses on any adjustments or transformations of data. This roughly covers the feature- and instance-based methods. The data-based strategies

include among others, instance weighting and feature transformation. Their respective objectives can be space adaptation, distribution adaptation, and data property preservation or adjustments.

2. The *model-based* category covers the model- or parameter-based approaches. Model-based strategies include but are not limited to model control, parameter control, model ensembles and deep learning techniques. Objectives of model-based approaches are, among others, prediction making, domain adaptation and generation of pseudo labels.

In the simulation study, we suggest several strategies for transfer learning with XGBoost. The *Combination* method can be considered a data-based or instance-based approach. Furthermore, we tested several model-based approaches - Freezing, Progressive Learning and Finetuning - that can be considered as XGBoost adaptions analogous, established methods for neural networks that go by the same names. See Section 4.1 for details on these methods, and Section 6 for the results of the simulation study.

3 Literature review

The literature review was methodically carried out using Google Scholar, IEEE Xplore, the ACM Digital Library and Springer Link. With keywords like "transfer learning", "machine learning", "deep transfer learning", "XGBoost", "tabular data," and their combinations, we aimed to find the most relevant studies. Additionally, we evaluated the references in these articles to capture other important works. This structured approach ensured a broad and detailed coverage of the topic.

The review charts the development of transfer learning over three overlapping periods and presents the articles in a roughly chronological order:

- 2010 - 2017: Transfer learning with classic machine learning algorithms, differentiated into general survey papers and discussions of tree-based approaches.
- 2017 - 2023: Surge of Deep Transfer Learning, for non-tabular and tabular data
- 2020 - 2023: Emergence of Transfer Learning with XGBoost

3.1 Transfer Learning in Classic Machine Learning

Classic machine learning algorithms comprise any algorithm type, that is not based on deep neural methods such as tree-based methods, Support Vector Machines and kernels. In this section, we will briefly name the most important survey articles of the period from 2010 to 2017 which cover transfer learning for any type of algorithm. Subsequently, we will present the transfer learning approaches that were proposed for the tree-based algorithms preceding XGBoost.

3.1.1 Survey Articles

Lu et al. (2015) examine transfer learning techniques and cluster-related developments in transfer learning. The approaches are classified into the categories neural network-based, Bayes-based and fuzzy transfer learning techniques. Weiss et al. (2016) surveys diverse and representative transfer learning solutions from the years 2011 to 2016, providing high-level descriptions, experimental results, and dedicated sections on homogeneous and heterogeneous transfer learning and negative transfer. A number of real-world applications employing transfer learning is examined and delineated. The authors observe that certain approaches are tailored to specific applications, rendering them inapplicable to others. Pan (2016) survey application of transfer learning techniques to the specific application of collaborative recommendations, which are embedded in many internet systems such as e-commerce and advertisement systems to provide personalized services. Day and Khoshgoftaar (2017) focus their survey on heterogeneous transfer learning. They categorize the techniques explicitly by their label- setting (see 2.4), which leads to the 38 surveyed methods being organized into the a number of categories: “Methods which require limited target labels”, “Methods which require limited target labels and accept unlabeled target instances”, “Methods which require no target labels”, “Methods which require limited target labels and no source labels”, “Methods which require no target or source labels”, as well as a “ pre-processing method for heterogeneous TL”.

3.1.2 Methodological Articles

AdaBoost and Random Forest were the dominant tree-based algorithms for classification problems before XGBoost was released in 2016. Multiple publications were dedicated to applying transfer learning to these algorithms. Table 1 lists the publications and their main contributions.

Dai et al. (2007) proposed **TrAdaBoost**, an extension of AdaBoost, to address inductive transfer learning. TrAdaBoost assumes source and target-domain data to have the same features and labels, but different distributions. After every boosting iteration, the weights of misclassified target instances are increased while the weights of correctly classified target instances are decreased. Dai et al. (2007) also provides a theoretical analysis of TrAdaBoost. TrAdaBoost belongs to the instance-based inductive algorithms with boosting. TrAdaBoost was later extended to cater for multiple source data sets (Yao and Doretto, 2010) and regression tasks (Pardoe and Stone, 2010). The shortcomings of TrAdaBoost and its extensions are discussed in Al-Stouhi and Reddy (2011). The authors suggest another extension, Dynamic-TrAdaBoost, that can overcome some of the limitations. They also provide experimental results using some real-world data sets. A combination of TrAdaBoost and Bagging was applied in Lv et al. (2014). TrAdaBoost was later successfully applied by Zhang et al. (2020) to the prediction of failures in data storage disks. When a new disk model is introduced, there is the availability of training data, making a prime example for transfer learning.

In parallel to AdaBoost, Random Forests were in the focus of transfer learning research and produced different approaches. For example, Rodner and Denzler (2008) and again Rodner and Denzler (2011) propose methods to retrain random forests when the target domain contains a new class that was not present in the source domain. Their method reuses estimated class probabilities in leaf nodes and performs a re-estimation based on a Bayesian framework.

Furthermore, the **TrBagg algorithm** is a simple approach to transfer learning proposed by Kamishima et al. (2009). TrBagg is composed of two stages: Many weak classifiers are first trained on random samples of size n_T that are drawn from $\mathcal{D}_T \cup \mathcal{D}_S$ (learning phase). These classifiers are then filtered based on their usefulness for the target task (filtering phase). Further, the algorithm is equipped with an algorithmic scheme to avoid negative transfer: They introduce a fallback classifier that is trained only on the target data set. This classifier will be used if no useful knowledge is found in the source data. Although they admit that there is no theoretical guarantee of its effectiveness, they claim that this trick worked well in their experiments.

Zhao et al. (2011) try to solve the cross-people activity recognition problem, and propose an algorithm known as TransEMDT (Transfer learning EMbedded Decision Tree) that integrates a decision tree and the k-means clustering algorithm for the adaptation of personalized activity-recognition models.

Sukhija et al. (2016) present a domain adaptation algorithm that learns the mapping between heterogeneous features of different dimensions. The shared label distributions and the relationship between the feature spaces and the label distributions are estimated in a supervised manner using random forests.

The **SER and STRUT algorithms** (Segev et al., 2017) are methods that refine a Random Forest model learned within the source domain using a training set sampled from the target domain. The target is assumed to be a variation of the source. The first algorithm searches greedily for locally optimal modifications of each tree structure by trying to locally expand or reduce the tree around individual nodes (SER: expansion/reduction algorithm). The second algorithm (STRUT: structure transfer) does not modify the structure, but only the parameter thresholds associated with the decision nodes. The authors also propose to combine both methods by considering an ensemble that contains the union of the two forests. Minvielle et al. (2019) study the impact of class imbalance on these algorithms and propose an adaptation for each of them, named SER* and STRUT*. Noteworthy are the conceptual similarities of these algorithms with the revision strategies for XGBoost, which were presented later in Fang et al. (2020).

This concludes the phase of transfer learning research for classic machine learning algorithms. Since its publication in 2016, XGBoost has been the prevalent tree-based algorithm.

However, the interest in transfer learning with classic machine learning algorithms, including XGBoost, was stifled by the emergence of deep transfer learning. Transfer learning with XGBoost re-emerged as a research topic in 2020 after it became apparent that deep learning was not competitive in problems involving tabular data (see Section 3.2).

3.2 Deep Transfer Learning

Starting form 2017, the focus of research shifts notably to *deep* transfer learning, accompanied by a decline in the number of papers about classic methods . Deep transfer learning is defined as any transfer learning endeavor that employs a deep neural network (Tan et al., 2018). In this section, we will present several survey articles on deep transfer learning. We will first provide an overview for non-tabular data, followed by tabular data.

3.2.1 Non-Tabular Data

Tan et al. (2018) provided the first representative survey focusing on deep transfer learning. Deep transfer learning is, similarly to earlier taxonomies, classified into the categories of instances-based, mapping-based, network-based and adversarial-based deep transfer learning. In most practical applications, multiple of these categories are often used in combination to achieve better results.

Some application-specific deep TL surveys followed: In Ball et al. (2017), the focus lies on deep learning in the context of remote sensing problems. Transfer learning is presented as one subtopic of that field. Liu et al. (2019) summarizes research results for sentiment analysis and focuses on the algorithms and applications of transfer learning for sentiment analysis. Sufian et al. (2020) review the status of deep transfer learning in the context of edge computing and IoT. Cheplygina et al. (2019) survey different learning scenarios for semi-supervised learning, multiple instance learning and transfer learning in medical image analysis.

The most extensive survey on deep transfer learning is the book published by Yang (2020), where they provide detailed comparisons of different deep transfer learning strategies across a variety of applications and problem settings.

Niu et al. (2020) presents the state of the art, current trends, applications and open challenges in transfer learning, and put special emphasis on cross-modality transfer learning. Cross-modality transfer learning operates on the assumption that knowledge can be transferred e.g. from text to image data, as can be observed in a human who reads a description about an animal he never encountered before, but will be able to recognize it when he sees it

Publication	Base algorithm	Contribution
Dai et al. (2007)	AdaBoost	Transfer Learning with AdaBoost
Yao and Doretto (2010)	AdaBoost	Multi-Source Transfer Learning
Pardoe and Stone (2010)	AdaBoost	Transfer Learning for Regression Tasks
Al-Stouhi and Reddy (2011)	AdaBoost	Dynamic Updates
Lv et al. (2014)	AdaBoost	Combination with Bagging
Rodner and Denzler (2008)	Random Forests	Re-calibration of trees
Kamishima et al. (2009)	Random Forests	Filtering of classifiers
Rodner and Denzler (2011)	Random Forests	Re-calibration of trees
Sukhija et al. (2016)	Random Forests	Domain Adaptation for Heterogeneous Features
Segev et al. (2017)	Random Forests	Structure adjustments of trees and thresholds
Zhao et al. (2011)	Decision Trees, k-Means	k-Means for selection of source instances for training
Fang et al. (2020)	XGBoost	Revision strategies for trees
Sun et al. (2022)	XGBoost, kernels	Theoretical guarantees in a parallel tree-building model
Bjelogrlic (2023)	XGBoost	Publicly available re-fitting algorithm
Woźnica et al. (2023)	XGBoost	Transferability of hyperparameters
This thesis	XGBoost	Reproducible experiments for comparing TL methods across different scenarios

Table 1: Publications about transfer learning with tree-based algorithms

for the first time. Later, Zhang and Gao (2022) unified the concepts of deep transfer learning and domain adaptation from a broader perspective.

Finally, Iman et al. (2023) analyze over thirty recently applied deep transfer learning studies and identify freezing, finetuning and progressive learning as the most common model-based techniques in deep transfer learning.

3.2.2 Tabular Data

The majority of research on tabular data is currently dedicated to advancing deep neural networks to the accuracy that XGBoost can achieve on tabular data. The reader is referred to Borisov et al. (2021) for an overview of the current state of DNNs for tabular data.

Examples of successfully applied deep learning models for tabular data are rare, but they exist. In his PhD thesis (Farhadi, 2020), the author focuses on deep transfer learning for *structured* healthcare data. The author also presents two case studies where they apply deep transfer learning to tabular breast cancer and diabetes data. They use XGBoost as one of several baseline algorithms to compare the performance of their approach. They claim that their deep TL approach outperforms all baseline algorithms. Unfortunately, they miss to explain how the baseline models were trained. It is not even apparent, whether they were trained on the source data, target data, or both, or if any TL ideas were applied to them.

Despite the recent interest in deep learning for tabular data, it has been shown repeatedly that XGBoost outperforms deep models on tabular data. Specifically, Shwartz-Ziv and Armon (2021) investigated the performance of recently proposed deep models for tabular data sets. In their study, the deep models were weaker on data sets that did not appear in their original publications, and they were weaker than XGBoost, the baseline model. They observed that an ensemble of XGBoost and Deep Learning models performed better than any individual model and the 'non-deep' classical ensemble. The survey by Borisov et al. (2021) comes to the same conclusion.

Grinsztajn et al. (2022) were the first to empirically investigate *why* tree-based models still outperform deep learning models on tabular data. They define 45 data sets from a variety of domains and a benchmarking method that accounts for model fitting and hyperparameter tuning. They show that tree-based models are superior on medium-sized data sets of around 10,000 samples, and also provide the reasons for this:

1. Neural networks are biased to overly smooth solutions. In comparison to trees, they struggle to fit these irregular functions. Some examples of non-smooth patterns which

neural networks fail to learn are provided in the paper.

2. MLP-like architectures are not as robust to uninformative features as tree-based models. Adding and removing uninformative features does not compromise the performance of tree-based models. The authors claim that uninformative feature are more prevalent in tabular data then in, for instance, image or signal data sets. They also empirically show that feature importance calculated via Random Forests is actually good proxy for informativeness.

3. If data are non-invariant to rotation, then so should be the learning procedures. MLPs are rotationally invariant. That means, the accuracy is unchanged if applying a rotation (unitary matrix) to the features of both training and testing set. However, tabular data is not invariant to rotations, which potentially mixes features with very different properties.

In summary, while XGBoost seems to have a head-start in the field of tabular data, deep neural networks are more advanced in the field of transfer learning (for text and image data). It is currently an open question whether XGBoost can leverage transfer learning as successfully as deep neural networks, or whether deep neural networks can catch-up in tabular data.

The competition between DNNs and XGBoost raises the question whether one can learn from the other. However, there is also a tendency to **combine deep transfer learning with XGBoost**. There are two sub-streams of this idea. The first is to train multiple deep transfer learning models and then create an ensemble of them with an XGBoost classifier as the final layer. This is pioneered by Jaiswal et al. (2022) who present a model for diagnosing multiple oral diseases from panoramic radiographs of patient's teeth. They use eight different pre-trained neural networks for image classification. Then, they feed the results of the eight fine-tuned models to eight XGBoost models. These models are then combined in a weighted ensemble model to obtain the final prediction. The second sub-stream is to train one deep transfer learning model and then use its weights, activation function values, or vector-valued model outputs as features for the XGBoost classifier. Examples of this sub-stream are Thi et al. (2022) for the classification of coronary artery diseases in medical images, Guram et al. (2021) for the detection of dementia in medical images, and Tawalbeh et al. (2020) for the detection of aggressive speech in texts.

3.3 Transfer Learning with XGBoost for Tabular Data

This last part of the literature review will focus on transfer learning methods that use XGBoost as their base algorithm. The mentioned algorithms are included in Table 1.

Adaptive Boosting is a finetuning approach suggested by Fang et al. (2020) who propose the TrainXGB wrapper function to train trees with levels based on a base model, by applying different types of revisions. They implement two strategies to conduct the training process, namely OneRound, and MultiRound. The main difference is how to train the source model. The strategy to be applied is chosen via cross-validation.

- OneRound trains a batch of trees in the source domain and transfer them to the target domain in bulk. Then, on the target side, the trees are revised one by one and become the base model for further training. The OneRound strategy captures more patterns in the source domain and maintains a lower transmission cost.

- MultiRound iteratively trains on the source domain and revises on the target domain to grow each source tree. Instead of the batch process, multiple model exchanges take place during the base model training. Then it continues training with only the target data set. The MultiRound strategy fits the target domain better and considers the statistics change after tree revising.

The revisions that are executed during the OneRound or MultiRound procedure are re-weighting, re-splitting and pruning/discounting of rare branches. We discuss these in detail Section 4.1.4.

The standard XGBoost API does not support the revision operations of Adaptive Boosting. The authors have developed a parser to manipulate the XGBoost models through the exported *dump* file. Unfortunately, the authors provide neither their implementation nor their data to replicate their results and investigate the general applicability of their approach.

TransBoost, developed by Sun et al. (2022), is a learning algorithm that combines the merits of tree-based models and kernel methods. The TransBoost algorithm is designed with a parallel tree structure and an efficient weights updating mechanism with theoretical guarantee. We can classify it as a combined instance-based and model-based method. The algorithm builds two parallel boosted tree models that share identical tree structures but different node weights.

The PyPI package **Transferboost** (Bjelogrlic, 2023), which unfortunately does not have an accompanying publication, works on a pre-trained XGBoost model and re-fits the leaf weights according to the target data. The overall structure of the individual trees remains unchanged.

Woźnica et al. (2023) explore **consolidated learning for XGBoost**, a technique of transferring the hyperparameters from one task to another. This approach is tailored for situations where multiple models are built using related data. Rather than optimizing each model individually, consolidated learning focuses on finding a common set of hyperparameters that perform well across many tasks. By using a consistent set of hyperparameters, it eases the tuning process and boosts efficiency. The authors validate their method using the XGBoost algorithm on medical benchmarks, showcasing its potential in various machine learning applications.

An applied example for transfer learning with XGBoost can be found in Cai et al. (2019), where an instance-based transfer learning method is combined with gradient boosting to develop a quantile regression model for wind power forecasting. Based on the spatial cross-correlation characteristic of wind power generations in different zones, the proposed model utilizes wind power generations in correlated zones as the source problems of instance-based transfer learning. By incorporating the training data of source problems into the training process, the proposed model successfully reduces the prediction error of wind power generation in the target zone. To prevent negative transfer, this paper proposes a method that properly assigns weights to data from different source problems in the training process, whereby the weights of related source problems are increased, while those of unrelated ones are reduced.

3.4 Conclusion

After reviewing the literature on transfer learning and its application, as well as its evolution into deep transfer learning and the integration of DNNs and XGBoost on tabular data, several key findings and observations emerge.

Firstly, the literature from 2010 to 2017, i.e. before the release of XGBoost, primarily focused on transfer learning with classic machine learning algorithms. Although these studies pioneered the comprehensive exploration of the underlying mechanisms and theoretical foundations in transfer learning, the limited capacities of the explored algorithms hindered the broader adoption and effective application of transfer learning in real-world scenarios.

The period from 2017 to 2023 witnessed a surge in research on deep transfer learning, primarily in non-tabular domains such as computer vision and natural language processing. While these advancements brought remarkable achievements, there remains a lack of standardization in methodologies, and benchmark data sets. Furthermore, several studies have shown that deep learning is often inferior to XGBoost when the problem involves tabular data.

After 2020, this performance gap has inspired researchers to investigate if transfer learning can be applied to XGBoost, which led to a number of suggested algorithms. However, these have neither been compared to one another nor to simple benchmarks, so that no recommendations can be made regarding which method to apply. Also, the implementations of these algorithms have not been made publicly available so that the results cannot be replicated.

Lately, the integration of DNNs and XGBoost on tabular data has shown promise. However, it is important to critically evaluate the claims made regarding the complementarity and cooperation between these models. The literature lacks rigorous comparative analyses and insights into the pitfalls of such combinations. Guidelines and best practices are needed to ensure the responsible and meaningful use of these combined approaches.

In conclusion, while transfer learning with XGBoost for tabular data, deep transfer learning, and the integration of DNNs and XGBoost hold potential for improving model performance and generalization across domains, critical gaps in the research persist. Future research should focus on ensuring the reliability and reproducibility of findings to support the emergence of proven best practices and standard procedures for common real-world applications.

This thesis closes multiple persisting gaps in the current research. Namely, in Section 5.2, we present a *reproducible* way to create synthetic tabular data which allows us to simulate a variety of scenarios of differing distributions between source and target. Furthermore, in Section 4.1, we suggest new strategies for transfer learning with XGBoost that can be easily implemented with the publicly available XGBoost packages. In Section 6, we also give practical advice on how to select the best strategy.

4 Transfer Learning Strategies for XGBoost

Deep transfer learning is a well-established method to adapt pre-trained neural networks to new tasks. The idea is to leverage acquired knowledge within the model (network) by employing various combinations of pre-trained layers. This set-up is often referred to as *deep* transfer learning and has been successful in problems involving image and text data. According to Iman et al. (2023), the most prevalent deep transfer learning techniques are freezing, fine-tuning, and progressive learning.

Freezing re-utilizes one or more layers from a pre-trained model. Freezing specific layers ensures that their parameters (i.e. the weights) remain unchanged, as inherited from the pre-trained model. On the other hand, fine-tuning initializes the parameters or weights with the pre-trained values, rather than random initialization, for either the entire network or select layers. The parameters may then be updated by the target data. Adding new layers to an existing model is also known as Progressive Learning.

This can be translated to XGBoost as follows: Freezing would preserve a number of trees that have been obtained by training the model on the source data, and replace the remaining trees based on the target data. Progressive learning would translate to adding more trees to an existing model. Finetuning a model would keep the tree structures unchanged, but allow to re-calculate split values and leaf values, and prune or discount branches of trees.

However, the question of whether freezing, progressive learning or finetuning can be applied to XGBoost remains largely unexplored, despite the importance of tabular data in economics, econometrics and many other fields, and XGBoost's outstanding performance on such data. This study aims to bridge that gap by investigating and comparing freezing, progressive learning and finetuning for XGBoost, as analogous techniques have proven successful for neural networks. In this section, we will first describe a data-based approach to transfer learning, followed by the details of how we translate freezing, progressive learning and finetuning to XGBoost.

4.1 Suggested Transfer Learning Strategies

4.1.1 Combination Strategy

The Combination strategy is a simple instance-based method inspired by Daumé III (2007). We merge the labeled data sets of both domains, which turns the problem into a standard machine learning problem with a single domain. As $n_T \ll n_S$, we should prevent that

X_S dilutes any effect that X_T might have. Therefore, we will re-weight the the samples accordingly: The source samples are given the weight n_T/n_S while the target samples keep the weight 1. Additionally, we add a new boolean feature, `is_from_target`, to help the algorithm distinguish between the domains whenever beneficial. This additional feature can potentially help to reduce negative transfer.

4.1.2 Freezing Strategy

The Freezing strategy is based on the idea of freezing the layers of a neural network, as presented in Warchal et al. (2019) for Convolutional Neural Networks (CNN). The authors suggest to train a CNN in the source domain and then freeze the weights of the first layers to leave only the last convolutional block and fully connected layer available for training. Then, the training continues on the so-far unseen target training data set for several epochs with a reduced learning rate. This is adopted for XGBoost as follows:

The Freezing strategy (see Algorithm 1) is designed to tune and train an XGBoost classifier model using two training data sets, followed by evaluating it on a test data set. It does this by first tuning the hyperparameters of the model on the source data set. Then, it aims to find the best point in the training process to switch from the source data set to the target data set by iteratively trying different swap times and re-training the model using the second data set.

The resulting model consists of trees trained on the source data set followed by trees trained on the target data. The model can also decide to replace all source trees, or keep all source trees and not add any target trees. The final model has the same number of estimators as the initial model.

In our simulation, we found that freezing is a very effective strategy in some specific scenarios (see Section 6).

4.1.3 Progressive Learning Strategy

Borisov et al. (2021) state that Progressive Learning is a popular transfer learning method for DNNs. Progressive Learning describes the idea of adding new layers to an existing DNN model, using a reduced learning rate.

Our Progressive Learning strategy (see Algorithm 2 translates this idea to XGBoost as follows: First, this method tunes the hyperparameters of an initial model on the source data. Then, we fit the model allowing early stopping after 20 rounds. We keep the tuned hyperparameters, except for the learning rate (reduced) and number of estimators (increased). With these updated parameters, we continue training the model on the target data, adding more trees until the early stopping criterion is met again.

The simulations showed that Progressive Learning can be advantageous although freezing often performs slightly better (see Section 6 again).

4.1.4 Finetuning Strategy

Finetuning in the context of Neural Networks is understood as using the tuned parameters (node weights) of a trained networks to initialize a new network (instead of random initialization). Then, all weights are allowed to change based on new data. To our knowledge, finetuning is the only DNN-inspired method that has been translated to XGBoost.

One such translation was provided by Fang et al. (2020) who suggest a strategy to revise trees that were constructed on the source data to better fit the target data. They identify three types of revisions and also provide justifications. However, in the light of small target data, they do not mention that all these different revisions might lead to over-fitting the target model while "forgetting" useful aspects we have learned from the source data. Before we present our Finetuning strategy, we will discuss the model revisions suggested by Fang et al., and explain where we will deviate from their approach:

1. **Re-weight:** In the XGBoost algorithm, every tree divides the data into partitions, which are represented by the leaves. The weight of each leaf is an estimate of the probability of the leaf samples belonging to a class.

   The authors argue that, to address label drift (inductive transfer problem, $P_S(Y \mid X) \neq P_T(Y \mid X)$), it is necessary to recalculate the leaf weights. Otherwise, the estimate will be biased. Hence, the weights should be updated based on the target samples falling into the leaf, resulting in an unbiased estimation.

   We would like to point out that re-weighting can easily lead to over-fitting if the number of target samples that fall into the leaf is very small. To address over-fitting, we experiment with adding bias to reduce the variance in the estimate. This bias can be achieved by adding a number of same-leaf samples from the source data to target data before re-weighting. For this reason, we introduced the option of *data augmentation* in our Finetuning strategy.

2. **Re-split:** Each tree that XGBoost constructs is specified by a) the partitioning features

and b) their respective decision thresholds.

The authors argue that, due to scale changes and shape changes in the features (non-inductive transfer problem, i.e. $P_S(X) \neq P_T(X)$, the source thresholds for splitting data are no longer optimal. They propose to traverse every tree top-down and recalculate the split values based on the target data. Furthermore, the authors empirically observe that, for some features, there is no split value with a positive gain in the target data. They attribute this phenomenon to a lack of samples and to pattern differences between source and target. If there is no positive split gain, it is pointless to split and the branch should be pruned.

We would like to reply that the problem of scale and shape change can, at least partly, be addressed by standard feature engineering practices such as scaling, centering and normalizing transformations. Therefore, re-splitting might not be necessary in all cases. Re-splitting is also currently not a functionality that is readily available in the XGBoost library. The authors have implemented a parser to manipulate the splits of a dumped XGBoost model. For these reasons, we have decided to not implement re-splitting in our Finetuning strategy.

3. **Pruning and discounting rare branches:** A branch that had plenty of instances in the source data may become rare in the target data. This can be handled by pruning or discounting the branch. The authors notice that pruning might discard a perfectly valid branch which was just not correctly represented in our target training data. In this case, they propose to discount the branch. Whether pruning or discounting is the best strategy should be determined by cross validation. We would like to point out that it is questionable if rare branches are actually harmful. The partitions created by the trees can have differing importance between source and target data. If the partition is small in the target data, the branch will only rarely have a an influence on the target predictions. If, on the other hand, the partition is large in the target data, but insufficiently represented in the target training data, it could be harmful to remove the branch.

In combination, these revisions aim to optimize the parameters of each tree locally, treating it in isolation from its predecessors. As each tree was initially constructed on the basis of the preceding residual error, it is possible that the sequential nature of the tree ensemble will be destroyed by revising the trees.

Although Fang et al. (2020) propose a valid implementation of a finetuning strategy for XGBoost, we will, for the aforementioned reasons, deviate from their approach and instead suggest Algorithm 3 for XGBoost.

As mentioned above, we assume that using a small set of target data for finetuning might lead to over-fitting. Therefore, we compare the performance of the finetuned model with and without introducing a bias by augmenting the target data with samples from the source data. If activated, data augmentation is done as detailed in Algorithm 4.

While pruning can reduce over-fitting, we argued above that it might just as well be harmful for the model performance if valid branches are cut. Therefore, we will compare the performance of the finetuned model with and without pruning. Algorithm 5 provides the details on this option.

Lastly, while re-weighting leafs appears mandatory for handling inductive transfer problems, we shall also validate if this option is always advisable. The details of re-weighting the leafs are provided in Algorithm 6.

Our algorithm provides the options to activate or deactivate data augmentation, pruning and re-weighting of leaves. This results in the six versions of the finetuning strategy presented in Table 2.

However, all of the six finetuning options are clearly inferior to all our other proposed strategies, and often lead to negative transfer (see again Section 6.

Method	Summary
Target Only	Traditional ML using only target data
Source Only	Traditional ML using only source data
Combination	Traditional ML using source and target data
Freeze	Keep a number of estimators of a Source Only model, retrain the rest
Progressive Learning	Add more estimators to a Source Only model
Finetuning	Adjust tree-parameters of a Source Only model
- P	Prune branches
- R	Re-weight leafs
- PR	Prune and re-weight
- AP	Augment data and prune
- AR	Augment data and re-weight
- APR	Augment data, prune and re-weight

Table 2: List of compared strategies

5 Design of the Simulation study

In this section, we will first explain our assumption of a 2-stage process when solving a transfer learning problem. This assumption justifies our approach to generating synthetic data, which we explain afterwards. We will first describe how we generate the feature data and which parameters are available to create pairs of similar but different domains. Next, we will describe how this feature data is used to create the labels, again detailing the parameters and how they can be used to simulate domain pairs. Last, we discuss our baseline algorithm and how we compare the different transfer strategies against the baseline and against each other.

5.1 Assumption of a Two-Stage Process

Zhuang et al. (2020) suggest that, for a successful transfer learning project, it is not sufficient to use just one transfer learning approach. Instead, they can and should be combined in a thoughtful manner. To achieve this, researchers or practitioners should begin by diligently defining the problem in terms of label setting and space setting. Next, they can choose and iteratively combine data- and model-based transfer techniques.

The combination of data- and model-based techniques can be achieved either in one-stage or in a two-stage process. A one-stage process performs domain adaptation and classifier learning concurrently. In their survey, Weiss et al. (2016) observed that the trend in transfer learning solutions leans towards a one-stage process, but this trend apparently did not stabilize. Instead, a two-stage process seems to be more prevalent nowadays. The two-stage process separates the tasks of data transfer and model transfer:

First, in the *data transfer stage*, the objective is to align the spaces and distributions of source and target features as well as assigning weights to the instances. Note that in some practical scenarios, there may be no source data but only a source model. In this case, the data transfer phase may be skipped. The data transfer phase could also be considered as a more thoughtful version of the data selection, feature engineering and feature selection phase in traditional machine learning projects. Hence, the data transfer phase can include transformations like scaling and centering, generating pseudo-labels, inferring missing data, augmenting the feature space, or calculating a sort-of propensity score estimate for instance weighting, all with the assumed domain discrepancy between source and target in mind.

Second, in the *model transfer stage*, the objective is to train, retrain, fine-tune or combine new and existing models. This phase can be seen as an extension of the usual training phase in traditional machine learning projects. The prevalent algorithms for this phase are currently XGBoost and neural networks. A detailed description of the current state of the art for transfer learning with XGBoost and neural networks can be found in Section 3.2 and Section 3.3.

The proposed two-stage process elucidates that different algorithms may be involved in each phase. On the other hand, a specific algorithm may also oscillate between the two phases automatically. For example, TrAdaBoost (see 3.1) iteratively learns a classifier and re-weights the training data. Depending on the problem setting, the researcher or practitioner needs to define their objective and then select an appropriate strategy for the data transfer and the model transfer, respectively. Note also, that the choices are interdependent and that an iterative approach must be applied to achieve a good combination of data and model transfer methods.

Building on Zhuang et al. (2020), we will assume a two-stage process with a data transfer phase and a model transfer phase. This simulation study will focus on the use of XGBoost in the *model* transfer phase. To this aim, we will treat the *data* transfer phase as a black box which provides us with source and target data that are as closely aligned as possible. We will assume that this black box provides us with the optimal solution and no further efforts will be made in the fields of domain adaption, distribution correction, missing value imputation, etc. We must, however, assume that the result of the data transfer phase is not perfect. Otherwise, the problem would translate into a traditional machine learning problem. Hence, the data will be sampled from two "similar but different" distributions. In Section 5.2, we explain how such data can be generated synthetically.

5.2 Creating Synthetic Tabular Data for a Simulation Study

Synthetic data is pivotal in simulation studies, particularly when introducing new algorithms. It ensures reproducibility, a quality often missed in contemporary research, as identical scenarios can be reliably recreated. Additionally, by generating synthetic data, we gain control over the parameters. This enables a deeper understanding of the underlying mechanisms, an insight that a singular real-world data set might not provide.

For our simulation study, we create synthetic data from 60 source-target domain pairs. All domains employ the same data generation process but differ in the parameters of that process. A source-target domain pair is identical in all but one parameter.

In this section, we will first describe the data-generation process, its parameters and how the parameters can be used to simulate non-inductive, inductive, and feature overlap problems. Table 3 provides an overview of the parameters.

This section draws inspiration from Zhuang et al. (2020) (label setting, data-based and model-based transfer), Zhu et al. (2021) (categories of non-i.i.d. data) and Grinsztajn et al. (2022) (data set inclusion criteria for benchmarking algorithms on tabular data).

5.2.1 Generation of Feature Data

The number of base variables, M, determines the size of our feature matrix: $\boldsymbol{X} = (X_1, ..., X_M)$. The M^2 interaction variables are all pairs $(X_i, X_j), i, j \in \{1, ...M\}$, resulting from the M base variables. We chose $M = 30$ in our simulation study.

The matrix $\boldsymbol{\Sigma}$ will govern the correlation structure of the features. The matrix $\boldsymbol{\Sigma}$ is an M-dimensional, randomly generated correlation matrix. The entries of $\boldsymbol{\Sigma}$ are drawn from $U(-0.5, 0.5)$ if we want to allow positive correlations to change to negative in the other domain. If we want to enforce same signs for all correlations, we will draw from $U(0, 0.5)$. This will allow us to examine the effect of differing correlation structures on the performance of transfer learning methods with XGBoost.

Figure 2 shows samples drawn from domains that vary in $\boldsymbol{\Sigma}$, but are otherwise identical.

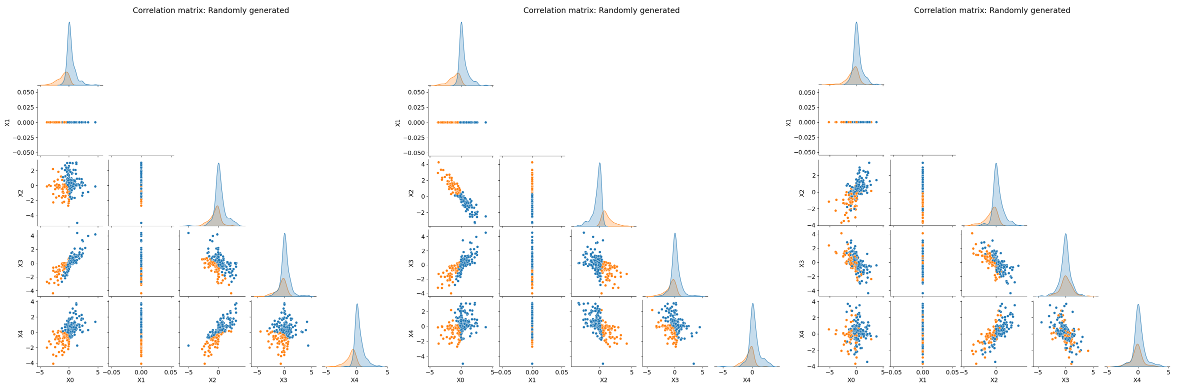

Figure 2: Samples from domains with differing correlation structures

Note how the sign of the correlation between X_2 and X_4 changed from the first to the second domain, while the univariate distribution shapes are hardly effected.

The exponent t governs the univariate distribution of the features. We draw t form $U(0.5, 2)$. If $t < 1$, the univariate distributions will be more pointy then the normal distribution. If $t > 1$, the univariate distributions will be more flat or even bimodal.

We obtain $\boldsymbol{X}$ in a three-step process: First, we draw a random sample (a_{mn}) from $N(\mathbf{0}, \mathbf{\Sigma})$. Second, we raise the sampled values to the power of t while keeping the original sign of the variable: $b_{mn} = sgn(a_{mn}) \mid a_{mn} \mid^t$. Third, we re-scale using the observed standard deviation to achieve a variance of 1: $x_{mn} = b_{mn}/\sigma_m$. This results in a feature matrix that still exhibits a similar correlation structure but is not normally distributed.

Figure 3 illustrates the resulting features for different values of t. Note how the shapes of the univariate distributions change while the correlation structure is mostly preserved.

Varying $\mathbf{\Sigma}$ and t between source and target data while keeping the remaining parameters fixed will enable us to simulate **non-inductive transfer problems**. In summary, $\Sigma_T \neq \Sigma_S$ and $t_T = t_S$ yields differing correlation and resembling univariate distributions; $t_T \neq t_S$ and $\Sigma_T = \Sigma_S$ yields differing univariate distributions and resembling correlation. The inductive transfer problems will be represented by the scenarios *UnivShape*, *CorrSame*, and *CorrAny* (see Table 6).

Another aspect of feature generation is **feature overlap**. The features of two domains can be fully overlapped, non-overlapped, or partially overlapped. In a fully overlapped feature space, the features of source and target domains are the same: $\mathcal{X}_S = \mathcal{X}_T$. If no other variation is introduced, this case represents a traditional machine learning problem. In a setting with

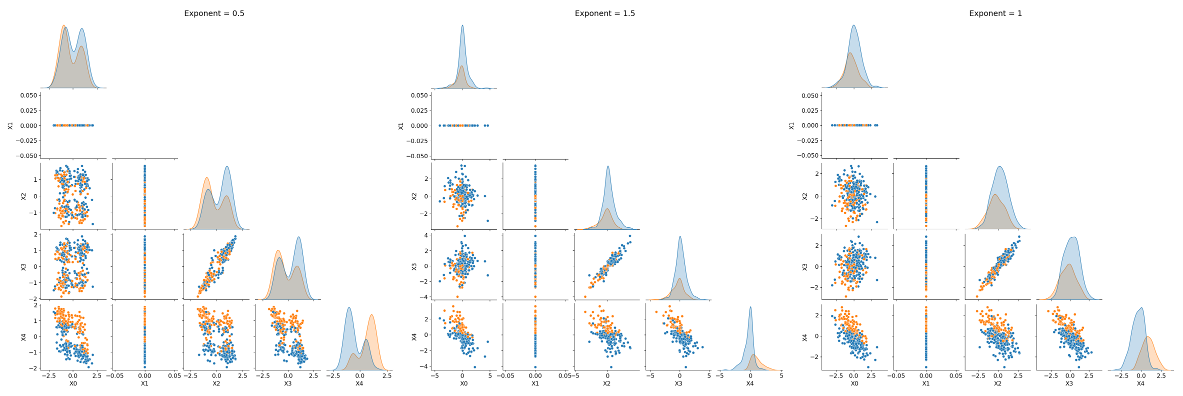

Figure 3: Samples from domains with differing exponents

non-overlapped feature spaces, the source and target features are mutually exclusive, i.e. $\mathcal{X}_S \cap \mathcal{X}_T = \emptyset$. This case can be considered as a heterogeneous or cross-modal transfer problem and will not be considered in this study. Lastly, in partially overlapped feature spaces, some of the features are shared between source and target domain, while others exist only in one of the domains: In our simulation, a feature overlap problem is characterized by $\mathcal{X}_S \cap \mathcal{X}_T \neq \emptyset$ but $\mathcal{X}_S \neq \mathcal{X}_T$.

We will simulate the feature overlap problems by randomly censoring 6 of the 30 features of the source and target domains after the labels were generated. This means, the feature will be rendered uninformative by setting all the values to 0. This covers partially overlapped feature spaces where $\mathcal{X}_S \subset \mathcal{X}_T$, $\mathcal{X}_S \supset \mathcal{X}_T$, or neither $\mathcal{X}_S \subset \mathcal{X}_T$ nor $\mathcal{X}_S \supset \mathcal{X}_T$.

Figure 4 shows exemplary feature overlap situations. The first domain provides the feature X_0, X_1, X_2, the second provides X_0, X_1, X_4 and the last provides X_1, X_2, X_4, while the others have been rendered uninformative. Note how the distributions are otherwise identical.

5.2.2 Generation of Label Data

We will simulate a binary classification problem, i.e. the label spaces of source and target data are assumed to be homogeneous: $\mathcal{Y}_S = \mathcal{Y}_D = \{0, 1\}$. The binary labels Y_S and Y_T are produced by a "reverse logistic regression" model.

First, we randomly generate the coefficients for the M base variables, $\boldsymbol{\beta}$, and for the interaction variables, $\boldsymbol{\gamma}$. We set $P(\beta_i = 0) = 0.8$ and $P(\gamma_{(i,j)} = 0) = 0.5$ and otherwise draw β and γ from $U(-10, 10)$ if we allow the sign of the coefficient to change between domains. Otherwise, we draw β and γ from $U(0, 10)$.

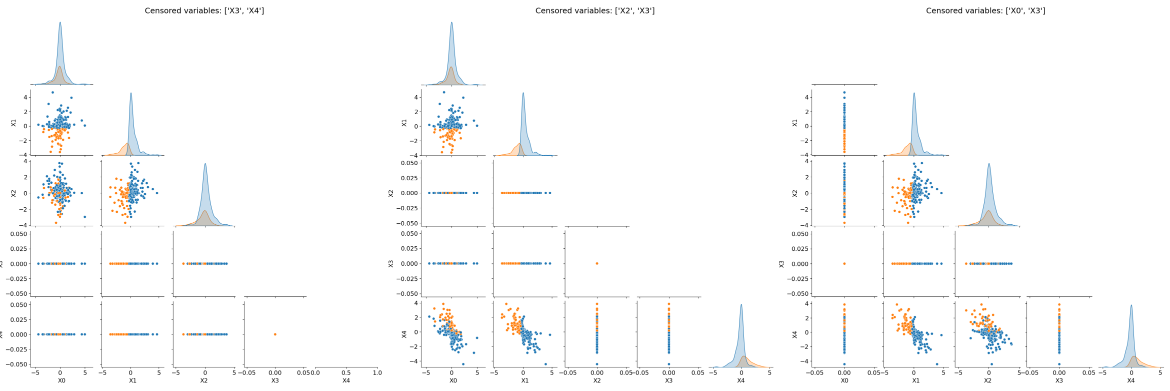

Figure 4: Samples from domains with differing feature spaces

We use the aggregation function $g : \mathcal{X}_S \to \mathbb{R}$

$$g(\boldsymbol{X}) = \sum_{k \in \{1 \ldots M\}} \beta_k X_k + \sum_{(i,j) \in \{1 \ldots M\}^2} \gamma_{(i,j)} X_i X_j,$$

and the sigmoid link function $h : \mathbb{R} \to (0, 1)$:

$$h(z) = \frac{1}{1 - e^z}$$

to model the class probabilities. The output of the sigmoid will be used as the probability of a Bernoulli-type randomized sampling function $r : (0, 1) \to \{0, 1\}$, which will randomly return 0 or 1 as class labels. The bias $b \in [-0.2, 0.2]$ will allow us to manipulate the class ratio:

$$P(r(z) = 1) = \max\ \{\min\{h(z) + b; 1\}, 0\}$$

Finally, we obtain:

$$Y(X) = r \circ h \circ g(X).$$

Inductive transfer problems are characterized by $P_S(Y) \neq P_T(Y)$ and $P_S(Y|X) \neq P_T(Y|X)$. In the simulation, we will keep $P_S(X) = P_T(X)$. The difference in the conditional probabilities will be introduced via the aggregation and sampling function.

We can impose $g_S() \neq g_T()$ by varying β_k and $\gamma_{(i,j)}$, which will change the conditional probability mechanism. Figure 5 shows samples from domains with different coefficients. Observe that the feature distribution is identical while the class allocation is different between the three domains. Also, a change in the sign of a coefficient can even reverse class allocation as seen in X_2 of the first and second domain.

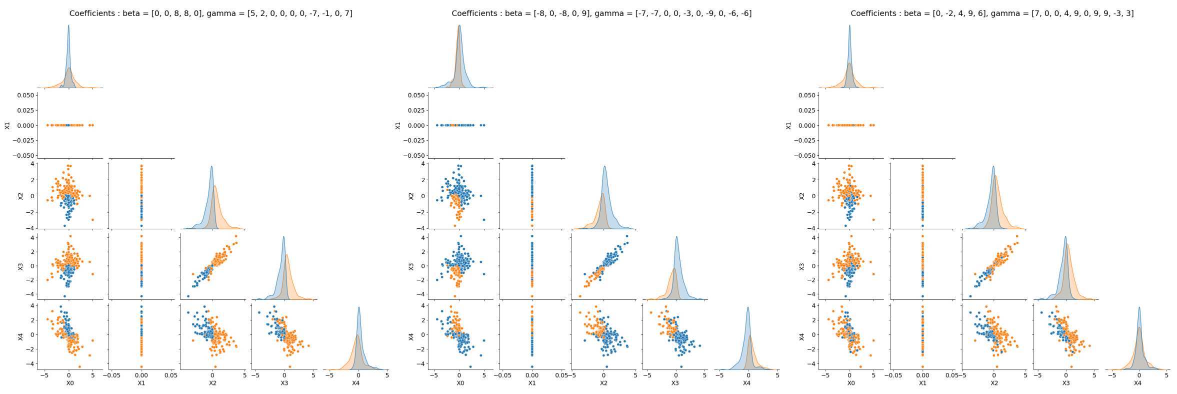

Figure 5: Samples from domains with differing coefficients

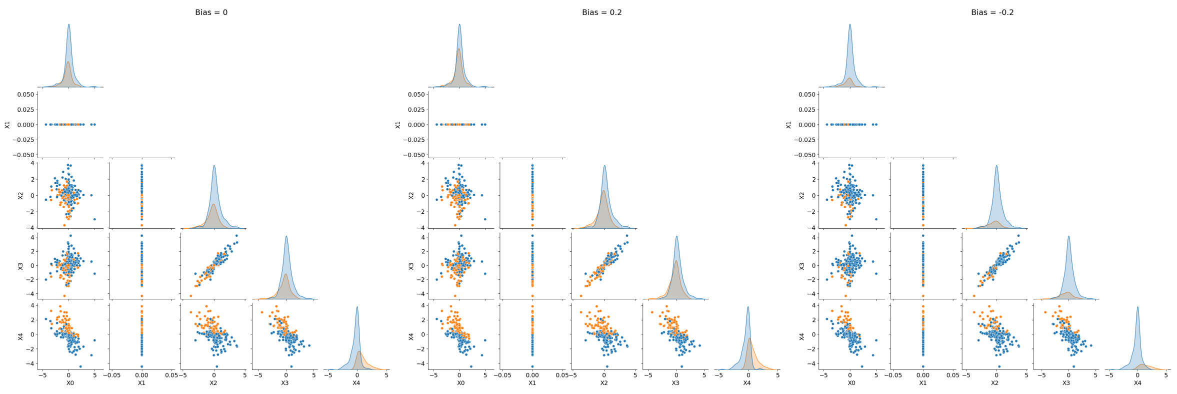

Figure 6: Samples from domains with differing class ratio

Furthermore, we can achieve $r_S \neq r_T$ by imposing $b_S \neq b_T$, which will change the class ratio, as can be seen in Figure 6.

The inductive problems will be represented by the scenarios *ClassRatio*, *CoeffSame*, and *CoeffAny* (see Table 6).

5.3 Parameters of the Simulation Study

5.3.1 Data Generation Parameters

Grinsztajn et al. (2022) define guardrails for data sets for benchmarking ML algorithms for tabular data and removing side-issues that are worth studying separately. Table 4 lists how our data-generating process implements these criteria. We have checked that the simulation

Transfer problem	Distributional differences	Source of difference	DGP Parameters
Non-inductive	$P_S(Y) \neq P_T(Y)$ $P_S(X) = P_T(X)$ $P_S(Y \mid X) = P_T(Y \mid X)$ $\mathcal{X}_S = \mathcal{X}_T$	Univariate distribution of X_i	t
		Correlation structure of X, same or any sign	Σ
Inductive	$P_S(Y) \neq P_T(Y)$ $P_S(X) = P_T(X)$ $P_S(Y \mid X) \neq P_T(Y \mid X)$ $\mathcal{X}_S = \mathcal{X}_T$	Coefficients of base features and coefficients of interactions, same or any sign	$\boldsymbol{\beta}, \boldsymbol{\gamma}$
		Class ratio	b
Feature overlap	$P_S(Y) = P_T(Y)$ $P_S(X) = P_T(X)$ $P_S(Y \mid X) = P_T(Y \mid X)$ $\mathcal{X}_S \neq \mathcal{X}_T$	Presence of different features in the domains	random censoring

Table 3: DGP parameters for experimental scenarios

results are not sensitive to exact parameter choices. This allows us to vary the parameters and simulate non-inductive, inductive, and feature overlap problems as explained in Section 5.2.

However, we would like to draw attention to the **sample size parameters** because they are central to both the general setting of transfer problems and to the performance that can be achieved by an ML algorithm on our synthetic data.

We assume that some source labels and target labels are available, but at a different price. Labels for the source data are cheap and readily available. The labeled source data is sufficient to obtain a "well-trained" source model. On the other hand, the available amount of target data will only be capable of generating an "insufficiently trained" target model. Neither additional labeled nor unlabeled target data can be obtained.

Our DGP creates random domains from which we can sample an arbitrary amount of data. Figure 7 illustrates the improvement of AUC when the sample size increases. We observe the same pattern across all our random domains: The AUC improves quickly up to a size of around 500 samples. Then, the improvement slows down and plateaus at around 2000 samples. More samples lead only to marginal improvements. Figure 7 also provides the normalized gain in ROC-AUC (NGL, see Section 5.4) of increasing the sample size from 200 to 2000. The NGL ranges from 0.59 to 0.8.

With this observation in mind, we have fixed the training sample sizes at 200 for the target data, and *2000 for the source* data. In order to obtain reliable estimates of the AUC on test data, we have set the sample size to 500 for the test data set - even though, in reality, one would probably dedicate more data to training than to testing.

5.3.2 Strategy Execution Parameters

Parameters of the Suggested Strategies. Our suggested strategies introduce several new parameters which might require additional tuning for optimal results.

For the Freezing strategies, we need to specify the number of trees to freeze. We chose to let the algorithm try different fractions and pick the best one. Specifically, we used the fractions $[0, 0.2, 0.4, 0.6, 0.8, 1]$. It could be advantageous to use a finer grid here, but our results show that the Freeze strategy performs well with these parameters (see Section 6).

For the Progressive Learning strategies, we need to specify three parameters. We decided to set the number of early stopping rounds to 20. The learning rate for the newly added trees is reduced by a factor of 0.75. The maximum number of trees increases the tuned maximum

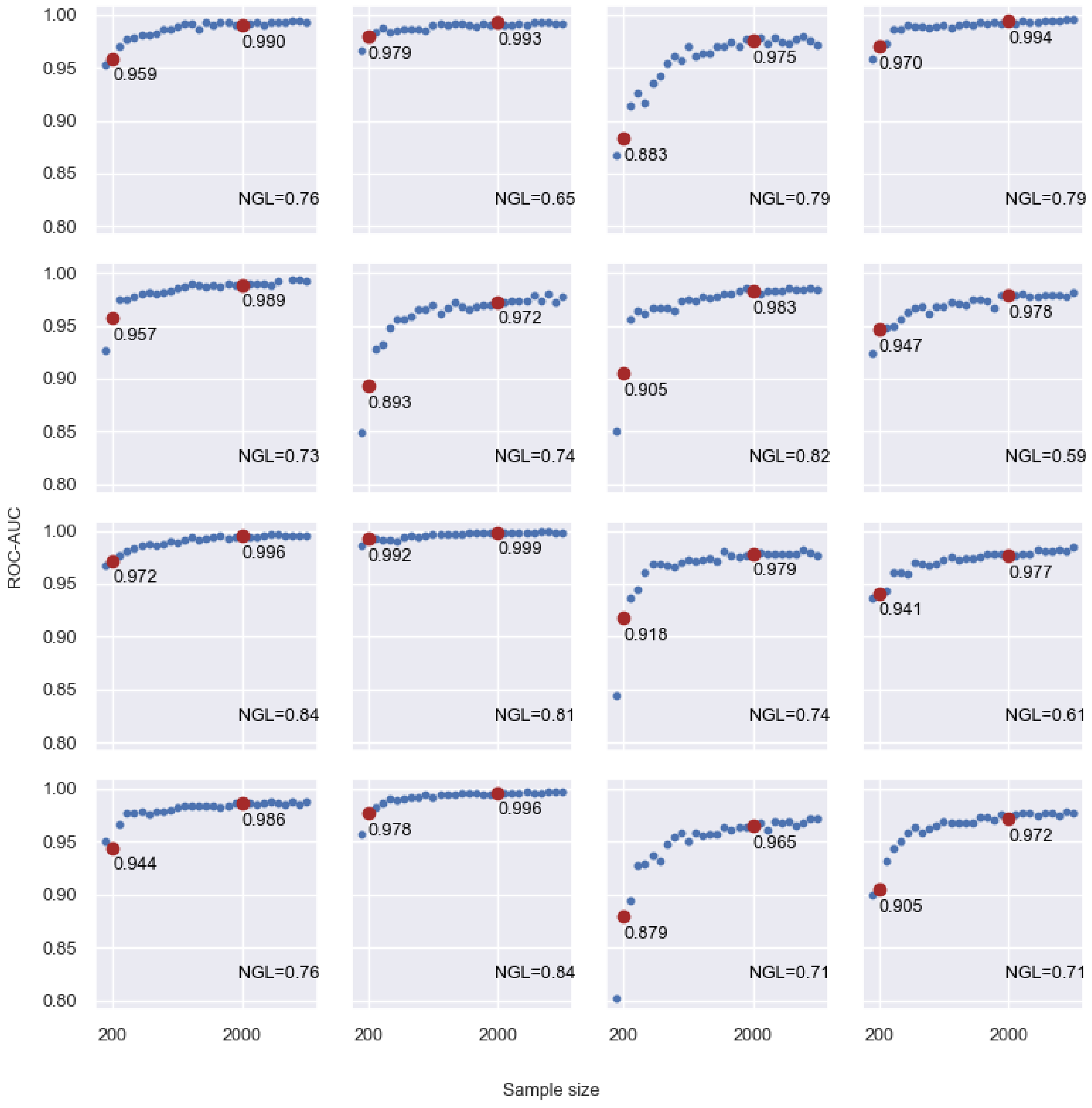

Figure 7: Effect of sample size on the AUC, and NGL of 200 vs. 2000 samples

Criterion	Comment	Enforced by
I.i.d data	This excludes stream-like or time-series data.	i.i.d. within source and within target, but not identical between source and target
No high dimensionality	The data sets should have at least 4 features , but the ratio of features to samples should be around 1/10.	$M = 30$, $n_S = 2000, n_T = 200$.
Medium-sized training data sets	less then 10,000 samples	$n_S = 2000, n_T = 200$
Not too easy	XGBoost cannot routinely achieve an AUC above 0.95	Overall design of the DGP at $n_T = 200$
Not deterministic	The label should not be deterministic function of the data.	Bernoulli sampling function r
No missing data	XGBoost has been shown to handle missing data well.	Within a domain, features are either complete or censored. We randomly censor 6 out of 30 features.
Balanced classes	Class ratio between 0.5 and 2.	$-0.3 < b < 0.3$
Low cardinality in categorical features	-	No categorical features, except for the binary `is_from_target` in the *Combination* strategy
High cardinality in continuous features	-	X drawn from $N(0, \mathbf{\Sigma})$, continuous transformations

Table 4: DGP parameter choices

number of trees of the source model by a factor of 2. Tuning both factors might be beneficial. However, we refrained from tuning them because the Combination and Freezing strategies were superior to Progressive Learning.

A tuneable parameter of the Finetuning strategy is the minimum required number of target samples in a leaf when augmenting the target training data. We set this parameter to 20. We refrained from tuning this parameter due to the obvious superiority of other strategies.

Hyperparameters. The comparison of different transfer strategies can only be fair if they receive the same amount of tuning. Therefore, whenever we train a basic XGBoost model, we use the same methodology:

The hyperparameters of all basic XGBoost models are tuned using the `RandomSearch` functionality provided by *sklearn*. We follow the recommendation of Bentéjac et al. (2021) who showed in their empirical study which parameters of XGBoost must be tuned, and which values can be set to good defaults to achieve the best performance. Specifically, the `RandomSearch` is configured as follows:

- Scoring: ROC-AUC (fixed)
- Number of cross validations: 2 (fixed)
- Number of search iterations: 70 (fixed)
- Number of estimators: $[50, 100, 150, 200]$
- Learning rate: $U(0.025, 0.03)$
- Gamma: $U(0.1, 0.5)$
- Maximum depth of tree: $[2, 3, 5, 7, 10, 30]$
- Column sample by level: 0.5 (fixed)
- Subsampling rate: 0.75 (fixed)

5.4 Comparing transfer learning strategies

5.4.1 Baselines

Any suggested transfer learning strategy should beat a simple baseline algorithm. We will compare all strategies against the following baselines:

The *Target only* baseline trains a single model only on the target data. This model ignores any information that can be obtained from the source data. It might suffer from overfitting due to the high-variance estimates obtained from a small dataset. This strategy corresponds to a traditional machine learning problem with a single, small dataset.

The *Source Only* baseline ignores the target data and trains a single model only on the source data. This model is the applied to make predictions on the target data. This strategy is equivalent to re-purposing a traditional machine model (that was trained on sufficiently large dataset) under the presence of data drift. It ignores any information that can be obtained from the target data. However, it can give us an idea of how close the target distribution is to the source distribution. If the Source Only baseline performs well, we can conclude that the benefits of a larger training set outweigh the risks of domain differences.

5.4.2 Normalized Gain/Loss of AUC

The Receiver Operating Characteristic Area Under the Curve (ROC-AUC) is a performance metric used in machine learning to evaluate the discriminative ability of a binary classification model. It is computed by plotting a curve that represents the true positive rate (sensitivity) against the false positive rate (1-specificity) at various threshold settings, and calculating the area under this curve (AUC). A ROC-AUC score of 0.5 suggests no discrimination (equivalent to random guessing), while a score of 1.0 indicates perfect discrimination. Typically, a higher AUC indicates a more effective model in distinguishing between positive and negative classes. The AUC can be increased by using a better model or by increasing the sample size. However, the AUC will typically exhibit diminishing marginal improvements. This means, that it will be easier to achieve an increase, for instance, from 0.85 to 0.90 than from 0.90 to 0.95.

To achieve comparability of strategies across models, sample sizes and domains, we will use the normalized gain or loss of the ROC-AUC. Let a be the AUC achieved by a baseline model and b the AUC achieved by the model to be compared. Then, we define the normalized gain or loss (NGL) as

a	b	NGL
0.80	0.85	0.25
0.90	0.95	0.50
0.85	0.80	-0.17
0.95	0.90	-0.13

Table 5: Examples of normalized gain/loss

$$NGL(a,b) = \begin{cases} \frac{b-a}{1-a}, & \text{if } a \leq b \\ \frac{b-a}{a-0.5}, & \text{if } a > b. \end{cases}$$

A negative NGL identifies cases with negative transfer, i.e. the compared model cannot beat the baseline and the application of the transfer learning strategy is actually harmful. Table 5 provides some examples of NGL for different values of a and b. We will use the NGL to compare our suggested methods (= b) against the Target Only baseline (= a).

Problem setting	Scenario Name	Source of distributional differences between domains
Non-inductive	UnivShape	Univariate distribution shape
	CorrSame	Correlations with same sign
	CorrAny	Correlations with any sign
Inductive	ClassRatio	Class ratio
	CoeffSame	Coefficients with same sign
	CoeffAny	Coefficients with any sign
Feature overlap	FeatOverlap	Availability of features

Table 6: List of scenarios

6 Results

This section aims to evaluate the performance of four distinct transfer learning strategies: Combination, Freezing, Progressive Learning, and Fine-tuning. The strategies are summarized in Table 2. Each of these strategies has been systematically applied across seven different scenarios to investigate their effectiveness. The scenarios are listed again in Table 6.

The primary metric for performance assessment is the normalized gain/loss in the ROC-AUC (NGL) compared to the Target Only baseline (see Section 5.4).

The objective is to identify the optimal transfer learning strategy across a range of scenarios, thereby providing insights that could facilitate the development of more effective models for tabular data.

The section is structured to first present the performance of each individual strategy across all scenarios, followed by a comparative analysis that evaluates the strategies in relation to each other. Subsequent discussions will delve into the role of distribution differences and summarize the key findings of this study.

6.1 Results by Transfer Learning Strategy

6.1.1 Source Only Baseline

Firstly, we present the results of the baseline strategy, Source Only. This will help us understand if a positive NGL is due to the transfer learning strategy or can be attributed to the larger sample size of the source training set. Figure 8 shows the NGL of the Source Only stratgy against the Target Only strategy. Every dot represents one source-target domain pair. We observe that the NGL is positive for the UnivShape and the ClassRatio scenario. In

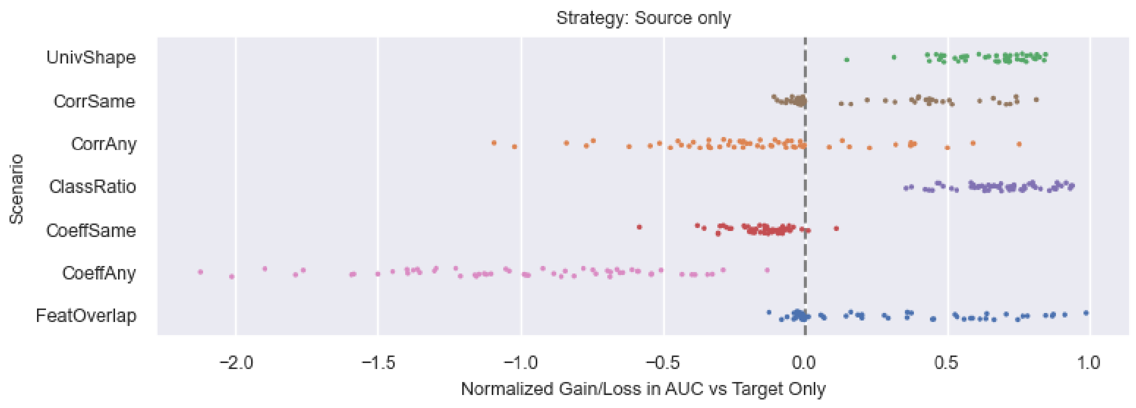

Figure 8: NGL for Source Only strategy

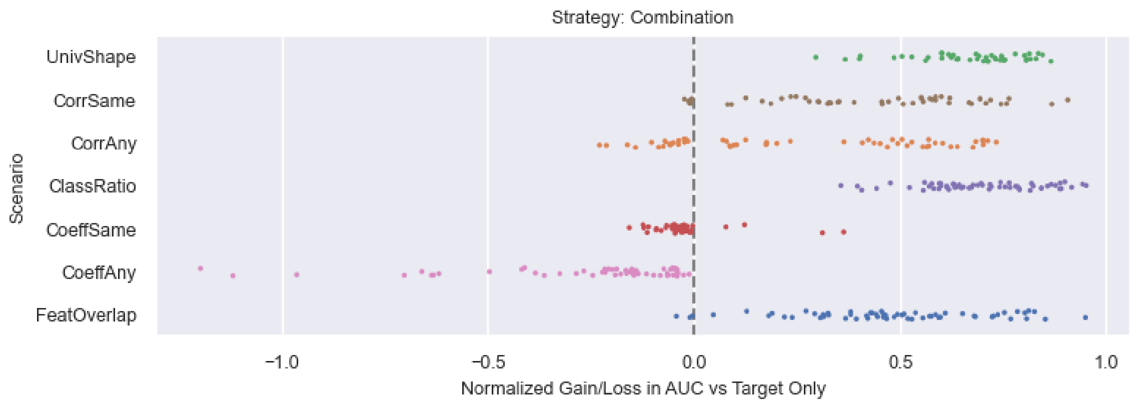

Figure 9: NGL for Combination strategy

the CorrSame and FeatureOverlap scenarios, the NGL is mostly positive, but negative NGL cannot be ruled out. In the CorrAny, CoeffSame and CoeffAny scenarios negative NGL is very likely.

We conclude that the Source Only strategy can be viable under the assumption that the domains vary only in class ratio or univariate distribution shape. If different correlation structures or coefficients are at play, it is not recommended to use the Source Only strategy.

6.1.2 Combination Strategy

Figure 9 shows that the Combination strategy performs similarly to the Source Only strategy. However, we note that negative NGL occurs less frequently and tends to be smaller.

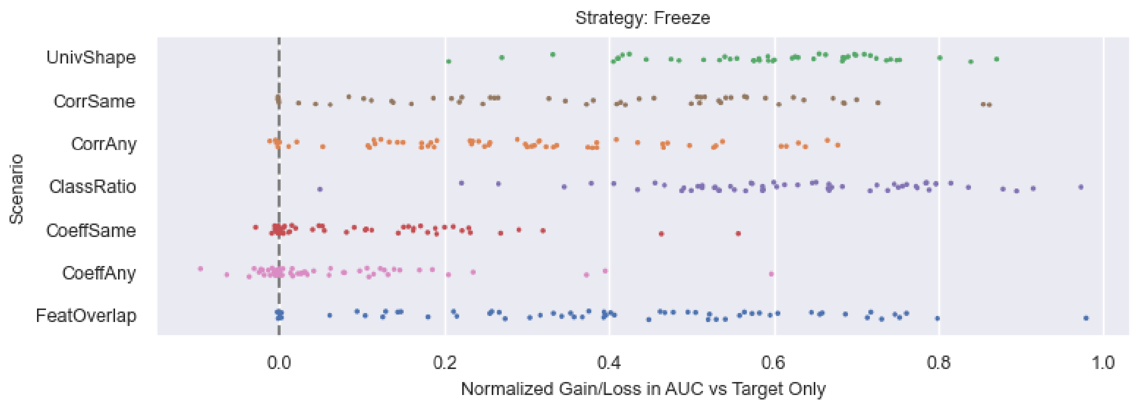

Figure 10: NGL for Freeze strategy

This improvement probably stems from the following reasons: First, the `weights` and the `is_from_target` feature help the algorithm to form meaningful partitions and focus on the more important target samples, thereby reducing the impact of contradictory patterns. Second, in the FeatOverlap scenario, this strategy can deal with features that are missing completely in one domain just as it would with only partially missing features in a regular ML model. XGBoost has been shown to handle such cases of missing values well.

Hence, if source data - and not only a source model - is available, it is always beneficial to train a new model instead of re-purposing the existing model. This is a low-effort way to reduce the probability and impact of negative transfer.

6.1.3 Freezing Strategy

The Freezing strategy is superior to all other strategies, as seen in Figure 11. Even in the CorrAny, CoeffSame and CoeffAny scenarios, negative NGL occurs rarely and, moreover, remains small. We attribute this to the additional tuning parameter `swap_time`. This parameter allows the model to include as many trees from the source domain as is beneficial. By allowing the model to include none of the source trees, we can achieve a performance on par with the Target Only model. A small negative NGL can still occur because the tuning of the remaining hyperparameters is done only on the source data and may therefor be slightly sub-optimal compared to the Target Only model. It could hence be beneficial to re-tune the remaining hyperparameters.

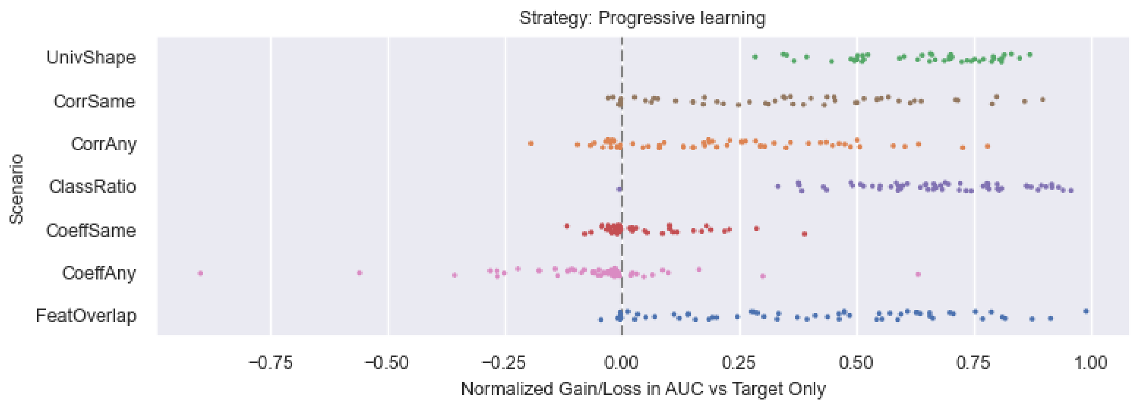

Figure 11: NGL for Progressive Learning strategy

6.1.4 Progressive Learning Strategy

The Progressive Learning strategy exhibits more frequent and larger negative NGLs than the Freeze strategy, while still achieving mostly positive results. We attribute this to the fact that the model is forced to keep all trees constructed from the source model, whether they are beneficial or not. However,the strategy may benefit from the possibility of adding as many new target trees as necessary.

6.1.5 Finetuning Strategy

The Finetuning strategy was implemented with options to activate data augmentation (A), pruning (P) and re-weighting (R). We have simulated all combinations of these options. However, no combination was superior to the others. Figure 12 shows that the Finetuning strategy is arguably inferior to the other strategies. Negative transfer is an even larger problem than in the Source Strategy. Also, the Finetuning strategy rarely achieves an NGL above 0.5 - a value which we observe frequently with the other strategies. Hence, we cannot recommend to use Finetuning if source data is available to train a new model.

6.2 Summary of Key Findings

Finally, we would like to decide which strategy should be employed in each scenario. Figure 13 depicts the probabilities of a strategy performing better than another strategy in any given scenario. For example, in the UnivShape scenario, the probability that the Combination strategy has a better NGL than the Freeze strategy is 0.74 across all experiments. From the

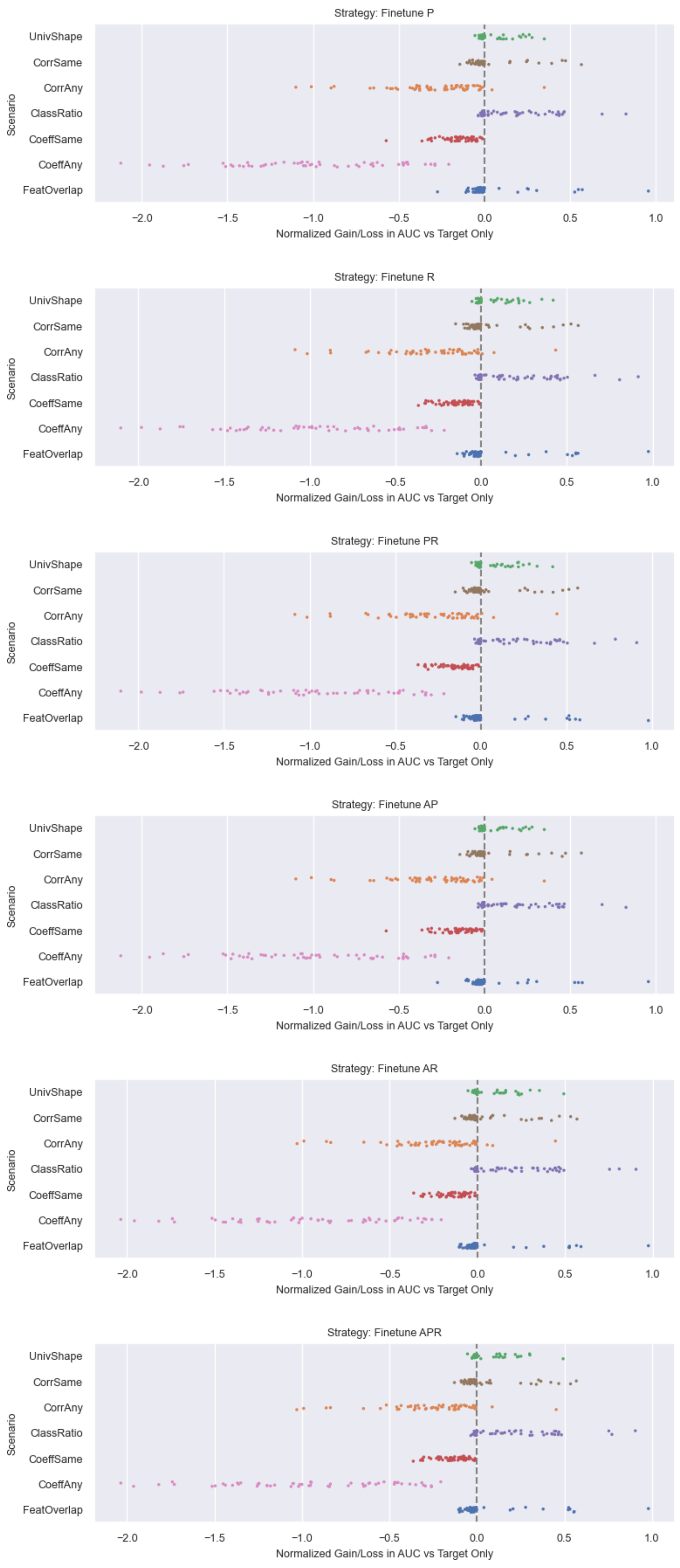

Figure 12: NGL for Fine-tuning strategy

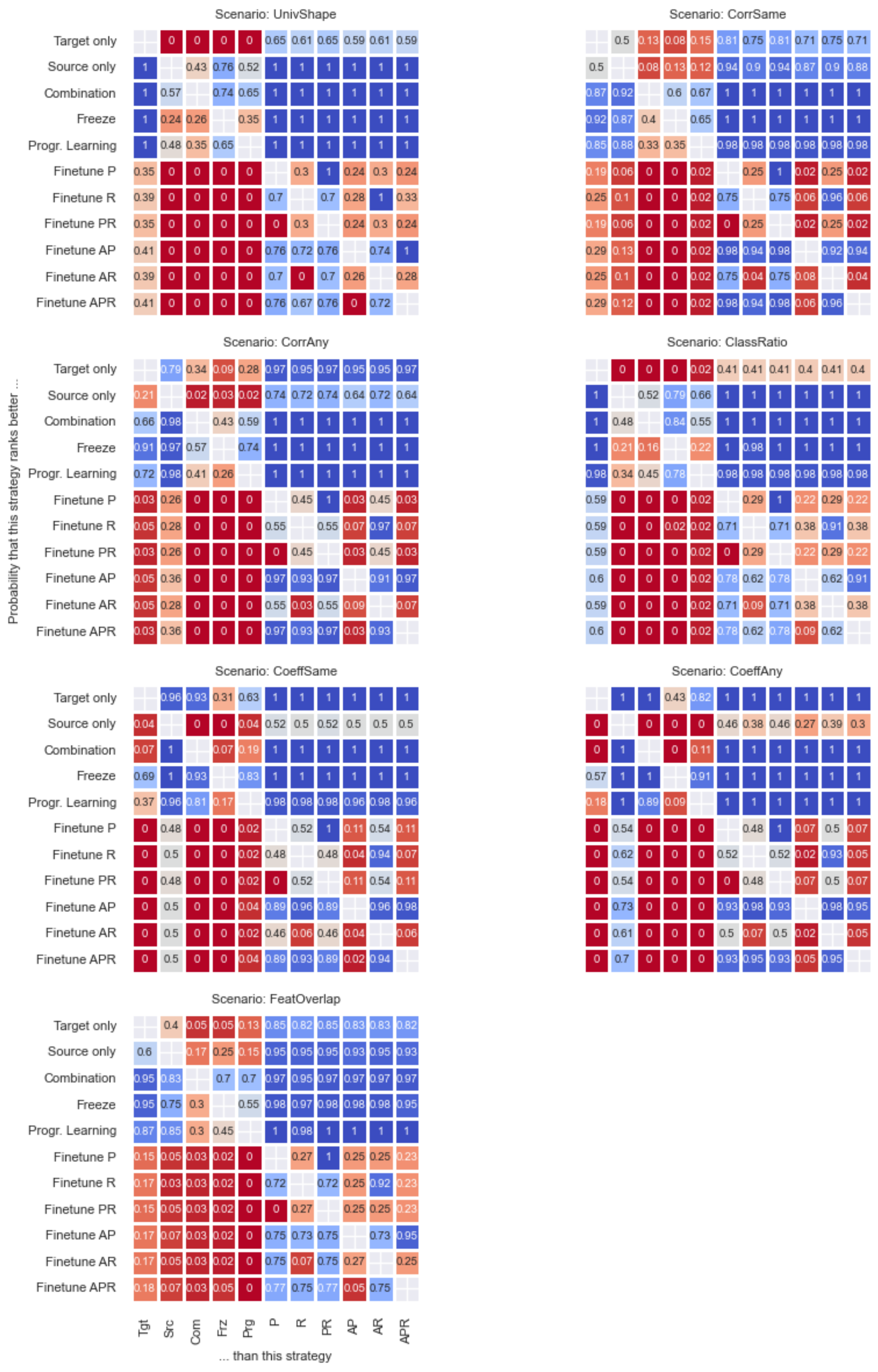

Figure 13: Ranking of strategies in different scenarios

ranking, we can recommend a strategy for each scenario.

In the cases of UnivShape, CorrSame and ClassRatio, the Combination strategy should be used. We can deduce that Combination strategy is a valuable approach to transfer learning if we assume that the underlying correlation structure and probability mechanism of source and target domain are very similar. In these cases, weighting and partitioning by data origin can greatly improve the classification performance in the target domain. In the case of a feature overlap problem, the Combination strategy allows us to treat it as a simple case missing data, which XGBoost handles sufficiently well.

If, on the other hand, we assume that the main domain difference is in the probability mechanism, we should turn to the Freeze strategy, as in the cases of CorrAny, CoeffSame and CoeffAny. The Freeze strategy allows the model to first learn the shared part of the mechanism and then focus on the part that is decisive only in the target domain. Also, a very dissimilar correlation structure where positive correlation in one domain can be negative in the other should prompt us to use the Freeze strategy. This is consistent with the intuition that the splitting criteria of a tree implicitly also split the ranges of correlated features. If the correlation differs strongly, this implicit split will lead to sub-optimal partitions.

We would like to point out that the distinction between non-inductive and inductive problems would not be sufficient to choose the best strategy. Consequently, researchers and practitioners should analyze their data thoroughly and identify the source of the distributional difference in order to choose the best strategy.

Overall, we found the best performance in the Freezing, Progressive Learning and the Combination strategies depending on the distributional differences. It is conceivable that, if more than one type of difference is present, a combination of these strategies could lead to better results. Successive research could investigate how the strategies can be combined. For example, instead of limiting the Freeze strategy to yield the same number of trees as the source model, we could combine it with Progressive Learning and allow the addition of more trees. Alternatively or additionally, we could train the base model of Freeze strategy on the combined data set instead of the source data set, as has been done in the Combination strategy.

6.3 Limitations

Despite the insights from our simulation study, it is important to acknowledge its limitations, especially given that it relies on synthetic data. One of the primary limitations is the lack of real-world complexity in synthetic data sets. Synthetic data cannot fully capture the intricacies and stochastic variability inherent in real-world data. For instance, unmeasured confounding factors, missing data patterns, and noisy signals are commonly observed in real-world scenarios but may not be adequately represented in our synthetic data sets. Therefore, the strategies we recommend based on these simulations may not be universally applicable, or may require adjustments when applied to real-world data.

Another limitation is the risk of over-fitting our recommendations to the particular synthetic scenarios we created. The assumptions used to generate the synthetic data cannot cover all possible conditions that could be encountered in real-world applications. For example, our synthetic scenarios may have defined correlation structures or probability mechanisms that are less nuanced than those in actual applications. As such, the strategies that performed well in our synthetic scenarios may not necessarily perform well under conditions that deviate from our initial assumptions. Hence, future research with real-world data is critical for validating the robustness and generalizability of our findings.

7 Conclusions

In this thesis, we conducted a focused study on transfer learning strategies utilizing XGBoost for tabular data. Through a carefully designed simulation study employing synthetic data, we tested various strategies such as Combination, Freezing, Progressive Learning, and Finetuning.

We found that the Combination strategy performs exceptionally well when the underlying correlation structures between the source and target domains are similar, whereas the Freeze strategy is more effective when dealing with domains that have diverging probability mechanisms. Progressive Learning is also a promising approach, but was not able to outperform Freezing and Combination in the context of our simulation. Lastly, our implementation of Finetuning was inferior to the other strategies.

Our findings suggest that the effectiveness of a transfer learning strategy depends on the specific characteristics of the source and target domains. Understanding these characteristics can guide the choice of strategy, thereby potentially enhancing performance in real-world applications. This adds a layer of nuance to the existing literature, suggesting that a one-size-fits-all approach to transfer learning is unlikely to be effective. Furthermore, our simulation study has shown that the source of distributional differences is a more important factor when choosing the best transfer strategy than the distance between two distributions. We recommend that any transfer learning endeavor should begin with a thorough understanding of the distributional differences (e.g. class ratio, correlation structure, feature overlap) which will help guide the choice of transfer strategies.

A limitation is that our study uses synthetic data and a limited set of scenarios, which may not capture the complexity and stochastic variability of real-world data. This may limit the generalizability of our findings.

Future research could validate these findings by using real-world data sets. Additionally, studying the effects of combining our suggested strategies could lead to a more holistic understanding of the problem. While our Finetuning approach failed, the algorithms by Fang et al. (2020) and Sun et al. (2022) can also be considered to be finetuning strategies. It might be worthwhile applying these algorithms to our synthetic data for a thorough evaluation. Lastly, there is also room to explore how varying degrees of domain dissimilarity affect the performance of different strategies within each scenario.

References

Al-Stouhi, S. and C. K. Reddy (2011): "Adaptive Boosting for Transfer Learning Using Dynamic Updates." *ECML/PKDD (1)*, 6.

Ball, J. E., D. T. Anderson, and C. S. Chan (2017): "Comprehensive survey of deep learning in remote sensing: Theories, tools, and challenges for the community," *Journal of Applied Remote Sensing*, 11, 042609.

Bentéjac, C., A. Csörgő, and G. Martínez-Muñoz (2021): "A comparative analysis of gradient boosting algorithms," *Artificial Intelligence Review*, 54, 1937–1967.

Bjelogrlic, S. (2023): "TransferBoost: Tooling for transfer learning with Boosting Models," Python package available from PyPI.

Borisov, V., T. Leemann, K. Sessler, J. Haug, M. Pawelczyk, and G. Kasneci (2021): "Deep neural networks and tabular data: A survey. arXiv 2021," *arXiv preprint arXiv:2110.01889*.

Cai, L., J. Gu, J. Ma, and Z. Jin (2019): "Probabilistic Wind Power Forecasting Approach via Instance-Based Transfer Learning Embedded Gradient Boosting Decision Trees," *Energies*, 12, 159.

Chen, T. and C. Guestrin (2016): "XGBoost: A Scalable Tree Boosting System," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 785–794, arXiv:1603.02754 [cs].

Cheplygina, V., M. de Bruijne, and J. P. Pluim (2019): "Not-so-supervised: a survey of semi-supervised, multi-instance, and transfer learning in medical image analysis," *Medical Image Analysis*, 54, 280–296.

Dai, W., Q. Yang, G.-R. Xue, and Y. Yu (2007): "Boosting for transfer learning," in *Proceedings of the 24th international conference on Machine learning*, Corvalis Oregon USA: ACM, 193–200.

Daumé III, H. (2007): "Frustratingly Easy Domain Adaptation," in *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, Prague, Czech Republic: Association for Computational Linguistics, 256–263.

Day, O. and T. M. Khoshgoftaar (2017): "A survey on heterogeneous transfer learning," *Journal of Big Data*, 4, 1–42.

Fang, W., C. Chen, B. Song, L. Wang, J. Zhou, and K. Q. Zhu (2020): "Adapted tree boosting for Transfer Learning," .

Farhadi, A. (2020): "Classification Using Transfer Learning on Structured Healthcare Data," Ph.D. thesis, University of Georgia.

Grinsztajn, L., E. Oyallon, and G. Varoquaux (2022): "Why do tree-based models still outperform deep learning on tabular data?" *arXiv preprint arXiv:2207.08815*.

Guram, M. H. et al. (2021): "Improved Demntia Images Detection And Classification Using Transfer Learning Base Convulation Mapping With Attention Layer And XGBOOST Classifier," *Turkish Journal of Computer and Mathematics Education (TURCOMAT)*, 12, 217–224.

Iman, M., H. R. Arabnia, and K. Rasheed (2023): "A review of deep transfer learning and recent advancements," *Technologies*, 11, 40.

Jaiswal, P., V. Katkar, and S. Bhirud (2022): "Multi Oral Disease Classification from Panoramic Radiograph using Transfer Learning and XGBoost," *International Journal of Advanced Computer Science and Applications*, 13.

Kamishima, T., M. Hamasaki, and S. Akaho (2009): "TrBagg: A Simple Transfer Learning Method and its Application to Personalization in Collaborative Tagging," in *2009 Ninth IEEE International Conference on Data Mining*, Miami Beach, FL, USA: IEEE, 219–228.

Liu, R., Y. Shi, C. Ji, and M. Jia (2019): "A survey of sentiment analysis based on transfer learning," *IEEE Access*, 7, 85401–85412.

Lu, J., V. Behbood, P. Hao, H. Zuo, S. Xue, and G. Zhang (2015): "Transfer learning using computational intelligence: A survey," *Knowledge-Based Systems*, 80, 14–23.

Lv, X., Y. Guan, and B. Deng (2014): "Transfer learning based clinical concept extraction on data from multiple sources," *Journal of biomedical informatics*, 52, 55–64.

Minvielle, L., M. Atiq, S. Peignier, and M. Mougeot (2019): "Transfer Learning on Decision Tree with Class Imbalance," in *2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)*, 1003–1010, iSSN: 2375-0197.

Niu, S., Y. Liu, J. Wang, and H. Song (2020): "A Decade Survey of Transfer Learning (2010–2020)," *IEEE Transactions on Artificial Intelligence*, 1, 151–166.

Pan, S. J. and Q. Yang (2010): "A survey on transfer learning," *IEEE Transactions on knowledge and data engineering*, 22, 1345–1359.

Pan, W. (2016): "A survey of transfer learning for collaborative recommendation with auxiliary data," *Neurocomputing*, 177, 447–453.

Pardoe, D. and P. Stone (2010): "Boosting for regression transfer," in *Proceedings of the 27th International Conference on International Conference on Machine Learning*, 863–870.

Rodner, E. and J. Denzler (2008): "Learning with Few Examples using a Constrained Gaussian Prior on Randomized Trees." 159–168.

——— (2011): "Learning with few examples for binary and multiclass classification using regularization of randomized trees," *Pattern Recognition Letters*, 32, 244–251.

Segev, N., M. Harel, S. Mannor, K. Crammer, and R. El-Yaniv (2017): "Learn on Source, Refine on Target: A Model Transfer Learning Framework with Random Forests," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39, 1811–1824, conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence.

Shwartz-Ziv, R. and A. Armon (2021): "Tabular Data: Deep Learning is Not All You Need," *CoRR*, abs/2106.03253.

Sufian, A., A. Ghosh, A. S. Sadiq, and F. Smarandache (2020): "A survey on deep transfer learning to edge computing for mitigating the covid-19 pandemic," *Journal of Systems Architecture*, 108, 101830.

Sukhija, S., N. C. Krishnan, and G. Singh (2016): "Supervised Heterogeneous Domain Adaptation via Random Forests." in *IJCAI*, 2039–2045.

Sun, Y., T. Lu, C. Wang, Y. Li, H. Fu, J. Dong, and Y. Xu (2022): "TransBoost: A Boosting-Tree Kernel Transfer Learning Algorithm for Improving Financial Inclusion," in *Proceedings of the AAAI Conference on Artificial Intelligence*, AAAI, vol. 36, 12181–12190.

Tan, C., F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu (2018): "A survey on deep transfer learning," in *Artificial Neural Networks and Machine Learning–ICANN 2018: 27th International Conference on Artificial Neural Networks, Rhodes, Greece, October 4-7, 2018, Proceedings, Part III 27*, Springer, 270–279.

Tawalbeh, S., M. Hammad, and M. AL-Smadi (2020): "SAJA at TRAC 2020 Shared Task: Transfer Learning for Aggressive Identification with XGBoost," in *Proceedings of the Second Workshop on Trolling, Aggression and Cyberbullying*, Marseille, France: European Language Resources Association (ELRA), 99–105.

Thi, H., H. Tuan, H. Thai, T. Thanh, C. T. Nguyen, N. H. Phung, and B. Viet (2022): "A Deep Learning framework with transfer learning and XGBoost for multi-label classification of Coronary Artery Disease from SPECT polar maps," .

Warchal, S. J., J. C. Dawson, and N. O. Carragher (2019): "Evaluation of Machine Learning Classifiers to Predict Compound Mechanism of Action When Transferred across Distinct Cell Lines," *SLAS Discovery*, 24, 224–233.

Weiss, K., T. M. Khoshgoftaar, and D. Wang (2016): "A survey of transfer learning," *Journal of Big data*, 3, 1–40.

Woźnica, K., M. Grzyb, Z. Trafas, et al. (2023): "Consolidated learning: a domain-specific model-free optimization strategy with validation on metaMIMIC benchmarks," *Mach Learn*.

Yang, Q. (2020): *Transfer learning*, Cambridge ; New York, NY: Cambridge University Press.

Yao, Y. and G. Doretto (2010): "Boosting for transfer learning with multiple sources," in *2010 IEEE computer society conference on computer vision and pattern recognition*, IEEE, 1855–1862.

Zhang, J., K. Zhou, P. Huang, X. He, M. Xie, B. Cheng, Y. Ji, and Y. Wang (2020): "Minority Disk Failure Prediction Based on Transfer Learning in Large Data Centers of Heterogeneous Disk Systems," *IEEE Transactions on Parallel and Distributed Systems*, 31, 2155–2169, conference Name: IEEE Transactions on Parallel and Distributed Systems.

Zhang, L. and X. Gao (2022): "Transfer adaptation learning: A decade survey," *IEEE Transactions on Neural Networks and Learning Systems*.

Zhao, Z., Y. Chen, J. Liu, Z. Shen, and M. Liu (2011): "Cross-people mobile-phone based activity recognition," in *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence-Volume Volume Three*, 2545–2550.

Zhu, H., J. Xu, S. Liu, and Y. Jin (2021): “Federated Learning on Non-IID Data: A Survey,” .

Zhuang, F., Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He (2020): “A comprehensive survey on transfer learning,” *Proceedings of the IEEE*, 109, 43–76.

A Algorithms

Algorithm 1 Freeze strategy

Function name: tune_freeze_model

Purpose: Train an XGBoost classifier using two distinct training datasets, freezing a fraction of estimators and retraining the remaining ones.

Input Parameters:

- `X_train_src`, `y_train_src`: The features and labels of the source training set.
- `X_train_tgt`, `y_train_tgt`: The features and labels of the target training set.
- `X_test_tgt`, `y_test_tgt`: The features and labels of the test set

Function Description:

Initial Model Tuning

- Tune the model hyperparameters on the source training dataset using the `tune_model` method
- Retrieve the best parameters from the tuning results
- Extract the optimal number of estimators, `t0`

Tune the number of freezed estimators

- Generate a sequence of potential `swap_times` based on `t0`: `[0*t0, 0.2*t0, 0.4*t0, 0.6*t0, 0.8*t0, 1*t0]`
- For each `st in swap_times`:
    - Initialize an XGBoost classifier model with the tuned hyperparameters
    - Train the model with `st` estimators on the source dataset
    - Continue training the model on the target training dataset for `t0 - st` rounds
    - Calculate the AUC-PR for the test dataset
    - If the current model's AUC-PR score is better than previous, update the best results.

Return Value:: Trained XGBoost model

Algorithm 2 Progressive Learning strategy

Function name: `tune_progressive_model`

Purpose:

Train an XGBoost classifier model first on the source dataset and then add more estimators based on the target dataset.

Input Parameters:

- `X_train_src`, `y_train_src`: The features and targets of the source training set
- `X_train_tgt`, `y_train_tgt`: The features and targets of the target training set
- `X_test_tgt`, `y_test_tgt`: The features and targets of the target test set

Function Description:

Initial Model Tuning

- Obtain the initial model from the `tune_model` method using the source training dataset
- Extract the best model parameters from the tuning result.

Training on Source Dataset

- Initialize the XGBoost classifier with the optimal parameters.
- Set parameters for early stopping
- Train the model on the source training dataset
- Retrieves parameters of the model after training as `step1_parameters`.

Training on Target Dataset

- Retrieve the learning rate and number of estimators from `step1_parameters` and adjust them for the next training phase:

  ```
  - eta = eta * 0.75
  - n_estimators = n_estimators * 2
  ```

- Continue model training on the target training dataset.
- The previous model's booster is passed to continue training from where it left off.

Return Value: Trained XGBoost model

Algorithm 3 Finetuning strategy

Function name: `tune_finetune_model`

Purpose:

Finetune an XGBoost classifier trained on the source data, using various strategies like data augmentation, model pruning, and leaf reweighting based on the target data.

Input Parameters:

- `X_train_src`, `y_train_src`: The features and labels of the source training set.
- `X_train_tgt`, `y_train_tgt`: The features and labels of the target training set.
- `X_test_src`, `y_test_src`: The features and labels of the source test set.
- `X_test_tgt`, `y_test_tgt`: The features and labels of the target test set.
- `augment`, `prune`, `reweight`: Booleans indicating whether to augment the data, prune the model, or reweight the leaf nodes.

Function Description:

Base Model Tuning

- The function initiates the tuning of the model on the source training dataset using the `tune_model` method.

Data Augmentation

- If `augment`, augment the target data using the `augment_target_data` method
- Else, use the source training data as it is

Model Pruning

- If `prune`, prune the base model using the `update_model_with_prune` method.
- Else, the base model remains unchanged.

Leaf Re-Weighting

- If `reweight`, updated the leaf weights of the model using the `update_model_leaf_weights` method.
- Else, the potentially pruned model remains unchanged.

Return Value: Trained XGBoost model

Algorithm 4 Data augmentation for finetuning

Function name: `augment_target_data`

Purpose:

The function augments the target training dataset with samples from the source training dataset based on leaf indices predicted by an XGBoost model. This is done to ensure that each leaf node in the decision trees contains at least a minimum number of samples. This aims to introduce bias and reduce the risk of overfitting during finetuning

Input Parameters:

- `model`: An instance of a trained XGBoost classifier.
- `X_train_src`, `y_train_src`: Feature matrix and labels of the source training dataset
- `X_train_tgt`, `y_train_tgt`: Feature matrix and labels of the target training dataset
- `min_leaf_cnt`: Minimum number of samples required in each leaf

Function Description:

- Predicts the leaf indices for both source and target datasets using the `model`
- Initialize the augmented dataset with the target dataset
- Iterate over each decision tree in the model:
  - Extract leaf indices for the current tree.
  - Compute the number of samples in each leaf of the target dataset
  - For each leaf, if the sample count is below `min_leaf_cnt`:
    - Determine how many additional samples are needed
    - Select the required number of samples from the source dataset with the same leaf index
    - Add selected source samples to the augmented dataset

Return Values:

- Augmented feature matrix
- Augmented labels

Algorithm 5 Tree pruning for finetuning

Function name: `update_model_with_prune`

Purpose:

This function updates an existing XGBoost model using the ”prune” updater, which trims branches from the trees based on a provided gamma threshold. Branches with gains lower than this threshold are removed from the model.

Input Parameters:

- `base_model`: The base XGBoost model to be updated.
- `X_train_new`, `y_train_new`: New training feature matrix and labels.
- `gamma`: Threshold for pruning. Branches with gain less than this value are pruned.

Function Description:

- Sets the updater to ”prune”, provides the gamma value, and sets the process type to ”update”.
- Updates the model with the new training data using the specified pruning parameters.

Return Values:

- `updated_model`: The updated XGBoost model after pruning.

Algorithm 6 Leaf re-weighting for finetuning

Function Name: `update_model_leaf_weights`

Purpose:

Updates the leaf weights of a given XGBoost model based on new training data. It uses the "refresh" updater to adapt the model without altering its structure, avoiding a complete retraining.

Input Parameters:

- `base_model`: The initial XGBoost model to be updated.
- `X_train_new`, `y_train_new`: New training feature matrix and corresponding labels.

Function Description:

- Configures parameters for the updater as "refresh", indicating the leaf weights should be updated without altering tree structure.
- Trains and updates the model, using the base model's structure.

Return Value:

- `updated_model`: The XGBoost model with recalculated leaf weights.

Declaration of Authorship

I hereby confirm that I have authored this Master's thesis independently and without use of others than the indicated sources. All passages which are literally or in general matter taken out of publications or other sources are marked as such.

Berlin, September 22, 2023

Ulrike Wöll