Efficient Calculation of Approximate Shapley Values in Decomposable Generalized Additive Neural Networks with Pairwise Interactions

Şimşek, M.G. (Gökhan)

Award date:
2020

Technische Universiteit Eindhoven University of Technology
Department of Mathematics and Computer Science
Architecture of Information Systems Research Group

Efficient Calculation of Approximate Shapley Values in Decomposable Generalized Additive Neural Networks with Pairwise Interactions

Master Thesis

Mehmet Gökhan Şimşek

Supervisors:
University Supervisor: Robert Peharz
Industry Supervisor: Nicoleta Onuta
Committee Member: Wouter Duivesteijn
Committee Member: Erik Quaeghebeur

Eindhoven, September 2020

Abstract

This thesis is a collaborative work between TU/e and De Lage Landen (DLL) to design an explainable model to solve the problem of prioritization of debtors for Collections and Recovery agents. In this work, we present a new glass box model architecture called Decomposable Generalized Additive Neural Network with Pairwise Interactions (DGANN^2). We further present an efficient algorithm to approximate Shapley values on DGANN^2 utilizing its decomposable nature for easy marginalization. We use a debtor information and behaviour data set provided by DLL to classify debtors using DGANN^2, and benchmark DGANN^2 against several other models. We find that DGANN^2 can be a useful alternative to existing black box and glass box models, due to its comparative performance and efficient explainability methods.

Preface

This thesis marks the end of a two-year adventure at the Eindhoven University of Technology, and possibly the end of my life in academia. It has been a tough process, especially the last few months with the quarantine, but I have made it. Not without the help of some important people in my life, though.

I would like to start by thanking my supervisors: Robert Peharz from TU/e for your very involved supervision and feedback, and Nicoleta Onuta from DLL for the opportunity that you gave me to work as part of an intimate team. I would like to thank you both for the freedom you gave me in executing the project, and in the writing process. I would like to especially thank a colleague and a fellow intern, Phuong Trinh, for her support and friendship during my time at DLL. Finally, thank you to Wouter Duivesteijn and Erik Quaeghebeur for accepting to be in the assessment committee.

I would like to thank my family here in the Netherlands for their invaluable support during this process. Starting with the closest friends that made up my support system at this new and uncharted territory, Irmak, Sonali, Selima, and Berk. We have been through this journey together, and I cannot think of a better group of people to have shared countless days, nights, and moments together. I'm looking forward to the day when we have celebratory drinks for all of our graduations. I would like to thank my housemate, Ruben, for enduring me during the stressful times of quarantine and thesis writing. Your support and encouragement, and the distractions you provided have kept me going. And last but definitely not least, I would like to thank Semih. You have been a brother to me for the past however many years, and I would not be here without you. You are the reason I applied to TU/e, and you have been there for me through the highs and the lows. I cannot thank you enough.

Lastly, I want to thank my mom, dad, and brother. I distinctly remember that I forgot to thank my mom when I was graduating from elementary school, and I will not make the same mistake again. I wouldn't be where I am today without your continuous, endless, immeasurable love and support. Even though we are currently apart, I feel your spirit close to me. Göktuğ, my brother, I am absolutely sure that I will see your thesis in a few years, and you will even do better than me. I love you all.

Mehmet Gökhan Şimşek Eindhoven, 2020

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Problem Formulation

Explainability in AI has come under focus especially in the recent years due to two main reasons: the rise of black box models with the recent developments in neural networks and deep learning, and regulatory action from lawmakers such as GDPR requiring clear explanations from model predictions. Explainability is a wide field with methods that span model-specific and model-agnostic approaches, and different ways of representing a model's decision making. A main reason for the variety of explainability methods is the term's ambiguity. It is not immediately clear what is meant by 'explainability', who the audience is, or how explainability can be measured. Thus, some of the methods have been developed for field experts such as machine learning engineers, while others have been developed for laypeople. The definition of explainability that we will use in this thesis comes from [5]:

> Given an audience, an explainable Artificial Intelligence is one that produces details or reasons to make its functioning clear or easy to understand.

This definition of explainability can be considered wide. However, it encompasses the two concepts of human-centric and model-centric explainability. Explainability and interpretability are two terms that are often used interchangeably in literature, while having slightly separate meanings. [23] define interpretation as the 'mapping of an abstract concept (e.g. a predicted class) into a domain that the human can make sense of', and define explanation as 'the collection of features of the interpretable domain, that have contributed for a given example to produce a decision'. While the difference is subtle, interpretability is human-centric. The interpretation of the model should be in a domain natural to a human being, such as text or images. Explainability, on the other hand, is model-centric, requiring an observer to be able to understand why a certain model made a certain decision starting from the interpretable domain, such as features in a data set or words for a natural language processing task. In this thesis, we will be using explainability as described by [5] above as an umbrella term. It is also important to note that the explainability in this context refers to more than a low-level understanding of the underlying algorithms of a model. Rather than the mathematical operations that make up the decision-making of a model, explainability requires a functional understanding of a model [15].

The upsurge of research around explainability coincides with the advancements in the field of deep learning. In critical fields such as medicine, finance, self-driving cars, and in the legal field, machine learning models and deep networks in particular are successfully utilized to either automate tasks or to help human experts. When the decisions made by models and algorithms affect human lives, the need for understanding how these decisions are taken arises. This wide deployment of machine learning (ML) has caused several issues to be raised in terms of confidence, fairness,

informativeness, and trustworthiness. Explainable Artificial Intelligence (XAI) as a broad field aims to address these issues.

In this thesis, we will be focusing on a specific method of providing explainability to machine learning models, namely Shapley values. Calculating Shapley values is a model-agnostic method that takes its roots from game theory, first used in machine learning in [42]. Shapley values are calculated per feature; the Shapley value for a feature can be interpreted as the contribution of the specific feature towards the prediction. Calculating Shapley values exactly is exponential in the number of features that the model takes as input. Shapley values are generally calculated using approximation techniques on black box models, i.e. models that are not inherently explainable. Black box models include complex, non-intuitive models including deep neural networks, such as recursive neural networks (RNNs). In this thesis, we are aiming to enhance the explainability of a glass box model inspired by generalized additive models with interactions, or GA^{2}Ms. We call this new type of models DGANN^{2}s, for Decomposable Generalized Additive Neural Networks with Pairwise Interactions. We will be focusing on the question of calculating approximate Shapley values efficiently on DGANN^{2}s utilizing an easy way of marginalization of variables inherent in DGANN^2's architecture.

1.2 Business Application

The thesis presented here has been done in collaboration with De Lage Landen (DLL), a global finance partner for equipment and technology assets, under the Rabobank group. As a financial company founded more than 50 years ago, DLL combines years of experience and data with artificial intelligence to help human agents in taking critical decisions. A key requirement for the models that are put into production at DLL is explainability, due to the regulations regarding handling of user data and financial decision making. This explicit requirement for explainability can be limiting in the type of models that can be put into production. The developers need to approach the balance between explainability and complexity of the models carefully. Traditionally, models such as linear or logistic regression have been deployed to comply with explainability requirements. However, these models are not complex enough to give accurate predictions, and more complex models with better predictive power can be combined with model agnostic methods to be put into production. Explainability can also be useful internally at DLL. Complex models can discover underlying trends in the data, and can lead to improving the internal processes of DLL.

This work has been done in the AI Lab, a new group inside DLL that's focused on providing value to the organization via artificial intelligence. AI Lab focuses on several projects, and operates with agile development practices. This thesis is in relation to the project 'Prioritization of Collection and Recovery Activities', also referred to as C&R. The C&R project focuses on the activities of collections and recovery agents. As a financing partner for assets, DLL has a large number of debtors, and a large number of invoices and contracts to keep track of. C&R agents are the first line of contact from DLL; they contact a debtor after an invoice goes past its due date, and try to set up a deal. If an invoice goes past due a certain number of days, then the debtor is moved to a different management section. C&R agents get a list of debtors every day that have gone past the due date on their invoices. However, this list is not sorted or prioritized. The aim of the C&R project within AI Lab is to prioritize the debtors and the corresponding actions of C&R agents based on the potential risk of the debtors.

Towards this goal, DLL has provided data on past behaviours of debtors, their contracts, and invoices. Based on this historical data, the goal is to predict which debtors are more likely to go past a certain number of days from their due date. To put it more formally, the problem at hand is a binary classification problem. The classifier should be able to accurately predict whether a debtor will go 17 days past due from the due date of the invoice, in which case they are called given a 'late' classification of 1. The 'on-time' debtors are denoted with a 0 label.

At DLL, currently an XGBoost model is developed for this particular use case. XGBoost (eXtreme Gradient Boosting) implements gradient boosting, creating an ensemble of multiple weaker learners into a single model. In the case of XGBoost, these weaker models are regression or decision trees. XGBoost is not inherently explainable by itself. However, as described later in this thesis, with a certain method called SHAP values for tree-based learners (Tree SHAP), feature importance values can be calculated from XGBoost models in polynomial time.

This thesis aims to construct a proof-of-concept for an alternative solution to the C&R project using DGANN^{2}s. The introduced model can in turn be used in other projects as well.

1.3 Proposed Solution and Motivation

Our work presents a specific model architecture that we call Decomposable Generalized Additive Neural Networks with Pairwise Interactions, or DGANN^2 that allows for efficient approximate calculation of Shapley values. This model architecture is inspired by generalized additive neural networks (GANN) and generalized additive models plus interactions (GA^{2}M).

Our aim is twofold: We show that this model architecture has comparative performance to its close alternatives and widely used black box models while retaining intrinsic explainability. We further show that subject to some general assumptions about the data, we can calculate approximate Shapley values much more efficiently compared to standard methods of calculation. This efficiency is due to the decomposable nature of the network, enabling easy marginalization of variables.

The motivation for this thesis is to present a glass box model with an additional efficient explainability technique, namely efficient Shapley value calculations. Each explainability method can enrich the understanding of a model further. While glass box additive models such as GA^{2}Ms allow the users to observe how the changes to a variable affects the model output, they do not natively support feature importance observations. Providing feature importances in an efficient way equips the data scientist with more power to better understand the model, and provides a better overview of the model for end users.

1.4 Contributions

We present a new type of model called DGANN^2 and test its performance on several data sets. We compare DGANN^2 against widely used models like GAMs, GA^{2}Ms or EBMs, and XGBoost. We further present D-Shapley, a closed form solution that utilizes probabilistic marginalization inherent in the decomposable structure of DGANN^2 to calculate approximate Shapley values. We find that DGANN^2 shows comparative performance, making it a strong solution to be used in the C&R project. We further show that DGANN^2 is equipped with several ways of providing explanations to the user. With its highly explainable nature, DGANN^2 is a valuable addition to the explainability literature.

1.5 Thesis Outline

The thesis is structured as follows: Chapter 2 will present an overview of explainability in AI and some of the models and methods to achieve explainability. Chapter 3 will describe the new type of model called DGANN^2 proposed in this work, and derive the closed form solution to calculate Shapley values efficiently on this model. Chapter 4 will describe the main data set used in this work, along with the experimental setup to test the performance of DGANN^2. Chapter 5 will provide explanations generated by the trained DGANN^2 on the DLL data set, and will present performance comparisons of DGANN^2. Lastly, Chapter 6 will provide conclusions and limitations of the thesis, and will present suggestions on future work.

Chapter 2

Literature Review

In this chapter, background on explainability as a field is given. In the following sections, certain influential glass box models and model-agnostic explainability methods for calculating feature importances are presented.

2.1 Explainability

2.1.1 Motivations for Explainability in AI

Explainability in AI has a variety of motivations, centered around the issues of fairness, confidence, informativeness, and trustworthiness.

Fairness in AI has been addressed in numerous publications. An article on the risk assessment tool COMPAS brought to daylight the racial discrimination automated decision making can lead to if not controlled and designed correctly [4]. Safiya Noble wrote about the negative effects of automated decision making on minority groups like people of color, women, LGBTQ+ individuals, religious minorities, and their intersections [25]. Computer vision algorithms and in particular automatic gender recognition came under scrutiny after Buolamwini's seminal paper on how commercial gender classification tools perform significantly worse for darker-skinned women when compared to light-skinned people [7]. These have pushed prominent organizations to publish responsible AI principles to make machine learning more transparent, fair, and explainable [1, 2]. Organizations are gradually becoming more aware of fairness and inclusivity practices, and this in turn is starting to reflect in their practices in artificial intelligence. The fairness concept is especially important for organizations dealing with human data, not only from an ethical standpoint, but also from a legal standpoint.

From a legal perspective, the GDPR has been influential in bringing AI explainability and transparency to the forefront. Since 2018, General Data Protection Regulation or the GDPR has been in place with regards to the citizens of the European Union. The GDPR changed how data is being handled worldwide, since any company handling the data of any EU citizen, regardless of their current residence, has to comply with the new regulations. While the GDPR doesn't directly address the right to explanation, Articles 13-15 and Article 22 touch upon the subject [28, 27]:

> Article 13(2): In addition to the information referred to in paragraph 1, the controller shall, at the time when personal data are obtained, provide the data subject with the following further information necessary to ensure fair and transparent processing:
>
> Article 13(2)(f): the existence of automated decision-making, including profiling, referred to in Article 22(1) and (4) and, at least in those cases, **meaningful information about the logic involved, as well as the significance and the envisaged**

consequences of such processing for the data subject.

Article 22(3): In the cases referred to in points (a) and (c) of paragraph 2, the data controller **shall implement suitable measures** to safeguard the data subject's rights and freedoms and legitimate interests, at least the right to obtain human intervention on the part of the controller, to express his or her point of view and to contest the decision.

Even though there is no clear definition of the right to explanation, the authors of [38] argue that in practice, these articles point to a right to explanation and should be treated as such. Thus, the GDPR forms a legal basis for companies to put more emphasis into explainability. Given the recent rising concerns around fairness, and the legal restrictions, an active emphasis should be put on explainability in AI at financial companies like DLL.

2.1.2 Taxonomy of Explainability

Explainability can be understood differently across applications and stakeholders.

In [6], authors have surveyed data scientists and representatives from different organizations to arrive at several different explainability needs: model debugging, model monitoring, model transparency, and model audit. While these needs often intersect for large organizations, debugging and monitoring generally concern internal stakeholders like ML engineers and quality assurance teams. Model transparency and audit at large can concern compliance teams and may be necessary to explain models to end users or management. In the financial sector, model auditing takes on an important role due to regulations.

In [32], authors identify four main stakeholder groups: developers, theorists, ethicists, and users. They argue that definitions and methods of explainability can change between the different stakeholders, even if they overlap at some parts. According to the authors, developers are concerned with explainability in terms of quality assurance and debugging. Theorists aim to use explainability to advance research and improve existing models, such as finding their weak points. Ethicists are interested in fairness and accountability of systems, concerned with model transparency and possibly auditing. The final group, users, are active consumers of AI systems. They require explainability to act rationally when faced with decisions made by an AI system. In financial systems, all these stakeholders are generally involved. Thus, it is important to take the definition of explainability as general as possible to include all the different stakeholders. Multiple methods of explainability can be appropriate to be able to explain the model to these different groups.

The taxonomy around explainability is large, and some terms can be used interchangeably in literature (e.g. explainability and interpretability, transparent and interpretable). However, the following terms are necessary for further discussion, and we present the most widely-used definitions:

Local Explainability: Local explainability methods are aimed towards explaining the model behaviour for a single instance. e.g. if a loan request is rejected for a particular applicant, which factors have caused this rejection? For a given application, changes to which variables can cause this application to be accepted instead of rejected?

Global Explainability: Global explainability refers to describing the behaviour of a model in its entirety, rather than explaining the decision-making process for a single instance. For example, the existence of which words in a spam detection model is important?

Ante-hoc Approaches to Explainability: These approaches consider explainability by design. Glass box models as described further in Section 2.2 fit into ante-hoc framework. The recent years have especially seen a push for new ante-hoc explainable models, such as Reverse Time Attention models (RETAIN) [10] and GA^{2}Ms [19]. The main challenge in developing ante-hoc models lies

in providing explainability while maintaining a high predictive performance. With the rise of explainable AI, the newer body of work aims to challenge the assumption that performance and explainability present a trade-off. This work falls under this category, presenting a performant and explainable architecture.

Post-hoc Approaches to Explainability: In post-hoc approaches, explainability is not designed into the system, and is provided after a model has been trained. These methods explain the model by what is readily interpretable, e.g. in terms of how much importance is gives to certain features [15]. Post-hoc approaches are explained more in detail in Section 2.3.

2.2 Glass Box Models

Glass box models, also called transparent models, are models that are understandable by themselves [5]. Several types of models fall under this category: linear/logistic regression, decision trees, K-nearest neighbours, Bayesian models, and Generalized Additive Models. The simplest one, linear regression, learns a scalar weight for each feature. The explanation is straightforward, as the weight for each feature lets the user explicitly understand how much of a change would result from the perturbation of input features. However, there usually is a performance and explainability trade-off especially when glass box models are considered. Linear regression and decision trees are generally less powerful than black box models such as neural networks, especially when extensive feature engineering has not been performed. Researching and developing glass box models that are also performant is an ongoing area of research.

In the following sections, we will describe Generalized Additive Models and some extensions, as these models lay the groundwork for the architecture of DGANN^2 that we will describe in Section 3.1.

2.2.1 Generalized Additive Models (GAM)

Generalized additive models, or GAMs, have been proposed in [14] as an improvement over linear models such as linear regression and logistic regression models. The GAM architecture replaces each variable in a linear regression model with a (potentially nonlinear) function. The general mathematical form of GAMs are:

$$g(\mathbb{E}[y]) = \beta_0 + \sum_{i=1}^{p} f_i(x_i) \tag{2.1}$$

where g is a link function such as identity or logit, $\mathbb{E}$ is the expectation operator, y is the target variable, β_0 is the intercept, and each x_i is a predictor variable. GAMs are considered to be glass box models: for local explainability, for each feature i, we can observe the result from $f_i(x_i)$. For global explainability, the functions f_i themselves serve as explanations over a domain for each feature i.

GAMs have traditionally been fit using splines [14], boosted stumps, and tree-based methods such as bagged or boosted ensembles of trees [18].

2.2.2 Generalized Additive Neural Networks (GANN)

Generalized additive neural networks, or GANNs, form a specific subset of generalized additive models. In a GANN model, the family of functions for each variable is constricted to artificial neural networks. First developed deeply in [31], GANNs have the same form as GAMs as described in Equation (2.1). The simplest version of GANN with only a single hidden layer of h units and a skip connection would have the following function for each variable:

$$f_j(x_j) = \sum_{k=1}^{h} w_{kj} \cdot g(w_{0kj} + w_{1kj}x_j) + x_j \tag{2.2}$$

where $w = \{w_{hj}, w_{0hj}, w_{1hj} | h = 1...H, j = 1...D\}$ denotes the network parameters, and g is a link function. Thus, each f_j is parameterized by a neural network.

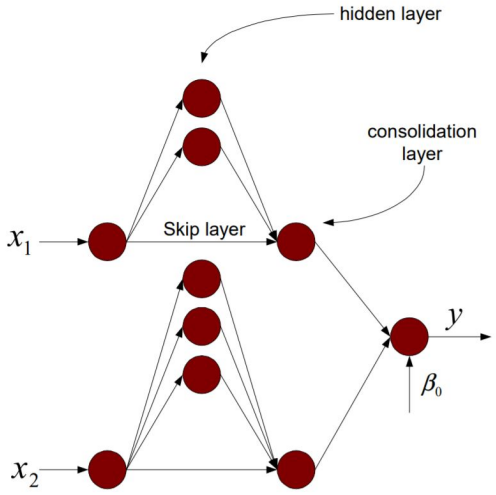

Figure 2.1: A simple GANN architecture with a single hidden layer for each feature [11]

2.2.3 Generalized Additive Models Plus Interactions (GA^{2}M)

A natural improvement to GAMs has come in the form of generalized additive models plus interactions [19]. The basic form of GAMs as in Equation (2.1) have been extended to explicitly consider pairwise interactions between features:

$$g(\mathbb{E}[y]) = \beta_0 + \sum_{i=1}^{p} f_i(x_i) + \sum_{i \neq j} f_{ij}(x_i, x_j) \tag{2.3}$$

GA^{2}M s have been used in numerous settings, including medical settings to predict pneumonia risk and hospital readmission [8]. Authors of [8] show GA^{2}Ms to be competitively accurate compared to black box models such as random forests, while retaining both local and global explainability. For local explainability, the contribution of each feature and each interaction to the prediction can be seen simply from inspecting the results of $f_i(x_i)$ and $f_{ij}(x_i, x_j)$ for a given data point x. Because of the inherent additivity of GA^{2}Ms, feature contributions can be sorted and visualized. For global explainability, the separate functions f_i and f_{ij} can be visualized as line graphs and heatmaps, respectively. However, GA^{2}Ms do not natively support global feature importance. Local feature importance for a given data point can be determined easily from the results of the additive functions, as it is clear how much each feature contributes to the prediction. However, to get global feature importances, other methods can be applied, such as observing the difference in evaluation metrics when a single feature is removed.

In GA^{2}Ms, the number of interactions to consider rises quadratically with the number of features. Modelling every interaction can hurt performance and accuracy of the model, as well as diminish the explainability of the model. A variety of interaction detection methods are available, such as an ANOVA test, RuleFit [13], Partial Dependence Functions, and FAST [19].

In [26], GA^{2}Ms are also referred to as Explainable Boosting Machines, or EBMs. EBMs are available as a Python library under the name `interpret`.

2.3 Post-hoc Approaches

Complex models such as deep neural networks are regarded as black-box models and are not intrinsically explainable. Black box models usually represent complicated functions that cannot easily understood by humans. As an example, deep neural networks, without restricting their form, are comprised of a large number of connections, layers, non-linear transformations, and interactions between the input features. A user cannot understand how a deep neural network makes decisions simply by looking at the weights of the network. To explain the predictions of black-box models, several post-hoc methods have been developed over the years [34, 43, 21, 36].

Model-agnostic post-hoc approaches can be applied to any type of model, and they generally treat the model at hand as a function that can be repeatedly probed. Some post-hoc methods can be model-specific as well, such as Tree SHAP, that is developed specifically for tree-based models.

Post-hoc approaches can take several forms of explanation, as can be seen in Figure 2.2:

- Generating textual explanations to explain the results of a model, either in textual or symbolic form.
- Training a simpler and explainable model to approximate the decisions of the original model.
- Providing explanations via visualizations. These visualizations can show the decision making process of a model, for example by showing how a model divides the data set.
- Generating feature relevance values to explain how each feature contributes to the model's decision making.
- Understanding the model behavior on a local level, in a simplified space. A common form of local explanations is sensitivity analysis.
- Selecting or constructing example instances to show the inner workings of a model. For example, a visual emotion recognition network from faces can explain its decision on an image by showing images that look similar that have the same emotion.

2.3.1 LIME

Presented in [34], Local Interpretable Model-Agnostic Explanations, or LIME, is an explanation method to find an interpretable substitute model that locally approximates the original model around a singular prediction. LIME provides locally faithful explanations by approximating the model's behavior in the neighborhood of a certain data point using an inherently interpretable model.

In mathematical terms, LIME operates on an interpretable representation $x' \in \{0,1\}^{d'}$ in place of the original data point $x \in \mathbb{R}^d$. d' is the number of features in the interpretable representation, whereas x' acts as an activation function denoting which features in the interpretable representation will be considered. We can define a function $h_x(x') = x$ to transform the simplified representation x' to the original input x. An explanation model g is a model that will locally approximate the original model around x'.

In the original paper, authors have taken g to belong to the class of linear models, such that $g(z') = \phi_0 + \sum_{i=1}^{d} {}'\phi_i z'_i$, where $\phi_i \in \mathbb{R}$, and $z' \in \{0,1\}^{d'}$. z' corresponds to a perturbed sample that is in the locality of x'. g thus becomes a linear regression problem. This is one of the underlying assumptions of LIME, that when the locality around an instance is considered, the complex model can behave in a linearly separable way.

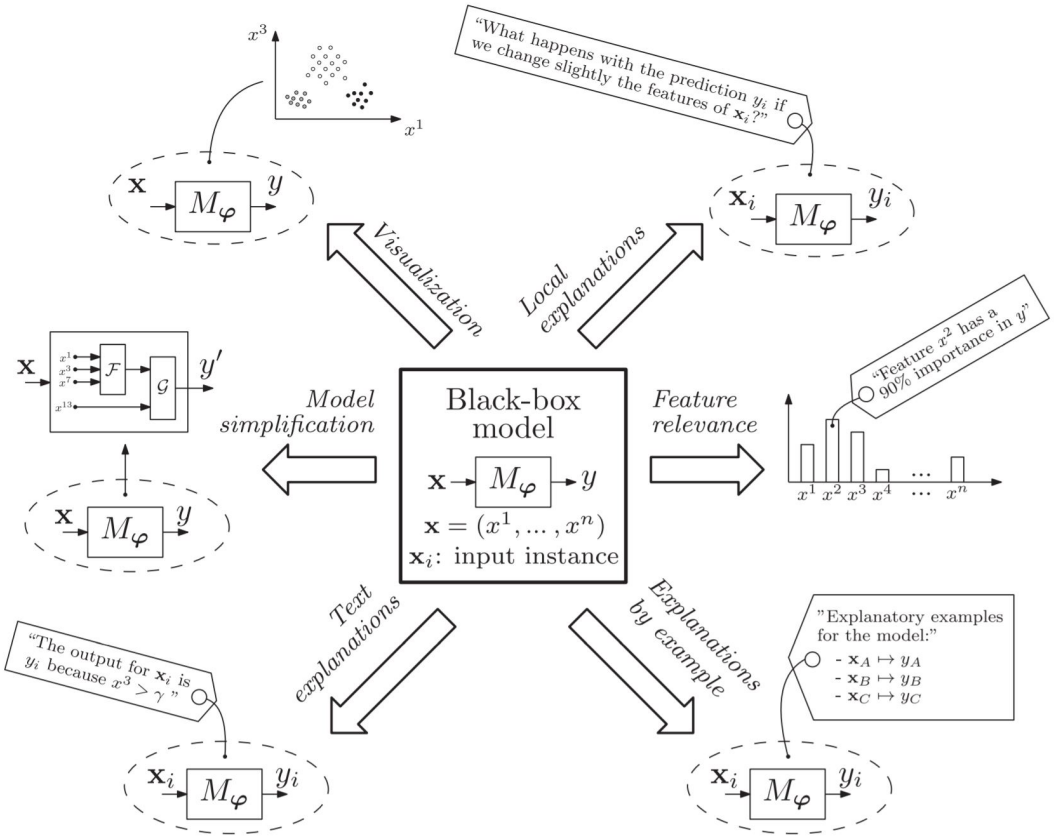

Figure 2.2: Several different post-hoc explainability approaches [5]

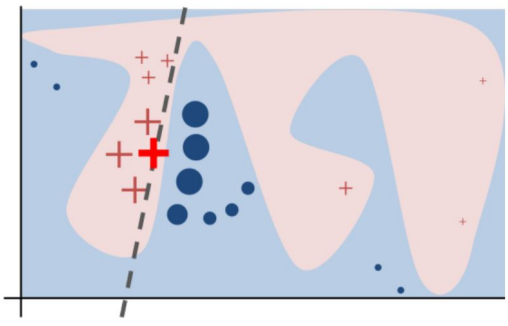

Figure 2.3: Blue/pink background represents the unknown decision function for a black box model. To explain the decision around the instance represented by the bold red plus, LIME samples instances in the locality, and finds an approximate explanation that is locally faithful, represented as the dashed line. Source: [35]

LIME essentially tries to minimize the complexity of the explanation g, denoted as $\Omega(g)$, and a loss function. This loss function is denoted as $\mathcal{L}(f, g, \Pi_x)$, where $\Pi_x(z)$ is a proximity measure between z and x. $\mathcal{L}$ measures local fidelity - how close is g to representing f in the locality around x.

The fidelity loss function is defined as

$$\mathcal{L}(f, g, \Pi_x) = \sum_{z,z' \in \mathcal{Z}} \Pi_x(z)\big(f(z) - g(z')\big)^2 \tag{2.4}$$

where $\mathcal{Z}$ is a data set of perturbed samples of z, z', around the locality of the original data point x.

LIME minimizes the objective function ξ comprised of the loss function $\mathcal{L}$ and the complexity function $\Omega(g)$ as described in Equation (2.5).

$$\xi(x) = \arg\min_{g \in \mathcal{G}} \mathcal{L}(f, g, \Pi_x) + \Omega(g) \tag{2.5}$$

Algorithmically, LIME works by approximating $\mathcal{L}$ through sampling perturbed samples from $\mathcal{Z}$. $f(h_x(z')) = f(z)$ serves as the labels to the explanation model g. Given this data set $\mathcal{Z}$, Equation (2.5) can be solved using Lasso regression or another penalized linear regression method.

LIME requires picking a kernel function to define the proximity measure $\Pi_x(z)$ and a complexity function $\Omega(g)$. These functions are chosen heuristically. The authors have defined $\Pi_x(z) = \exp(-D(x, z)^2/\sigma^2)$ in [34], where D is the cosine distance. However, picking a kernel function is not straightforward especially in the cases of tabular data. Cosine distance can work for textual data better, but in case of tabular data, defining the distance between two instances becomes harder, since a mix of categorical and numerical features can be present. Furthermore, LIME has been shown to be sensitive to minor perturbations [3], and this sensitivity can be utilized in adversarial attacks to create a desired explanation [41].

2.3.2 Shapley Values

Shapley values have their roots in game theory, and have first been introduced by Lloyd Shapley [40]. The main problem Shapley values are developed to solve is fairly distributing surplus payoff among players in a coalitional game setting. In the coalitional, or cooperative game setting, there is a finite set of $N = \{1, 2, ..., n\}$ players, where each non-empty subset of N forms a coalition [39]. $v : 2^N \mapsto \mathbb{R}$ denotes a payoff function, also called a characteristic function, such that $v(\emptyset) = 0$. The payoff function v gives each coalition $S \subseteq N$ a real-valued payoff to be distributed among players in the coalition. $v(S)$ can also be thought as the worth of S. The individual payoff for each player $i \in N$ is denoted as ϕ_i.

Shapley has first introduced 4 axioms to solve the problem of distributing $v(S)$ fairly and uniquely among S:

1. Efficiency axiom: The payoffs for each player must add up to $v(N)$.

$$\sum_{i=1}^{n} \phi_i = v(N) \tag{2.6}$$

2. Symmetry axiom: If two players always contribute the same to each coalition, these players should receive the same payoff.

$$\text{If } v(S \cup \{i\}) = v(S \cup \{j\}) \text{ for all } S \subseteq N \setminus \{i, j\}, \text{ then } \phi_i = \phi_j \tag{2.7}$$

3. Additivity axiom: If there are two games with payoff functions v_1 and v_2, and another game is created where $v_3(S) = v_1(S) + v_2(S)$ for all coalitions S, then a player's payoff under the single game (ϕ_i^3) should be the sum of the payoff in two separate games.

$$\text{If } v_3(S) = v_1(S) + v_2(S) \text{ for all } S \subseteq N, \text{ then for all } i \in N, \phi_i^3 = \phi_i^1 + \phi_1^2 \tag{2.8}$$

4. Dummy player axiom: If there is a player that does not contribute to the payoff of any coalition, then that player should receive a 0 payoff.

$$\text{If } v(S \cup \{i\}) = v(S) \text{ for every } S \subset N \setminus \{i\}, \text{ then } \phi_i = 0 \tag{2.9}$$

Shapley has produced a unique solution that satisfies the axioms listed above to distribute surplus fairly among players in a coalitional game with N players and the payoff function v. The result is called the Shapley value of each player:

$$\phi_i = \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!(|N| - |S| - 1)!}{|N|!} \Big(v(S \cup \{i\}) - v(S) \Big) \tag{2.10}$$

The Shapley value for each player can be understood as the average marginal contribution over all the different coalitions they can be a part of.

Shapley values were first used in [42] in a machine learning setting to explain individual classifications. In a machine learning setting, the concept of players is mapped to features. We assume a model to have $N = \{1, 2, ..., n\}$ features.

Authors of [43] starts by defining a model's prediction, $f(x)$ as conditional to a subset of features' values being known:

$$f_S(x) = \mathbb{E}[f | X_i = x_i, i \in S], \text{ where } S \subseteq N \tag{2.11}$$

Using Equation (2.11), we can define the contribution of S to be:

$$\Delta_S(x) = f_S(x) - f_\emptyset(x) \tag{2.12}$$

$\Delta_S(x)$, further called as contribution function, corresponds to the payoff function $v(S)$ in the game theory explanation. With this definition of contribution, [42] defines the contribution of a feature for a given data point x to be:

$$\varphi_i(x) = \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!(|N| - |S| - 1)!}{|N|!} \Big(\Delta_{S \cup \{i\}}(x) - \Delta_S(x) \Big) \tag{2.13}$$

Since we know that from Equation (2.12) $\Delta_S(x) = f_S(x) - f_\emptyset(x)$, we can further simplify Equation (2.13) as:

$$\varphi_i(x) = \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!(|N| - |S| - 1)!}{|N|!} \Big(f_{S \cup \{i\}}(x) - f_S(x) \Big) \tag{2.14}$$

Since calculating Shapley values exactly for a single data point requires re-training the model and evaluating the result for each possible coalition, the operations required for exact calculation increase exponentially with respect to the number of features. A sampling-based approximation algorithm has been proposed in [43]. This approximation assumes that individual features in a given data point are independently distributed.

The approximation works by sampling: given a data point x, we are trying to assign Shapley values to feature i. In each iteration, we sample a data point $w \in \mathcal{X}$, where $\mathcal{X}$ is the data set, and a permutation of the feature indices, uniformly at random, $O \in \pi(n)$, where $\pi(n)$ is the set of all permutations of n items. We construct two data points z_1 and z_2, that only differ in their values for feature i. For features that come before i in the permutation O, $Pre^i(O)$, the values are taken from x, and for features that come after i in the permutation O, $Post^i(O)$, the values are taken from w. In z_1, the value of the i^{th} feature comes from x, whereas in z_2, the value of the i^{th} feature comes from w. We evaluate the model for z_1 and z_2, and take the differences. The mean of these differences gives us an approximate value. The proof can be found in [42].

Algorithm 1: Shapley value approximation as described in [43]

Input: Data point to explain x, data set $\mathcal{X}$, model function f

for $i \leftarrow 0$ **to** n **do**
  $\varphi_i(x) \leftarrow 0$
  for $k \leftarrow 1$ **to** m **do**
    select a random permutation $O \in \pi(n)$
    select a random data point $w \in \mathcal{X}$
    construct two instances:
    $z_1 \leftarrow (x_{Pre^i(O)_1}, ..., , x_i, w_{Post^i(O)_1}, ...)$
    $z_2 \leftarrow (x_{Pre^i(O)_1}, ..., , w_i, w_{Post^i(O)_1}, ...)$
    $\varphi_i(x) \leftarrow \varphi_i(x) + f(z_1) - f(z_2)$
  end
  $\varphi_i(x) \leftarrow \varphi_i(x)/m$
end

Shapley values are widely used in practice. In [6], authors have found that Shapley values were the most common method for providing feature importance explanation.

Global Shapley values can be calculated to reach at global feature importances. Since Shapley values represent feature importances for a single instance, global Shapley values can be calculated by iterating over instances in a data set. Denoting global feature importance for feature i to be Φ_i, the explicit approximate calculation of global Shapley values is given in Equation (2.15).

$$\Phi_i = \frac{1}{K} \sum_{k=i}^{K} |\phi_i(x^{(k)})| \tag{2.15}$$

where K is the number of instances in the data set. K can also be chosen to be a random sample from the data set, however, the approximation will be less accurate.

2.3.3 Shapley Additive Explanations (SHAP)

Presented in [21], SHAP values combine several model-agnostic explanation methods, including LIME and Shapley values under a unified measure to determine feature importance. They define additive feature attribution methods as methods that have an explanation model that is a linear function of binary variables:

$$g(z') = \phi_0 + \sum_{i=1}^{m} \phi_i z'_i \tag{2.16}$$

where z' is an input of binary variables $z' \in \{0,1\}^m$, m is the dimension of the binary input z', and $\phi_i \in \mathbb{R}$ is the coefficient denoting the attribution or contribution of the i^{th} feature. In essence, $g(z')$ can be seen as a simple linear regression model, where the input z' has m binary features, and ϕ_i is the coefficient denoting the contribution of feature i.

Kernel SHAP

Kernel SHAP is a model-agnostic method under the SHAP framework. Kernel SHAP modifies LIME as an additive feature attribution method, and reduces approximate Shapley value calculation to finding ϕ_i for each feature in a penalized linear regression model. Authors prove that when complexity, proximity, and loss functions are set as described in Equation (2.17), the solution to the Equation (2.5) gives Shapley values.

$$\begin{aligned} \Pi_{x'}(z') &= \frac{m-1}{\binom{m}{|z'|}|z'|(m-|z'|)} \\ \Omega(g) &= 0 \\ \mathcal{L}(f,g,\Pi_x) &= \sum_{z'\in\mathcal{Z}} \Pi_{x'}(z')\big(f(h_x(z')) - g(z')\big)^2 \end{aligned} \tag{2.17}$$

where $|z'|$ is the number of non-zero elements in z'. Like in LIME, $\mathcal{Z}$ represents a data set constructed of randomly perturbed data points in the simplified input space that are locally close to x', and h_x is the function used to convert from simplified input space to original input space, such that $h_x(x') = x$. The proof can be found in the supplementary material of [21].

Kernel SHAP reduces computing Shapley values to a penalized linear regression problem. Like LIME, the algorithm works by randomly sampling data points in the simplified input space around the locality of x', and uses the predictions $f(h_x(z'))$ as labels. Then, the binary vectors z' and labels $f(h_x(z'))$ are fit using linear regression.

Tree SHAP

While we cover Tree SHAP under post-hoc approaches, unlike the previous approaches that we have described, Tree SHAP is not a model-agnostic method. Tree SHAP is a model-specific method for computing exact Shapley values for tree-based models [20]. Tree based models include random forests, decision trees, gradient boosted trees, and the variations of gradient boosted trees such as XGBoost (eXtreme Gradient Boosting) [9], LightGBM [16], CatBoost [33], etc. Tree SHAP is very commonly used, since tree-based models are popular when dealing with tabular data. Tree SHAP is currently used at DLL to explain the predictions of various XGBoost models.

2.4 Sum Product Networks

In this work, we describe DGANN^2, with a decomposable architecture allowing the efficient calculation of approximate Shapley values. This calculation is motivated from Sum Product Networks (SPNs) [30]. To this end, we introduce them here.

Sum product networks are a newer type of probabilistic model similar to probabilistic graphical models (PGMs) such as Bayesian and Markov networks. In their essence, sum product networks are 'networks of sum and product operations with certain numeric inputs' [29]. They represent probability distributions over random variables, and can be seen as deep neural networks with only sum and product nodes. A simple sum product network representing a distribution over three independent binary random variables X, Y, Z looks like Figure 2.4. The joint distribution is represented at the product node, and its children each represent a marginal.

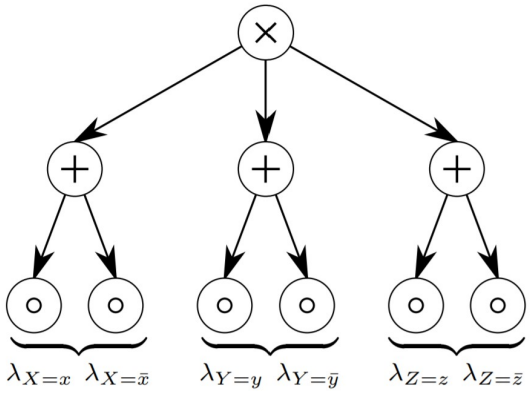

Figure 2.4: A simple SPN of three independent binary random variables [29]

The one aspect of SPNs that we will make use of in this work is marginalization of random variables. In probabilistic terms, a marginal distribution refers to the probability distribution of a subset of random variables S in a larger set X, while not assuming any knowledge about the other random variables that are not in S. Intuitively, marginalization calculates the probability distribution of the subset by considering all possible values of the variables that are not in the subset, and summing over them.

For a set of continuous random variables $X = \{X_1, X_2, ..., X_n\}$ with an integrable probability density function p_X, for a set $S \in X$ where $M = X \setminus X = \{M_1, M_2, ..., M_k\}$, we can find the marginal distribution of $p_S(s)$ as described in Equation (2.18).

$$p_S(s) = \int_{m_1} \int_{m_2} ... \int_{m_k} p_X(s, m_1, m_2, ..., m_k) dm_1 ... dm_k \tag{2.18}$$

In SPNs, marginalization of random variables can be done in a single pass through the network on the condition that decomposability is met [29]. A sum product network is decomposable if and only if a random variable appears no more than once in the children of any product node.

Chapter 3

Proposed Approach

3.1 Description of DGANN^2

In this work, we propose a generalized additive model that is inspired by GANNs and GA^{2}Ms, as described in Section 2.2.2 and Section 2.2.3, respectively.

The proposed model is a generalized additive model that has the following general form:

$$g(\mathbb{E}[y]) = \sum_{i=1}^{n} f_i(x_i) + \sum_{(i,j)\in\mathcal{I}} f_{ij}(x_i) \cdot f_{ji}(x_j) \tag{3.1}$$

where g is a link function, n is the number of features, $\mathcal{I}$ is a set of tuples denoting which interactions will be modeled and each of f_i and f_{ij} denotes a neural network with a single input and a single output neuron.

DGANN^2 is essentially a glass box model: for each data point x, we can clearly see how much each feature has contributed to the result, by examining the results of $f_i(x_i)$ and $f_{ij}(x_i)$. Furthermore, we can construct graphs of f_i and heatmaps of f_{ij} over their respective domains, and get an understanding of how feature interactions behave.

To select which interactions to include in the model, we first fit DGANN^2 without considering any interactions. Looking at the coefficients of the variables that DGANN^2 has learned, we select k features with the highest coefficients, and take the pairwise combinations from this subset.

The model architecture is quite similar to GAMI-Nets as described in [45]. The key difference between GAMI-Net and DGANN^2 is the construction of interaction terms. GAMI-Net architecture trains a single neural network for each interaction term, taking as input two values corresponding to the interacting variables. In contrast, DGANN^2 architecture trains a neural network for each interactive variable and then multiplies the results from each neural network at the end. The architecture described in this thesis allows for decomposability, which aids us in the efficient calculation of Shapley values as described below in Section 3.2.

As we pointed out before, in GAMs certain explainability measures such as getting feature importances for local explainability is not straightforward when interactions are present. Moreover, ranking features with global importance is also not directly possible from the model structure. Thus, for feature importance calculation, we turn to Shapley values.

3.2 Efficient Calculation of Shapley Values

The calculation of Shapley values and the corresponding approximation algorithm have been described in Section 2.3.2. The algorithm can be thought of as two nested loops:

1. Loop over different subsets of features $S \in N \setminus \{k\}$.

2. Sample a number of instances $w \in \mathcal{X}$, and evaluate $f(z_1) - f(z_2)$ for z_1 and z_2 as described in Algorithm 1.

In its core, the algorithm is constructing subsets $S \in N \setminus \{k\}$, and for each subset S, tries to find the approximate value of the prediction of f marginalized over features that are not in S. In essence, to find $f_{S\cup\{k\}}(x) - f_S(x)$, we would need to construct two models, one built with features in $S \cup \{k\}$, and one built with features in S, and find the difference of their predictions. Viewing from a holistic perspective of our model f with all features N, we are trying to marginalize f over features that are not in S.

$$f_S(x) = \int f(x_1, x_2, ..., x_n) d\,\mathbb{P}_{x_a, a \notin S} \tag{3.2}$$

We take integrals over the domains of each variable x_a such that $a \notin S$. Equation (3.3) re-writes Equation (3.2) as nested integrals.

$$f_S(x) = \int_{val(x_{a_1})} \int_{val(x_{a_2})} ... \int_{val(x_{a_k})} f(x_1, x_2, ..., x_n) p(x_{a_1}, x_{a_2}, ..., x_{a_k}) dx_{a_1} ... dx_{a_k} \tag{3.3}$$

where $k = |N \setminus S|, x_{a_j} \in N \setminus S$ for $j \in \{1, 2, ..., k\}$, and $val(x_{a_j})$ denotes the domain of variable x_{a_j}. We use the more concise version in Equation (3.2) in the rest of the section.

Algorithm 1 uses sampling from the data set to approximate $f_S(x)$, and also samples random subsets $S \in N$ to emulate coalitions. With the model described in Section 3.1, we can calculate Shapley values without the need for sampling, as we can arrive at a closed form solution. We make the assumption first that all the features are independent, as in the original approximation algorithm as described in Algorithm 1.

We show how the efficient marginalization works:

$$
\begin{aligned}
f_S(x) &= \int f(x_1, ..., x_n) d\,\mathbb{P}_{x_a, a \notin S} && (3.4)\\
&= \int \Big[\sum_{i=1}^{n} f_i(x_i) + \sum_{(i,j)\in\mathcal{I}} f_{ij}(x_i) \cdot f_{ji}(x_j) \Big] d\,\mathbb{P}_{x_a, a \notin S} \\
&= \int \sum_{i \in S} f_i(x_i) d\,\mathbb{P}_{x_a, a \notin S} + \int \sum_{i \notin S} f_i(x_i) d\,\mathbb{P}_{x_a, a \notin S} + \int \Big[\sum_{(i,j)\in\mathcal{I}} f_{ij}(x_i) \cdot f_{ji}(x_j) \Big] d\,\mathbb{P}_{x_a, a \notin S} \\
&= \sum_{i \in S} f_i(x_i) \overbrace{\int d\,\mathbb{P}_{x_a, a \notin S}}^{=1} + \int \sum_{i \notin S} f_i(x_i) d\,\mathbb{P}_{x_a, a \notin S} + \int \Big[\sum_{(i,j)\in\mathcal{I}} f_{ij}(x_i) \cdot f_{ji}(x_j) \Big] d\,\mathbb{P}_{x_a, a \notin S} \\
&= \sum_{i \in S} f_i(x_i) + \int \sum_{i \notin S} f_i(x_i) d\,\mathbb{P}_{x_a, a \notin S} + \int \Big[\sum_{(i,j)\in\mathcal{I}} f_{ij}(x_i) \cdot f_{ji}(x_j) \Big] d\,\mathbb{P}_{x_a, a \notin S} \\
&= \sum_{i \in S} f_i(x_i) + \sum_{i \notin S} \int f_i(x_i) p(x_i) dx_i + \int \Big[\sum_{(i,j)\in\mathcal{I}} f_{ij}(x_i) \cdot f_{ji}(x_j) \Big] d\,\mathbb{P}_{x_a, a \notin S} && (3.5)\\
&= \sum_{i \in S} f_i(x_i) + \sum_{i \notin S} \mathbb{E}[f_i(x_i)] + \int \Big[\sum_{(i,j)\in\mathcal{I}} f_{ij}(x_i) \cdot f_{ji}(x_j) \Big] d\,\mathbb{P}_{x_a, a \notin S} && (3.6)
\end{aligned}
$$

In Equation (3.5), we make use of the assumption that the variables are independently distributed to decompose $p(x_i, x_{i+1}, ..., x_k) = p(x_i) \cdot p(x_{i+1}) ... \cdot p(x_k)$.

From Equation (3.6), we will focus only on the remaining integral part:

$$
\begin{aligned}
\int \Big[\sum_{(i,j)\in\mathcal{I}} f_{ij}(x_i) \cdot f_{ji}(x_j) \Big] d\,\mathbb{P}_{x_a, a \notin S} = &\int \sum_{(i,j)\in\mathcal{I}\ \&\ i,j\in S} f_{ij}(x_i) \cdot f_{ji}(x_j) d\,\mathbb{P}_{x_a, a \notin S} \\
&+ \int \sum_{(i,j)\in\mathcal{I}\ \&\ i\in S\ \&\ j\notin S} f_{ij}(x_i) \cdot f_{ji}(x_j) d\,\mathbb{P}_{x_a, a \notin S} \\
&+ \int \sum_{(i,j)\in\mathcal{I}\ \&\ i\notin S\ \&\ j\in S} f_{ij}(x_i) \cdot f_{ji}(x_j) d\,\mathbb{P}_{x_a, a \notin S} \\
&+ \int \sum_{(i,j)\in\mathcal{I}\ \&\ i,j\notin S} f_{ij}(x_i) \cdot f_{ji}(x_j) d\,\mathbb{P}_{x_a, a \notin S} && (3.7)
\end{aligned}
$$

We look at each component of Equation (3.7) separately:

$$
\begin{aligned}
\int \sum_{(i,j)\in\mathcal{I}\ \&\ i,j\in S} f_{ij}(x_i) \cdot f_{ji}(x_j) d\,\mathbb{P}_{x_a, a \notin S} &= \sum_{(i,j)\in\mathcal{I}\ \&\ i,j\in S} f_{ij}(x_i) \cdot f_{ji}(x_j) \int d\,\mathbb{P}_{x_a, a \notin S} \\
&= \sum_{(i,j)\in\mathcal{I}\ \&\ i,j\in S} f_{ij}(x_i) \cdot f_{ji}(x_j) && (3.8)
\end{aligned}
$$

$$
\begin{aligned}
\int \sum_{(i,j)\in\mathcal{I}\ \&\ i\in S\ \&\ j\notin S} f_{ij}(x_i) \cdot f_{ji}(x_j) d\,\mathbb{P}_{x_a, a \notin S} &= \sum_{(i,j)\in\mathcal{I}\ \&\ i\in S\ \&\ j\notin S} f_{ij}(x_i) \cdot \int f_{ji}(x_j) d\,\mathbb{P}_{x_a, a \notin S} \\
&= \sum_{(i,j)\in\mathcal{I}\ \&\ i\in S\ \&\ j\notin S} f_{ij}(x_i) \cdot \mathbb{E}[f_{ji}(x_j] && (3.9)
\end{aligned}
$$

$$\int \sum_{(i,j)\in\mathcal{I}\ \&\ i\notin S\ \&\ j\in S} f_{ij}(x_i)\cdot f_{ji}(x_j)d\,\mathbb{P}_{x_a,a\notin S} = \sum_{(i,j)\in\mathcal{I}\ \&\ i\notin S\ \&\ j\in S} f_{ji}(x_j)\cdot\int f_{ij}(x_i)d\,\mathbb{P}_{x_a,a\notin S}$$
$$= \sum_{(i,j)\in\mathcal{I}\ \&\ i\notin S\ \&\ j\in S} f_{ji}(x_j)\cdot\mathbb{E}[f_{ij}(x_i)] \tag{3.10}$$

$$\int \sum_{(i,j)\in\mathcal{I}\ \&\ i,j\notin S} f_{ij}(x_i)\cdot f_{ji}(x_j)d\,\mathbb{P}_{x_a,a\notin S} = \sum_{(i,j)\in\mathcal{I}\ \&\ i,j\notin S}\int f_{ij}(x_i)\cdot f_{ji}(x_j)d\,\mathbb{P}_{x_a,a\notin S}$$
$$= \sum_{(i,j)\in\mathcal{I}\ \&\ i,j\notin S}\int f_{ij}(x_i)\cdot f_{ji}(x_j)\cdot p(x_i)\cdot p(x_j)dx_i dx_j d\,\mathbb{P}_{x_a,a\notin S\cup\{i,j\}}$$
$$= \sum_{(i,j)\in\mathcal{I}\ \&\ i,j\notin S}\mathbb{E}[f_{ij}(x_i)]\cdot\mathbb{E}[f_{ji}(x_j)] \tag{3.11}$$

Putting everything together, we arrive at the following result for Equation (3.2):

$$\begin{aligned} f_S(x) &= \int f(x_1,x_2,...,x_n)d\,\mathbb{P}_{x_a,a\notin S} \\ &= \sum_{i\in S} f_i(x_i) + \sum_{i\notin S}\mathbb{E}[f_i(x_i)] + \sum_{(i,j)\in\mathcal{I}\ \&\ i,j\in S} f_{ij}(x_i)\cdot f_{ji}(x_j) + \sum_{(i,j)\in\mathcal{I}\ \&\ i\in S\ \&\ j\notin S} f_{ij}(x_i)\cdot\mathbb{E}[f_{ji}(x_j)] \\ &+ \sum_{(i,j)\in\mathcal{I}\ \&\ i\notin S\ \&\ j\in S} f_{ji}(x_j)\cdot\mathbb{E}[f_{ij}(x_i)] + \sum_{(i,j)\in\mathcal{I}\ \&\ i,j\notin S}\mathbb{E}[f_{ij}(x_i)]\cdot\mathbb{E}[f_{ji}(x_j)] \end{aligned} \tag{3.12}$$

Thus, as can be seen in Equation (3.12), in our model we can marginalize over features without having to repeatedly sample for approximation. The expected runtime for this calculation is $O(n+|\mathcal{I}|)$, provided that the expected values of each of the functions in the additive model are calculated once during training, and are available in the form of a dictionary. To calculate the expected values of each function, or each neural network, we use the training set to arrive at an empirical expected value. Explicitly, we sum the results of each network over the data points in the training set, and divide by the number of data points. Another approach we have tried is simulating a Gaussian distribution from the data, from each feature, to arrive at an expected value. However, our experiments have shown that using empirical expected value gives better results. Lastly, we only need to make one pass through the model with the data point x to get the results of 1-D and 2-D functions. However, to approximate Shapley values, we still need to iterate over possible subsets S of the feature set, $N=\{1,2,..,n\}$.

A way to incorporate these subsets would be sampling random permutations, as in Algorithm 1, and constructing two subsets. However, the formulation in Equation (3.12) allows us to calculate this in closed form, without resorting to sampling.

Recall that we calculate $f_S(x)$ in Equation (3.12) to plug it into the right-hand side of Equation (2.14). We are effectively marginalizing out the features that are not in S, to arrive at a substitute for $f_S(x)$. Let us denote the feature index we are interested in as k, and per the right-hand side of Equation (2.14), we take the difference $f_{S\cup\{k\}}(x)-f_S(x)$. We can see that $f_{S\cup\{k\}}(x)$ and $f_S(x)$ share some common terms, and we can simplify this difference further. First of all, looking at the first two additive terms that are not related to interactions in Equation (3.12), we can observe that when taking the difference $f_{S\cup\{k\}}(x)-f_S(x)$, the terms $\sum_{i\in S} f_i(x_i)$ cancel each other out. Furthermore, we see that the terms $\sum_{i\notin S\cup\{k\}}\mathbb{E}[f_i(x_i)]$ also cancel each other out. Thus, in building the difference we arrive at:

$$f_{S\cup\{k\}}(x) - f_S(x) = f_k(x_k) - \mathbb{E}[f_k(x_k)] + R \tag{3.13}$$

where R is a placeholder for the remainder terms. Next, observe that for pairwise interaction tuples $(i,j) \in \mathcal{I}$, if $i \neq k$ and $j \neq k$, the multiplicative terms cancel each other out in the difference as well, since k is the only member of the sets S and $S \cup \{k\}$ that is different. Thus, we are only interested in pairwise interactions where k is one of the features interacting. Thus, when finding R:

$$f_{S\cup\{k\}}(x) = ... + \sum_{(k,j)\in\mathcal{I}\ \&\ j\in S} f_{kj}(x_k)\cdot f_{jk}(x_j) + \sum_{(k,j)\in\mathcal{I}\ \&\ j\notin S} f_{kj}(x_k)\cdot\mathbb{E}[f_{jk}(x_j)] \tag{3.14}$$

$$f_S(x) = ... + \sum_{(k,j)\in\mathcal{I}\ \&\ j\in S} f_{jk}(x_j)\cdot\mathbb{E}[f_{kj}(x_k)] + \sum_{(k,j)\in\mathcal{I}\ \&\ j\notin S} \mathbb{E}[f_{kj}(x_k)]\cdot\mathbb{E}[f_{jk}(x_j)] \tag{3.15}$$

Thus, when finding the remainder R in the difference:

$$R = \sum_{(k,j)\in\mathcal{I}\ \&\ j\in S} \Big(f_{kj}(x_k) - \mathbb{E}[f_{kj}(x_k)]\Big)\cdot f_{jk}(x_j) + \sum_{(k,j)\in\mathcal{I}\ \&\ j\notin S} \Big(f_{kj}(x_k) - \mathbb{E}[f_{kj}(x_k)]\Big)\cdot\mathbb{E}[f_{jk}(x_j)] \tag{3.16}$$

Putting it all together, we arrive at the difference:

$$\begin{aligned} f_{S\cup\{k\}}(x) - f_S(x) = f_k(x_k) - \mathbb{E}[f_k(x_k)] &+ \sum_{(k,j)\in\mathcal{I}\ \&\ j\in S} \Big(f_{kj}(x_k) - \mathbb{E}[f_{kj}(x_k)]\Big)\cdot f_{jk}(x_j) \\ &+ \sum_{(k,j)\in\mathcal{I}\ \&\ j\notin S} \Big(f_{kj}(x_k) - \mathbb{E}[f_{kj}(x_k)]\Big)\cdot\mathbb{E}[f_{jk}(x_j)] \end{aligned} \tag{3.17}$$

We reach a closed form for the difference. Recall that S as we use it here refers to the $Pre^k(O)$, where O is a permutation in the set of permutations of n features, $\pi(n)$. Thus, S denotes the features in O that come before k, the feature we're computing the Shapley value for. Observe that in Equation (3.17), for each feature j where $(k,j) \in \mathcal{I}$, we have two possible options: either $j \in S$ or $j \notin S$. Our goal is to iterate over every single permutation $O \in \pi(n)$, where $S = Pre^k(O)$, and divide by the number of iterations. Observe that exactly in half of these permutations, $j \in S$, and in the other half, $j \notin S$. In other words, exactly in half of these permutations j will be before k, and exactly in half of these permutations j will be after k. Thus, when calculating the Shapley value for feature k, we will have:

$$\phi_k(x) = f_k(x_k) - \mathbb{E}[f_k(x_k)] + \sum_{(k,j)\in\mathcal{I}} \Big(f_{kj}(x_k) - \mathbb{E}[f_{kj}(x_k)]\Big)\Big(0.5\cdot f_{jk}(x_j) + 0.5\cdot\mathbb{E}[f_{jk}(x_j)]\Big) \tag{3.18}$$

Equation (3.18) gives us a closed form of calculating Shapley values for a single instance. We denote this closed form calculation as D-Shapley, as in Decomposable Shapley approximation. Recall that these values are not exact, since in the original Shapley construction, $f_{S\cup\{k\}}(x)$ would be a model trained using the subset of features $S\cup\{k\}$, and f_S would be a model trained using the subset of features S. Having pointed this out, the values calculated using Equation (3.18) are exact for the approximation algorithm, since we do not need to sample data points nor subsets $S \in N$.

The closed form solution thus is able to arrive at the exact values without explicit iterations that would have an exponential complexity in the number of features.

DGANN^2 and D-Shapley can further be generalized to include interactions of higher order. While we will not derive the equation here, we make the following observations for 3-way interactions: For each 3-way interaction $(i, j, k) \in \mathcal{I}_3$, we have 3 functions $f_{ijk}(x_i), f_{jik}(x_j), f_{kij}(x_k)$. Our model would look like Equation (3.19).

$$g(\mathbb{E}[y]) = \beta + \sum_{i=1}^{n} f_i(x_i) + \sum_{(i,j)\in\mathcal{I}_2} f_{ij}(x_i) \cdot f_{ji}(x_j) + \sum_{(i,j,k)\in\mathcal{I}_3} f_{ijk}(x_i) \cdot f_{jik}(x_j) \cdot f_{kij}(x_k) \quad (3.19)$$

We can plug this equation in to Equation (3.2), and again assuming independence of variables, we can deconstruct the function in a similar way. This is also apparent when we construct a SPN to represent the model. The SPN would still be decomposable, and so we can easily marginalize out a given subset of variables. This fast marginalization leads to a closed form equation, just as in Equation (3.18).

Chapter 4

Data and Methodology

4.1 Description of Data Sets

4.1.1 Data from DLL

Data from DLL comes from a consolidation of different tables. The features of interest can be broadly categorized into debtor information and invoice information. Features on debtors include identifying personal information, such as debtor's address and contact information, along with information about debtor's standing within DLL, such as the debtor's internal risk rating. Features on invoices include information about particular invoices, such as the amount of invoice, due date, overdue interest from the given invoice. The data from different tables are merged, and a consolidated file is created. The granularity of the data is on a daily level for each debtor, as the C&R agents receive daily lists of debtors to reach. Since each debtor can have multiple contracts and multiple invoices over these contracts, variables related to these contracts and invoices are aggregated on a daily level. Feature selection is then performed with expert input and identifying features are discarded from the data set. Several rounds of row selection are performed in accordance with the scope of the project and recommendations of the experts. A list of the model features can be found in Appendix A.

After the row selection, aggregation, and feature selection steps, the resulting data set contains 47943 rows, each row representing a debtor's situation on a given day, and 49 features for each row. The target variable, y, is a binary variable to predict whether a debtor will go 17 days past due on their invoice. When $y = 0$, i.e. the debtor will not go 17 days past due, we call them an on-time debtor, whereas when $y = 1$, we call them a late debtor.

Out of the 47493 rows, 38763 (81.6%) have the target variable 1, while 9180 (19.4%) have the target variable 0. The data set thus has an imbalance. As per this imbalance, we check different metrics and not simply accuracy while training and testing DGANN^2 and the other models.

Since neural networks cannot naturally handle categorical features, they are encoded using one-hot encoding for DGANN^2. One-hot encoding is commonly used to represent a categorical variable x_i of d distinct values as d binary variables, where only the binary variable corresponding to x_i's value is set to 1 [17]. This one-hot encoding is not used for GAMs and GA^{2}M s, since they can handle categorical features natively. Furthermore, numerical features in the data set are scaled using a standard scaler.

4.1.2 Other Classification Data Sets

Bank Marketing Data Set

Bank marketing data set [24] contains information about the marketing campaigns of a Portuguese bank. The marketing campaign was run through phone calls. The goal is to predict whether a client that is called will subscribe a term deposit. The data set consists of 41188 rows, with 20 features. Like the data set from DLL, bank marketing data set also shows a class imbalance, with only 6557 (12.38%) instances with positive target variables. The feature 'duration' is not considered in the model, since it is very highly correlated to the output, as duration of 0 indicates a negative output.

Diabetes Data Set

Obtained from UCI ML repository [12], Pima Indians Diabetes data set contains patient information and medical readings related to diabetes. The target is a binary variable denoting whether a patient tested positive for diabetes.

Banknote Authentication Data Set

Banknote authentication data set [12] consists of 4 features obtained from the wavelet transform of images of banknotes. The goal is to determine whether a banknote is authentic.

MAGIC Gamma Telescope Data Set

This data set [12] contains simulated observations to be able to differentiate between gamma rays (signal) and hadronic showers (background) when a Major Atmospheric Gamma-Ray Imaging Cherenkov (MAGIC) Gamma Telescope is used to register high energy gamma particles. Since the data is simulated, the ratio of positive events, i.e. gamma ray registrations are higher than normal. For the data set, simple accuracy measures are not suitable, since it is important to not classify background events as a signal. The data set contains 19020 instances, with 10 features.

4.1.3 Regression Data Sets

Parkinsons Data Set

Parkinsons data set [44] consists of biomedical voice measurements from 42 patients with early stage Parkinson's disease. The goal of the data set is to predict the Parkinson's disease symptom score on the UPDRS scale from 16 voice measures. For each patient, there are around 200 recordings, for a total of 5875 recordings.

California Housing Data Set

A well-known data set in the field of machine learning, California Housing data set is used to train a model to predict housing prices in California in 1990, based on location-related and house-related features. The goal is to predict the median house value for a given block. The description of the variables in the data set can be found in Appendix A under Table A.2. The data set contains 20640 instances, with 9 features.

Abalone Data Set

This data set [12] contains features about the characteristics of abalones, such as length, height, and weight, in order to predict their age. The data set contains 4177 instances, with 8 features.

Airfoil Self-Noise Data Set

This data set [12] is from a series of aerodynamic and acoustic tests, aimed at predicting the sound pressure level of airfoil blade sections. With only 5 features, this data set can be tested using all available interactions. The data set contains 1503 instances.

4.2 Practical Implementation of DGANN^2

To implement the model, we utilize Keras with Tensorflow 2.0 backend. The model has several hyperparameters that can be tweaked:

1. For each feedforward neural network, the number of hidden layers and the number of neurons in each hidden layer
2. Learning rate
3. Dropout rate
4. Number of interactions to consider

To determine the hyperparameters, we use grid search with 5-fold cross validation. However, since the number of neural networks in the model can be quite large, i.e. one for each feature and two for each interaction, we restrict our search space. We divide the neural network architectures into two groups, so that all feature networks have the same structure, and so do all interaction networks. Thus, instead of $n + 2|\mathcal{I}|$, we will only search for 2 architectures. While neural architecture search is being continuously researched, we deem it to be out of scope for this thesis.

Another point of interest is choosing the interactions to be included in the model. We achieve this by first training a DGANN^2 without pairwise interactions to arrive at a GANN. We use the contributions of each feature to be representative of the importance of the feature when the whole training data set is considered. We pick the top k features and get the pairwise combinations of these k features to arrive at the interactions. k can be tweaked as a hyperparameter and thus can be found using cross validation.

4.3 Experimental Setup

We test the performance of DGANN^2 against several baselines:

1. GAM: To model GAMs, we use Explainable Boosting Machines (EBMs), which are implemented using gradient boosting of bagged trees. To emulate GAMs, we explicitly restrict EBMs to not consider interactions. EBMs are open-sourced under the Python library `interpret` [26]. When training, the default parameters are used.
2. GA^{2}M: We again use EBMs with default parameters, only changing the number of interactions to be considered to fit the model. We set the number of interactions to be the same number as the interactions considered in DGANN^2 for the same data set.
3. XGBoost: eXtreme Gradient Boosting [9], a tree-based ensemble algorithm utilizing the gradient boosting framework, is one of the most popular ML algorithms in competitions like Kaggle. XGBoost is also currently used for the C&R project at DLL. The hyperparameters for XGBoost models in the experiments are tuned using randomized search.

While evaluating the models, we look at several metrics. For regression tasks, we use Root Mean Squared Error (RMSE). For classification tasks we look at:

1. AUC [22]: Area Under the ROC Curve. A receiver operating characteristic (ROC) curve measures the performance of a classifier at all classification thresholds. ROC curve plots true

positive rate vs. false positive rate. AUC is the calculation of the area under the ROC curve. AUC values range from 0 to 1, where higher values indicate a better performing model.

2. F1 Score [37]: Also referred to as F score, this metric measures the accuracy of a model based on precision and recall. The metrics below are calculated using True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN).

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{4.1}$$

$$Precision = \frac{TP}{TP + FP} \tag{4.2}$$

$$Recall = \frac{TP}{TP + FN} \tag{4.3}$$

$$F1 = \frac{2}{1/Recall + 1/Precision} = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{4.4}$$

We prefer F1 score since there is a class imbalance for the DLL data set, as simply looking at accuracy as defined in Equation (4.1) can give misleading results. For an extreme example, imagine that the model predicts 0 for every data point in the DLL data set. Since 81.6% of data points have 0 as a label, the model would have 81.6% accuracy. However, the model would not have discriminatory power, as we cannot identify and late debtors. F1 score, on the other hand, punishes such non-discriminatory models since it takes into account precision and recall explicitly. F1 score can be between 0 and 1, just as precision and recall, and a higher F1 score indicates better discriminatory power for a model.

Chapter 5

Results and Discussion

5.1 Performance Comparisons for DGANN^2

Table 5.1: AUC and F1 on classification data sets for different models. Each value represents the mean AUC/F1 ± one standard deviation from 5-fold cross validation. Higher AUC and higher F1 score is better.

Data Set	Metric	DGANN^2	EBM-GAM	EBM-GA^{2}M	XGBoost
DLL	AUC	$\mathbf{0.760 \pm 0.004}$	0.750 ± 0.009	0.751 ± 0.009	0.731 ± 0.018
	F1	0.863 ± 0.007	$\mathbf{0.892 \pm 0.006}$	$\mathbf{0.892 \pm 0.006}$	0.887 ± 0.008
Bank	AUC	0.712 ± 0.010	0.738 ± 0.022	$\mathbf{0.740 \pm 0.017}$	0.724 ± 0.021
	F1	0.322 ± 0.027	0.194 ± 0.027	0.239 ± 0.024	$\mathbf{0.360 \pm 0.018}$
Diabetes	AUC	0.821 ± 0.040	0.832 ± 0.029	$\mathbf{0.835 \pm 0.028}$	0.822 ± 0.032
	F1	$\mathbf{0.659 \pm 0.018}$	0.608 ± 0.029	0.640 ± 0.042	0.612 ± 0.058
Banknote Authentication	AUC	$\mathbf{1.0 \pm 0.0}$	$\mathbf{1.0 \pm 0.0}$	0.999 ± 0.0006	0.999 ± 0.0005
	F1	0.998 ± 0.003	$\mathbf{1.0 \pm 0.0}$	0.997 ± 0.003	0.983 ± 0.011
MAGIC	AUC	0.916 ± 0.006	0.903 ± 0.003	0.920 ± 0.007	$\mathbf{0.932 \pm 0.005}$
	F1	0.788 ± 0.012	0.781 ± 0.005	0.803 ± 0.006	$\mathbf{0.811 \pm 0.007}$

It can be seen from Table 5.1 that DGANN^2 performs better in terms of AUC on the primary DLL data set, while it's slightly worse than the other methods when it comes to F1 score. On the regression tasks, as can be seen from Table 5.2, DGANN^2 performs slightly worse than the models it's compared against. We can attribute this performance drop to possible convolution of the model by forcing the model to explicitly consider certain interactions. This drop can be possibly addressed with a more advanced automated interaction detection algorithm.

Generally speaking, DGANN^2 shows comparative performance to the benchmark models in classification tasks. In regression tasks, XGBoost outperforms the other models. However, DGANN^2 can still be considered as an alternative when the additional explainability can be useful.

Table 5.2: RMSE on regression data sets for different models. Each value represents the mean RMSE ± one standard deviation from 5-fold cross validation. Lower RMSE is better.

Data Set	DGANN^2	GAM	GA^{2}M	XGBoost
Parkinsons	1.224 ± 0.132	1.216 ± 0.209	1.259 ± 0.236	$\mathbf{1.205 \pm 0.186}$
Housing	0.789 ± 0.105	0.732 ± 0.073	0.709 ± 0.056	$\mathbf{0.666 \pm 0.048}$
Abalone	2.209 ± 0.556	2.289 ± 0.526	2.217 ± 0.488	$\mathbf{2.162 \pm 0.523}$
Airfoil	4.133 ± 1.051	5.132 ± 1.346	$\mathbf{3.714 \pm 0.696}$	$\mathbf{3.676 \pm 1.087}$

5.2 Explainability of DGANN^2

Since DGANN^2 is essentially a glass box model, we can visualize how each variable contributes to the housing price in a line plot for 1D functions, and in heatmaps for interaction functions. On top of these visualizations, we can efficiently calculate Shapley values to get feature importances, both on a local and global level, using D-Shapley. In the following subsections, we look into two data sets: DLL data and California Housing data.

5.2.1 DLL Data

The initial GAM model, trained as DGANN^2 without interactions, identified `country`, `min_principle_amount_invoice`, `max_days_past_due`, `max_principle_amount_invoice`, `sum_principle_amount_invoice`, `min_amount_invoice` as the features to be considered for pairwise interactions. Most of these features are related to invoice information, such as the principal amount and total invoice amount. The explanations of these variables can be found in Table A.1.

DGANN^2 trained with the pairwise interactions of features mentioned above, yields several explanations. The contributions of a few selected features and interactions can be seen in Figure 5.1 and Figure 5.2, respectively. The contribution plots for all features and all interactions can be found in the appendix, specifically Figures A.1 to A.4.

Recall that a classification of 1 means that the debtor will be more than 17 days past due on their invoices, i.e. they will be classified as late. The positive contribution of a feature indicates that it is pushing the prediction towards a late classification. Looking at the global feature plots in Figure 5.1, we can identify maximum principle amount invoice and overdue invoices to be important in deciding a debtor's payment behavior, particularly in deciding that a debtor will pay in time, since a large part of their contributions negatively influence the outcome. We can also see that the features maximum days past due and average days between due and invoice date positively influence the outcome. These explanations can be used at DLL to validate the behaviour of the model with expert input. They can also lead to the discovery of patterns that were not considered by experts before.

Looking at the interaction plots in Figure 5.2, we can make a couple of observations:

- From the top left plot, we observe that debtors with large invoice amounts and short payment periods are more likely to pay their debts late.
- From the bottom middle plot, we see that debtors with invoices close to passing more than 17 days on their due date and large invoice amounts are more likely to pay their debts late.
- From the bottom right plot, we observe that the model considers debtors at the beginning of their payment period and with large principal payments are less likely to go past due on their invoices.

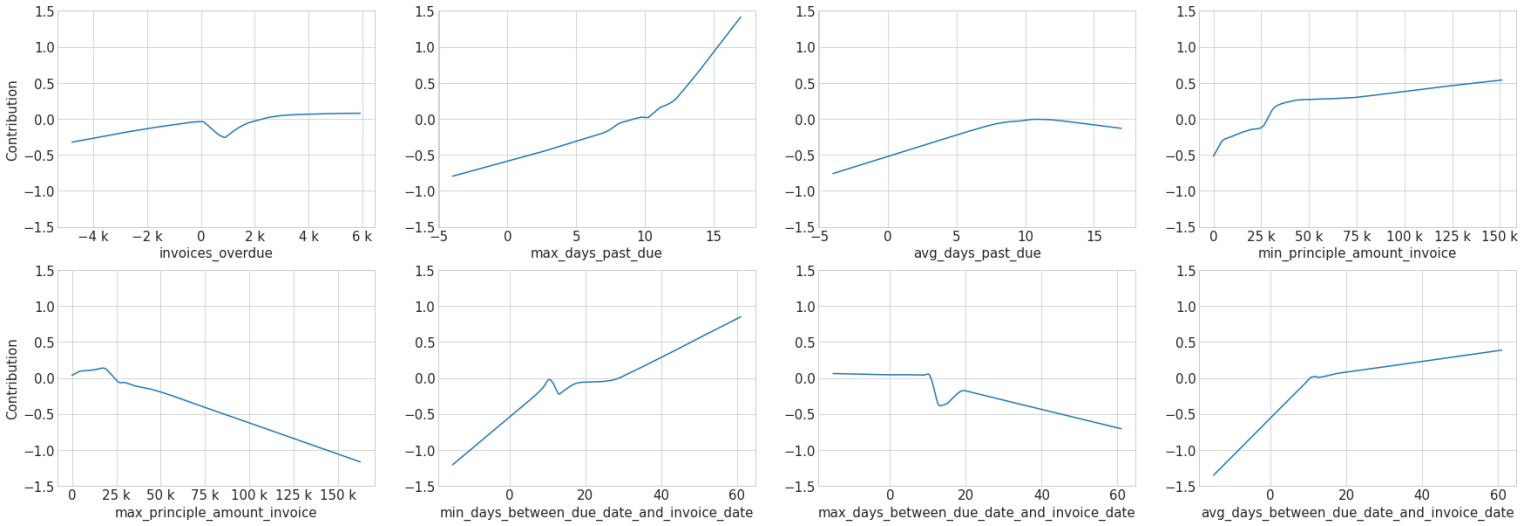

Figure 5.1: Global interpretation of a subset of 1D features for DGANN^2 trained on DLL data set. All plots share the same y-scale.

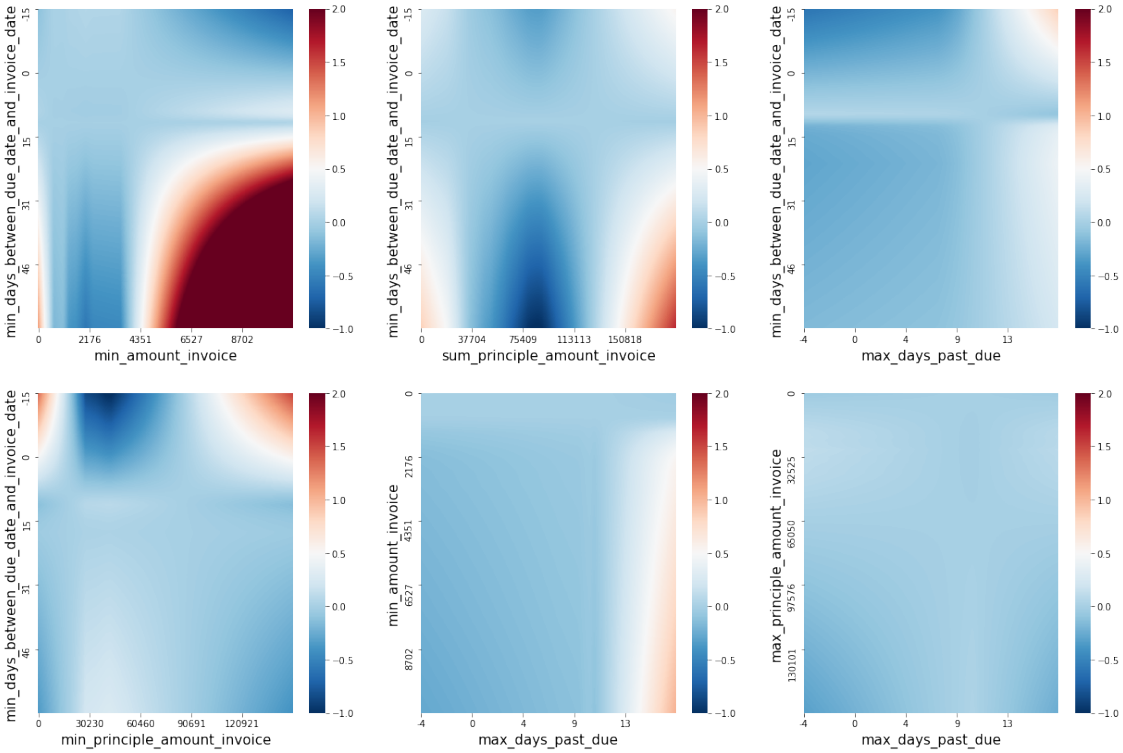

Figure 5.2: Global interpretation of a subset of interaction features for DGANN^2 trained on DLL data set.

Furthermore, to get feature importances for the main features, global Shapley values can be calculated using D-Shapley. The calculated global Shapley values can be seen in Figure 5.3. Since the categorical features are one-hot encoded, each categorical feature is represented as separate features, one for each class. Note that the Shapley values as presented in Figure 5.3 are approximately 1000 times the actual Shapley values. The actual Shapley values are rather small, since the problem at hand is a binary classification problem, and so each value is scaled to allow for easier interpretation. Globally, the trained model considers `max_days_past_due, x1_724V7, max_principle_amount_invoice` to be the most important features. This is consistent with the plots that have been gathered from the 1D functions. The global Shapley values can provide insight into the decision making processes of the model. For example, `x1_724V7` refers to the categorical variable 'reason for technical default', and the value `724V7` refers to a Corona payment plan. Even though the global Shapley values do not show in which direction the features affect the outcome, it can be hypothesized that when debtors have defaulted on their previous payments due to Corona, the model considers this as a sign that they might not be able to pay newer invoices on time. Another influential feature, `max_days_past_due` refers to the maximum days past after the due date among the invoices of a debtor. It can be hypothesized that the closer a debtor gets to being 17 days past due, the more likely the model is to consider the debtor to be late. This can also be in the 1D feature plot for the feature `max_days_past_due` in Figure 5.1. On the other end of the spectrum, the irrelevant features can be observed, such as `indicate_aftercare` and `x3_AI-Direct-Debit`. These binary features share the same value across all data points in the data set, and should have been excluded from the development data set in the feature selection steps. Similarly, `indication_forbearance` also doesn't influence the outcome of the model, and can be excluded as well. Such observations from global Shapley values can help the developers perform better feature selection.

It is further possible to delve into a single instance and explain how the model behaves specifically for that instance. For a particular instance, referred to as Debtor A from now on, the calculated Shapley values can be seen in Figure 5.4. Debtor A is a late debtor, i.e. they will be more than 17 days past due on their payment. The prediction for Debtor A is 0.96, meaning the model is quite certain in its prediction that Debtor A is a late debtor. As can be observed, the most influential feature is `x1_724V7`, which refers to the Corona payment plan. This debtor has previously defaulted on a payment due to Corona, and the model considers this to be highly informative. It can also be seen that 'maximum days past due' and 'minimum days past due' are influential. Debtor A has 3 invoices, each 14 days past due, so the values for 'maximum days past due' and 'minimum days past due' are both 14. Since Debtor A is close to going past 17 days on the due date for each of their invoices, it makes sense that the model considers these features to be influential.

On the other hand, the values as shown in Figure 5.5 are obtained when Shapley values are calculated for an on-time debtor. This debtor, referred to as Debtor B from now on, will pay their debt before going 17 days past due. The model is able to predict this with a prediction of 0.05. As can be observed, the most influential features are related to the unpaid invoice amount and maximum days past due. This debtor has a single invoice, and this invoice is only 1 day past due. The model considers this to be an indicator that Debtor B is still likely to pay their invoice. It is not immediately clear how the model considers unpaid invoice amount as an indicator, however, Debtor B's unpaid invoice amount is in the 80th percentile, meaning they have a rather large unpaid invoice amount. This can be a direction of investigation to determine whether debtors with larger unpaid invoice amounts are more likely to pay their debts, perhaps because the interest they would pay would be higher. Another observation can be made towards days between due date and invoice date. For Debtor B, the payment period is rather small at 6 days. Since Debtor B has a single invoice, all the maximum, minimum, and average values point to 6 days, which is in the 25th percentile. The model could have discovered the trend in the data that debtors with smaller payment periods are more likely to pay their debts before going significantly past due.

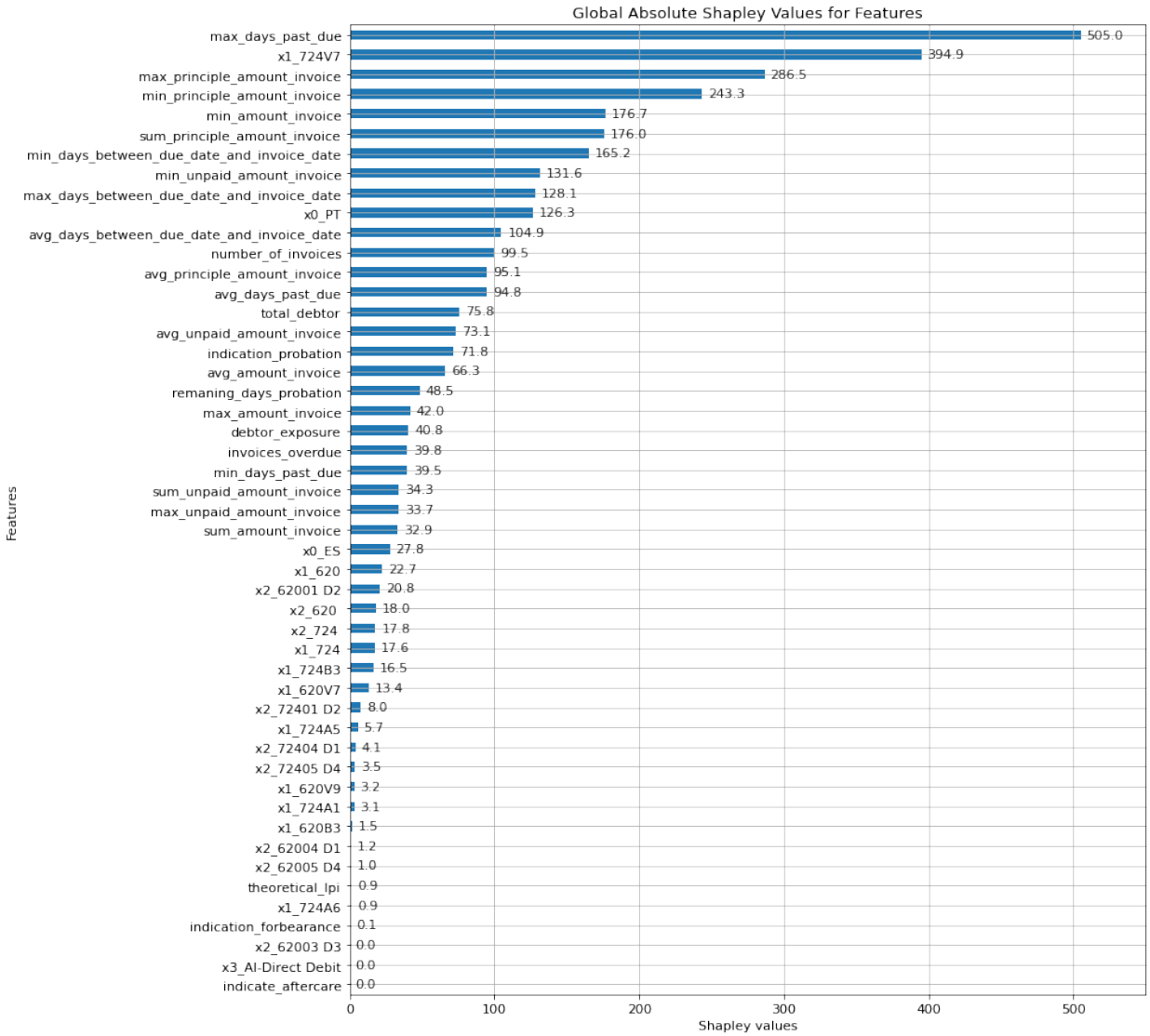

Figure 5.3: Scaled global Shapley values of features for DGANN^2 trained on DLL data set. The features are ranked from most important to least important according to their respective global Shapley values.

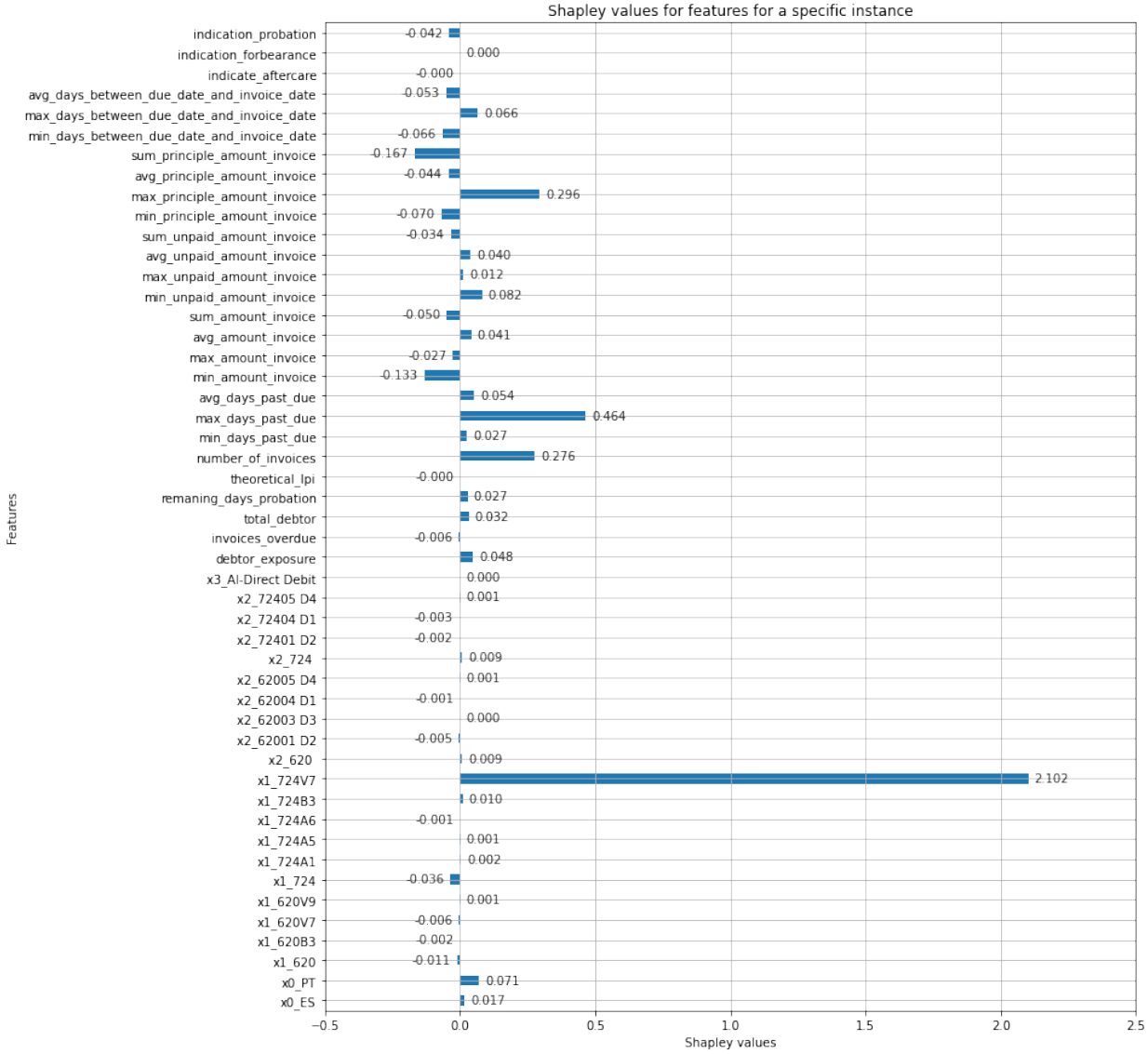

Figure 5.4: Local Shapley values for a specific late debtor, i.e. outcome = 1. The values are calculated on a DGANN^2 that was trained on the DLL data set using D-Shapley.

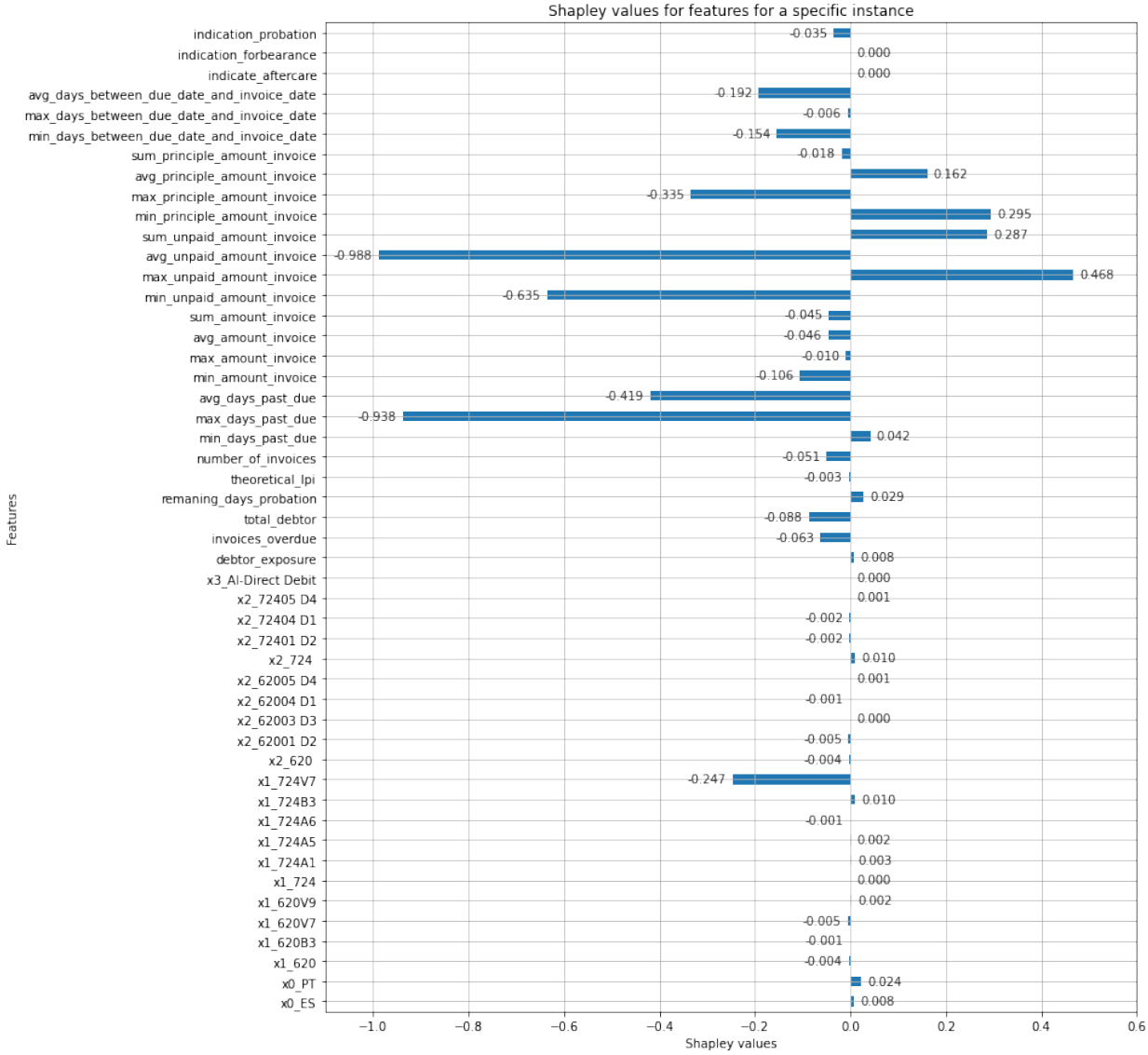

Figure 5.5: Local Shapley values for a specific on-time debtor, i.e. outcome = 0. The values are calculated on a DGANN^2 that was trained on the DLL data set using D-Shapley.

It can be seen with the examples provided that DGANN^2 provides several ways of explainability, both on a global and local level. These methods can help developers to validate and audit the model as well as to discover trends within the data. At DLL, currently deployed XGBoost paired with TreeSHAP is able to provide local and global Shapley value explanations as well. However, being a glass box model, DGANN^2 can also easily provide the feature and interaction plots in Figure 5.1 and Figure 5.2, along with Shapley values calculated with D-Shapley.

5.2.2 California Housing Data

For California housing data, we can interpret how each feature contributes to price from a global point of view with the plots in Figure 5.6. As can be seen, longitude, latitude, median income, and population have strong effects on the outcome. When checked, the peaks in the longitude and latitude graphs coincide with Los Angeles and San Francisco areas. LA area corresponds to the peak around longitude -118.4, and latitude 33.8. SF area corresponds to the peak around longitude -122.4, and the small stagnation around latitude 37.5. Another interesting observation comes from the plot for total rooms. It appears that to the model trained on the California housing data set, the number of total rooms within a block has absolutely no effect on the outcome. This can also be observed in the global Shapley values calculated on a subset of the data set as plotted in Figure 5.8. It appears that the number of total rooms has a global Shapley value of 0. We have run the experiments again, using a different subset of the data to train the model, and using a yet different subset to calculate global Shapley values. However, the effect of total rooms on the outcome stays at a minimal level. This warrants further investigation into the data.

The most influential feature for the trained model seems to be median income, as can be seen from Figure 5.8. This would make sense, as income and housing prices can be mutually reinforcing factors: neighbourhoods higher income households inhabit will have more luxurious houses built, and will in turn attract richer households.

From the top left plot of the interaction plots in Figure 5.7, it can be deduced that when population is low with higher median income, house values are higher. However, in the same plot, there is an interesting behaviour where house values also seem to be higher when population is higher and median income is lower. Such an observation can lead to doing extra sanity checks for the available data, or can influence the data selection process by pointing out the existence of outliers.

It is also possible to look at a single instance, and inspect which features influence the prediction of the model on that particular instance. For a particular instance, the local Shapley values are plotted in Figure 5.9. The values for this particular instance are given in Appendix A, in Table A.3. The predicted median house value for this instance is 300771, while the real value is 310900. Being in the 83$^{\text{rd}}$ percentile, this value is quite high for the data set. It can be seen that latitude and longitude are both influential for the rise in value. When checked, these coordinates correspond to Oxnard, a city very close to LA. As it was already observed on 1D plots, LA and SF areas have higher housing prices. A feature that brings the prediction down is median income. Even though the median income of the block is higher compared to the data set at 43693 USD, compared to the area this income is at the 60$^{\text{th}}$ percentile. The population is also a feature that brings the prediction lower, and this can be explained in a similar way to the income. The population in the block is at the 80$^{\text{th}}$ percentile, and judging from the 1D plots, larger populations tend to bring the housing prices lower.

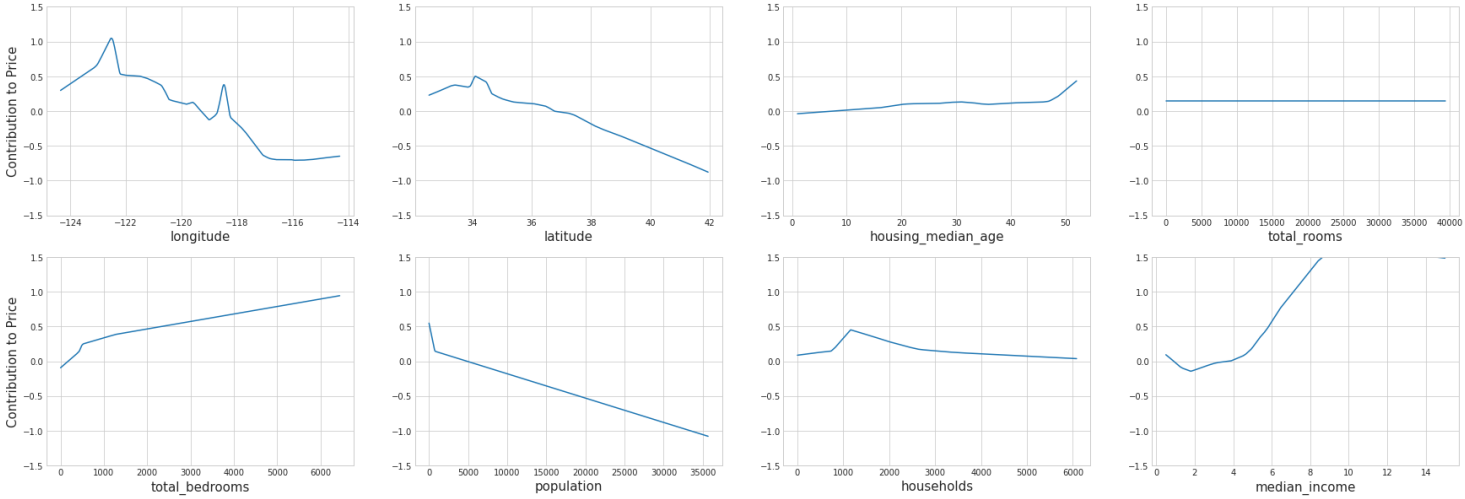

Figure 5.6: Global interpretation of 1D features for DGANN^2 trained on California housing data set

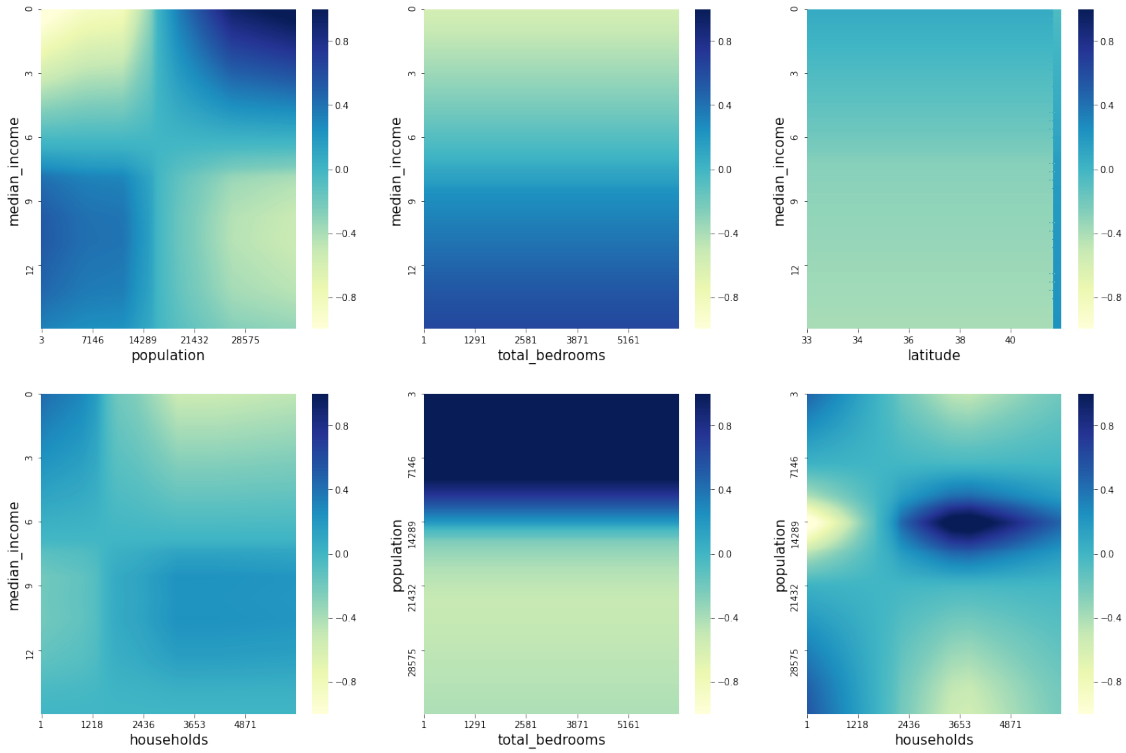

Figure 5.7: Global interpretation of a subset of interaction features for DGANN^2 trained on California housing data set

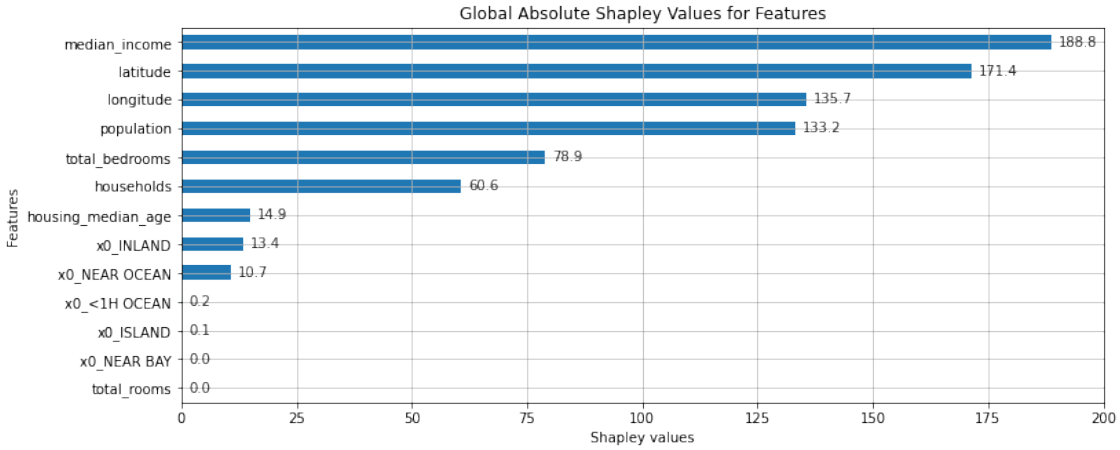

Figure 5.8: Scaled global Shapley values of features for DGANN^2 trained on California housing data set. The features are ranked from most important to least important according to their respective global Shapley values.

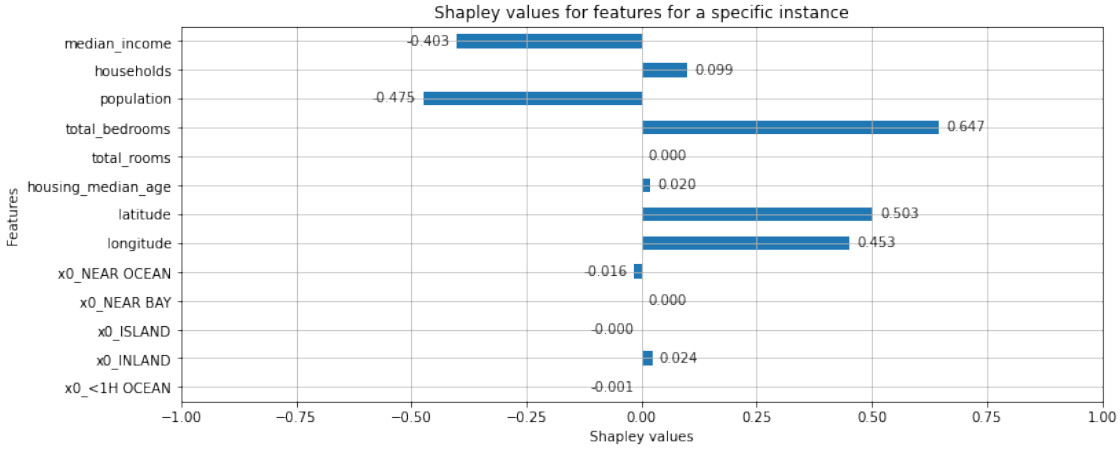

Figure 5.9: Local Shapley values of features for a specific instance. The values are calculated using D-Shapley on a DGANN^2 trained on the California housing data set. The details of this particular instance can be seen in Table A.3

Chapter 6

Conclusions

In this work, we have presented a new decomposable glass box model called DGANN^2. By limiting GA^{2}Ms to only consider neural networks, and forcing pairwise interactions to have a multiplicative form, we made DGANN^2 decomposable. Due to its decomposable architecture, we presented a faster method to calculate approximate Shapley values for DGANN^2, equipping it with an additional and efficient method of explainability.

In Chapter 3, we describe the architecture of DGANN^2 and provide the algorithm D-Shapley to calculate approximate Shapley values on DGANN^{2}s. In Chapter 5, we use DGANN^2 to provide explanations for the data set obtained from DLL. We see that due to its glass box nature, DGANN^2 can provide feature and interaction plots in the form of line graphs and heatmaps, respectively. Furthermore, using D-Shapley we can calculate global Shapley values to get feature importances. These give us a ranking of the features in terms of their influence on the model outcome. The same algorithm can be used to provide local explainability on a single data point as well. In Figure 5.4 and Figure 5.5, we look at the Shapley values for two data points with different outcomes. The approximate Shapley values in these plots allow us to provide explanations on how the model came to a decision.

We further tested DGANN^2 against a few other models on 9 data sets. We can see that DGANN^2 has comparative performance to similar GAM-based models, and outperforms its competitors in a few classification data sets. However, DGANN^2 performs worse than XGBoost models in all provided regression data sets. We suggest ways to improve DGANN^2 to possibly improve its performance in Section 6.2.

The glass box nature of DGANN^2 and the efficient calculation of Shapley values using D-Shapley make DGANN^2 a desirable alternative to existing models and methods used. Furthermore, as can be seen from Table 5.1, DGANN^2 shows comparative performance in the DLL data set. We believe that DGANN^2 can be used in the C&R project.

6.1 Limitations

DGANN^2 has several limitations with how it is described in this work. The biggest limitation is the lack of a structured way of deciding how many and which interactions to include. The performance of the model depends heavily on the choice of interactions, since forcing the model to consider wrong interactions can lead the model away from learning. Another limitation is the possible long training times. Since each function in DGANN^2 is a neural network, the training time for DGANN^{2}s especially on data sets with a large number of features can be quite long.

6.2 Future Work

Throughout this work, we have endeavored to provide a proof of concept for DGANN^2. While this work can lay the basis for such decomposable models, there can be made numerous improvements.

One direction is improving the implementation of DGANN^2, by introducing constraints to consider only a subset of the features, and using a smarter algorithm such as FAST [8] to select which pairwise interactions to include in the model. Furthermore, the architecture can be extended to consider not only pairwise interactions, but higher order interactions as well. Another technical implementation detail could be in automatic architecture selection in the neural networks that make up the 1D and 2D functions. Neural architecture search (NAS) is a rapidly developing area of research, and methods from NAS can be integrated into DGANN^2 to reach better performance.

Bibliography

[1] Responsible AI Practices - Google. https://ai.google/responsibilities/responsible-ai-practices/. Accessed: 2020-23-07. 4

[2] Responsible AI Practices from Microsoft. https://www.microsoft.com/en-us/ai/responsible-ai. Accessed: 2020-23-07. 4

[3] D. Alvarez-Melis and T. S. Jaakkola. On the Robustness of Interpretability Methods. *arXiv preprint arXiv:1806.08049*, 2018. 10

[4] J. Angwin, J. Larson, S. Mattu, L. Kirchner, and ProPublica. Machine Bias - ProPublica. https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing, 23 May 2016. Accessed: 2020-07-24. 4

[5] A. B. Arrieta, N. Díaz-Rodríguez, J. D. Ser, A. Bennetot, S. Tabik, A. Barbado, S. Garcia, S. Gil-Lopez, D. Molina, R. Benjamins, R. Chatila, and F. Herrera. Explainable artificial intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Information Fusion*, 58:82–115, June 2020. doi:10.1016/j.inffus.2019.12.012. vi, 1, 6, 9

[6] U. Bhatt, A. Xiang, S. Sharma, A. Weller, A. Taly, Y. Jia, J. Ghosh, R. Puri, J. M. Moura, and P. Eckersley. Explainable machine learning in deployment. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, pages 648–657, 2020. 5, 12

[7] J. Buolamwini and T. Gebru. Gender shades: Intersectional accuracy disparities in commercial gender classification. In S. A. Friedler and C. Wilson, editors, *Proceedings of the 1st Conference on Fairness, Accountability and Transparency*, volume 81 of *Proceedings of Machine Learning Research*, pages 77–91, New York, NY, USA, 23–24 Feb 2018. PMLR. URL http://proceedings.mlr.press/v81/buolamwini18a.html. 4

[8] R. Caruana, Y. Lou, J. Gehrke, P. Koch, M. Sturm, and N. Elhadad. Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1721–1730, 2015. 7, 36

[9] T. Chen and C. Guestrin. XGBoost. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, Aug. 2016. doi:10.1145/2939672.2939785. 13, 23

[10] E. Choi, M. T. Bahadori, J. Sun, J. Kulas, A. Schuetz, and W. Stewart. Retain: An interpretable predictive model for healthcare using reverse time attention mechanism. In *Advances in Neural Information Processing Systems*, pages 3504–3512, 2016. 5

[11] D. De Waal, J. Du Toit, and T. De La Rey. An investigation into the use of generalized additive neural networks in credit scoring. In *Proceedings of IX Credit Scoring & Credit Control Conference*, pages 1–10. Citeseer, 2005. vi, 7

[12] D. Dua and C. Graff. UCI machine learning repository, 2017. URL `http://archive.ics.uci.edu/ml`. 22, 23

[13] J. H. Friedman, B. E. Popescu, et al. Predictive Learning via Rule Ensembles. *The Annals of Applied Statistics*, 2(3):916–954, 2008. 7

[14] T. J. Hastie and R. J. Tibshirani. *Generalized Additive Models*, volume 43. CRC press, 1990. 6

[15] A. Holzinger, C. Biemann, C. S. Pattichis, and D. B. Kell. What do we need to build explainable AI systems for the medical domain? *CoRR*, abs/1712.09923, 2017, 1712.09923. URL `http://arxiv.org/abs/1712.09923`. 1, 6

[16] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu. Lightgbm: A highly efficient gradient boosting decision tree. In *Advances in neural information processing systems*, pages 3146–3154, 2017. 13

[17] Y. Liu. *Python machine learning by example : easy-to-follow examples that get you up and running with machine learning*. Packt Publishing, Birmingham, UK, 2017. 21

[18] Y. Lou, R. Caruana, and J. Gehrke. Intelligible models for classification and regression. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 150–158, 2012. 6

[19] Y. Lou, R. Caruana, J. Gehrke, and G. Hooker. Accurate Intelligible Models with Pairwise Interactions. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 623–631, 2013. 5, 7

[20] S. M. Lundberg, G. Erion, H. Chen, A. DeGrave, J. M. Prutkin, B. Nair, R. Katz, J. Himmelfarb, N. Bansal, and S.-I. Lee. From local explanations to global understanding with explainable AI for trees. *Nature Machine Intelligence*, 2(1):56–67, Jan. 2020. doi:10.1038/s42256-019-0138-9. 13

[21] S. M. Lundberg and S.-I. Lee. A unified approach to interpreting model predictions. In *Advances in neural information processing systems*, pages 4765–4774, 2017. 8, 12, 13

[22] F. Melo. *Area under the ROC Curve*, pages 38–39. Springer New York, New York, NY, 2013. doi:10.1007/978-1-4419-9863-$7_2$09. 23

[23] G. Montavon, W. Samek, and K.-R. Müller. Methods for Interpreting and Understanding Deep Neural Networks. *Digital Signal Processing*, 73:1–15, Feb 2018. doi:10.1016/j.dsp.2017.10.011. 1

[24] S. Moro, P. Cortez, and P. Rita. A data-driven approach to predict the success of bank telemarketing. *Decision Support Systems*, 62:22–31, 2014. 22

[25] S. U. Noble. *Algorithms of Oppression: How Search Engines Reinforce Racism*. NYU Press, 2018. URL `http://www.jstor.org/stable/j.ctt1pwt9w5`. 4

[26] H. Nori, S. Jenkins, P. Koch, and R. Caruana. Interpretml: A Unified Framework for Machine Learning Interpretability. *arXiv preprint arXiv:1909.09223*, 2019. 8, 23

[27] E. Parliament and C. of European Union. European Union General Data Protection Regulation, Article 22. `https://www.privacy-regulation.eu/en/22.htm`, 2018. Accessed: 2020-07-23. 4

[28] E. Parliament and C. of European Union. European Union General Data Protection Regulation, Articles 13-15. `https://www.privacy-regulation.eu/en/13.htm`, 2018. Accessed: 2020-07-23. 4

[29] R. Peharz. *Foundations of sum-product networks for probabilistic modeling*. PhD thesis, PhD thesis, Medical University of Graz, 2015. vi, 13, 14

[30] H. Poon and P. Domingos. Sum-product networks: A new deep architecture. *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, Nov 2011. doi:10.1109/iccvw.2011.6130310. 13

[31] W. J. Potts. Generalized Additive Neural Networks. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 194–200, 1999. 6

[32] A. D. Preece, D. Harborne, D. Braines, R. Tomsett, and S. Chakraborty. Stakeholders in explainable AI. *CoRR*, abs/1810.00184, 2018, 1810.00184. URL `http://arxiv.org/abs/1810.00184`. 5

[33] L. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush, and A. Gulin. CatBoost: unbiased boosting with categorical features, 2017, 1706.09516. 13

[34] M. T. Ribeiro, S. Singh, and C. Guestrin. ' Why should I trust you?' Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144, 2016. 8, 10

[35] M. T. Ribeiro, S. Singh, and C. Guestrin. Model-agnostic interpretability of machine learning. *arXiv preprint arXiv:1606.05386*, 2016. vi, 9

[36] M. T. Ribeiro, S. Singh, and C. Guestrin. Anchors: High-precision model-agnostic explanations. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018. 8

[37] C. J. V. Rijsbergen. *Information Retrieval*. Butterworth-Heinemann, USA, 2nd edition, 1979. 24

[38] A. D. Selbst and J. Powles. 7(4):233–242, Nov. 2017. doi:10.1093/idpl/ipx022. 5

[39] R. Serrano. Cooperative games: Core and Shapley value. Technical report, Working Paper, 2007. 10

[40] L. S. Shapley. A value for n-person games. *Contributions to the Theory of Games*, 2(28):307–317, 1953. 10

[41] D. Slack, S. Hilgard, E. Jia, S. Singh, and H. Lakkaraju. Fooling LIME and SHAP: Adversarial attacks on post hoc explanation methods. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, pages 180–186, 2020. 10

[42] E. Strumbelj and I. Kononenko. An Efficient Explanation of Individual Classifications Using Game Theory. *The Journal of Machine Learning Research*, 11:1–18, 2010. 2, 11, 12

[43] E. Štrumbelj and I. Kononenko. Explaining prediction models and individual predictions with feature contributions. *Knowledge and information systems*, 41(3):647–665, 2014. 8, 11, 12

[44] A. Tsanas, M. A. Little, P. E. McSharry, and L. O. Ramig. Accurate telemonitoring of Parkinson's disease progression by noninvasive speech tests. *IEEE transactions on Biomedical Engineering*, 57(4):884–893, 2009. 22

[45] Z. Yang, A. Zhang, and A. Sudjianto. GAMI-Net: An Explainable Neural Network based on Generalized Additive Models with Structured Interactions. *arXiv preprint arXiv:2003.07132*, 2020. 15

Appendix A

Data Set Descriptions and Plots

Table A.1: Features in the DLL data set. The features indicated with a * are related to singular invoices. They are aggregated to get minimum, maximum, average, and sum values. These aggregate features are included in the modeling data set.

Variables	Type	Description
country	Categorical	The country a debtor operates in.
reason for technical default	Categorical	A technical default indicates that a debtor cannot uphold an aspect of the loan terms. This field gives the reason in the case of a technical default.
reason default masterscale	Categorical	A master scale is an organization's own rating board. Reasons for default are collected under certain risk groups.
debtor exposure	Numerical	Debtor exposure is the maximum potential loss to DLL in case the debtor defaults on a payment.
theoretical lpi	Numerical	Theoretical late payment interest.
total debtor	Numerical	The total amount of money a debtor owes.
invoices overdue	Numerical	The amount of money owed on overdue invoices.
remaining days probation	Numerical	The number of remaining days of probation for a debtor. A probation period can be thought of as a trial time during which DLL assesses a debtor.
number of invoices	Numerical	The number of invoices a debtor has on a day. Aggregated variable.
days past due *	Numerical	Number of days past due for an invoice, calculated as current date - due date.
amount invoice *	Numerical	The amount of money owed on an invoice.
unpaid amount invoice *	Numerical	The unpaid amount on an invoice.
principal amount invoice *	Numerical	A principal amount on an invoice is the amount owed without calculating the interest.
days between due date and invoice date *	Numerical	Self-explanatory.
indicate aftercare	Binary	When 1, a debtor has an aftercare status.
indicate forbearance	Binary	When 1, a debtor is under forbearance. During a forbearance period, DLL can reduce or halt payments due to extraordinary causes.
indicate probation	Binary	When 1, a debtor is in their probation period.

Table A.2: Features in the California Housing data set.

Variables	Type	Description
longitude	Numerical	Longitude of the house.
latitude	Numerical	Latitude of the house.
housing median age	Numerical	Median age of houses within a block.
total rooms	Numerical	Total number of rooms in the houses within a block.
total bedrooms	Numerical	Total number of bedrooms in the houses within a block.
population	Numerical	Total number of people residing within a block.
households	Numerical	Total number of households within a block.
median income	Numerical	Median income for households within a block.
ocean proximity	Categorical	Proximity of the house to the ocean.

Table A.3: Values of features for a singular instance from the California housing data set.

Feature	Value
longitude	-119.27
latitude	34.17
housing median age	15
total rooms	11403
total bedrooms	2131
population	3327
households	1585
median income	4.3693
ocean proximity	NEAR OCEAN
median house value (target)	423300

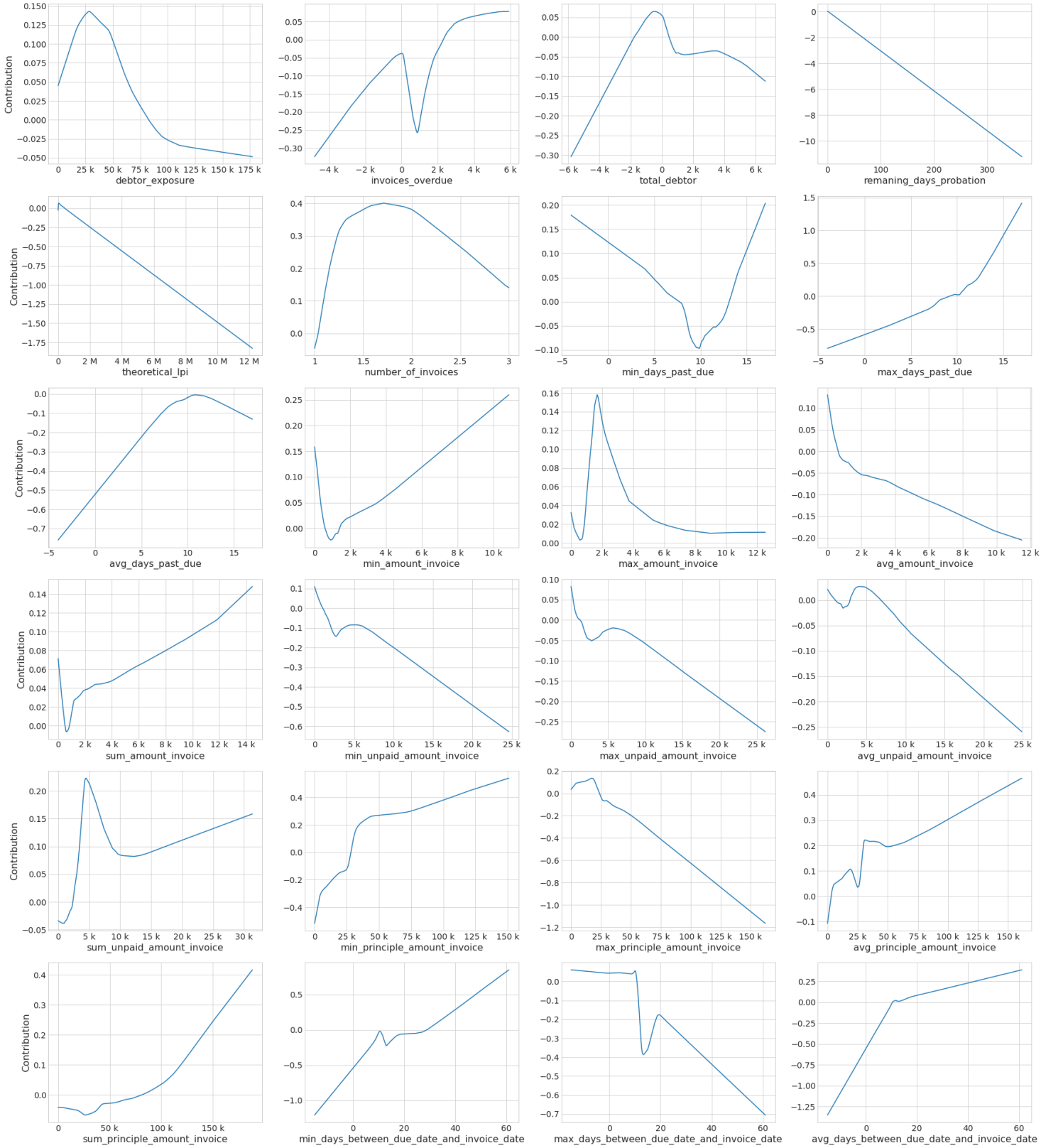

Figure A.1: Global interpretation of 1D features for DGANN^2 trained on DLL data set.

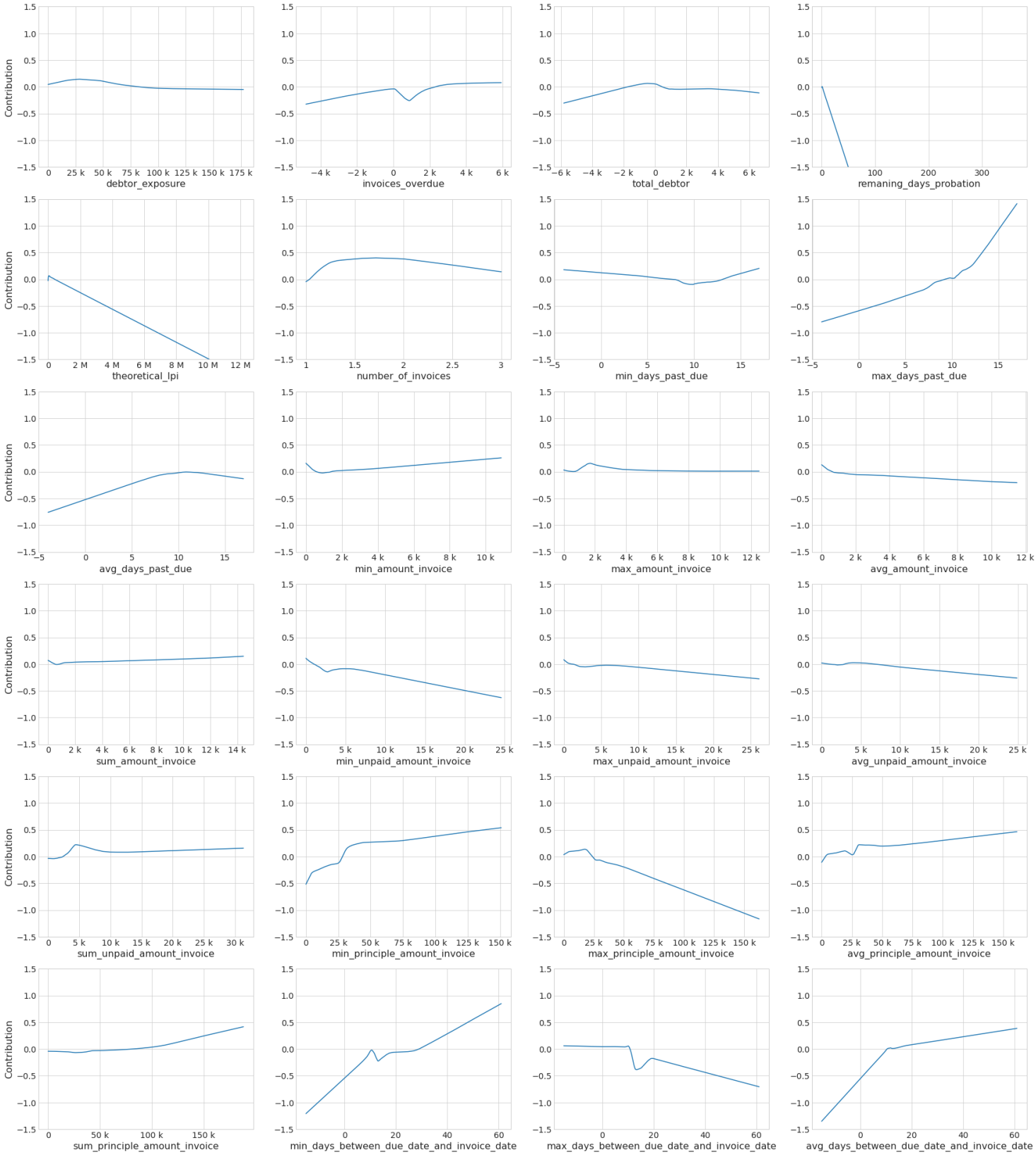

Figure A.2: Global interpretation of 1D features for DGANN^2 trained on DLL data set, scale shared between the plots.

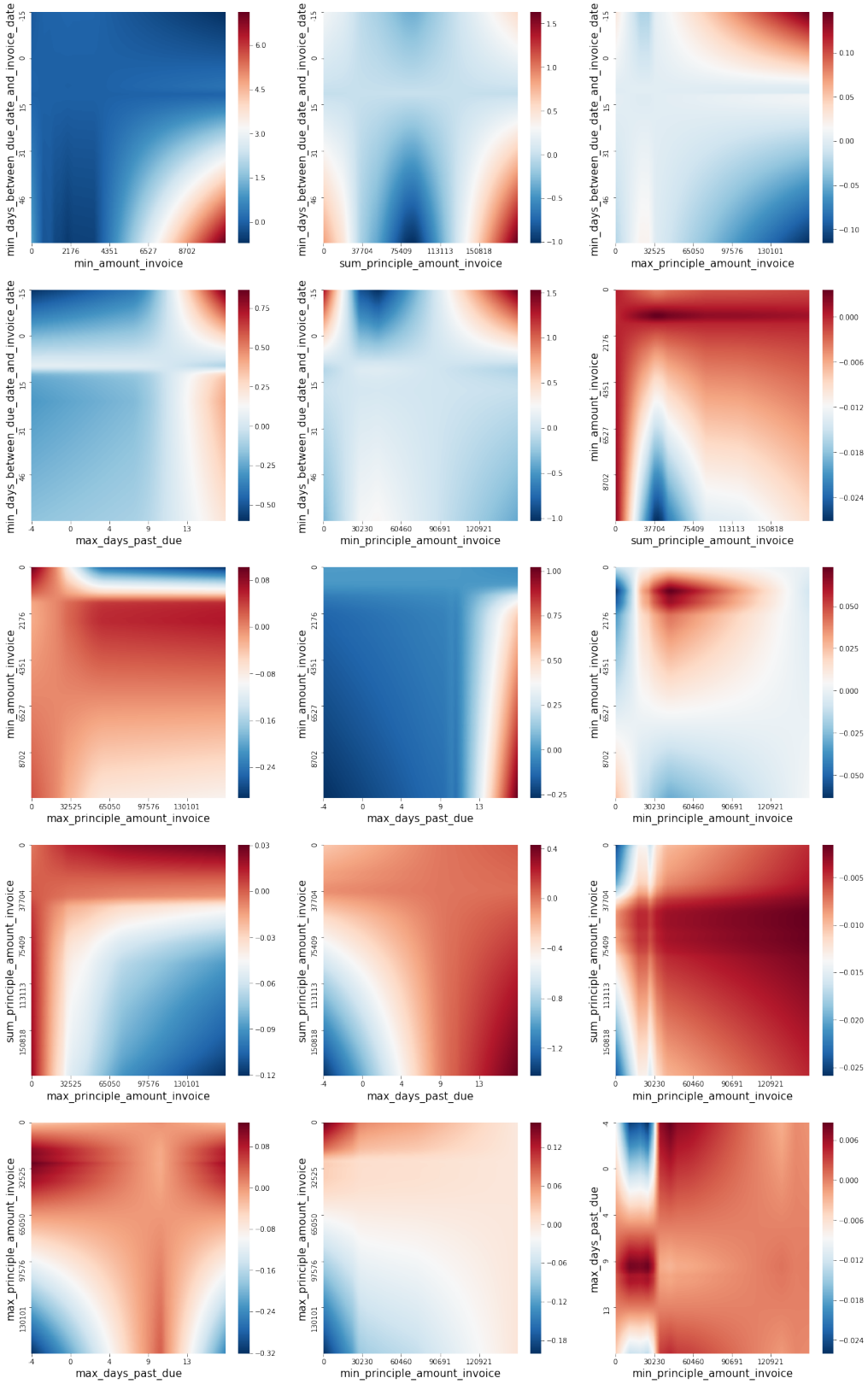

Figure A.3: Global interpretation of interaction features for DGANN^2 trained on DLL data set.

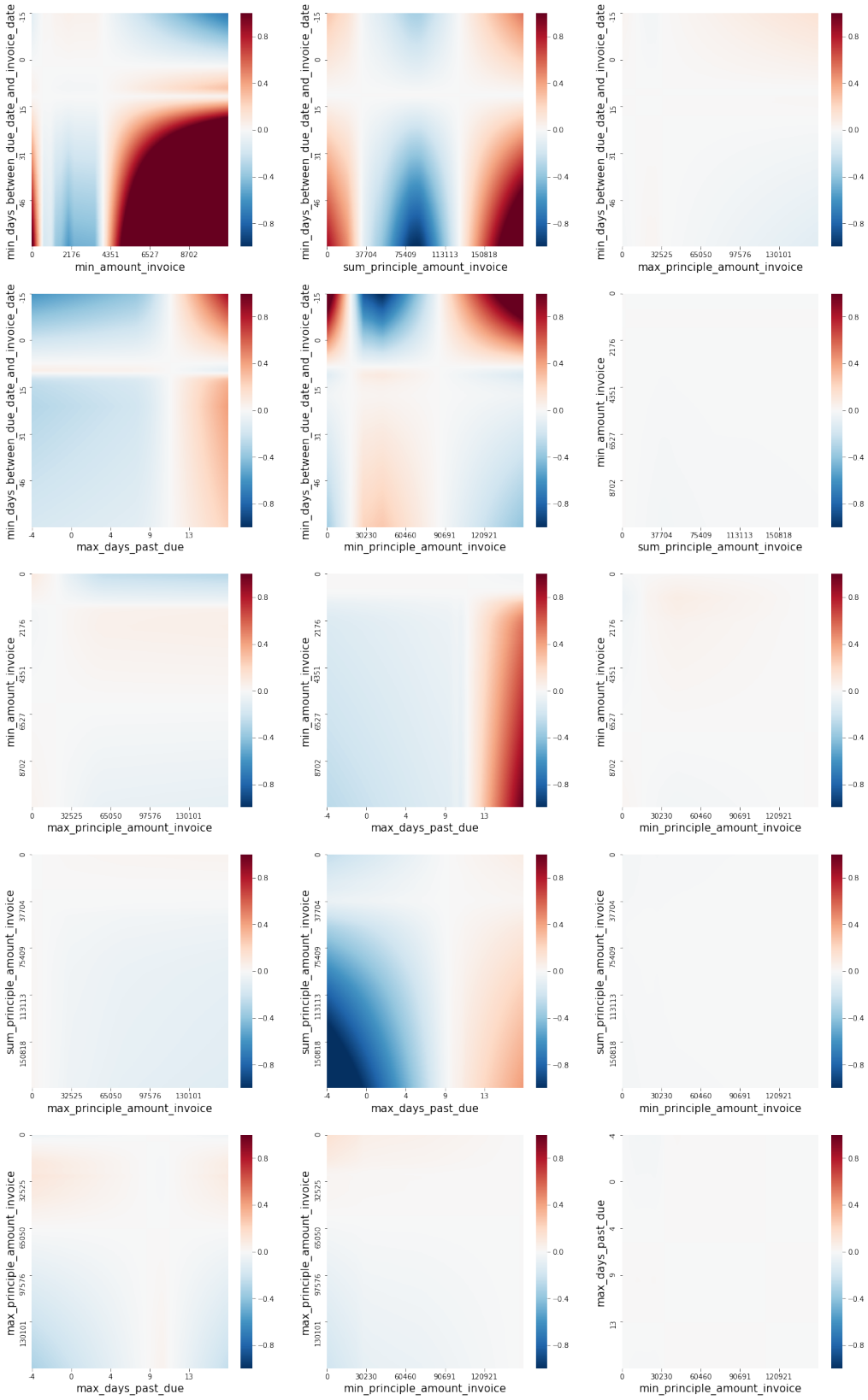

Figure A.4: Global interpretation of interaction features for DGANN^2 trained on DLL data set, scale shared between the plots.