universität wien

MASTERARBEIT / MASTER'S THESIS

Titel der Masterarbeit / Title of the Master's Thesis

„Interpreting Neural Networks Under Latent Confounding“

verfasst von / submitted by

Patrick Pollek, BSc

angestrebter akademischer Grad / in partial fulfilment of the requirements for the degree of

Master of Science (MSc)

Wien, 2023 / Vienna, 2023

Studienkennzahl lt. Studienblatt / degree programme code as it appears on the student record sheet:	UA 066 645
Studienrichtung lt. Studienblatt / degree programme as it appears on the student record sheet:	Master Data Science
Betreut von / Supervisor:	Ass.-Prof. Dr.techn. Sebastian Tschiatschek

Acknowledgements

I wish to express my heartfelt gratitude to my academic advisor, Sebastian Tschiatschek, for his continuous support and insightful feedback throughout the course of my research. The periodic consultations and guidance have been immensely helpful in shaping this thesis.

I also owe a great debt of gratitude to Günther Hörmann, who has been an unwavering source of support and belief in both my academic and personal life from the onset of my academic career. Thanks to you I was able to apply and pursue this Master's degree!

I am grateful for the collaborative and stimulating environment fostered by my colleagues, collectively known as Die Denker [Anton, Camillo, Clemens, Fabs, Freddy, Kathi, Lena, Lorena, Marijana, Quiqua, Ralf, Sophie, Stella, Vivi]. Your constructive criticism, late nights at the university, and countless discussions have enriched my scholarly development.

On a personal note, my heartfelt thanks go to my family, especially to my mother and sister, whose constant support and love have been my stronghold. Your faith in my abilities has been a continuous source of strength throughout my academic pursuits.

Finally, I am deeply thankful to my girlfriend Kathi. Your constant presence, emotional support, and love have made this whole journey a lot easier, allowing me to confront and overcome the various challenges of this endeavor.

Patrick Pollek

Abstract

Machine learning models are increasingly used in various fields for high-risk decision-making processes. While these models often exhibit high predictive accuracy, their complex nature often results in a lack of interpretability, creating barriers to human understanding and raising ethical issues. In this context, confounding variables serve as a complicating factor that can further undermine the interpretability of these models and can affect interpretability techniques. These often latent factors can influence both independent and dependent variables and can affect the reliability and trustworthiness of any derived interpretations. In general, researchers frequently control for confounders but often omit interpretive analyses of these models. When interpretive studies are conducted, the impact on the interpretations remains largely unexplored and unquantified. Thus there is an investigative gap in understanding how the presence of confounding variables might influence the interpretability of the models. To narrow this gap this thesis assesses the impact of confounding variables on the interpretation of neural networks in a supervised setting by juxtaposing and analyzing ICE plots and SHAP value visualizations for neural networks. Experiments show that simple feed-forward neural networks are sensitive to hidden confounding, particularly in terms of feature importance. To compensate for these shortcomings a model based on a variational autoencoder is introduced which provides evidence that the interpretations from this new model are more consistent and intuitive compared to those from other models, as verified across three different datasets. The findings indicate that confounding variables have the most impact on the importance of features that are highly correlated.

Kurzfassung

Modelle des maschinellen Lernens werden zunehmend in verschiedenen Bereichen für entscheidungsintensive Prozesse hohen Risikos eingesetzt. Obwohl diese Modelle oft eine hohe Vorhersagegenauigkeit aufweisen, führt ihre komplexe Natur häufig zu einem Mangel an Interpretierbarkeit, was Barrieren für das menschliche Verständnis schafft und ethische Fragen aufwirft. In diesem Kontext wirken Störfaktoren (Confounder) als ein erschwerender Faktor, der die Interpretierbarkeit dieser Modelle weiter untergraben kann und sich auf Interpretationstechniken auswirken kann. Diese oft versteckten Faktoren können sowohl unabhängige als auch abhängige Variablen beeinflussen und die Zuverlässigkeit und Vertrauenswürdigkeit jeder abgeleiteten Interpretation beeinträchtigen. Üblicherweise regulieren oder berücksichtigen Forscher Störfaktoren, vernachlässigen jedoch oft interpretative Analysen solcher Modelle. Wenn interpretative Studien durchgeführt werden, bleibt die Auswirkung der Störfaktoren auf die Interpretationen weitgehend unerforscht und nicht quantifiziert. Daher besteht eine Untersuchungslücke im Verständnis, wie das Vorhandensein von konfundierenden Variablen die Interpretierbarkeit der Modelle beeinflussen könnte. Um diese Lücke zu schließen, bewertet diese Arbeit den Einfluss von konfundierenden Variablen auf die Interpretation von neuronalen Netzwerken in einem überwachten Umfeld, indem ICE-Diagramme und SHAP-Wert-Visualisierungen für neuronale Netzwerke gegenübergestellt und analysiert werden. Experimente zeigen, dass einfache vorwärts gerichtete neuronale Netzwerke empfindlich auf versteckte Konfundierung reagieren, insbesondere in Bezug auf die Relevanz der Variablen. Um diesen Problemen entgegenzuwirken, wird ein Modell auf Basis eines variationalen Autoencoders vorgestellt, das nahelegt, dass die Interpretationen aus diesem neuen Modell konsistenter und intuitiver sind als die aus anderen Modellen, wie anhand von verschiedenen Datensätzen verifiziert wurde. Die Ergebnisse zeigen, dass konfundierende Variablen den größten Einfluss auf die Bedeutsamkeit von Varaiblen haben, die stark korreliert sind.

Contents

1. Introduction

As machine learning (ML) systems become increasingly integrated into our daily lives, they often serve as crucial aids in decision-making processes [LZHH14]. In healthcare for example, several computer-aided diagnosis systems utilize deep neural networks to accurately identify medical conditions while also enhancing the efficiency of the diagnostic process [MMOR+21], [ALPM+13]. However, assessing these systems against critical criteria like safety, avoidance of discrimination, and provision of explanations can be challenging. Interpretability, or the ability of a system to explain its reasoning, is often sought after in cases, where these criteria cannot be completely quantified [TLS+22], [BYC+17].

Consider autonomous driving, for example. A deep learning model might attempt to learn how to detect cyclists with the intent to stop the vehicle before a potential accident occurs. The goal would be to ensure flawless and safe operation every time. If upon interpreting the model, it is revealed that the most important feature recognized is the helmet of the cyclist - because every image of a cyclist in the training data featured a helmet - it would be a good idea to check how the model responds to cyclists not wearing helmets [DVK17].

Much like in the realm of autonomous driving, the stakes are similarly high in healthcare, where the accurate interpretation of machine learning models can have life-altering consequences. In medical settings, interpretability serves as a critical asset for understanding diagnostic processes and as a safeguard against potential errors. However, the presence of confounders, which arise in all sorts of areas, might further complicate the interpretability of these systems.

A confounding variable is an often hidden factor that affects both the independent and dependent variables, thereby complicating the interpretation of the true relationships involved [Pea09]. In the study of the impact of smoking tobacco on human health, for instance, variables such as alcohol consumption and diet are also lifestyle factors that are interrelated. A risk assessment focusing solely on the effects of smoking could erroneously overestimate the health risks attributed to smoking (Figure 1.1). This makes these additional factors confounding variables, as they interfere with the ability to accurately interpret the relationship between smoking and health,[KZEW04], [HMKG17].

The issue of confounding variables is a longstanding one, and researchers have developed various methodologies to account for them or model them. For instance, in assessing the impact of medication on a specific patient, a study introduced the Causal Effect Variational Autoencoder, a model that incorporates confounders while simultaneously evaluating the causal effect [LSM+18]. Strategies for mitigating the impact of confounders are not exclusive to the medical field but extend to various other domains as well [SMS20], [dHJL19]. However, both types of models - those that address confounding variables and

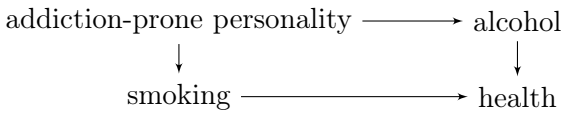


Figure 1.1.: An addiction-prone personality can impact both alcohol consumption and smoking habits. If an assessment focuses solely on smoking to determine its health risks, the evaluation may overestimate the negative effects. This is because latent factors, such as alcohol consumption and personality traits like an addictive personality, are not taken into account. In this context, these obscured variables serve as confounders, complicating the true understanding of smoking's impact on health.

those that do not - are often insufficiently analyzed for interpretability. When attempts at interpretation are made, the impact of confounding variables on the model's reasoning remains often unknown [TLS+22]. This lack of clarity is particularly problematic because the outcomes of these models, or the importance attributed to certain features, may be used as guidelines, potentially leading to flawed or misleading conclusions. This creates a research gap that the present thesis aims to address. This gives rise to the first research question.

Research Question 1:
What is the impact of confounding variables on the interpretability of a neural network, via feature importance, and ICE plots? To what extent do we observe differences in the interpretability results across distinct datasets?

To address the challenges identified in RQ1, we introduce a second research question focused on the potential for enhancing model interpretability through specific modeling techniques.

Research Question 2:
Can a model based on a Variational Autoencoder (VAE) improve the interpretability when confounding is evident in the sense of being more intuitive or more in line with the model that has access to a hidden confounder on distinct datasets?

This second research question stems from the desire to improve model interpretability by employing a Variational Autoencoder-based framework, inspired by the Causal Effect Variational Autoencoder discussed earlier [LSM+18].

Approch and contribution To address the aforementioned research gap concerning the influence of confounding variables on interpretability we conducted a comparative analysis between two feed-forward neural network (FFNN) models: one that includes all variables (full model) and another that omits one variable (confounder model), thereby introducing

a latent confounder. We will use ICE plots and SHAP values to interpret the models and discuss differences. Subsequently, we introduce a model (VAE model) constructed on the foundation of a variational autoencoder (VAE) which is inspired by causal inference to better handle latent confounding with respect to interpretability. We show that latent confounding can significantly alter the interpretations based on ICE plots and SHAP values of neural networks. This manifests in very sensitive feature importance values as well as overestimation of variable effects in ICE plots. The proposed model yields interpretations that are both more intuitive, and more consistent than other analyzed models. Specifically, the SHAP summary plots show consistent clusters of SHAP values which indicate an isolated and more intuitive relationship of variables and targets.

Thesis structure First, we discuss relevant literature in chapter 2. After that explanations of the interpretability methods and a short introduction to VAEs will be given (Chapter 3). The methodology, outlined in chapter 4, on how we aim to answer the research questions will follow as well as the used datasets (Chapter 5) and the experimental setup (Chapter 6) containing hyperparameters of used models. The results section (Chapter 7) extensively shows conducted experiments which are then discussed (Chapter 8) and summarized (Chapter 9).

Used tools Chat GPT [Ope23] was used to reformulate text and to generate code snippets in Python.

2. Literature review

2.1. Interpreting Blackbox Models

In conducting this research on Neural Network interpretability, we will utilize a variety of well-established methods with the objective of increasing the transparency of black-box models. The complexity and opacity of such models present challenges in practical applications, necessitating the development of tools that can shed light on their internal operations.

According to Murdoch et al., 2019 [MSK+19] there are two main ways to understand how models make decisions: Model-based interpretability and Post hoc interpretability. Model-based interpretability is when the model itself is easy to understand. For instance, in linear regression models, the significance of a feature can be assessed by examining the weights attributed to it. Decision trees too are inherently interpretable, as at each step (or node), one can clearly see the decision being made. On the other hand, post hoc interpretability involves methods that can be used on any kind of trained model, even those that are usually hard to interpret, like black box models. These methods aim to explain how the model made decisions or what was learned after it had been trained. Since this thesis focuses on Neural Networks (NNs) we will use post hoc interpretability methods to understand the trained models.

The following interpretability methods are comprehensively discussed in "Interpretable Machine Learning: A Guide for Making Black Box Models Explainable" by Christoph Molnar [Mol22]. This work provides a comprehensive overview of state-of-the-art interpretation methods for various machine learning models.

A very popular method for enabling post hoc interpretability is to generate individual conditional expectation plots (ICE plots), introduced by Goldstein et al. (2014)[GKBP15]. They extend the concept of Partial Dependence Plots introduced by Friedman in 2001 [Fri01] by plotting individual instances instead of averages, thus offering a more granular view of model behavior. In this research, ICE plots serve as an intuitive way of discovering heterogeneous relationships in the data, especially considering the confounding variable, and getting a good overview of the model on an instance-based level. In this context, heterogeneous relationships refer to the potential identification of associations within the data that are influenced by a confounding variable and a particular attribute. Utilizing ICE plots, which permit color-coding of individual lines, can effectively reveal such intricate relationships.

Another interpretability method is the calculation of Shapley values. Shapley values [Sha53] are a concept from cooperative game theory, adapted to machine learning, that offers a mathematically fair method for attributing the contribution of each feature to

the prediction of an individual instance in a model. In essence, Shapley values estimate the contribution of a feature by comparing the model's predictions with and without the feature. They work by iterating through all possible combinations of features (coalitions) in the model and averaging the difference in the prediction outcome when a particular feature is included versus when it is not. This results in an additive feature attribution method where each feature gets a score that represents its contribution towards the prediction for a particular instance. These scores can then be used to interpret the model's decisions, to understand the relationships between features, or to validate the fairness of the model's predictions. Given the computational intensity of precisely calculating Shapley values, which necessitates training a new model for all 2^d subsets of features (where 'd' represents the dimensionality of our data points), even for relatively low dimensional data, this approach quickly becomes impractical. To circumvent this issue, we employ an efficient approximation technique known as KernelSHAP.

KernelSHAP (Kernel SHapley Additive exPlanations)[LL17] is an algorithm to approximate Shapley values based on the coefficients of a weighted linear model. The weighting scheme is designed to give greater emphasis to both small coalitions, comprising fewer features, and large coalitions, encompassing many features. This is achieved using the SHAP kernel, a specialized form of weighting function. By calculating the mean of all SHAP values across all instances, a global metric of feature importance can also be obtained.

Other typical methods for Post hoc interpretability are for example counterfactual explanations or influential Instances. Counterfactual explanations [WMR17] are explanations that describe the minimal change of a feature to change the prediction or classification of the model. This enables us to create explanations of the form: „If you would earn 300$ more per month, then this would increase your probability of getting the loan from 25% to 50% ". Influential Instances, a concept first introduced in the field of statistics in the 1970s by Hampel (1974) [Ham74] and Jaeckel (1972) [Jae72], refers to particular data points in the training set that significantly impact the predictions or parameters of a trained model. If the removal of a single instance from the training set leads to substantial changes in the model's outputs, this instance is labeled as 'influential', indicating that the model is significantly dependent on that specific data point. While the method of identifying Influential Instances can offer useful insights and assist in model debugging, it is important to note that it is computationally intensive, requiring the model to be retrained multiple times. This, coupled with the fact that the gain in insights is often limited, can make this method a less optimal choice compared to others for this thesis.

2.2. Confounding

Confounding is a fundamental concept in research across numerous fields, especially in medicine [BSG+10], [CHH+09], and epidemiology [Van04]. By distorting the true relationship between an independent and dependent variable, confounding can lead to inaccurate results and misleading conclusions about the relationship between variables.

Because some variables are correlated to the confounder (to some degree), they will be correlated to each other themselves. This creates some dependencies among them which is known as multicollinearity. Multicollinearity can influence the stability of model training and can result in unstable interpretations and parameter estimations (See A.4). A lot of real-world datasets show some degree of multicollinearity.

Recently, there has been a notable increase in papers within the Machine Learning community that address the issue of confounding in supervised learning problems. For instance, a study conducted by John R. Zech and colleagues in 2018 [ZBL+18] revealed that Convolutional Neural Networks (CNNs), trained on pneumonia screenings, demonstrated significantly higher performance on internal screenings (originating from the same hospital) compared to external ones (from different hospitals). It was observed that these CNNs were identifying hospital systems and even specific departments within a hospital, which acted as confounding variables, affecting the prediction accuracy of the disease.

In causality research, causal graphs are frequently employed to pinpoint confounding variables, often utilizing algorithms such as the Fast Causal Inference (FCI) algorithm. A 2021 study [SMS20] by Numair Sani et al. used this method to understand how groups of individual features that together represent an interpretable feature are connected in a causal graph. Specifically, this was done by grouping pixels and pixel information in an image together to form features such as the wing shape of a bird or belly color. The approach also took into account unseen or unmeasured confounding variables that could influence the outcomes based on these features. This approach holds promise in revealing key features that essentially drive or influence the outcome generated by the neural net.

Another case where one can observe confounding was presented in a paper by Pim de Haan et al. [dHJL19]. In this paper, they describe a scenario where the aim is to train a Reinforcement Learning (RL) agent to operate a vehicle through the method of imitation learning. This involves training the RL agent to predict subsequent actions based on given state-action pairs from an expert driver. Two distinct situations were tested: in scenario A, the RL agent was provided an image containing both the dashboard and the windshield; in scenario B, the image was similar, but the dashboard was obscured (2.1). Despite achieving low training losses in both scenarios, the policies learned by the RL

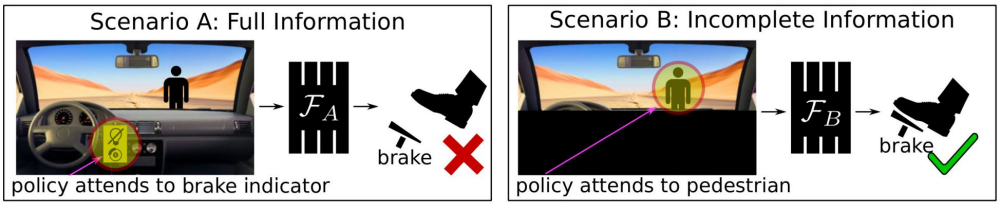


Figure 2.1.: Model A relies on the braking indicator to decide whether to brake. Model B instead correctly attends to the pedestrian. Taken from [dHJL19]

agents exhibited different behaviors. In scenario A, the RL agent associated the activation of the brake light on the dashboard with the necessity to apply the brakes, even though the light's illumination is a result, not a cause, of braking. In contrast, the RL agent in

scenario B correctly learned to brake when detecting a pedestrian ahead, demonstrating accurate causal understanding. Therefore, the agent in scenario A misattributed cause and effect in its decision-making process. In this case, the confounding variable is the brake light on the dashboard. They propose a novel idea to also take the already-mentioned causal graph into account to solve this causal misidentification problem.

In certain studies such as the one by Pryzant et al. (2018) [PBS18], machine learning models are trained and interpreted while controlling for confounding variables. In this particular case, they utilized a Convolutional Neural Network (CNN) to analyze advertisement text to determine optimal writing styles. A confounding variable in this context is the presence of brand names. During the training phase, the aim was not only to select features (n-grams) that are predictive of the outcome 'y' but minimize the predictive power of confounding variable 'C', represented by the brand names. By controlling for this confounder, they ensure that the selected n-grams signify the true contributors towards a successful ad, rather than being influenced by the associated brand name. Subsequent interpretation of the CNN enables a clearer understanding of which n-grams are instrumental in producing an effective ad, providing valuable insights into advertising strategies instead of traditional A/B testing.

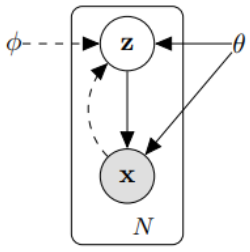
Despite the extensive literature on confounding in machine learning, there seems to be no work in which the authors try to understand and improve the interpretability of NNs in the context of confounding. Although this is a common practice in fields such as medicine, where features are analyzed in the context of known confounders, the outcomes of these analyses are usually interpreted through the lens of clinical expertise. For instance, in an article by Zeng et al. [ZGR+22], the patient's age was identified as a confounding variable. Age is a well-acknowledged confounder in treatment decisions and survival outcomes in healthcare. Older patients are often more likely to receive radiation treatment rather than surgery due to the heightened risks that surgical procedures pose to them. The difference to this thesis is that they observe the confounder and have them in their features, which also allows for a fairly simple interpretation, while we remove features that make the confounder hidden.

2.3. Variational Autoencoder

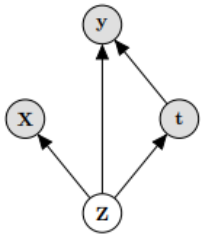
Some elements of this research are closely related to Variational Autoencoders (VAE) which were first described by Kingma and Welling [KW22]. A VAE is a type of generative model that uses deep learning techniques to produce complex models of data distribution. It works by encoding input data into a lower-dimensional latent space and then reconstructing the original data from this latent representation. Unlike traditional autoencoders, which often do not impose any structure on the latent space, a VAE imposes the condition that the latent variables follow for example a Gaussian distribution.

Figure 2.2a visually represents this concept.

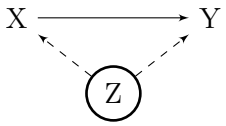
Building on the principles of VAE, another approach to managing hidden confounders in a different setting, is presented by Louizos et al. [LSM+18]. They proposed the Causal Effect Variational Autoencoder (CEVAE) which is designed to learn about the



(a) Solid lines denote the generative model $p_\theta(z)p_\theta(x|z)$, dashed lines denote the variational approximation $q_\phi(z|x)$ to the intractable posterior $p_\theta(z|x)$. The variational parameters ϕ are learned jointly with the generative model parameters θ [KW22].



(b) Example of a proxy variable. t is a treatment, e.g. medication; y is an outcome, e.g. mortality. Z is an unobserved confounder, e.g. socioeconomic status; and X is noisy views on the hidden confounder Z, say income in the last year and place of residence.(Taken from [LSM+18])



(c) This figure shows a typical confounding variable Z which influences both the independent variable X and the dependent variable Y.

Figure 2.2.

posterior distribution when there is an intervention or treatment in addition to dependent and independent variables. Figure 2.2b illustrates the setup. Their VAE also takes the treatment and the outcome as inputs which requires them to have these values at hand when using their models. Thus for out-of-sample predictions which are for example new subjects, they have to predict these values beforehand.

Yet another compelling approach to the problem of confounding, particularly in the context of biased data, emerges from causal modeling. A recent paper promotes learning from biased data by exploring the relationship between fair classification and intervention [MCPZ19]. The proposed model innovatively views a sensitive attribute as an observed confounder contributing to dataset bias, extending previous methodologies like the one employed in the CEVAE model by Louizos et al. Furthermore, to tackle the issue of unobserved or hidden confounders, they adopt deep latent variable models, allowing for approximate inference of these hidden elements. They described a method to learn the parameters of this causal model purely from observational data. Their work reveals that fairness-aware causal modeling provides superior estimates of the causal effects between the sensitive attribute, treatment, and outcome. Furthermore, by estimating these causal effects, they demonstrated that it is possible to derive policies that are both more accurate and fairer when confronted with a historically biased dataset.

The methodological framework of this thesis can be considered as related to the classical VAE as well as to the setting of having an additional treatment variable. The aim is to utilize information from the dependent variable (y) to infer a more informative latent

representation, thereby enhancing the interpretability of the model. Figure 2.2c shows the already-known setup that will be used in this thesis.

From the first introduction of VAEs by Kingma and Welling, we have seen a significant evolution of this framework with a variety of promising variants. For example, Beta-VAEs (Higgins et al., 2017 [HMP+17]) tweak the standard VAE's objective function by introducing a hyperparameter β to control the balance between the reconstruction and latent space regularization terms. This tweak enables Beta-VAEs to achieve more disentangled representations, improving interpretability and downstream tasks. One other extension is the Hierarchical VAE (HVAE). In HVAEs (Ranganath et al., 2016 [RTB16]), the latent space is structured hierarchically, allowing the VAE to model more complex data distributions. In contrast to traditional VAEs, where each latent variable operates independently, Hierarchical VAEs (HVAEs) guide the parameters of the latent variables with a prior distribution, facilitating the formation of complex structures. Similar to Beta-VAEs, a TC-VAE [MGD23], or Total Correlation Variational Autoencoder, is a type of Variational Autoencoder designed to encourage the latent variables in the model to be statistically independent of each other and more entangled. In a standard VAE, there's no guarantee that the latent variables will be independent, which can limit the model's ability to generate diverse and realistic data. The TC-VAE introduces an additional term to the loss function to minimize the Total Correlation (a measure of the statistical dependence between variables) in the latent space. Remarkably, the TC-VAE demonstrates the capability to expose out-of-distribution (OOD) generative factors, revealing consistent and interpretable attributes absent in the original dataset.

3. Background

3.1. Interpretability

The popularity of big data and artificial intelligence has increased dramatically in the past decade [BMR+20],[Ope23],[MKS+13],[KSH12]. Along with this great success, often based on complex black-box models such as deep neural networks, there is also growing awareness for improving model transparency and interpretability [ZTLT20]. This awareness is strongly needed since these models are used in more sensitive areas such as healthcare, autonomous driving, criminal justice, hiring people, and many more. This immediately requires the model to have traits such as fairness, safety or trust. For instance, it is critical that our model upholds fairness, meaning its predictions should remain uninfluenced by sensitive attributes such as gender or race, thereby avoiding discrimination against underrepresented groups. Employing an interpretable model aids in exposing underlying biases that might otherwise remain unnoticed. In settings where safety is key, like in autonomous driving, it is essential to make sure the model's learned behavior matches what we want. The ability to interpret the model can help with this. For example, it can help to identify whether the model has only learned to spot a helmet in a picture, or if it can actually recognize a cyclist. So, interpretability helps us check and understand what the model has learned.

Before we explain the different forms of interpretability, we will provide two important definitions related to accuracy. [MSK+19]

The first definition defines the term predictive accuracy, which is the error in the model's fit to the data. If we have a poor approximation of the underlying relationship, we might get accurate information about the model when interpreting it but the true relationship in the data might be different.

On the other hand, descriptive accuracy in the context of interpretability is the degree to which an interpretation method objectively captures the relationships learned by machine-learning models.

Oftentimes these two accuracies require a trade-off. Simple models are easy to understand, so they usually have good descriptive accuracy. However because they are simple, they might miss out on some complex patterns in the data. This can make them less accurate at making predictions. Conversely, more sophisticated models may excel in predictive performance due to their ability to capture complex patterns within the data. However, this increased complexity often renders them less interpretable, posing challenges to understanding the underlying mechanisms.

Relevancy, which determines the suitability of an interpretation for a specific audience or domain, joins predictive accuracy and descriptive accuracy to form the three essential

desiderata of the PDR framework. For an interpretation to be both trustworthy and insightful, it is essential to optimize these three dimensions.

Given the established importance of interpretability, it is essential to discuss its two primary types: model-based interpretability and post hoc interpretability. Each of these approaches highlights different aspects of the two forms of accuracy [MSK+19].

Model-based interpretability Model-based interpretability takes place in the modeling phase of the data-science life cycle. We chose models that are already interpretable by design and that provide insight into the relationship it has learned without any additional measures. One of the most intuitive models, from an interpretability standpoint, is the linear regression model. The innate interpretability of a linear regression model stems from its weights, which can serve as measures of feature importance for example. Moreover, it straightforwardly explains the effect of a variable increment, indicating the impact it has on the outcome. Oftentimes, this comes with a reduction of predictive accuracy since we now can only learn linear relationships, but the descriptive accuracy is high. Another common type of model-based interpretability is to impose sparsity on the model by restricting only a few feature weights to be non-zero. A common example of an interpretable model that induces sparsity is LASSO regression, which is a linear model. When dealing with biomolecular interactions, the datasets often contain features ranging from thousands to millions. LASSO regression can help pinpoint key features by identifying those with non-zero coefficients in the linear model, thereby enhancing interpretability. However, this method might lead to a trade-off in terms of predictive accuracy. By focusing on key features and not incorporating all available features, the model might not predict outcomes as accurately as it could otherwise. Note that there are also There are also other model-based methods that can be used such as modularity where portions of a model can be interpreted independently or a simulatable model where a human can reason about the entire decision process. A classic example of a simulatable model is a decision tree. In a decision tree, each rule or step in the decision-making process is clear and easy to understand.

Post hoc interpretability Post hoc interpretability is applied after a model has been trained. In this methodology, the emphasis is on analyzing the already-trained model to extract insights about the relationships it has learned, without modifying its architecture. This stands in contrast to model-based interpretability, where the model itself may be altered for the sake of clarity or comprehensibility. With post hoc interpretability, the model can be of any complexity, and its predictive accuracy remains unchanged since there's no need to simplify the model for interpretation purposes. For problems in high dimensions, like image classification, complex models are essential. Therefore, such scenarios tend to rely significantly on post hoc methods for interpretability. Post hoc methods for model interpretation primarily fall into two categories: local and global interpretations. Local interpretations concentrate on individual instances within the dataset and their respective predictions. On the contrary, global interpretations aim to provide an overall understanding of the entire dataset. For instance, a local interpretation

method could involve the use of Individual Conditional Expectation (ICE) plots, where a line is plotted for each instance in the dataset. A global interpretation method, however, could involve assessing feature importance across the whole dataset to gain a broader understanding of the model's behavior. Feature importance measures can also be used as a local method which then shows what feature was the most important one to arrive at this prediction or how much each feature contributed compared to a baseline to the final outcome.

In this thesis, our primary emphasis is on deep neural networks. Accordingly, we will exclusively use post-hoc methods, which are inherently model-agnostic. This means that these methods are designed to decouple the model's interpretation from the model itself. This separation offers us significant flexibility. More specifically, we can apply any function that accepts the features as input and provides a prediction as output, without needing to alter the method we employ for interpretation. This approach ensures consistency and adaptability in our analysis, regardless of the specific predictive function used. Figure 3.1 summarizes the discussed topics.

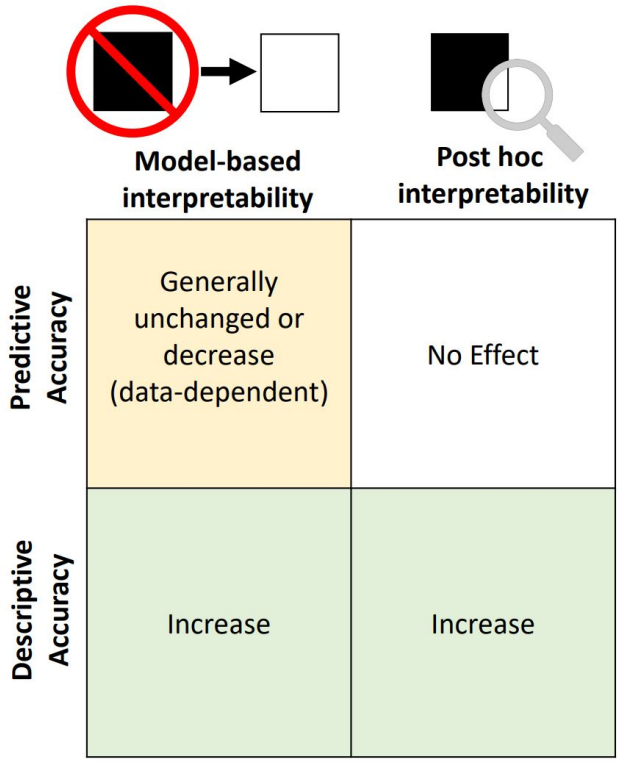


Figure 3.1.: The figure referenced is sourced directly from [MSK+19]. It illustrates the relationship between model-based and post hoc interpretability in the context of both predictive and descriptive accuracy.

3.2. ICE-Plots

Individual Conditional Expectation (ICE) plots offer an intuitive way of visually representing how a specific prediction changes as a given feature changes. Developed by Goldstein et al. in 2017 [GKBP15], these plots are designed to counter a limitation of the Partial Dependence Plots (PDP) [Fri01], which only show the average effect of a feature without focusing on individual instances. An ICE plot, on the other hand, provides a detailed view of the relationship between the prediction and the feature for each instance separately. This is done by changing the feature of interest and holding every other feature fixed and plotting the resulting prediction of the changed data point.

The mathematical formulation of the Individual Conditional Expectation plot can be rigorously described as follows:

Given a dataset X of size N, where each instance is denoted as a pair (x_i^V, x_i^F) for $i = 1, \ldots, N$. Here, x_i^V represents the variable of interest whose values we wish to modify, while x_i^F constitutes the portion of the instance that remains constant. Let ϕ be our predictive model. An ICE plot can be generated using the ensuing algorithm:

Algorithm 1 ICE Plot Calculation

```
1: procedure ICEPlot(X, φ, N, V) ▷ V indicates the part which will be changed and φ is our model
2:     for i = 1 to N do
3:         for j = 1 to N do
4:             Calculate φ(x_j^V, x_i^F)                ▷ ICE line for the sample i
5:             if j = i then
6:                 Plot dot to indicate the original value
7:             end if
8:         end for
9:     end for
10: end procedure
```

Another way in which the ICE line could be computed is to calculate the range of the feature x^V as $\min(x_i^V)$ and $\max(x_i^V)$ for all i and split the space evenly between them and evaluate the model on this grid points then. In this thesis, we will however use the first approach which only uses observed values as stand-ins. An example of an ICE plot can be seen in figure 3.2.

Interpretation Interpreting an ICE plot involves understanding the variations in the lines and what these imply about the model's predictions.

Trend and direction: The direction of the lines (upward, downward, horizontal) can indicate how changes in the feature affect the prediction. For instance, an upward trend suggests that an increase in the feature value leads to an increase in the predicted outcome while a horizontal line indicates that there is not much interaction and the outcome does not depend much on this variable.

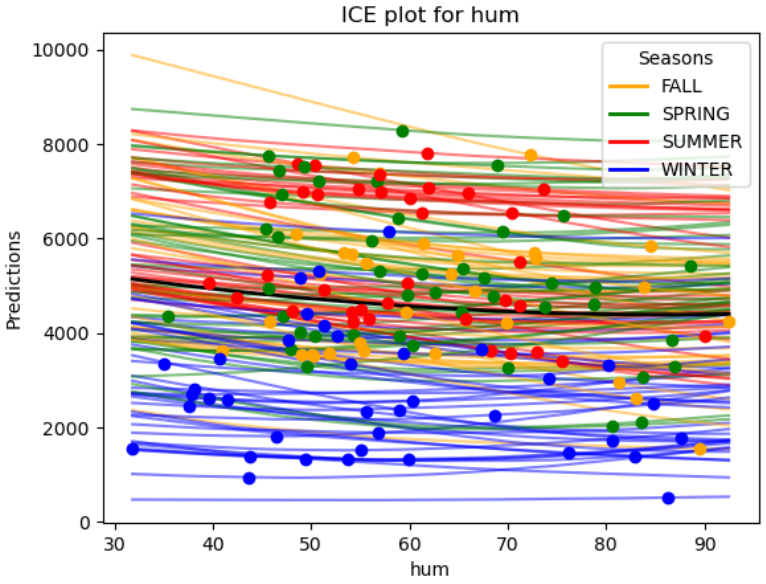


Figure 3.2.: This is a typical ICE plot for the variable humidity. The black line is the average of all lines (PDP line). The dots represent the original value of the instance. We also colored each lane based on another attribute (season).

Slope of the lines: The slope of the lines provides information about the rate of change in the prediction as the feature value changes. A steep slope indicates a high sensitivity of the prediction to changes in the feature value.
Variation across instances: Variations in the lines for different instances can hint at interactions between the primary feature under consideration and other features. When the lines for various instances diverge significantly, it suggests that the influence of the feature of interest is not uniform across all instances. Such inconsistencies might arise from interplay with other features.

Advantages One advantage of ICE plots is that they are very intuitive to understand. They also give an insight into the behavior of individual data points compared to PDP. ICE plots can uncover heterogeneous relationships that PDPs might obscure.

Disadvantages While ICE plots offer an intuitive and robust way to visualize the impact of a feature on model predictions for individual instances, they do come with some limitations. Firstly, the nature of ICE plots restricts us to examining one feature at a time. This could be limiting when trying to understand the interplay between multiple features in high-dimensional datasets. Secondly, when dealing with large datasets, the visualization may become cluttered due to the sheer volume of lines plotted. This high density of data points can make it challenging to discern clear patterns or interpret the plot. Finally, the approach used to create the grid for model evaluation can introduce

potential issues. By using a uniform distribution within the feature range to establish the grid, we risk generating invalid data points that do not align with the original distribution of the feature. This could lead to an inaccurate representation of the feature's influence and thus distort the interpretability of the model's behavior.

3.3. Shapley values

The calculation of Shapley values is an idea derived from game theory, which is primarily concerned with assigning a payout to players in a game based on their individual contributions. In the context of our work, the players are represented by the features of our model, and the game we are playing involves making the best possible prediction of the target variable. In this scenario, the payout can be reinterpreted as a measure of the relative importance of each feature. This allocation signifies how much each feature contributes to the prediction of the target. Therefore, the higher the absolut Shapley value, the more critical the feature is in the predictive process.

Because Shapley values can be calculated and applied after a model has been trained, they fall into the category of post-hoc methods. Importantly, they are model-agnostic, meaning they can be used with any type of predictive model. This flexibility and adaptability make Shapley values a powerful tool for understanding and interpreting model behavior.

Mathematically we define the shapley value θ of feature j by :

$$\theta_j(val) = \sum_{S \subseteq \{1,\dots,p=|V|\} \setminus \{j\}} \frac{|S|!(p-|S|-1)!}{p!} \left(val(S \cup \{j\}) - val(S)\right) \tag{3.1}$$

where S is a subset of all features (the number of available features is p) and val is a value function that calculates the worth or significance of the included features. Note that this value function can take different shapes and we will see one one specific choice shortly.

This formulation satisfies 4 very desirable properties which are called Efficiency, Symmetry, Dummy, and Additivity. Efficiency ensures the full value is allocated without any excess or shortfall. Symmetry maintains that variables with the same contribution receive equivalent shares. The dummy principle states that variables who don't contribute receive no payout. Lastly, additivity dictates that in combined scenarios, the total payout is the cumulative sum from individual situations. Together they are defining what a fair payout means. Importantly shapley values are uniquely characterized by these rules.

As easily seen, the calculation of the summation in Equation 3.1 requires a total of 2^d calculations, where d represents the number of features in the dataset. This results in an exponentially large number of subsets, even when d is relatively small. This combinatorial explosion makes the exact calculation impractical for most real-world datasets, given their size and complexity. For example, if the value function would be the accuracy or the mean squared error, we would need to retrain this model 2^d times (each time with a different combination of available features) which is infeasible. Therefore, to make the

computation manageable, it's often necessary to resort to approximations of these values.

Advantages Shapley values are based on a well-understood theory which is always good and the fact the payout is fairly distributed between the features makes it able to bring a full explanation to the table.

Disadvantages As already mentioned the enormous computing time makes them almost always infeasible to calculate exactly but they can be approximated. Additionally, the method needs to use all features so when we have a large number of features this makes it additionally challenging to interpret.

Interpretation: **Magnitude:** The magnitude of a Shapley value indicates the extent of a feature's contribution to the prediction outcome. A larger absolute value means the feature has a stronger impact.

Sign: The sign of a Shapley value indicates the direction of the feature's impact. A positive Shapley value implies the feature increases the prediction outcome, while a negative value means it reduces the prediction outcome.

Comparison: The Shapley values can be compared across features. The feature with the highest absolute Shapley value is deemed to be the most important one.

3.4. SHAP

SHapley Additive exPlanations (SHAP) provide individual explanations for predictions, drawing inspiration from Shapley values. Unlike traditional methods that necessitate retraining the model multiple times, SHAP leverages the model itself as its value function. When a feature for a specific instance is absent from the selected subset, also known as the coalition, SHAP allocates it a baseline value. This approach obviates the need for exponential retraining of the model. The overarching objective of SHAP is to interpret a model's prediction for a specific data point by breaking it down into contributions from each feature. Our discussion centers on KernelSHAP, which offers an approximation of SHAP values. It employs a weighted linear surrogate model, underpinned by a suitable weighting kernel. The derived weights from this surrogate model stand as the approximated SHAP values.

The implementation of KernelSHAP can be broken down into five sequential steps:

- First, we randomly select K coalitions $z_k \in Z \quad \forall k \in \{1, \ldots, K\}$ where $z_k \in \{0,1\}^d$. If $z_{k_i} = 1$, it indicates that the feature is included in the coalition, and if $z_{k_i} = 0$, it's not included.
- Next, we utilize the function $h_x(z_k)$, which maps the '1's to the original values of x and '0's to baseline values, to derive the model's predictions $f(h_x(z_k))$ (f is the trained model to interpret).
- We then calculate the weight for each coalition using the SHAP kernel π.

- The next step involves fitting a linear model where the data points $h_x(z_k)$, weighted by π, and the targets $f(h_x(z_k))$ are incorporated. Specifically, we aim to solve the following optimization problem:

$$argmin_g(L(f, g, \pi_x)) = min_g(\sum_{z \in Z}[f(h_x(z)) - g(z)]^2 \pi(z)) \tag{3.2}$$

  Here, $g(z) = \theta_0 + \sum_{j=1}^{M} \theta_j z_j$ represents a linear model and Z is the sampled set of coalitions.

- Finally, the coefficients derived from this linear model are an approximation to the Shapley values θ_k.

The SHAP kernel π is defined as:

$$\pi(z) = \frac{(M-1)}{\binom{M}{|z|}|z|(M-|z|)} \tag{3.3}$$

Here, M denotes the maximum coalition size or the number of features, and $|z|$ represents the 1 norm of z. This kernel gives high weight when $\|z\|_1$ is very small or very high. The implication is that we gather considerable information about a feature when it's considered in isolation, and when we have nearly all features available, we can also determine the total effect of a particular feature.

SHAP values obtained in this way satisfy properties similar to those inherent to the game-theoretic concept of Shapley values. We can then proceed to estimate a global feature importance measure using the SHAP methodology. This method is based on the straightforward principle that features with large absolute Shapley values are considered significant. The global importance of each feature is determined by averaging the absolute Shapley values per feature across the entire dataset, as represented by the equation:

$$I_j = \frac{1}{n} \sum_{i=1}^{n} |\theta j^{(i)}| \tag{3.4}$$

In this equation, I_j is the importance of the j-th feature, n is the number of data points, and $\theta j^{(i)}$ is the Shapley value of the j-th feature for the i-th data point. By focusing on the absolute value of the Shapley values, we assess their overall influence irrespective of their direction (positive or negative), which provides a measure of the magnitude of each feature's contribution to the overall model output, thus reflecting its importance.

Interpretation This method of computing Shapley values offers an intuitive approach to understanding a model's predictions. Each Shapley value can be interpreted as the amount either subtracted from or added to the prediction.

Advantages Since SHAP values are essentially shapley values they share the same advantages. A benefit is that this method can also be used for global measures such

as feature importance. Since we are using an approximation it is faster than the full computation of Shapley values. This method can also explain single predictions.

Disadvantages Kernelshap can be slow especially when using it to compute shapley values for a lot of instances as it would be needed when calculating global importance for example but as already said much faster than the full computation. Another point to be cautious about when interpreting the values is that it can destroy the feature dependence of features when setting one feature to the baseline while the other feature is in the coalition.

Visualizations One visualization that captures both local and global aspects is the SHAP summary plot 3.3. The summary plot combines feature importance and feature effects. Each point on the plot corresponds to a Shapley value for a particular feature-instance pair.

The y-axis of the summary plot is determined by the feature itself, while the x-axis represents the corresponding Shapley value. The color coding of the points provides additional information about the actual value of the feature, with a color gradient being used to represent values ranging from low to high. Due to overlap, points may be slightly adjusted (or "jittered") along the y-axis. This jittering aids in visualizing the distribution of the Shapley values for each feature, providing a sense of their spread and concentration.

Furthermore, the features are ordered in terms of their importance. This ordering typically means that features towards the top of the plot have higher global importance (according to the Shapley values), while those towards the bottom have less.

A typical summary plot can be seen in Figure 3.3.

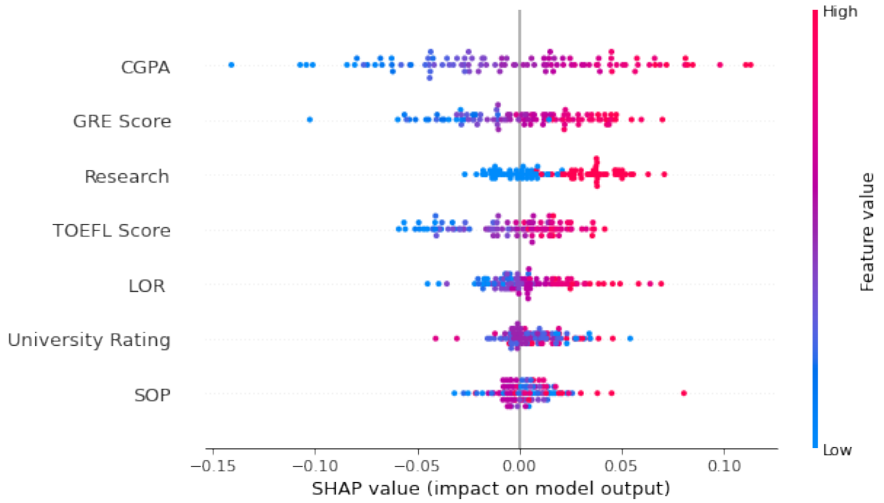


Figure 3.3.: This plot shows a SHAP summary plot. The analyzed model is a NN trained on the admission dataset.

Another frequently used visualization technique is the SHAP waterfall plot. This plot

is a graphical representation that shows the contribution of each feature to the prediction for a specific instance in the dataset.

This visualization resembles a waterfall: it starts with the expected prediction value (base value), and each feature either increases or decreases this base value based on its contribution to the overall prediction. The colors of the bars in a waterfall plot can be used to indicate whether the contribution is positive or negative. Red bars indicate positive contributions, while blue bars indicate negative contributions. The ordering of the features typically starts with the one having the most significant impact, gradually descending to those with less influence.

Through the SHAP waterfall plot, one can better understand how different features sway the model's decision, thereby enhancing interpretability. It is particularly beneficial in providing a granular, instance-specific interpretation, making it an effective tool for post-hoc interpretability.

An example of a SHAP waterfall plot is depicted in Figure 3.4. In this case, it is flipped upside down.

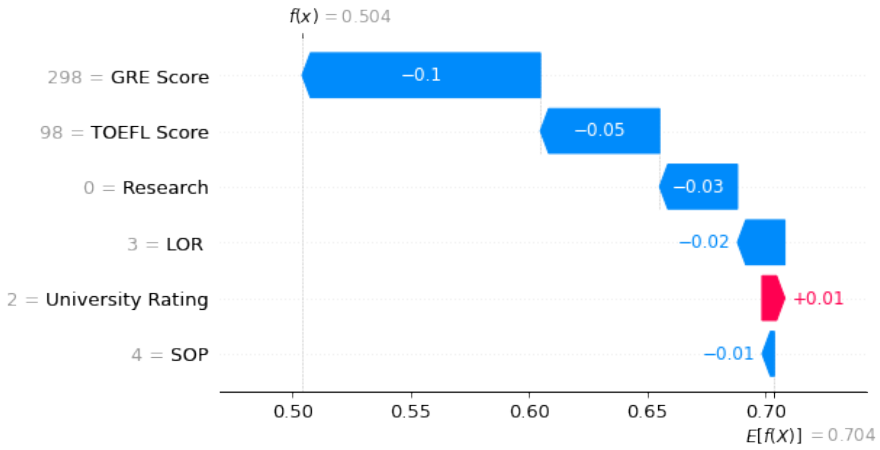


Figure 3.4.: This Figure shows a SHAP waterfall plot of a NN trained on the admission dataset.

3.5. Variational Autoencoders

A Variational Autoencoder (VAE) is a type of autoencoder that is designed to learn probabilistic mappings between the data space and a latent space. It is a generative model, which means it can generate new data samples that are similar to the input data. It works by encoding input data into a lower-dimensional latent space and then reconstructing the original data from this latent representation. Often a traditional autoencoder does not impose any structure on the latent space, whereas a VAE imposes the condition that the latent variables follow some distributions (a Gaussian for example). Thus we learn a distribution p of our data over our latent space. The generative model is represented by

$p_\theta(z)p_\theta(x|z)$ and it simulates the process of data creation. Initially, it produces z using $p_\theta(z)$ Subsequently, it generates x based on the value of z through $p_\theta(x|z)$. Nevertheless, determining the actual distribution of z when given x, denoted as $p_\theta(z|x)$ is frequently intricate and challenging to compute. Therefore, VAE introduces a simpler, variational approximation to this posterior, denoted as $q_\phi(z|x)$ which is modeled by a neural network and called the probabilistic encoder or $Enc_\theta(x)$. Similar $Dec_\theta(z)$ is the parameterization of the probabilistic decoder $p_\theta(x|z)$ which is also a neural network. Figure 3.5 shows this architecture.

Figure 3.5.

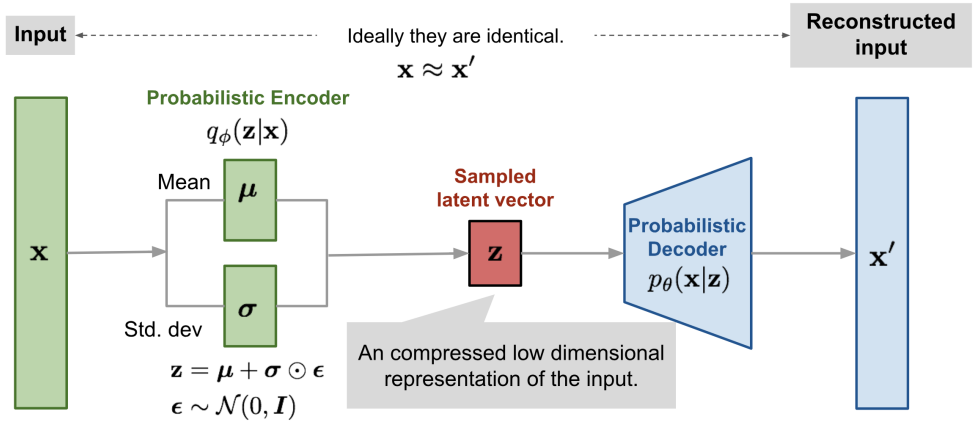


Figure 3.6.: This image is taken from [Wen18].

Since a loss function is needed to train the nets we will present how to derive the ELBO loss which is the standard choice. The following derivation is partially from [Wen18]

Instead of mapping the input into a fixed vector like the traditional autoencoder, we want to map it into a distribution p_θ.

The relationship between the data input x and the latent encoding vector z can be fully defined by:

- Prior: $p_\theta(z)$
- Likelihood: $p_\theta(x|z)$
- Posterior: $p_\theta(z|x)$

We could then generate new data points from this if we would know the real parameter θ^* for p_θ by first sampling z from the prior distribution $p_\theta(z)$ and then generating x from the conditional distribution $p_\theta(x|z)$.

The optimal parameter θ^* (maximum likelihood estimation) is the one which maximises $\theta^* = \arg\max_\theta \prod_{i=1}^n p_\theta(\mathbf{x}^{(i)})$. Taking the logarithm we get $\log(\theta^*) = \arg\max_\theta \sum_{i=1}^n \log(p_\theta(\mathbf{x}^{(i)}))$.

For a single summand, we can derive the following lower bound.

$$\begin{aligned}
\log(p_\theta(x)) &= \log \int_z p_\theta(x,z) \\
&= \log \int_z p_\theta(x,z) \frac{q_\phi(z|x)}{q_\phi(z|x)} \\
&= \log \mathbb{E}_{q_\phi(z|x)}[\frac{p_\theta(x,z)}{q_\phi(z|x)}] \\
&\geq \mathbb{E}_{q_\phi(z|x)}[\log(\frac{p_\theta(x,z)}{q_\phi(z|x)})] \\
&= \mathbb{E}_{q_\phi(z|x)}[\log(\frac{p_\theta(x|z)p_\theta(z)}{q_\phi(z|x)})] \\
&= \mathbb{E}_{q_\phi(z|x)}[\log(p_\theta(x|z))] + \mathbb{E}_{q_\phi(z|x)}[\log(\frac{p_\theta(z)}{q_\phi(z|x)})] \\
&= \mathbb{E}_{q_\phi(z|x)}[\log(p_\theta(x|z))] + \int_z q_\phi(z|x) \log(\frac{p_\theta(z)}{q_\phi(z|x)}) \\
&= \mathbb{E}_{q_\phi(z|x)}[\log(p_\theta(x|z))] - D_{KL}[q_\phi(z|x)||p_\theta(z)] := L(\theta,\phi,x)
\end{aligned} \tag{3.5}$$

We have introduced $q_\phi(z|x)$, which approximates the often intractable posterior $p_\theta(z|x)$. L is called the evidence lower bound (ELBO) and is our loss function for the VAE and we want to maximize this. The term $D_{KL}[q_\phi(z|x)||p_\theta(z)]$ is the Kullback-Leibler divergence which is a measure of the similarity of distributions. Since there is a minus in front of the KL divergence we want this to be as small as possible which means we want the learned distribution by $q_\phi(z|x)$ to be as close as possible to $p_\theta(z)$. If we assume that $p_\theta(z) \sim \mathcal{N}(0,1)$ we then get a closed-form solution of the D_{KL} (since $q_\phi(z|x)$ is also modeled as a gaussian) which is $-D_{KL}[q_\phi(z|x)||p_\theta(z)] = \frac{1}{2}\left(1 + \log(\sigma_j^2) - \mu_j^2 - \sigma_j^2\right)$ where μ and σ are taken from the $q_\phi(z|x)$ [KW22]. The term $\log(p_\theta(x|z))$ can be implemented as $-\frac{1}{2}|x - Dec_\theta(z)|^2$ which is the reconstruction loss. Implementing it like this is a design choice.

This means we want to minimize :

$$\begin{aligned}
-L &= -\mathbb{E}_{q_\phi(z|x)}[\log(p_\theta(x|z))] + D_{KL}[q_\phi(z|x)||p_\theta(z)] \\
&= \text{Reconstruction Loss} - \text{Difference of } q_\phi(z|x) \text{ and } p_\theta(z)
\end{aligned} \tag{3.6}$$

To calculate the reconstruction loss we have to generate samples z from $q_\phi(z|x)$ which is a probabilistic process where we can not use standard backpropagation. Thus we use the reparameterization trick where we learn the mean and standard deviation and then use the following formula to calculate z.

$$z = \mu + \sigma * \epsilon, \text{ where } \epsilon \sim N(0, I) \tag{3.7}$$

and $*$ is element-wise. This basically outsources the randomness to ϵ and thus enables us

to use backpropagation on the remaining deterministic variables.

Beta-VAE A popular variation of an autoencoder is the Beta-VAE. The difference to the traditional variational autoencoder is that in the ELBO loss, we add a hyperparameter β in front of the KL-Divergence loss term.

$$L_{BETA} = -\mathbb{E}_{q_\phi(z|x)}[log(p_\theta(x|z))] + \beta D_{KL}[q_\phi(z|x)||p_\theta(z)] \tag{3.8}$$

It is also common to add a hyperparameter in front of the reconstruction loss which would result in :

$$L_{BETA} = -\beta_1 \mathbb{E}_{q_\phi(z|x)}[log(p_\theta(x|z))] + \beta_2 D_{KL}[q_\phi(z|x)||p_\theta(z)] \tag{3.9}$$

Of course, this is equivalent to the first formulation except for scaling.

An increased β_2 value can promote more disentangled representations. This enhancement results in a more interpretable latent space, where each dimension independently encodes distinct and meaningful characteristics of the data.

4. Methods

In this section, we outline the methodology we used to answer the research questions presented in chapter 1. We will divide the methodology into two segments, each addressing its respective question. It should be noted, however, that there will be areas of overlap, particularly in relation to methods of interpretability.

4.1. Research Question 1

To address our first research question, which pertains to the impact of a confounding variable on the interpretability of neural networks (complete question in chapter 1), we used the following methodology.

Confounder Selection: We initially identify suitable confounders that exhibit a high correlation with both the target variable and other features within the dataset. For the scope of this study, only one confounder is selected for each dataset. Further details on the chosen confounders are provided in chapter 5.

Dataset Preparation: Subsequent to confounder selection, two versions of each dataset are prepared. The first version incorporates all variables, while the second excludes the selected confounder, effectively rendering it hidden.

Data Partitioning: To ensure a consistent and fair comparative analysis, the same training, validation, and testing datasets are employed for both versions of each dataset.

Optimal Model Identification: To find the most suitable models for our datasets, we employ a multi-step methodology that incorporates a variety of techniques. See 4.1.1 for details.

Model Interpretation: Armed with these refined models, we employ several interpretable methods, such as ICE plots, truth-prediction plots, and KernelSHAP. The models are analyzed and compared using these methods to identify disparities in their behavior. The findings are subsequently examined and clarified by the author. Comparative plots from all models are arranged side-by-side to enable a nuanced evaluation of differences. Special focus is given to the potential effects arising from the exclusion of a confounder. In the case of Individual Conditional Expectation plots, lines are color-coded based on the value of the confounder variable, adding an extra layer of interpretive depth.

4.1.1. Multi-step methodology to find best models for RQ1

1. We first establish a hyperparameter search space. The exact parameter space can be found in chapters 6 and A.2.

Next, we train 100 models with hyperparameters uniformly sampled from this search space. We employ 5-fold cross-validation, only using train data of a random 80/20 train-test split, to calculate the R^2 score and loss for each model. The ASHAScheduler [LJR+18] is utilized for early stopping in cases where the R^2 score is particularly low.

2. We then select the hyperparameters corresponding to the highest R^2 score and proceed to fine-tune the learning rate, batch size, and epochs using scikit-optimize [HKN+21] again using cross-validation. When using scikit-optimize we specifically employ the gp_minimize function, which performs Bayesian optimization with a Gaussian Process. We run 100 iterations for this optimization process.

3. Armed with these optimized parameters, and the consistent train, test, and validation sets (meaning the train, validation and test sets with which we train the full models and the confounding models are the same) at hand we train the final models and incorporate early stopping to prevent overfitting. These resulting models are saved as our final optimized models.

All models are implemented in PyTorch [PGM+19], and hyperparameter tuning is facilitated by Ray [MNW+17]. The exact implementation can be found on Github `https://github.com/patrickpollek1/Masterthesis-Interpreting-Neural-Networks-Under-Latent-Confounding`.

4.2. Research Question 2

The second question that the research aimed to address was whether we can find evidence that a variational autoencoder which also takes the variable y into account improves the interpretability in the sense of learning more intuitive relations and interpretations that are more aligned with reality (complete question in chapter 1).

To answer this we have to adapt the VAE model described in the background chapter (3.5) to also include y in some sense. The reason why we also want y in our model comes naturally when we look again at the setting of our confounding variable. In Figure 4.1 we see that z, the confounder, is influencing x and y simultaneously. Thus theoretically we want to learn z from x and y which would mean our input to the VAE would be x and y concatenated. This approach was also taken by [LSM+18]. The idea is to acquire a latent representation that is both more entangled and independent and potentially model hidden confounders. This representation should make it easier for the model to predict y and improve interpretability.

Figure 4.1 would lead us to a factorization of the distribution $p(x, y, z)$ as $p(x, y, z) = p(z)p(x|z)p(y|x, z)$.

Using the same techniques as in the background section we want to derive a lower bound on $p_\theta(x, y)$ in order to maximize it. Thus we have:

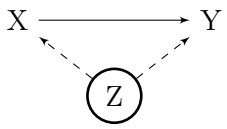


Figure 4.1.: This figure shows a typical confounding variable Z which influences both the independent variable X and the dependent variable Y.

$$
\begin{aligned}
\log(p_\theta(x,y)) &= log \int_z p_\theta(x,y,z) \\
&= \log \int_z p_\theta(x,y,z) \frac{q_\phi(z|x,y)}{q_\phi(z|x,y)} \\
&= \log \mathbb{E}_{q_\phi(z|x,y)}[\frac{p_\theta(x,z)}{q_\phi(z|x,y)}] \\
&\geq \mathbb{E}_{q_\phi(z|x,y)}[\log(\frac{p_\theta(x,y,z)}{q_\phi(z|x,y)})] \\
&= \mathbb{E}_{q_\phi(z|x,y)}[\log(\frac{p_\theta(z)p_\theta(x|z)p_\theta(y|x,z)}{q_\phi(z|x)})] \\
&= \mathbb{E}_{q_\phi(z|x,y)}[\log(p_\theta(x|z)p_\theta(y|x,z))] - D_{KL}[q_\phi(z|x,y)||p_\theta(z)]
\end{aligned}
\tag{4.1}
$$

This result mirrors the traditional loss mechanism utilized by the standard variational autoencoder, albeit with an additional layer of complexity wherein the variable y is incorporated as an input to our VAE and subsequently reconstructed. During the model's training phase, this process is easily managed due to the availability of target variables. However, this approach becomes significantly more challenging during the testing stage when y is not readily available.

We can either learn an additional integrated model that also predicts y to then use the model and get the latent representation or we could incorporate the target in a different way, without using y as an input to our model. In the context of this research, we have chosen to adopt the latter strategy and propose a different model.

The proposed strategy involves learning a separate latent variable z' from our existing latent space z, which might potentially model some information about a hidden confounder. Given that we learn z from x and z' from z, we expect z' to contain some information from x. Following this, we attempt to predict y using a combination of z and z', ensuring z' also encapsulates information about y. This process emulates to some degree the structure found in the factorization of the joint distribution $p(x,y,z)$ and the causal setting.

To develop an appropriate loss function, we first resort to calculating a lower bound on

the likelihood :

$$\begin{aligned}
log(p_\theta(x)) &= log \int_z \int_{z'} p_\theta(x,z,z') \\
&= log \int_z \int_{z'} p_\theta(x,z,z') \frac{q_\phi(z,z'|x)}{q_\phi(z,z'|x)} \\
&= log\, \mathbb{E}_{q_\phi(z,z'|x)}[\frac{p_\theta(x,z,z')}{q_\phi(z,z'|x)}] \\
&\geq \mathbb{E}_{q_\phi(z,z'|x)}[log(\frac{p_\theta(x,z,z')}{q_\phi(z,z'|x)})] \\
&= \mathbb{E}_{q_\phi(z,z'|x)}[log(\frac{p_\theta(z)p_\theta(z'|z)p_\theta(x|z,z')}{q_\phi(z,z'|x)})] \\
&= \mathbb{E}_{q_\phi(z,z'|x)}[log(\frac{p_\theta(z)p_\theta(z'|z)p_\theta(x|z,z')}{q_{\phi 1}(z|x)q_{\phi 2}(z'|z)})] \\
&= \mathbb{E}_{q_\phi(z,z'|x)}[log(p_\theta(x|z,z'))] - D_{KL}[q_{\phi 1(z|x)}||p_\theta(z)] - D_{KL}[q_{\phi 2}(z'|z)||p_\theta(z'|z)] \\
&:= ELBO_{z'}
\end{aligned} \tag{4.2}$$

which we want to maximize.

In the calculations above we have split the encoder (q_ϕ) into 2 parts. $q_{\phi_1}(z|x)$ decodes x into z and $q_{\phi 2}(z'|z)$ decodes z into z'. This way we can learn z' from z and model a potential confounder. (Actually, it might only learn the portion of the confounder that is not already in some form in x already and only the part that is needed for y.) The derived lower bound ($ELBO_{z'}$) can be interpreted in the following way. The first expectation is implemented as the reconstruction error of x using z and z'. We then also have the KL-divergence terms which measure the similarity of distributions, which we will assume to be both normal Gaussian's, allowing for an explicit solution and straightforward implementation.

But this is not enough since we also want to incorporate the loss that arises when we predict the target variables from z and z'. Luckily this is straightforward and can be implemented as an MSE between the prediction and the true value. This requires an additional part to our VAE which we call $\psi(z,z')$, and which is a feed-forward NN.

The complete loss is then

$$\begin{aligned}
L := \beta_1\, &\mathbb{E}_{q_\phi(z,z'|x)}[log(p_\theta(x|z,z'))] - \beta_2 D_{KL}[q_{\phi 1}(z|x)||p_\theta(z)] \\
&- \beta_3 D_{KL}[q_{\phi 2}(z'|z)||p_\theta(z'|z)] + \beta_4[\psi(z,z') - y]^2
\end{aligned} \tag{4.3}$$

, where y is the true target value and the β's are hyperparameter.

The whole architecture of this model can be summarized as follows:

- $Enc_z(x) = q_{\phi 1}(z|x)$ is the encoder of x into z. This means $z|x$ is normally distributed as $N(\mu_{\phi 1}(x), \sigma_{\phi 1}(x)^2)$.

- $Enc_{z'}(z) = q_{\phi 2}(z'|z)$ is the encoder of z into z'. This means $z'|z$ is normally distributed as $N(\mu_{\phi 2}(z), \sigma_{\phi 2}(z)^2)$.
- $Dec(z, z')$ is the decoder from z and z' to x
- $\psi(z, z')$ is the predictor from z and z' to y

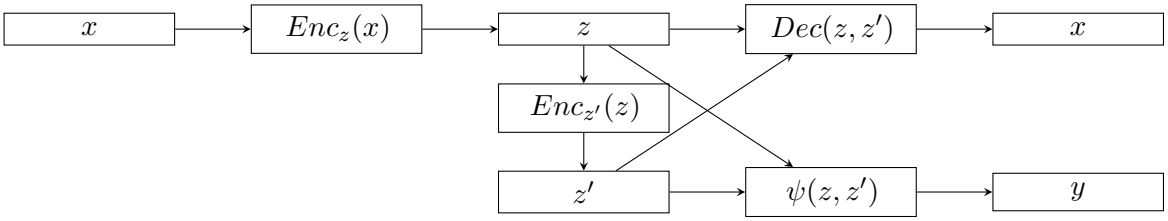


Figure 4.2.: This figure shows the flow of a sample through our VAE

Since we employ the parameterization trick we also have 2 additional layers per decoder which directly parameterize the mean and the logarithmic variance.

We used 2 different approaches on how to train this model. The first approach is the standard one where all parameters are trainable all the time and the loss function incorporates all terms. The second approach only trains the auto-encoder part first where we leave out the loss term of the prediction of y. After T_f epochs we freeze the weights of the encoder $Enc_z(x)$. After that, our loss is simply $\beta_4[\psi(z, z') - y]^2$. We then train an additional number of epochs like this, specified in O_f, and after that, we freeze now also $Enc_{z'}(z)$ and $Dec(z, z')$. Finally, we let this train for the remaining number of epochs.

For training, we first used a 5-fold CV to select the best hyperparameters. We again used PyTorch for the implementation and used Adam as our optimizer. The exact parameter spaces and further information can be found in the section 6.

The single best-performing model of both approaches is then trained again several times and then one is selected based on performance and the structure of the learned latent space. Structure means we investigate how each latent variable is correlated with the input and target variables and look for a nonuniform structure, meaning different variables encode different parts of the data.

Because we sample each forward pass through our model from a probability distribution, we will have (slightly) different latent variables and outputs each time. This affects our evaluation and interpretations of this model. For example, when reporting the R^2 score of the model this will be an average score of multiple evaluations (10 times). The same is true for the train and test loss and also for the creation of ICE plots and shapley values. We ran the prediction of the output multiple times which makes the plots cleaner and more comparable. Specifically, we averaged the shapley value calculation over 100 runs and the ICE plot calculations over 10 runs for the admission dataset and toy dataset and 20 runs for the bike rental dataset.

5. Datasets

We are going to use 3 different Datasets to answer the research questions.

5.1. Bike Rentals

The first dataset, known as the Bike Rentals Dataset, was initially released by Capital Bikeshare and later enriched by Fanaee-T and Gama [FTG14] with the inclusion of weather data. The primary objective of this dataset is to predict the daily bike rental demand, accounting for various factors such as weather conditions or if the day is a holiday or not. Note that we do not use every feature that is present in the original data but leave some out and use the preprocessing as in [Mol22]. The complete list of features we use in this thesis can be found in Table 5.1. A Table of Summary Statistics can be found in the appendix [A.1]

Feature	**Description**
season	Categorical: spring, summer, fall, or winter
yr	Categorical: 2011 or 2012
holiday	Binary: holiday (1) or not (0)
workingday	Binary: working day (1) or weekend (0)
weathersit	Categorical: weather conditions
temp	Continuous: temperature in degrees Celsius
hum	Continuous: relative humidity (0-100%)
windspeed	Continuous: wind speed in km/h
cnt	Continuous: total bike rentals (target variable)
days_since_2011	Continuous: days elapsed since 01.01.2011

Table 5.1.: Features of the Bike Rentals Dataset

We one-hot encoded weather situation into 3 separate features namely: weathersit_GOOD, weathersit_MISTY, and weathersit_RAIN/SNOW/STORM. The same was done for season (eg. into season_SPRING,season_SUMMER, and so on).

Figure 5.1 shows the course of the number of rented bikes over the whole period of the dataset. We can see a strong upward trend year over year.

Because we need to select suitable confounders, we have to look at the correlation of the features to see which are correlated with both other features and the target variable. Table 5.2 summarizes this. It is not surprising that days_since_2011 and year are highly correlated as well as the weather situation since it can not be both at the same time and

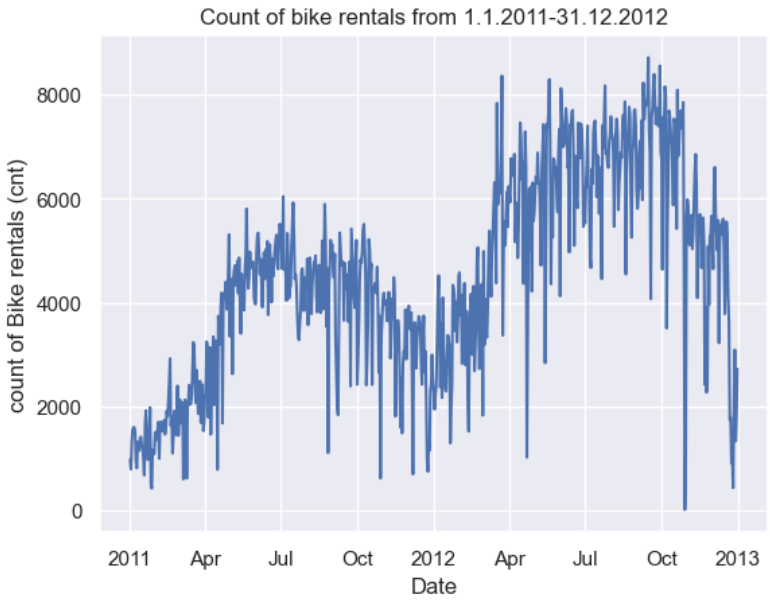
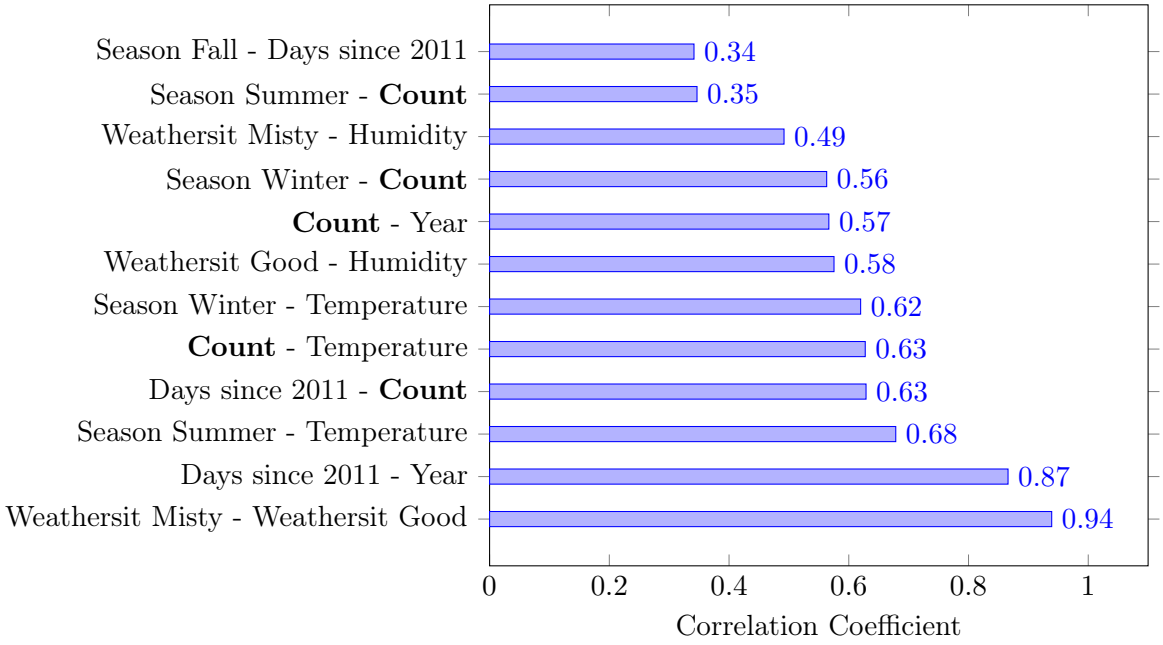


Figure 5.1.: This figure shows the number of rented bikes on each day.

weathersit_RAIN/SNOW/STORM happens not so often. An interesting variable are the Season variables. For example Season Summer correlates with Temperature with a coefficient of 0.6784 which makes sense since typically the temperature goes up in the summer. But Season Summer also correlates with our target variable count (0.3464), since a lot of people enjoy biking much more in Summer than in Winter. So it is plausible that the variable Season Summer influences both the variable Temperature and the variable Count. This means that the season might be a good candidate to be chosen as a confounding variable in our setting and this is what we did.

Figure 5.2.: (absolute) Correlation Coefficients



5.2. Graduate Admissionss

This dataset, inspired by the UCLA Graduate Dataset, aims to predict graduate admissions for Indian students by considering various factors such as test scores, research experience, and the quality of the statement of purpose. The full list of features used in this study is provided in Table 5.2, while a summary statistics table can be found in the appendix [A.2]. The dataset was made publicly available by Mohan S. Acharya et al. (2019) [AAA19] and can be accessed on Kaggle at the following link: `https://www.kaggle.com/datasets/mohansacharya/graduate-admissions`.

Feature	**Description**
GRE Score	Continuous: GRE Scores (out of 340)
TOEFL Score	Continuous: TOEFL Scores (out of 120)
University Rating	Categorical: University Rating (out of 5)
SOP	Ordinal: Statement of Purpose (out of 5)
LOR	Ordinal: Letter of Recommendation (out of 5)
CGPA	Continuous: Undergraduate GPA (out of 10)
Research	Binary: Research Experience (either 0 or 1)
Chance of Admit	Continuous: Chance of Admit (ranging from 0 to 1)

Table 5.2.: Features of the Graduate Admissions Dataset

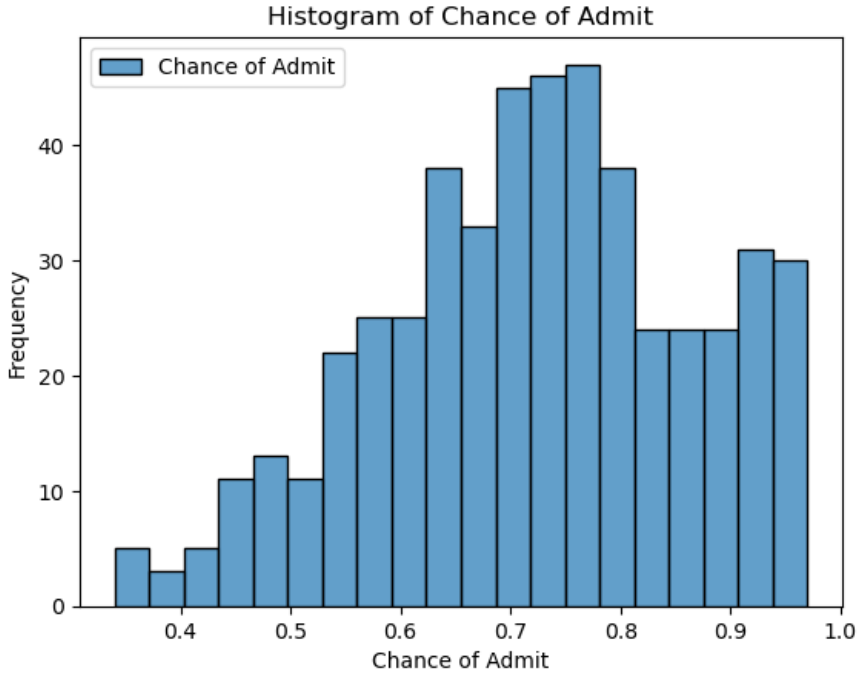


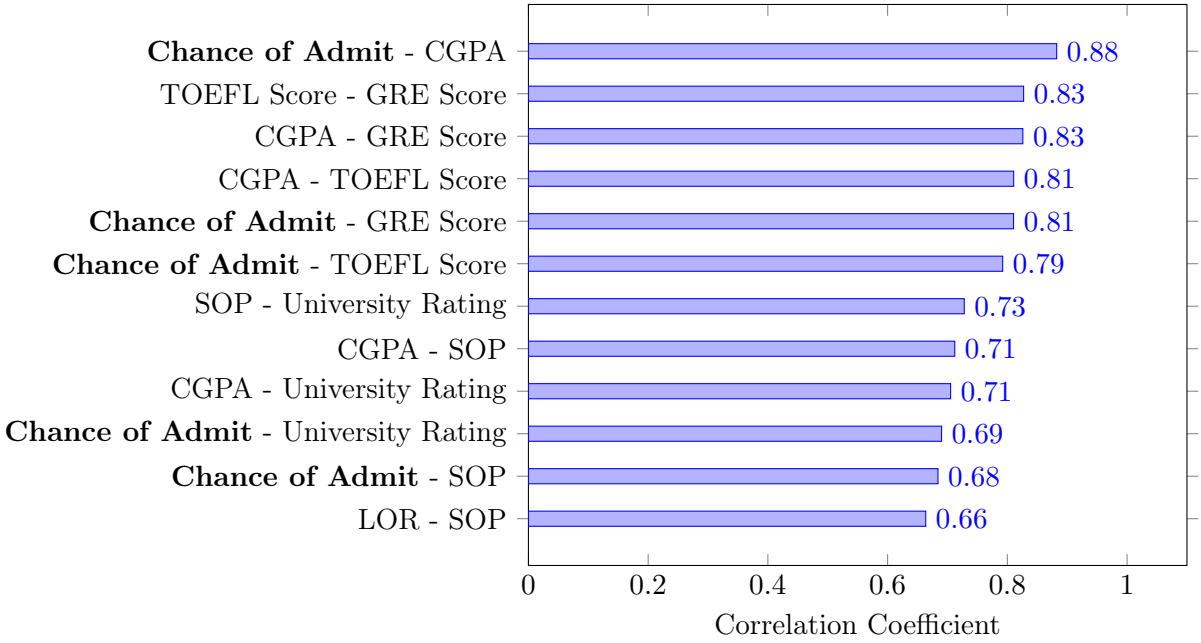
Figure 5.3.: This figure shows the frequency of the chance of admission feature.

The GRE score is a standardized test result, which assesses a candidate's verbal reasoning, quantitative reasoning, and analytical writing abilities for graduate school admissions. This test also includes a Verbal section where vocabulary usage is a part of it. This test is taken after finishing the bachelor's degree. On the other hand, the TOEFL score is a standardized assessment that evaluates a candidate's English language proficiency in reading, listening, speaking, and writing skills. The Cumulative Grade Point Average (CGPA) score serves as a comprehensive metric for evaluating a student's overall academic performance throughout their bachelor's degree program.

Therefore, it is plausible to contend that the CGPA score may exert a significant influence on both the GRE score, with a correlation of 0.83, and the Chance of Admit, with a correlation of 0.88 (Figure 5.4). This suggests that strong academic performance during a bachelor's program is likely to positively impact the likelihood of gaining admission to a master's program, as well as achieving a high GRE score.

Thus we select the CGPA score as our confounder.

Figure 5.4.: (absolute) Correlation Coefficients



5.3. Toy Dataset

This dataset is a synthetically generated dataset created to test our hypothesis and to ensure full control over the impacts of all attributes. In contrast to the other smaller datasets, which consist of 500 and 731 instances, we will generate a significantly larger number (200k) of samples to create a comparatively extensive dataset.

The following list describes the generation process :

1. X_0: A continuous attribute derived from X_1, $X_0 = \frac{X_1 - 0.8\epsilon_1}{2}$, where $\epsilon_1 \sim \mathcal{N}(2, 1)$.
2. X_1: A continuous attribute sampled from a normal distribution $X_1 \sim \mathcal{N}(3.6, 4.64)$.
3. X_2: A continuous attribute sampled from a normal distribution, $X_2 \sim \mathcal{N}(0, 0.8)$.
4. X_3: A continuous attribute obtained by taking the square root of the absolute product of the second and third attributes, $X_3 = \sqrt{|X_1 X_2|} + 0.2\epsilon_2$, where $\epsilon_2 \sim \mathcal{N}(0, 1)$.
5. X_4: A continuous attribute generated from a normal distribution, $X_4 = \mathcal{N}(160, 4) + X_5\mathcal{N}(5, 4)$.
6. X_5: A binary attribute generated from a binomial distribution, $X_5 \sim \text{Binomial}(1, 0.5)$.

The target variable Y was constructed using a linear combination of the attributes, along with some non-linear transformations and interactions:

$$Y = 1400 + X_0 + 30X_5 + 10X_1X_2 + 7\left(\frac{X_3}{2} + 5\right)^2 - 2\sin(X_4 + 8) + 3X_1,$$

The design of X_0 demonstrates that X_1 is an appropriate selection for a hidden confounder. This is because X_0 is directly influenced by X_1 and additionally, it is observable from Y that X_1 also has an impact on the target variable.

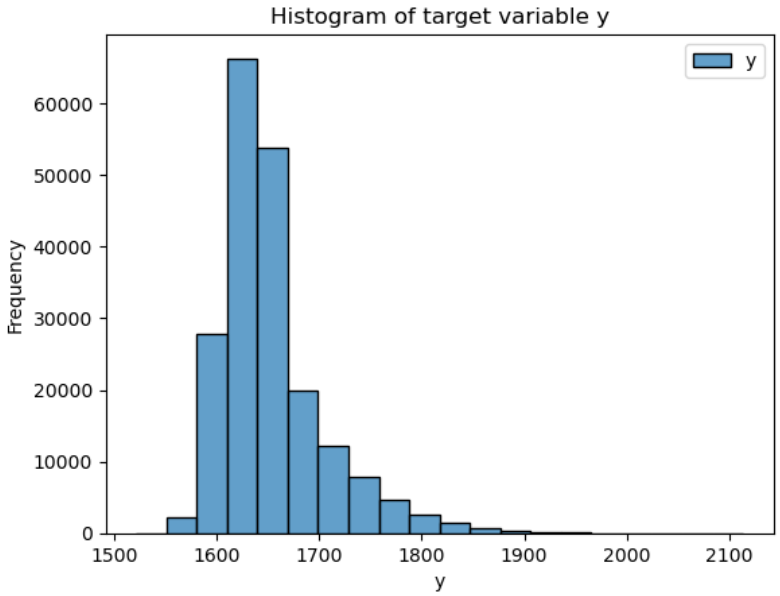
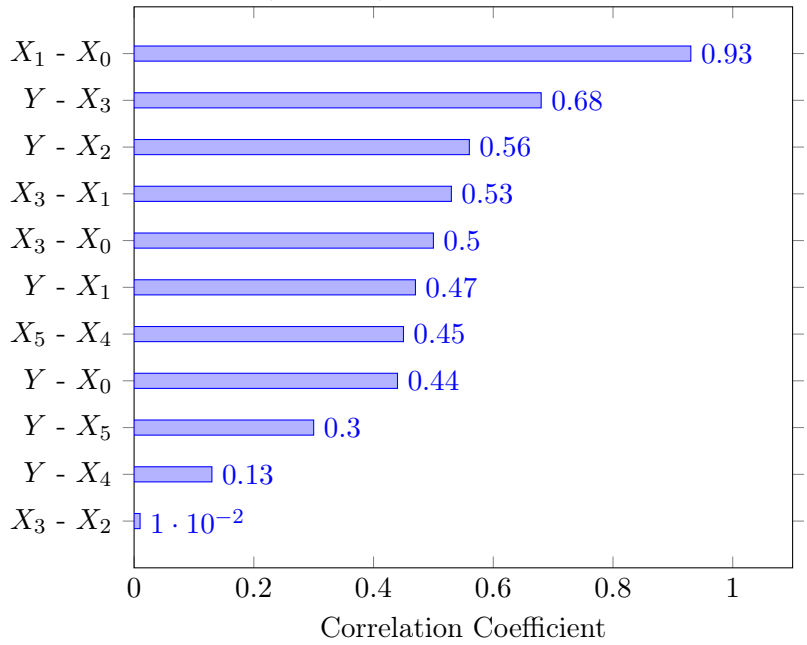


Figure 5.5.: This figure shows the frequency of the target variable Y.

Figure 5.6.: (absolute) Correlation Coefficients



6. Experimental Setup

In this section, we will specify all hyperparameter spaces that we used in this thesis.

6.1. Training of FFNN

The hyperparamter space to identify the optimal FFNN's (as in 4.1.1) were :

- **Batch size**: For the toy dataset, we consider values in the range [128, 256, 512, 1024, 2048]. For the remaining datasets, we use the range [8, 16, 32, 64, 128]. The toy dataset has a different range due to its larger size.
- **Learning rate**: We use a loguniform distribution between 10^{-7} and 10^{-2}.
- **Epochs**: For the toy dataset, we consider values in the range [10, 25, 50, 75]. For the other datasets, we use the range [25, 50, 100, 150, 300, 500].
- **Hidden layers**: We select a random integer between 1 and 4. This integer determines the number of hidden layers.
- **Hidden units**: For each layer, we consider values in the range [16, 32, 64, 128].
- **Dropout**: We select either True or False. This determines if we use dropout or not.
- **Dropout array**: If dropout is set to True, we sample a uniform value between 0.1 and 0.8 for each layer.
- **BatchNorm**: We select either True or False to determine whether a BatchNorm1d layer is included at the beginning.

The parameter space for further fine-tuning (step 2 in 4.1.1) of these models using scikit-optimize are :

Bike-Rental dataset

Hyperparameter	Range	Sampling Procedure
batch_size	$(8, 256)$	uniform
learning rate (lr)	$(10^{-4}, 10^{-2})$	log-uniform
epochs	$(100, 500)$	uniform

Admission dataset

Hyperparameter	Range	Sampling Procedure
batch_size	$(8, 256)$	uniform
learning rate (lr)	$(10^{-4}, 10^{-2})$	log-uniform
epochs	$(30, 250)$	uniform

Toy dataset Note that for the toy dataset, we only used 50 iterations of the optimization algorithm (gp_minimize) for further fine-tuning because of the long computation time. We also subsampled the data further by a factor of 10. This left us with a train set of 16k data points.

Hyperparameter	Range	Sampling Procedure
batch_size	$(25, 100)$	uniform
learning rate (lr)	$(10^{-4}, 10^{-2})$	log-uniform
epochs	$(25, 100)$	uniform

6.2. Training of VAE models

In this section, we detail the parameter spaces employed for training our VAE models. Through initial tests, it became clear that predictive accuracy is greatly influenced by the choice of betas. Specifically, when betas were of similar magnitude, the resulting latent space consistently showed a uniform correlation structure, which was undesirable and led to worse predictive accuracy. We discovered that elevating betas for reconstruction errors and prediction loss led to a more varied latent space. Due to limited time and computational resources, only a very small parameter space was chosen.

Bike rental dataset training For the bike rental dataset, we also scaled the target values y by $\frac{1}{10000}$. This solved some issues during training when we encountered NAN and inf values.

The parameters for the Bikerental Dataset can be found in table 6.1:

The settings for freezing the weights were $T_f = 500$ and $O_f = 200$. For the bike rental dataset, we took the 3rd best model based on R^2 score where we froze the parameters since we saw that the latent space of the no-freeze model was very uniform and highly correlated with the target variable already although the performance was superior to the freeze models.

Parameter Name	Possible Values
z_dim_values	[4]
z'_dim_values	[1]
$Enc_z(x)$	[[12,12,12],[128,128]]
$Enc_{z'}(z)$	[[5,5,5]]
$\psi(z, z')$	[[28,28,1]]
BETAS	[[β_1: 10000, β_2: 1, β_3: 1,β_4: 1000], [β_1: 1, β_2: 1, β_3: 1, β_4: 1]]
learning_rate	[10^{-3}]
batch_size	[16,32]
num_epochs	[700,1000]

Table 6.1.: [12,12,12] means that we have 3 layers each with 12 neurons fully connected. The decoder has always the same size as the encoder with the last layer having the dimension of the input. (If the encoder has [12,12,12] then the decoder has [12,12,12,d] where d is the dimension of the input data)

Admission dataset training The hyperparameter space for the VAE model can be found in table 6.2

Parameter Name	Possible Values
z_dim_values	[3,4]
z'_dim_values	[1]
$Enc_z(x)$	[[10,10,10],[64,64,64]]
$Enc_{z'}(z)$	[[10,10]]
$\psi(z, z')$	[[12,12,1],[32,32,32,1]]
BETAS	[[β_1: 10000, β_2: 1, β_3: 1, β_4: 10000]]
learning_rate	[10^{-3}]
batch_size	[16,32]
num_epochs	[600,800,1000]

Table 6.2.: Again the decoder has the same size as the encoder for each experiment.

The settings for freezing the weights were $T_f = 400$ and $O_f = 200$. For the admission dataset, we took the 3rd best model based on R^2 score where we did not freeze the parameters. We selected this model because we saw that the latent space had a clear highly correlated structure between the variables.

Toy dataset training The hyperparameter space for the VAE model can be found in table 6.3.

Parameter Name	Possible Values
z_dim_values	[3,4]
z'_dim_values	[1]
$Enc_z(x)$	[[10,10,10],[64,64,64]]
$Enc_{z'}(z)$	[[10,10]]
$\psi(z, z')$	[[12,12,1],[32,32,32,1]]
BETAS	[[β_1: 1, β_2: 1, β_3: 1, β_4: 1], [β_1: 10000, β_2: 1, β_3: 1, β_4: 10000]]
learning_rate	[10^{-3}]
batch_size	[512,1024]
num_epochs	[300,500]

Table 6.3.: Again the decoder has the same size as the encoder for each experiment.

For the toy dataset, we took the best model based on the R^2 score where we again did not freeze the parameters. We saw that freezing the parameters did not result in very good models. We used $T_f = 400$ and $O_f = 200$.

6.3. Interpretation methods settings

Kernel SHAP settings The number of sampled collations was set to $n\_samples = 200$ for all datasets and all calculations of SHAP values. Theoretically increasing this number would give more accurate approximations of Shapley values but the computation time would also increase dramatically.

7. Results

This section presents the performance of the trained models and describes the findings of interpreting them. We have selected 3 models for each dataset.

- The full model: This is the feed-forward neural network (FFNN) which was trained with all available features.
- The confounder model: This is again an FFNN that was trained without the previously selected feature which we call the confounding feature which should act as a hidden confounder.
- The VAE model: This is the proposed model which we outlined in previous sections. This model does not have access to the confounding feature.

After stating the architecture of each model we will use ICE plots to analyze the learned relationship between variables and prediction. We then use Kernel SHAP to approximate shapley values and calculate the feature importance for each model of each dataset. We also define baselines that will allow us to analyze the prediction of single instances. Last but on least we will show the resulting SHAP summary plots for each model.

7.1. Architecture of Models

7.1.1. Bike rental

The optimal architecture that we found for these models can be found in 7.1 for the full model and in 7.2 for the confounder model. The architecture for the VAE model can be found in 7.3. Further in formations regarding other hyperparameters and truth-prediction plots are provided in the appendix (see Appendix A.2). The performance of these models is summarized in table 7.4.

Layer (type)	Param #
Linear (in_features=14, out_features=128)	1,920
ReLU	0
Linear (in_features=128, out_features=128)	16,512
ReLU	0
Linear (in_features=128, out_features=64)	8,256
ReLU	0
Linear (in_features=64, out_features=1)	65
	Total params: 26,753

Table 7.1.: Neural Network Structure of the full model.

Layer (type)	Param #
Linear (in_features=10, out_features=32)	352
ReLU	0
Linear (in_features=32, out_features=32)	1,056
ReLU	0
Linear (in_features=32, out_features=128)	4,224
ReLU	0
Linear-7 (in_features=128, out_features=1)	129
	Total params: 5,761

Table 7.2.: Neural Network Structure of the confounder model.

Hierarchy	Layer (type)	Description	Param #
$Enc_z(x)$	Linear	in_features=10, out_features=128	1,408
	ReLU		0
	Linear	in_features=128, out_features=128	16,512
	ReLU		0
	Linear	in_features=128, out_features=4	516
	Linear	in_features=128, out_features=4	516
$Enc_{z'}(z)$	Linear	in_features=4, out_features=5	25
	ReLU		0
	Linear	in_features=5, out_features=5	30
	ReLU		0
	Linear	in_features=5, out_features=5	30
	ReLU		0
	Linear	in_features=5, out_features=1	6
	Linear	in_features=5, out_features=1	6
$Dec(z, z')$	Linear	in_features=5, out_features=128	768
	ReLU		0
	Linear	in_features=128, out_features=128	16,512
	ReLU		0
	Linear	in_features=128, out_features=10	1,290
$\psi(z, z')$	Linear	in_features=5, out_features=28	168
	ReLU		0
	Linear	in_features=28, out_features=28	812
	ReLU		0
	Linear	in_features=28, out_features=1	29
			Total params: 38,628

Table 7.3.: Neural Network Structure of the VAE model.

Model	**Train RMSE Loss**	**Test RMSE Loss**	**R^2 Score**
Full model	581.562	729.221	0.852
Confounder model	885.610	921.290	0.764
VAE model	492.185	742.28	0.845

Table 7.4.: Train and test losses with R^2 scores where the R^2 score is calculated on the test set.

7.1.2. Admission

The optimal architecture that we found for these models can be found in 7.5 for the full model and in 7.6 for the confounder model. The architecture for the VAE model can be found in 7.7. Further information regarding other hyperparameters is provided in the appendix (see Appendix A.2). The performance of these models is summarized in table 7.8.

Layer (type)	Param #
BatchNorm1d	14
Linear (in_features=7, out_features=128)	1024
ReLU	0
Linear (in_features=128, out_features=16)	2064
ReLU	0
Linear (in_features=16, out_features=32)	544
ReLU	0
Linear (in_features=32, out_features=1)	33
	Total params: 3,679

Table 7.5.: Neural Network Structure of the full model.

Layer (type)	Param #
BatchNorm1d	12
Linear (in_features=6, out_features=128)	896
ReLU	0
Linear (in_features=128, out_features=1)	129
	Total params: 1,037

Table 7.6.: Neural Network Structure of the confounding model.

Hierarchy	Layer (type)	Description	Param #
$Enc_z(x)$	Linear	in_features=6, out_features=10	70
	ReLU		0
	Linear	in_features=10, out_features=10	110
	ReLU		0
	Linear	in_features=10, out_features=10	110
	ReLU		0
	Linear	in_features=10, out_features=3	33
	Linear	in_features=10, out_features=3	33
$Enc_{z'}(z)$	Linear	in_features=3, out_features=10	40
	ReLU		0
	Linear	in_features=10, out_features=10	110
	ReLU		0
	Linear	in_features=10, out_features=1	11
	Linear	in_features=10, out_features=1	11
$Dec(z, z')$	Linear	in_features=4, out_features=10	50
	ReLU		0
	Linear	in_features=10, out_features=10	110
	ReLU		0
	Linear	in_features=10, out_features=10	110
	ReLU		0
	Linear	in_features=10, out_features=6	66
$\psi(z, z')$	Linear	in_features=4, out_features=12	60
	ReLU		0
	Linear	in_features=12, out_features=12	156
	ReLU		0
	Linear	in_features=12, out_features=1	13
			Total params: 1,026

Table 7.7.: Neural Network Structure of the VAE model.

Model	**Train RMSE Loss**	**Test RMSE Loss**	**R^2 Score**
Full model	0.054	0.067	0.725
Confounder model	0.060	0.077	0.635
VAE model	0.062	0.072	0.681

Table 7.8.: Train and test losses with R^2 scores where the R^2 score is calculated on the test set.

7.1.3. Toy-Data

The optimal architecture that emerged for our models can be found in 7.9 for the full model and in 7.10 for the confounder model. The architecture for the VAE model can be found in 7.11. Further information regarding other hyperparameters is provided in the appendix (see Appendix A.2). The performance of these models is summarized in table 7.12.

Layer (type)	Param #
Linear (in_features=6, out_features=128)	896
ReLU	0
Linear (in_features=128, out_features=64)	8256
ReLU	0
Linear (in_features=64, out_features=32)	2080
ReLU	0
Linear (in_features=32, out_features=1)	33
	Total params: 11,265

Table 7.9.: Neural Network Structure of the full model.

Layer (type)	Param #
BatchNorm1d	10
Linear (in_features=5, out_features=64)	384
ReLU	0
Linear (in_features=64, out_features=32)	2080
ReLU	0
Linear (in_features=32, out_features=32)	1056
ReLU	0
Linear (in_features=32, out_features=1)	33
	Total params: 3,563

Table 7.10.: Neural Network Structure of the confounding model.

Hierarchy	Layer (type)	Description	Param #
$Enc_z(x)$	Linear	in_features=5, out_features=64	384
	ReLU		0
	Linear	in_features=64, out_features=64	4160
	ReLU		0
	Linear	in_features=64, out_features=64	4160
	ReLU		0
	Linear	in_features=64, out_features=3	195
	Linear	in_features=64, out_features=3	195
$Enc_{z'}(z)$	Linear	in_features=3, out_features=10	40
	ReLU		0
	Linear	in_features=10, out_features=10	110
	ReLU		0
	Linear	in_features=10, out_features=1	11
	Linear	in_features=10, out_features=1	11
$Dec(z, z')$	Linear	in_features=4, out_features=64	320
	ReLU		0
	Linear	in_features=64, out_features=64	4160
	ReLU		0
	Linear	in_features=64, out_features=64	4160
	ReLU		0
	Linear	in_features=64, out_features=5	325
$\psi(z, z')$	Linear	in_features=4, out_features=32	160
	ReLU		0
	Linear	in_features=32, out_features=32	1056
	ReLU		0
	Linear	in_features=32, out_features=32	1056
	ReLU		0
	Linear	in_features=32, out_features=1	33
			Total params: 22,355

Table 7.11.: Neural Network Structure of the VAE model.

Model	**Train RMSE Loss**	**Test RMSE Loss**	R^2 **Score**
Full model	2.027	2.040	0.998
Confounder model	6.924	6.857	0.982
VAE model	5.332	5.398	0.989

Table 7.12.: Train and test losses with R^2 scores where the R^2 score is calculated on the test set.

7.2. Model interpretation

7.2.1. Bike rental

The first interpretation we will do is on the prediction-truth plot. Our observations suggest that the full model (Figure 7.1) performs better than the confounding model (Figure 7.2), exhibiting a more linear relationship between predicted and actual values. Specifically, during the summer months, the full model displays accurate predictions, as evidenced by the close alignment of the depicted trend line with the actual data points. Conversely, the confounding model exhibits a tendency to overestimate bike rentals, particularly on hot summer days. This overestimation, visible in the top-right quadrant of Figure 7.2, may be attributable to the model's inability to account for seasonal variations. The VAE model (Figure 7.3) generally parallels the performance of the full model, especially during the summer. However, similar to the confounding model, it shows a proclivity for overestimation during the winter months, as observed in the bottom-left section of the corresponding plot.

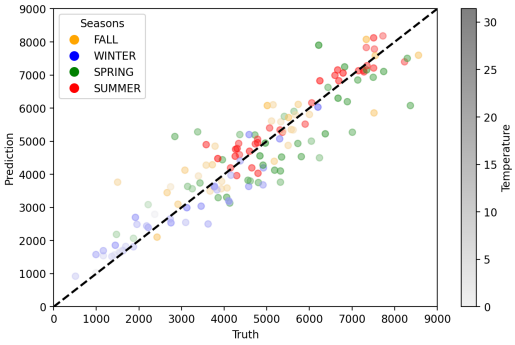


Figure 7.1.: Truth-prediction plot of the full model.

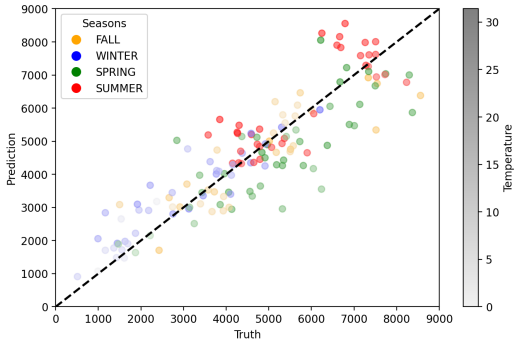


Figure 7.2.: Truth-prediction plot of the confounding model.

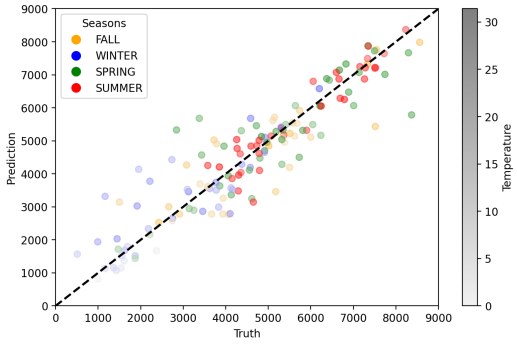


Figure 7.3.: Truth-prediction plot of the VAE model.

ICE-Plots

In the following section, we will present a series of Individual Conditional Expectation (ICE) plots for examination and discussion. Our analysis will start with an exploration of one of the most significantly correlated features in our study - the temperature. The depicted results in Figures 7.4 - 7.6 demonstrate several key insights. The full model associates summer with a negative correlation between temperature and bike rentals, while it finds a positive correlation for all other seasons. This also shows that the model is invalid for data points outside of the distribution of the training data since it does not make sense that the company would rent the most bikes on a summer day at -5 degrees. This downward trend helps the model to not overestimate very hot summer days thus producing a better performance. These observations contrast the confounding model, which establishes a positive correlation between temperature and rentals across all seasons, leading to an overestimation during hot summer days. Interestingly, the VAE model consistently shows a downward trend for each season when the temperature is high. If we operate under the assumption that high temperatures deter customers from renting bikes, then the VAE model's predictions seem more realistic. For instance, during an unusually warm spring day, one would anticipate reduced bike rentals, aligning with the VAE model's trend. We also immediately notice that the curves are not as smooth as in the other picture which is due to the stochastic nature of the VAE model.

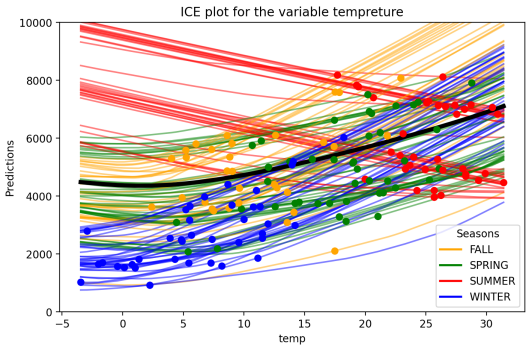


Figure 7.4.: ICE plot of the full model.

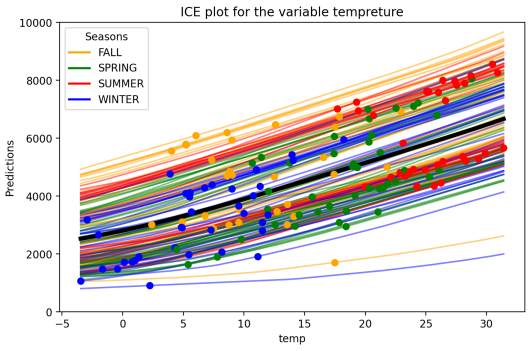


Figure 7.5.: ICE plot of the confounding model.

We see in all three plots 7.4 - 7.6 a strong incline in some parts of the average line (black) which indicates the high importance of the variable.

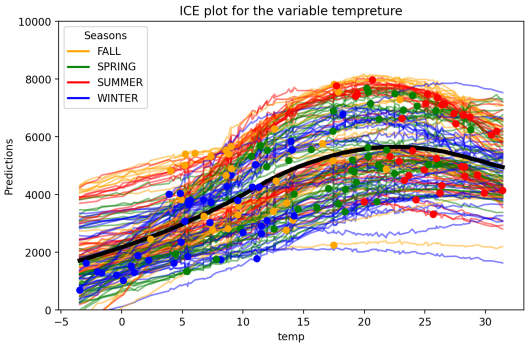


Figure 7.6.: ICE plot of the VAE model.

The next attribute we look at is the year. Since we know that we have a quite strong upward trend of rented bikes from 2011 to 2012 (5.1) we expect that to be seen in the ICE plots. In this ICE plot of the full model (Figure 7.7) we see a relatively good mapping from 2011 to 2012. Broadly speaking we map for example the winter season of 2011 to the winter season of 2012 and so on in terms of bike rentals. The average black line indicates that we increase the count when changing from 2011 to 2012 which aligns with what we know to be true in reality. The ICE plot for the confounding model (Figure 7.8) on the other hand shows a worse mapping and a very small upward trend from 2011 to 2012 compared to the full model. We observe that if we change the year from a hot summer day in 2012 to a hot summer day in 2011 we will end up way higher than what we would predict for a hot summer day in 2011. The ICE plot for the variable year of the VAE model (Figure 7.9) is similar to the confounding model but we have a lot of crossing lines there which we do not have in any other model. The black line is also not as steep as in the full model which indicates that it is a less important feature for the prediction.

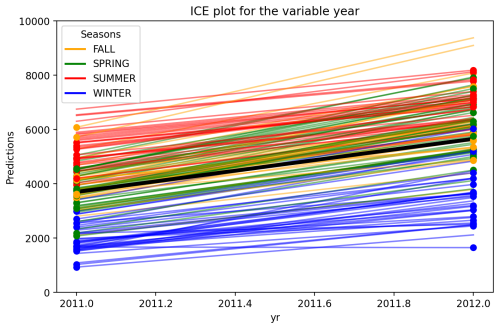


Figure 7.7.: ICE year plot of the full model.

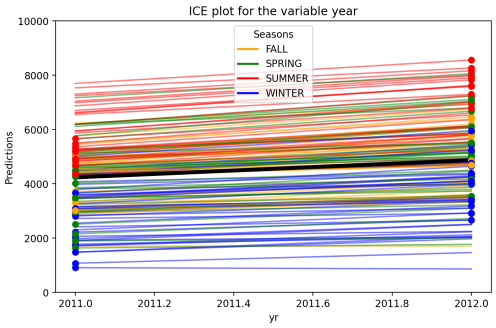


Figure 7.8.: ICE year plot of the confounding model.

The plots indicate that the full model captures the relation of the year better than the confounding model and VAE model. But this is not the full story. A section in the

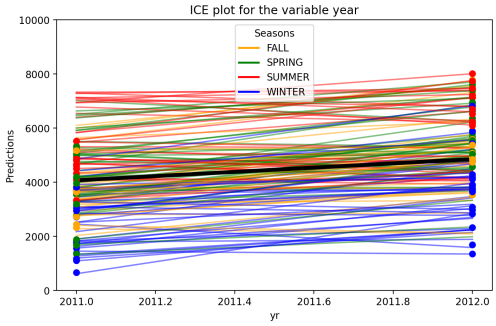


Figure 7.9.: ICE year plot of the VAE model.

appendix A.4 shows that the importance of variables can be distributed to different highly correlated features. The same thing happens here if we also look at the ICE plots of the variable days_since_2011. Since we can accurately calculate the year from this variable, all the information of the year is also available in days_since_2011. Subsequently, they have a high correlation. The ICE plots show this importance shifting perfectly. The plot for the full model (Figure 7.10) shows that the lines are mostly flat or sometimes show a small incline. The average line is nearly flat. This shows that this feature is not so important for this model when making the prediction. In contrast to the full model, we see in the plot of the confounding model (Figure 7.11) a very strong correlation between days_since_2011 and the prediction. Here we roughly map the summer of 2011 to the summer of 2012. The ICE plot of the VAE model (Figure 7.12) differs strongly from the previous two plots. Here we see lines that even went to negative predictions (Winter bottom right) and slightly wave line shape of the lines which means they go up until summer, then they do not rise that much until winter, and then go up again.

The ICE plots of other variables differ not significantly and thus are omitted but can be found in the Github repo(`https://github.com/patrickpollek1/Masterthesis-Interpreting-Neural-Networks-Under-Latent-Confounding`).

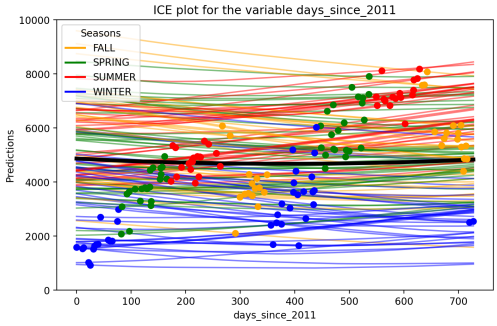


Figure 7.10.: ICE days_since_2011 plot of the full model.

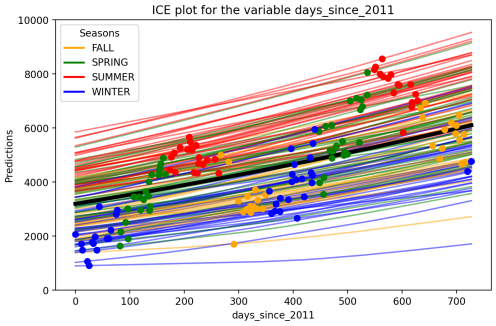


Figure 7.11.: ICE days_since_2011 plot of the confounding model.

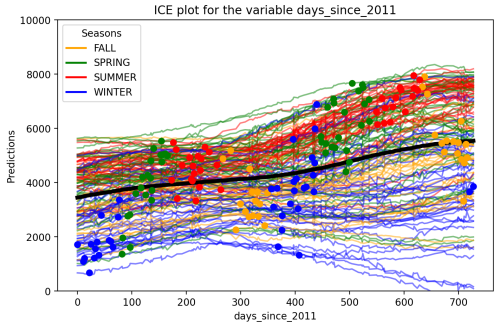


Figure 7.12.: ICE days_since_2011 plot of the VAE model.

Feature-importance / Kernel SHAP

In this section, we will discuss and present Kernel SHAP values as a tool for interpreting individual data points to gain an understanding of the impact of different features.

SHAP waterfall plots Initially, we will focus on the day with the highest and lowest number of rented bikes. By utilizing the SHAP waterfall plot, we will make a comparison against the baseline, which is the mean of all data points. Of course, the confounding models will not have access to the season variables. (Note that we only use the test set for this analysis.)

The baseline, the mean of all instances, can be found in table 7.13. From the test data set, the maximum number of bikes rented in a single day reached 8555, while the minimum dropped to 506. The features of these days are also shown in table 7.13.

Feature	Baseline	Best Day Value	Worst Day Value
yr	2011.510	2012	2011
temp	15.322	17.49	2.22
hum	60.256	54.29	86.25
windspeed	13.208	15.24	19.68
days_since_2011	361.142	637	25
season_FALL	0.210	1	0
season_SPRING	0.292	0	0
season_SUMMER	0.238	0	0
season_WINTER	0.258	0	1
holiday_HOLIDAY	0.020	0	0
workingday_WORKING DAY	0.700	0	1
weathersit_GOOD	0.639	1	0
weathersit_MISTY	0.346	0	0
weathersit_RAIN/SNOW/STORM	0.013	0	1
cnt	-	8555	506

Table 7.13.: Baseline summary

The predicted count of bikes rented on this baseline instance is summarized in table 7.14.

Model	Baseline Prediction
Full Model	1655.73
Confounder Model	2965.39
VAE Model	4787.01

Table 7.14.: Baseline Predictions of Models.

To better visualize the impact of various features on these extreme days, we turn to SHAP analysis. Figure 7.13 - 7.15 presents the SHAP waterfall plot for the day with the highest number of bike rentals for all 3 models. These graphs allow us to see how each feature contributes towards pushing the model's prediction away from the baseline. In the plot of the full model (Figure 7.13) we see that the season fall is the most important factor for this data point as it adds 1690.6 rented bikes to the total prediction. Closely behind we have the year which adds around 1621 bikes. This is expected since we saw a high trend when we looked at the ICE plots of the variable year for this model. We also observe that we have all season attributes which add a lot of bikes. The waterfall plot of the confounding model (Figure 7.14) shows that day_since_2011 is the most important attribute which is also expected since we saw earlier that the confounding model puts more importance on this variable instead of the year. In the plot of the VAE model (Figure 7.15) we again see that the variable day_since_2011 is much more important and adds 4 times the bike to the prediction than the year. We can observe that one of the most important factors in all of these plots is time (either year or day day_since_2011).

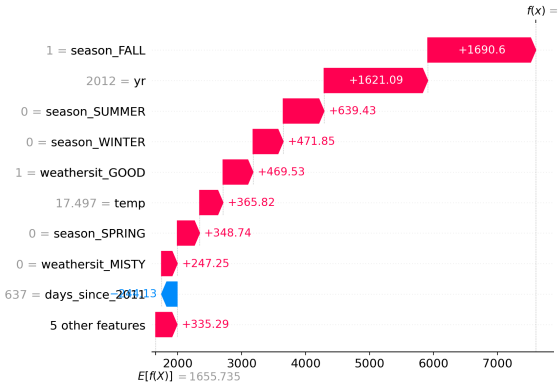


Figure 7.13.: SHAP Waterfall Plot of the full model.

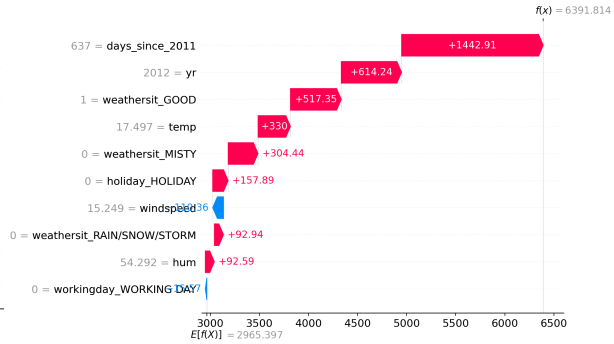


Figure 7.14.: SHAP Waterfall Plot of the confounding model

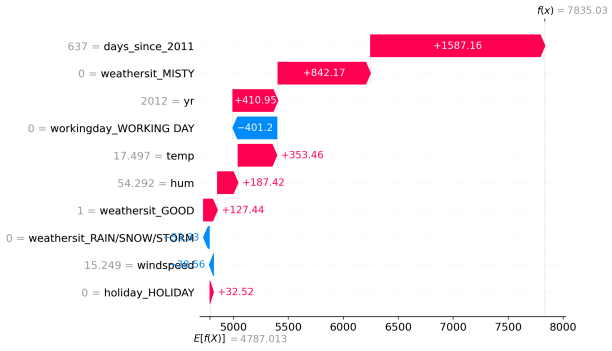


Figure 7.15.: SHAP Waterfall Plot of the VAE model

The next three figures 7.16 - 7.18 show the SHAP waterfall plot for the day with minimum Bike rentals, one for each model. We see that in all plots the weather situation and the temperature play a big role as well as the year respectively day_since_2011. In the plot of the full model (Figure 7.16) we see a reduction of bikes because of the year and the bad weather situation. Also, the near-freezing temperature further lowers the prediction. In Figure 7.17 which shows the plot of the confounding model we see that temperature has a much bigger impact compared to the full model. We also notice that the baseline is higher which in turn needs a bigger reduction to arrive at a count which is close to the true value. In the plot of the VAE model (Figure 7.18) we observe that the weather situation rain/storm has the most impact. If we add the year and day_since_2011 we would arrive at a similar decrease as in the confounding model. Because of the higher baseline, this reduction is not enough to arrive at a good prediction. The prediction of the VAE model for this instance is very bad at 1771.92 (compared to 506 for the true value).

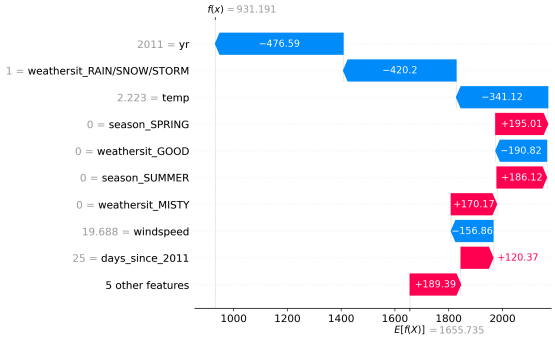


Figure 7.16.: SHAP Waterfall Plot of the full model.

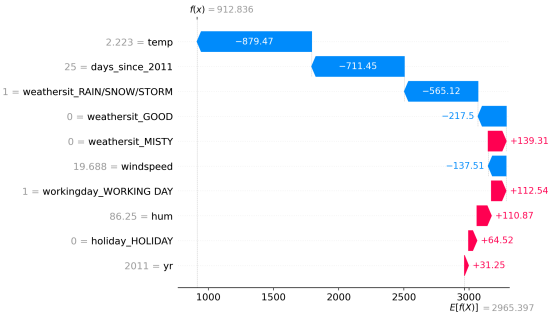


Figure 7.17.: SHAP Waterfall Plot of the confounding model.

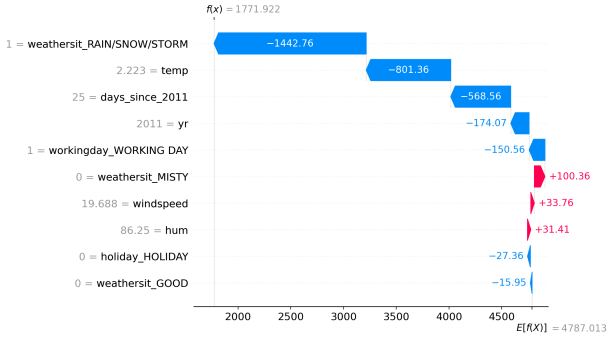


Figure 7.18.: SHAP Waterfall Plot of the VAE model.

Reflecting on the baseline data, it's evident that temperature had a more significant impact on the worst day with lower rentals than on the highest rental day. This is because the baseline temperature (15.3°C) is much closer to the temperature on the highest rental day (17.5°C). In contrast, the temperature on the lowest rental day is substantially lower, at just 2.2°C.

Feature importance All these previous KernelSHAP values were tied to a specific instance. We either selected the best day or the worst day and looked at how each variable affected the outcome. If we sum up all these values for all instances and average them, we will have a global feature importance measure as already described in the background chapter (Chapter 3). As a baseline, we again take the average of all instances.

This results in the following feature importance plot (Figure 7.19). For the full model, the variable year is the most important feature. On average it adds or takes about 1000 bikes based on this feature. We also observe that all season attributes have a high importance assigned. In contrast to the full model, the confounder and the VAE model have the temperature as the most important feature. When we also take the correlation structure of the variables into consideration it makes sense that the temperature is now more important since it is highly correlated with the season attributes. In the confounding model, the temperature holds an average absolute importance exceeding 1000, while the VAE model's importance stands at roughly 700. Again we see that the confounding as well as the VAE model weigh the day_since_2011 variable more.

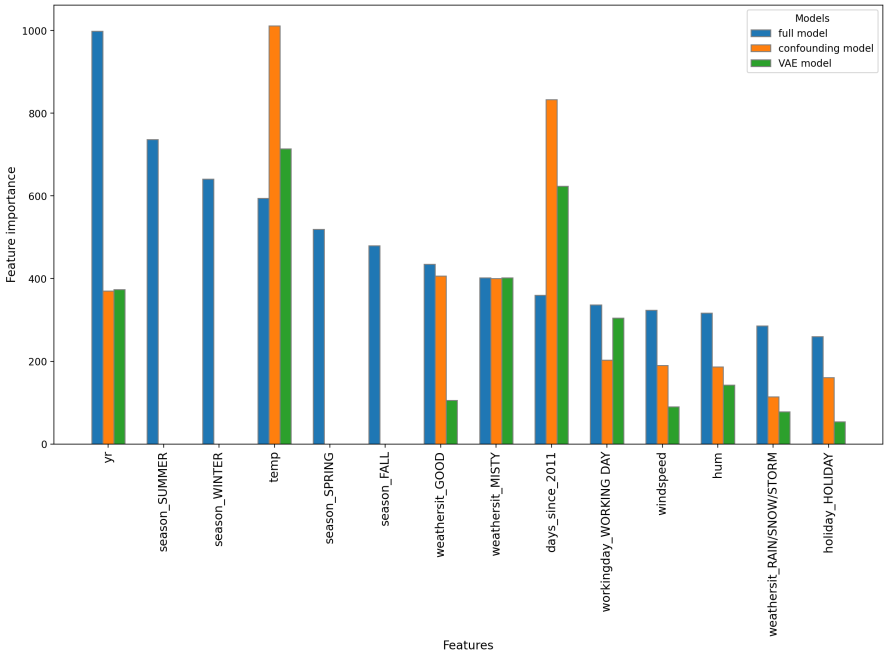


Figure 7.19.: Feature importance of all models.

SHAP summary plots Since we have calculated each SHAP value for every instance we can also use another visualization of these values. For each feature, we plot the SHAP value and color code the dot based on the value of the feature. This results in the SHAP summary plot which gives a global overview of how the SHAP values are distributed but also allows the examination of individual instances. The SHAP summary plot of the full model can be found in Figure 7.20. Since the year is the most important feature of this model, it is placed at the top. We see that a high value for the year (2012) leads to a higher positive SHAP value. In contrast, a low year (2011) leads to a (mostly) negative SHAP value. The distribution of the SHAP values for the variable temperature reveals that we also have instances where the feature value is high (high temperature) but the SHAP value is not. These could be the instances where the full model learned to not overshoot because it is too hot.

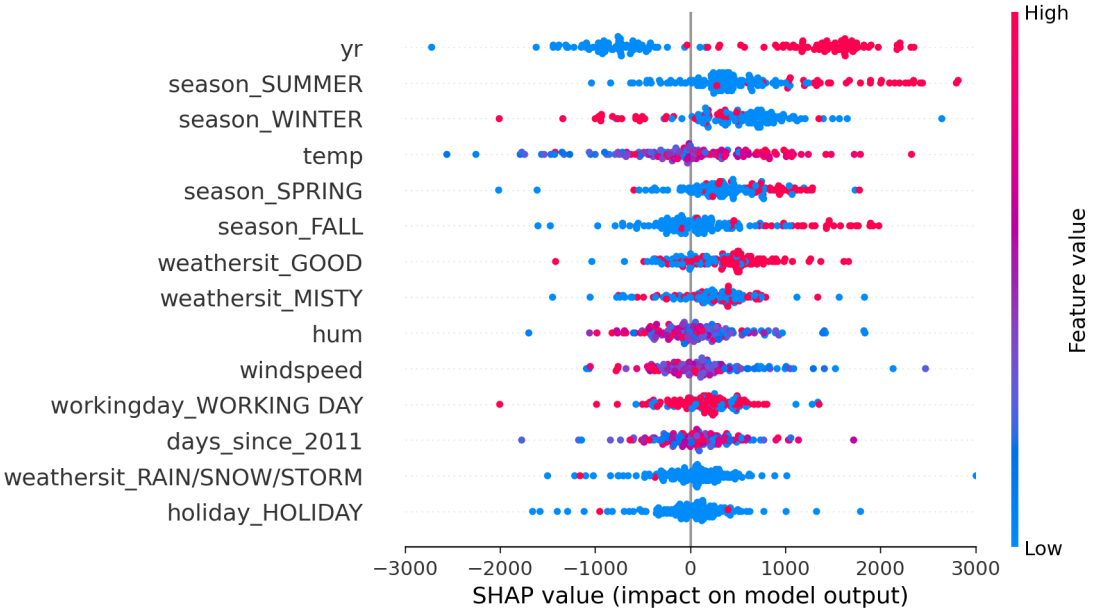


Figure 7.20.: SHAP summary plot of the full model.

In contrast to the full model we see that the temperature is monotonically ordered based on feature value and SHAP value for the confounding model (Figure 7.21). This means the higher the temperature, the higher the SHAP value. We see that a higher value in day_since_2011 leads to higher shap values. This relationship is more granular and we see that it is not clustered as in the variable year for example where we see two clusters. Looking at the variable humidity we see that high humidity will lead to fewer bikes rented and low humidity to the contrary. The same is true for wind speed. The shap summary plot of the VAE model (Figure 7.22) shows different results for the temperature compared to the confounding model. We see that we also have high feature values which result in a decrease in the prediction which is closer to the full model. We also have the variable day_since_2011 with a similar structure as the confounding model. The misty weather situation has a significantly different structure than the confounding model. Here we have a reduction in bikes when the weather is misty and the addition of bikes when the weather is not misty.

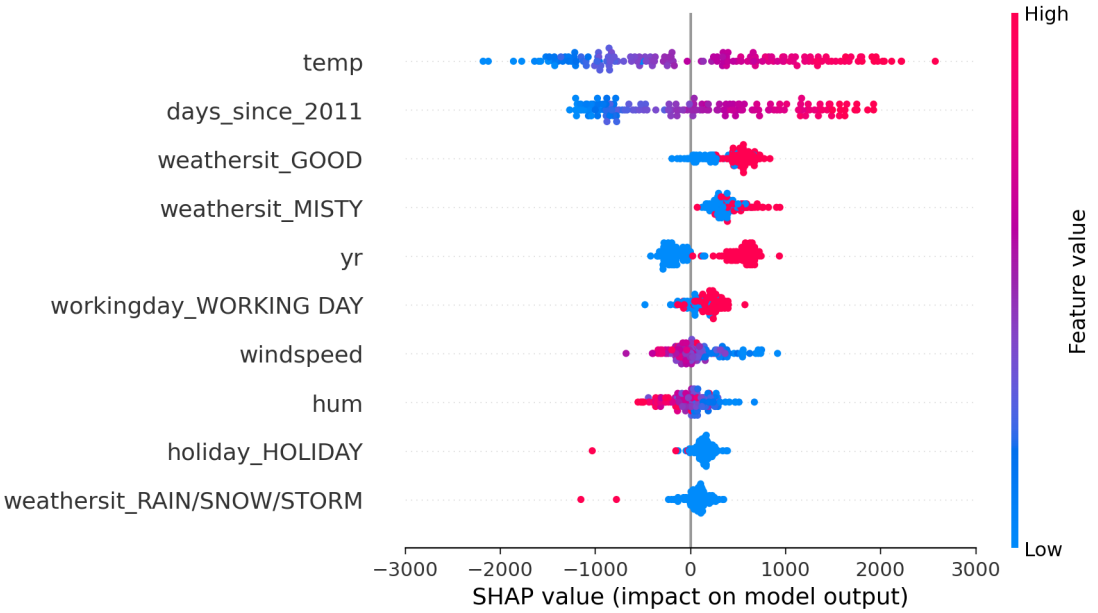


Figure 7.21.: SHAP summary plot of the confounding model.

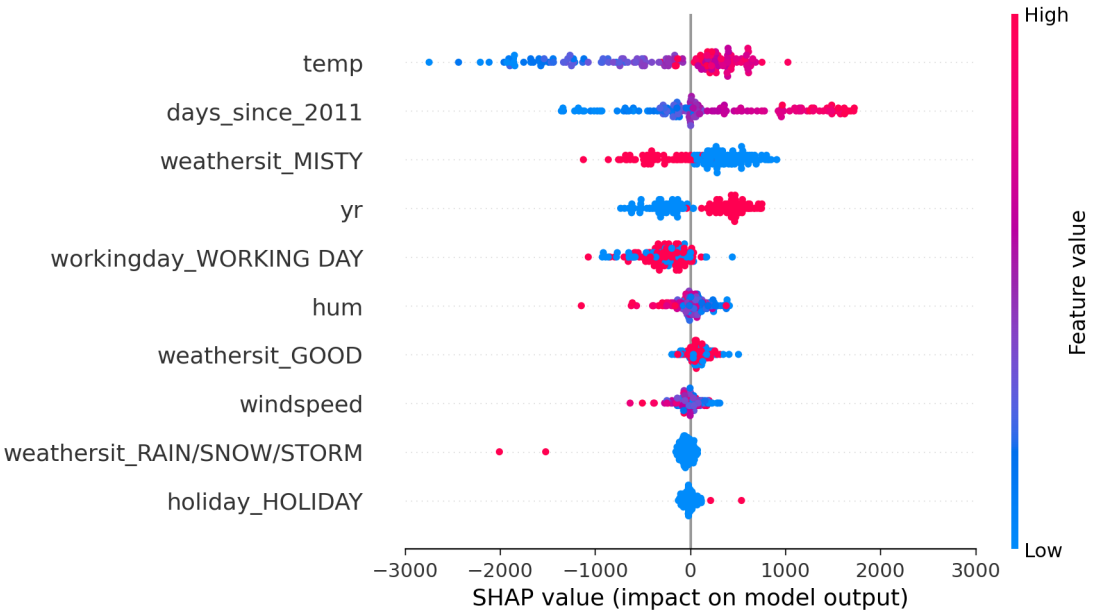


Figure 7.22.: SHAP summary plot of the VAE model.

Conclusion

In our analysis, we have identified varying levels of importance for different features in the models. Specifically, the full model heavily relies on the year feature, while the confounding model and the VAE model utilize the days_since_2011 feature. This observation is consistent across all plots and can be present due to multicollinearity, as these features are highly correlated. In fact, one feature can be precisely calculated using the other, as demonstrated by the equation:

$$year = 2011 + \mathbb{1}_{days\_since\_2011>356}$$

where $\mathbb{1}$ represents the indicator function.

Furthermore, the inclusion of the season feature versus its exclusion revealed notable distinctions in the ICE plots for temperature and the SHAP waterfall plots, particularly regarding the summer season for the two FFNNs. We saw that the full model differentiated between different seasons and captured a downward trend for extremely hot summer days whereas the confounding model exhibited an ever-increasing trend across all seasons, resulting in a structural overestimation of hot summer days. On the other hand, the VAE model learned to decrease the count across all seasons if the temperature is very high. The full model gives the impression that the decrease must happen because of the summer season, which is under the hypothesis that the decrease happens because it is too hot to bike, wrong. The VAE model successfully captures the more realistic relationship between the temperature and the count, independent of the season. In the SHAP plot, we also observe a more consistent distribution of SHAP values which again indicates a more clearer and isolated relationship learned between the variables.

With respect to the interpretability methods, it is important to also have the correlation structure in mind when looking at the feature importance. A grouping of several highly correlated variables such as days_since_2011 and year could potentially make the interpretation clearer.

7.2.2. Admission

Again we will first look at the prediction-truth plot where we colored the dots based on the CGPA value. In Figures 7.23 - 7.25 we see the plots of all models and observe that they all look similar. Looking at the values in the range of 0.7 to 0.9 we observe a slightly bigger discrepancy in the confounding model than in the full model. (Figures 7.24 and 7.23). It is also obvious that a higher CGPA score leads to a better chance of admission. This is captured by all models. Remember that the confounding model and the VAE model (Figure 7.25) have no access to the feature CGPA. What can be seen on all plots is that all models have better accuracy on students with high CGPA scores and high admission values than for students with low CGPA scores and admission values (More deviation from the black dotted line.).

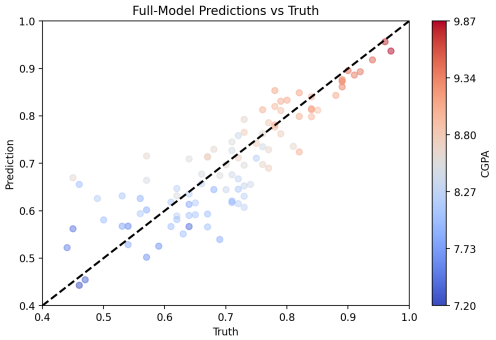


Figure 7.23.: Truth-prediction plot of the full model.

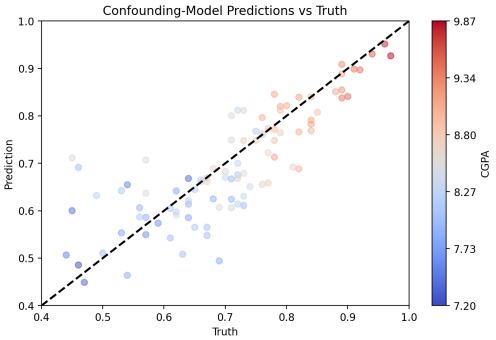


Figure 7.24.: Truth-prediction plot of the confounder model.

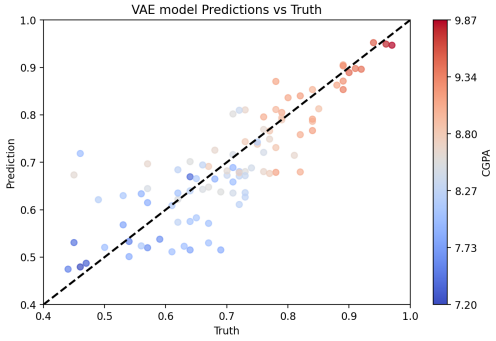


Figure 7.25.: Truth-prediction plot of the VAE model.

ICE-Plots

Figures 7.26 - 7.28 show three plots that present Individual Conditional Expectation plots for the variable GRE Score. The full model only has a very slight positive trend on average which translates to the interpretation that improving the GRE score does not significantly increase the probability of admission. On the other hand, the confounding model we see especially for students with a low CGPA score a massive increase in the probability if we would increase the GRE score. This could be due to the fact the full model relies on the CGPA score to predict the admission probability and not so much on the GRE score. In the ICE plot of the VAE model, we see an increase in the admission probability of low CGPA students up to some plateau. For high CGPA students, we see a linear increase.

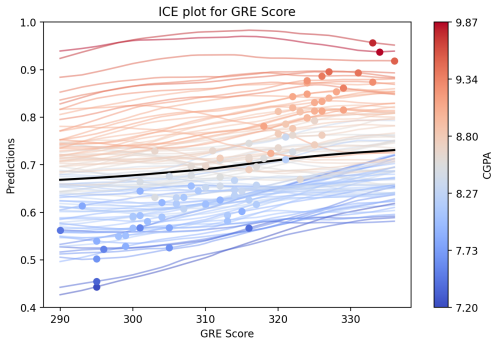


Figure 7.26.: ICE GRE plot of the full model.

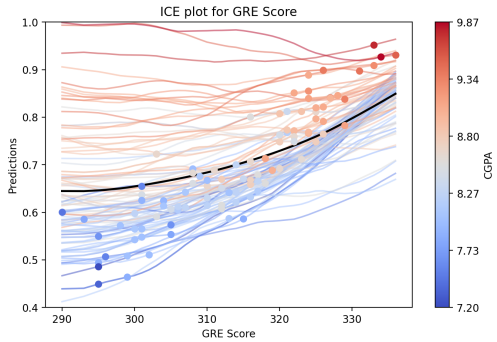


Figure 7.27.: ICE GRE plot of the confounder model.

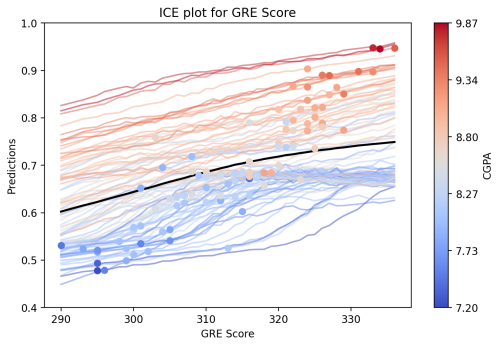


Figure 7.28.: ICE GRE plot of the VAE model.

In the ICE plots for the TOEFL score, we see a similar trend. The full model (Figure 7.29) has a positive trend which can be observed in all students. The confounding model (Figure 7.30) has a slightly stronger relation than the full model. For some high CGPA students, we see a steep decline in admission probability in the TOEFL score range of 103-108. Interestingly the ICE plot for the VAE model (Figure 7.31) shows similar behavior to the ICE plot of the variable GRE of the VAE model (7.28). We see a positive trend but for most low CGPA students it seems that there is again a plateau. In contrast, high CGPA students don't show such a plateau.

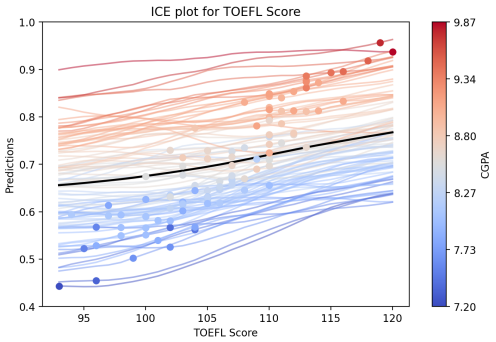


Figure 7.29.: ICE TOEFL plot of the full model.

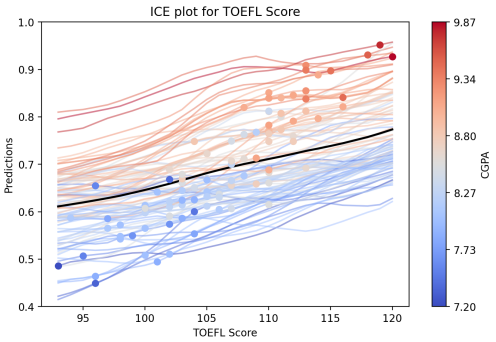


Figure 7.30.: ICE TOEFL plot of the confounder model.

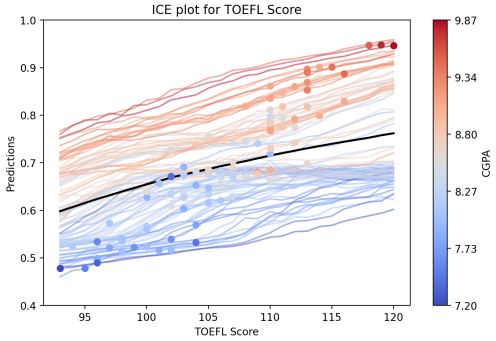


Figure 7.31.: ICE TOEFL plot of the VAE model.

The plots for university ranking are almost constant for the full model (Figure 7.32) and the confounding model (Figure 7.33), while the VAE model indicates a positive trend as seen in Figure 7.34. From the ICE plot, we also see that a high CGPA score (dark red) only exists in our training data with a university rating of 4 to 5.

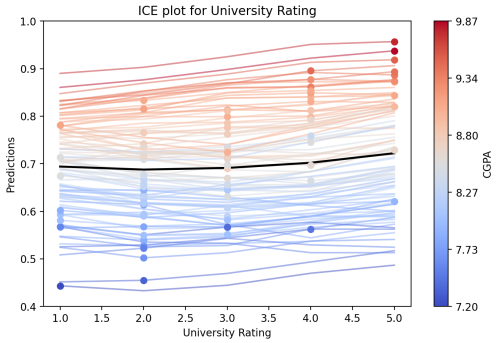


Figure 7.32.: ICE University plot of the full model.

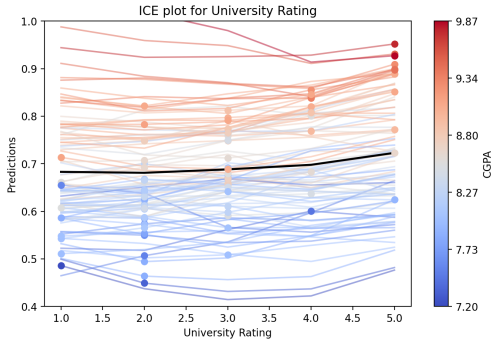


Figure 7.33.: ICE University plot of the confounder model.

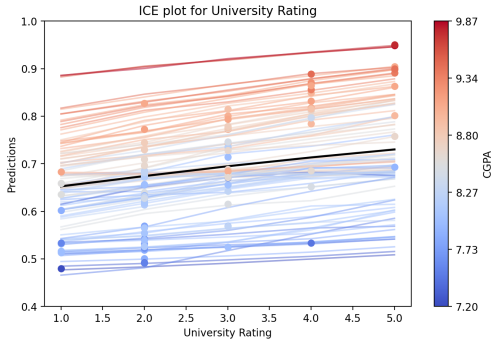


Figure 7.34.: ICE University plot of the VAE model.

For the variable Letter Of Recommendation (LOR), our plots once more exhibit a positive trend, most pronounced within the confounding model (Figure 7.36). Upon examining the ICE plot for this model, it becomes evident that students with lower CGPA scores benefit more significantly from a strong Letter of Recommendation than those with higher CGPA scores. This strong trend is exclusive to the confounding model. Meanwhile, the characteristics of both the full (Figure 7.35) and VAE model (Figure 7.37) appear largely comparable.

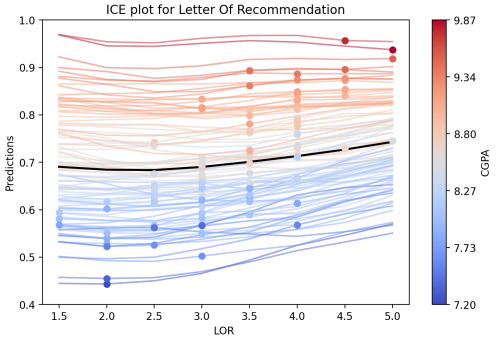


Figure 7.35.: ICE LOR plot of the full model.

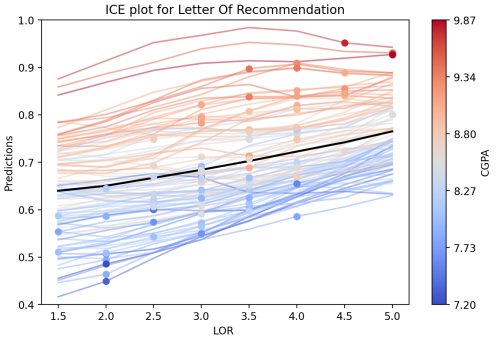


Figure 7.36.: ICE LOR plot of the confounder model.

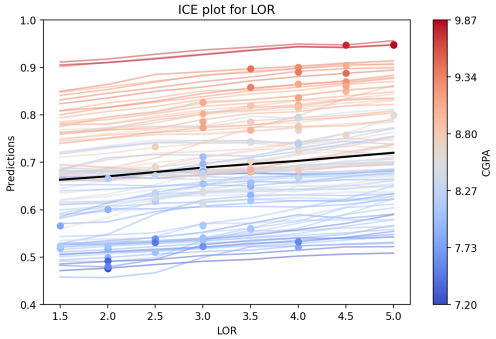


Figure 7.37.: ICE LOR plot of the VAE model.

The ICE plots for the strength of the Statement Of Purpose (SOP) (Figures 7.38 - 7.40) reveal intriguing differences between the confounding model and the remaining two models. In the full model (Figure 7.38), there is no significant increase in the chance of admission as the SOP improves. This trend is consistent for almost every individual line in the plot. However, the confounding model (Figure 7.39) displays a distinct pattern: for some high CGPA students, there is a sharp decline in the chance of admission when the SOP is not good. Conversely, low CGPA students do not experience an increase in the chance of admission when the SOP improves. Actually, the majority of students with low CGPA see a decrease in admission probability if they have a strong SOP. These contrasting observations could be attributed to the fact that the evaluated data points are out of distribution since we can see that high CGPA students do not have SOP scores of 3 and below and low CGPA students do not have SOP scores of 3 and above. The VAE model (Figure 7.40) has a positive trend on average and for all individual instances.

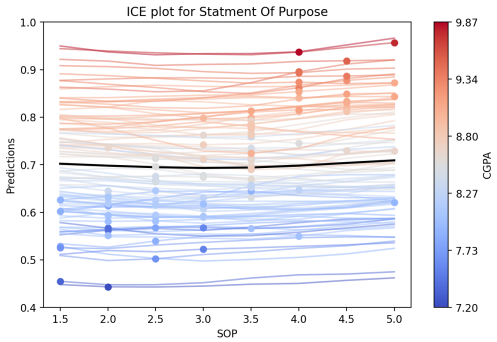


Figure 7.38.: ICE SOP plot of the full model.

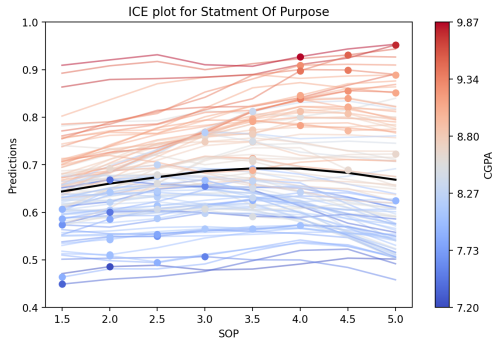


Figure 7.39.: ICE SOP plot of the confounder model.

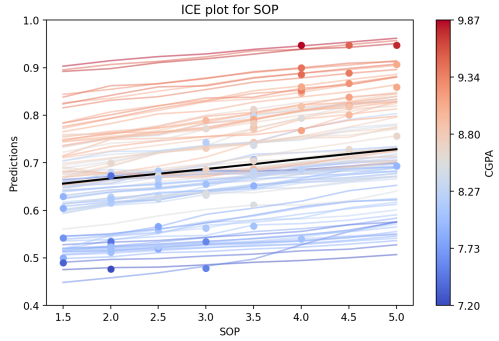


Figure 7.40.: ICE SOP plot of the VAE model.

The final ICE plots for this dataset were generated by perturbing the feature research experience. As this feature is binary, we assessed the model exclusively with values of either 0 or 1 for that specific feature. Across all models, as shown in Figures 7.41 - 7.43, there is a negligible average increase, suggesting that having research experience marginally boosts a student's chances of admission. Given the minuscule rise and nearly flat average line, we infer that this feature does not significantly influence the outcomes of any model.

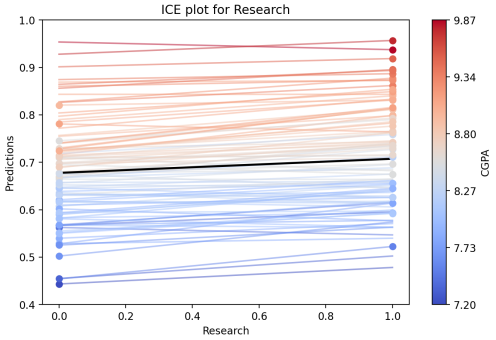


Figure 7.41.: ICE Research plot of the full model.

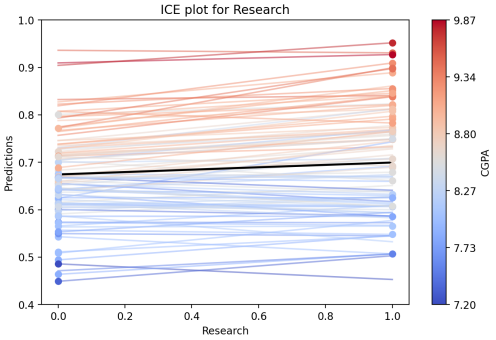


Figure 7.42.: ICE Research plot of the confounder model.

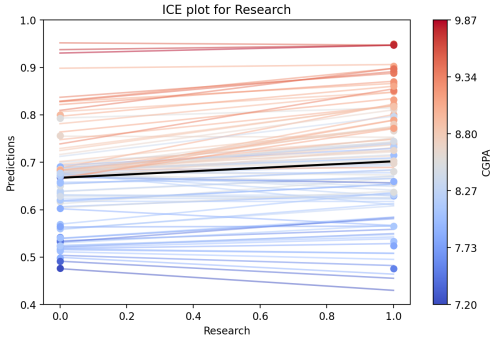


Figure 7.43.: ICE Research plot of the VAE model.

Feature-importance / Kernel SHAP

In this section, we will again discuss and present Kernel SHAP values as a tool for interpreting individual data points to gain an understanding of the impact of different features.

This time we will look at the students with the highest and lowest admission probability and try to understand which feature contributed to the prediction. By utilizing the SHAP waterfall plot, we will make a comparison against the baseline, which is the mean of all (test) data points. Of course, the confounding models will not have access to the CGPA variable. (Note that we only use the test set for this analysis.)

The baseline, the mean of all instances, can be found in table 7.15 as well as the features of the students with the highest and lowest admission probability.

Feature	**Baseline**	**Highest admission prob.**	**Lowest admission prob.**
GRE Score	313.910	334.00	298.00
TOEFL Score	106.260	120.00	98.00
University Rating	2.840	5.00	2.00
SOP	3.210	4.00	4.00
LOR	3.280	5.00	3.00
CGPA	8.481	9.87	8.03
Research	0.480	1.00	0.00
Admission probability	-	0.97	0.34

Table 7.15.: Baseline summary

The predicted admission probability of each model for the baseline can be found in table 7.16.

Model	Baseline Prediction
Full Model	0.653
Confounder Model	0.696
VAE Model	0.705

Table 7.16.: Baseline Predictions of Models

SHAP waterfall plots The first SHAP waterfall plot will look at how the prediction of each model comes together for the student with the highest admission probability in the test set. We can see that in the full model (Figure 7.44), the CGPA score is the most important factor for this prediction. Even though the ICE plot for the University rating 7.32 shows a relatively horizontal line, which would indicate no strong feature interaction with the prediction, we see a relatively big amount of probability added because of it. The confounding model (Figure 7.45) shows that the variable TOEFL score and the strength of the LOR added the most to this prediction. The VAE model (Figure 7.46) also indicates that the TOEFL score plays a huge part in the prediction. We also see a reduction in admission probability even though this particular student has research experience which is rather counterintuitive.

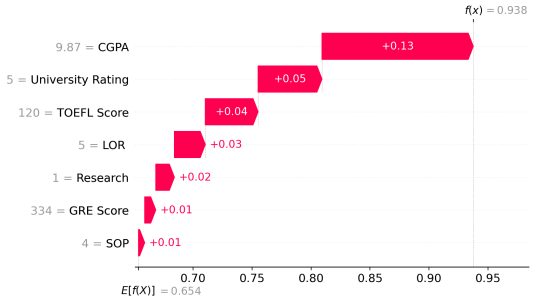


Figure 7.44.: SHAP Waterfall Plot of the full model.

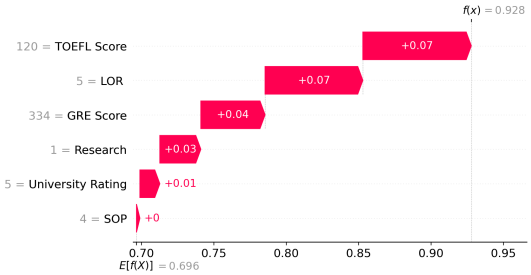


Figure 7.45.: SHAP Waterfall Plot of the confounder model.

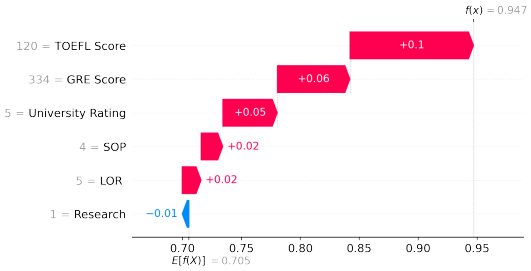


Figure 7.46.: SHAP Waterfall Plot of the VAE model.

We also look at how the prediction of the student with the lowest admission probability is composed. We see that especially the three features which represent tests and grades are mainly responsible for this prediction of the full model (Figure 7.47). This waterfall plot of the confounding model (Figure 7.48) is very similar to the full model with the twist that the importance from the variable CGPA got shifted to the GRE score. The plot for the VAE model (Figure 7.49) is similar to the confounding model. The ordering of the 4 most important variables is the same. One thing to point out is that we see that for an above-average SOP the probability gets boosted by a bit which is not present in the confounder model for example but can also be observed in the full model.

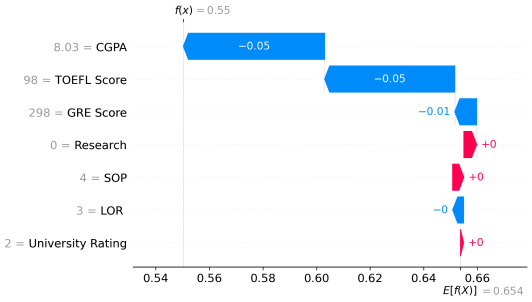


Figure 7.47.: SHAP Waterfall Plot of the full model.

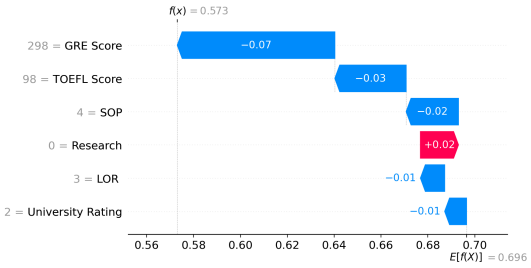


Figure 7.48.: SHAP Waterfall Plot of the confounder model.

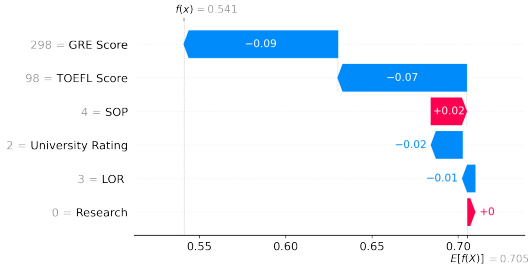


Figure 7.49.: SHAP Waterfall Plot of the VAE model.

Feature importance We again calculate the SHAP values for each instance and then average these values to arrive at a global feature importance measure. The plot of all feature importances can be seen in figure 7.50. The most important feature of the full model is the CGPA score. Interestingly the feature GRE Score is the penultimate variable in the ordering of importance. This is very different from other models. For example, the confounding model shows that the GRE score is the most important feature. This makes sense since the variables CGPA and GRE score are highly correlated, and thus can be seen and interpreted as some kind of proxy for the CGPA score. The same argument holds for the feature TOEFL score. The most important feature of the VAE model is also the GRE score. The magnitude of importance is also comparable to the magnitude in the confounding model. One difference between the VAE model and the confounding model is that research as well as the strength of the letter of recommendation plays a bigger role in the confounding model.

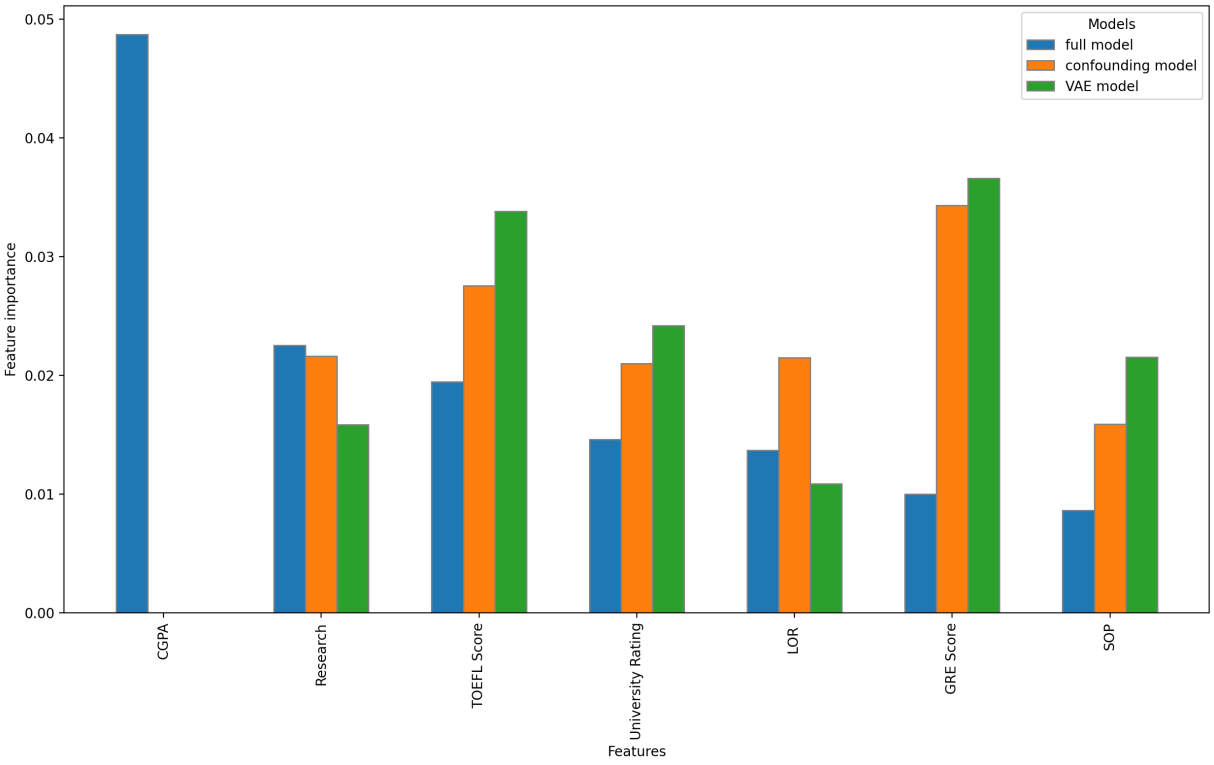


Figure 7.50.: Feature importance plot of all models.

SHAP summary plots Again if we plot the shapley values for each instance in a summary plot we also get to see a more detailed view of how the importance is distributed with respect to the feature value.

In the plot for the full model (Figure 7.51) we see that the shapley value and the feature value have a positive linear relationship for the first variable (CGPA). Second, we also see that having research experience adds more to the prediction than not having research experience. The last four variables have all clusters around 0 and do not show a meaningful structure. Especially the variable GRE score is very centered around 0. Contrary to to this the summary plot for the confounding model (Figure 7.52) shows that the GRE score now has a similar structure to the CGPA score. We also observe that everything is more spread out and we don't have any big clusters as in the full model.

The first two variables of the summary plot for the VAE model (Figure 7.53) are very similar to the confounding model. But the big difference can be seen when we look at the features University rating and SOP. We can see a clear and consistent clustering of based on the values of the features. In some sense, this could be interpreted as a specific factor for each feature value is added or removed from the prediction. Since this is a quite consistent meaning, for example, all students with university ratings of 3 will get 0.01 admission probability added and there is not one which gets significantly more or less, we could argue that this means that the model has a more independent view of this variable and its impact which is good. One thing to note is that we see in the plot that the feature Research indicates that having research experience will mostly lead to a decrease in probability which is counterintuitive.

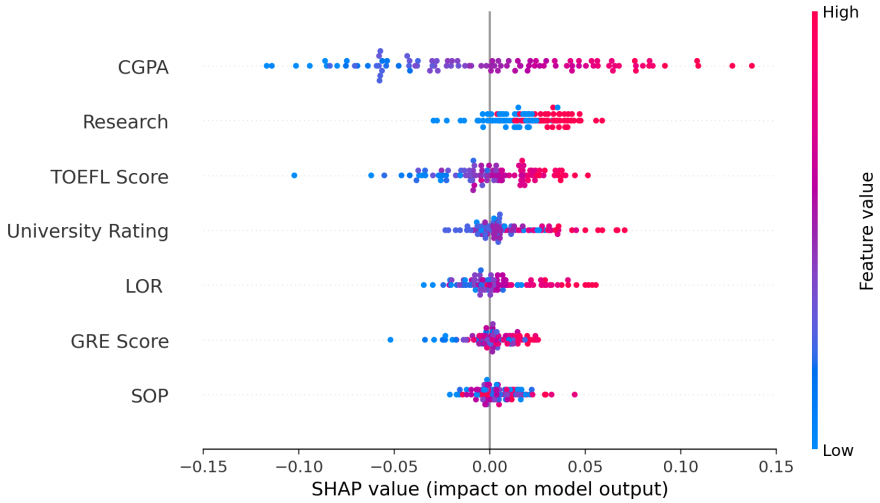


Figure 7.51.: SHAP summary plot of the full model.

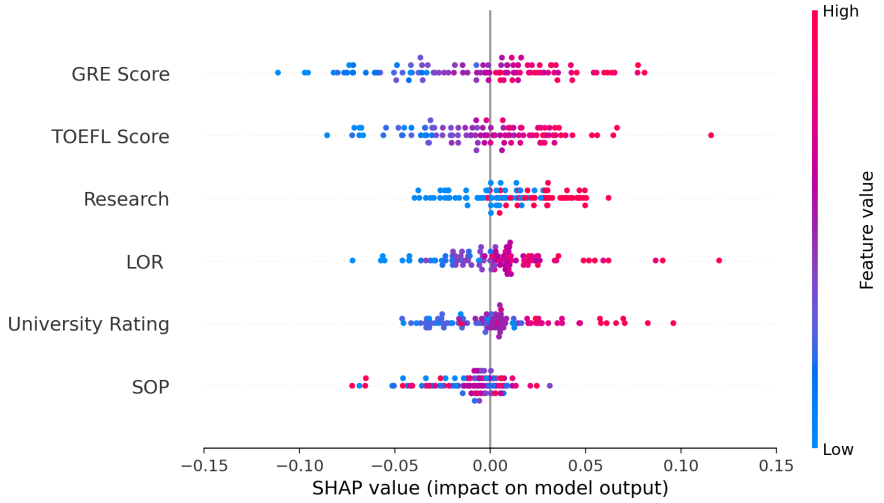


Figure 7.52.: SHAP summary plot of the confounding model.

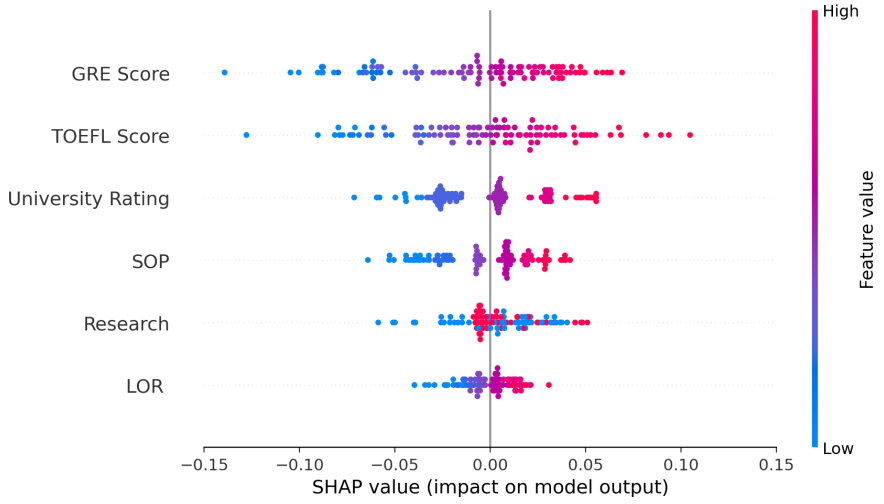


Figure 7.53.: SHAP summary plot of the VAE model.

Conclusion

In summary, the absence of information from the feature CGPA score is counterbalanced by the increased significance of the GRE variable. This observation is logical, given the strong correlation between these variables. This effect is evident in both the confounding model and the VAE model. In examining the GRE ICE plot for the confounding model (Figure 7.27), it became evident that students with lower CGPAs experienced a significant surge in admission probability with an increase in their GRE scores. Conversely, in the VAE model (Figure 7.28), a rise in the GRE score also increased the admission probability, but this enhancement plateaued and remained below the admission probabilities observed for students with the highest CGPA/GRE combinations. This pattern was mirrored in the ICE plots for the TOEFL score (Figure 7.31). This suggests that improving a single test score, whether it's GRE or TOEFL, can boost admission chances. However, a single impressive score is insufficient to guarantee a high probability of admission. Optimal outcomes likely require strong performances across multiple test scores. We believe this interpretation aligns more closely with real-world scenarios than the relationships we saw that the confounding model learned. With this hypothesis in mind, we then explored a 3-dimensional ICE plot, wherein we simultaneously adjusted the GRE and TOEFL scores, keeping all other features constant for a single instance. The resulting visualization can be referenced in Figure 7.54. The plot distinctly illustrates that increasing a single feature would reproduce the aforementioned increase-to-plateau relationship. Yet, by elevating both the GRE and TOEFL scores, it's possible to surpass this plateau and attain exceptionally high admission probabilities. In the SHAP summary plot, the Shapley values for certain features, notably SOP and University rating, within the VAE models displayed consistent clustering (Figure 7.53). This suggests that these features have a learned relationship that is more independent. This observation resonates with our hypothesis that specific attributes, such as a superior SOP, should enhance admission prospects irrespective of other features.

3D ICE plot for the variables GRE score and TOEFL score of the VAE model

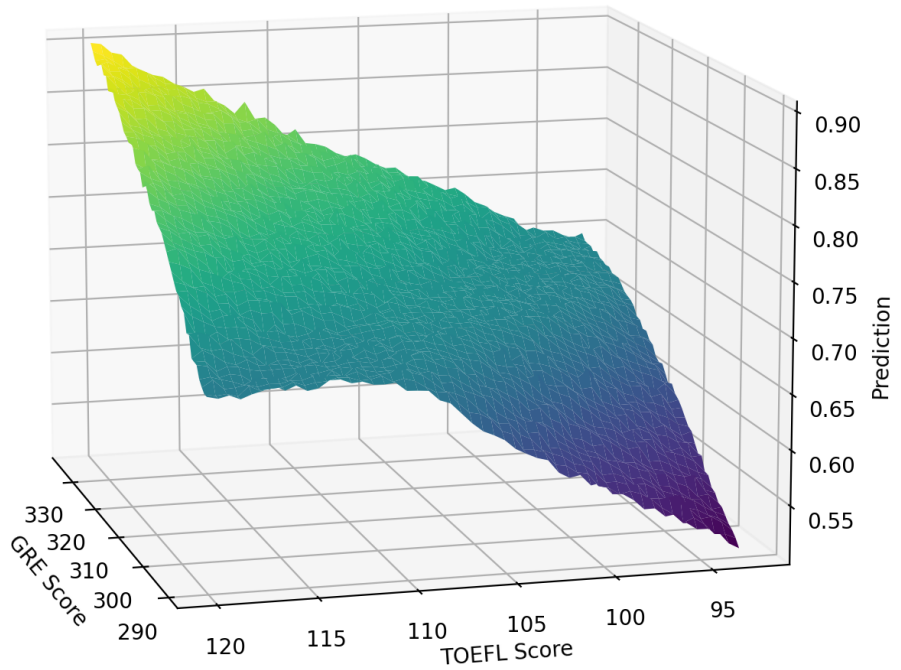


Figure 7.54.: This figure shows the 3d ICE plot for a single student (low CGPA) where we permutated both TOEFL score and GRE score and predicted the admission prob. with the VAE model.

7.2.3. Toy-Data

This time we use variable $x1$ as a confounder, hence the truth-prediction plot is color-coded accordingly. It's important to note that for the creation of these visualizations, we have utilized a subset of 300 randomly selected samples from the unseen test set. This strategy ensures clarity and avoids the over-saturation of data points that could potentially obscure the visualization, thus providing a more interpretable and meaningful graphical representation. In Figures 7.55 - 7.57 we see all three truth-prediction plots and observe that both the confounding model and the VAE model show slightly worse results than the full model over the full range of values of $x1$. Given that our analysis is restricted to a sample of 300 data points, it is advisable to consult Figures A.17 and A.18 located in the appendix. These figures present the truth-prediction plots encompassing all test data. Upon examination, it becomes evident that the predictions deviate significantly, particularly when the predicted values are low. This is more pronounced in the confounding model than in the VAE model.

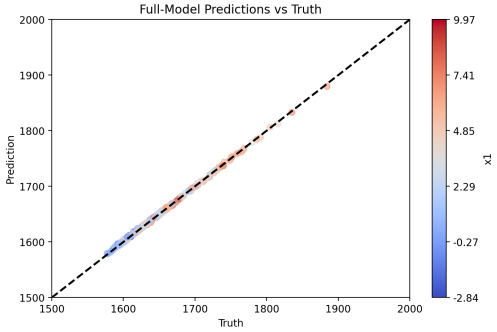


Figure 7.55.: Truth-prediction plot of the full model.

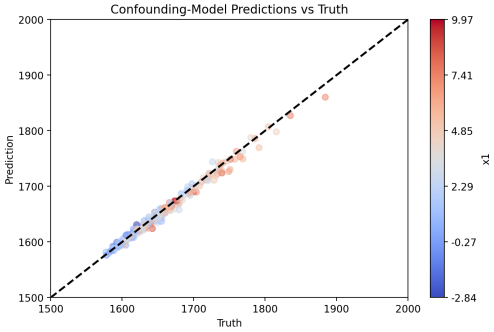


Figure 7.56.: Truth-prediction plot of the confounder model.

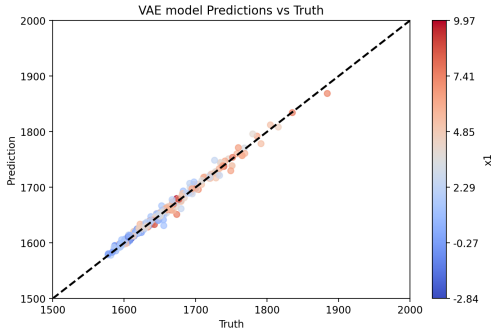


Figure 7.57.: Truth-prediction plot of the VAE model.

ICE-Plots

For our toy dataset, we are in a fortunate position where we know the underlying truth. This allows us to create ICE plots that directly relate to this truth. We start by looking at the ICE plots for feature $x0$. Given that this feature follows a Gaussian distribution with a mean of 1 and a variance of 1, and is merely added to the dependent variable y, we anticipate observing straight lines characterized by a modest positive slope. This expectation is confirmed by the ground truth model, as depicted in Figure 7.61. In the case of the full model, we notice that the ICE plots (Figure 7.58) exhibit primarily linear trends around the original values. However, where the model is evaluated at points outside the training distribution, we observe lines that either rise or fall. This serves as a clear reminder that when our model is extrapolating beyond the distribution of the training set, its behavior can become unpredictable and arbitrary. The confounding model, as seen in Figure 7.59, shows an upward trend for some instances which is also captured by the average line. This is probably related to the fact that $x0$ is adjusting for $x1$ not available since this pair has the highest correlation (Remember, $x0$ is derived from $x1$.). We see a similar behavior for the VAE model as seen in Figure 7.60. The blue lines are rather straight with some slight positive slope but some of the red lines show a strong positive trend. This might again be due to the fact that the VAE model has no access to $x1$.

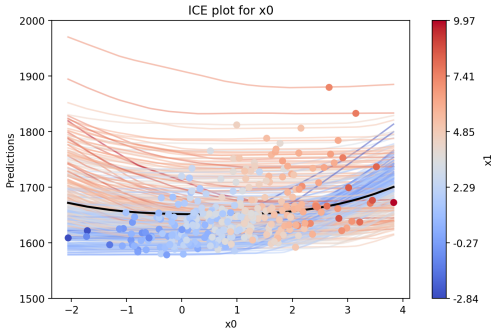


Figure 7.58.: ICE $x0$ plot of the full model.

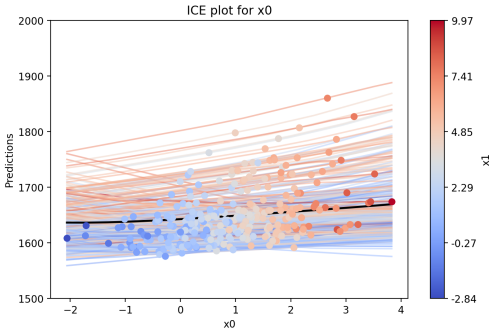


Figure 7.59.: ICE $x0$ plot of the confounder model.

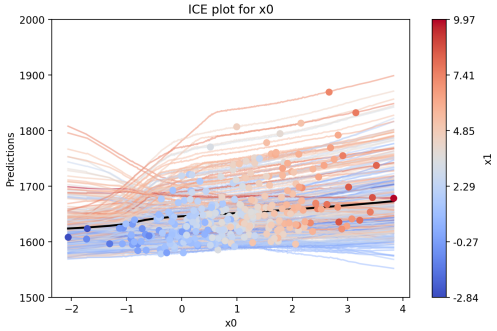


Figure 7.60.: ICE $x0$ plot of the VAE model.

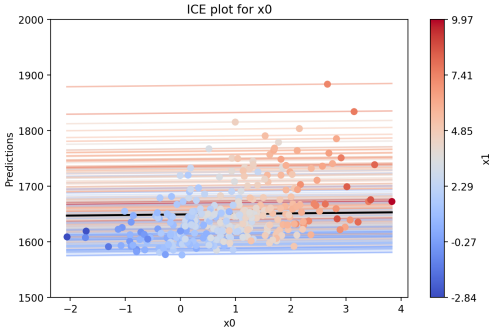


Figure 7.61.: ICE $x0$ plot of the true model.

The next variable under consideration is $x2$, which interacts with $x1$ to influence y. This relationship is represented in the equation as a product of $x1$ and $x2$, magnified by a factor of 10, that is, $10X_1X_2$. The full model (Figure 7.62) is nearly identical to the truth ICE plot in Figure 7.65. The confounding model (Figure 7.63) also presents some variations however, it is consistent with the primary trend. Specifically, we observe that as the values of $x1$ increase, the model predictions correspondingly show a rising trend. Conversely, for lower values of $x1$, the model predictions either remain constant or exhibit a declining trend. The same is true for the VAE model. Interestingly both models that do not have access to $x1$ show a kink in the range of 0 to 1 of $x2$.

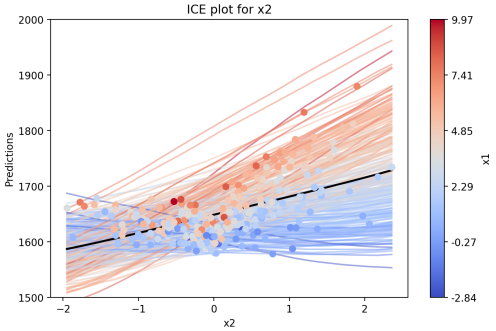


Figure 7.62.: ICE $x2$ plot of the full model.

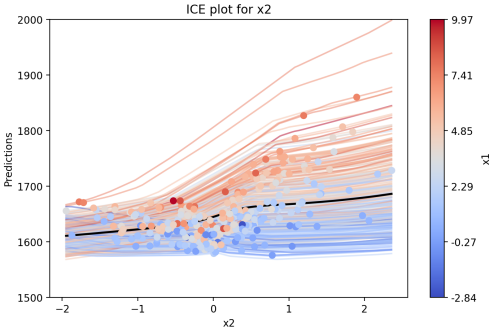


Figure 7.63.: ICE $x2$ plot of the confounder model.

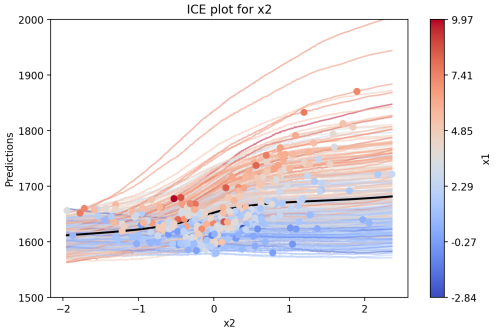


Figure 7.64.: ICE $x2$ plot of the VAE model.

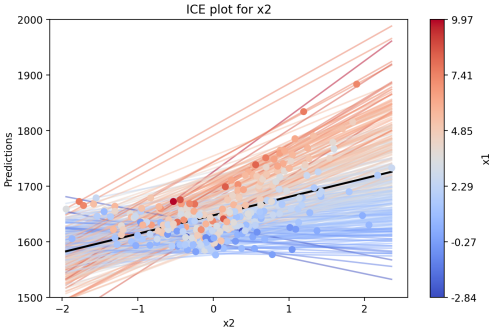


Figure 7.65.: ICE $x2$ plot of the true model.

We now turn our attention to x_3, which is defined as $\sqrt{|X_1 X_2|}$ augmented by noise. In the dependent variable y, this feature undergoes scaling and shifting, followed by squaring. The ICE plots for x_3 exhibit a general resemblance across models (Figures 7.66 - 7.69). However, it is noteworthy that the plots corresponding to the confounding model and the VAE model manifest a more cone-like shape as x_3 increases in value. As x_3 diminishes, the bundle of lines narrows, which may be an artifact attributable to the absence of x_1.

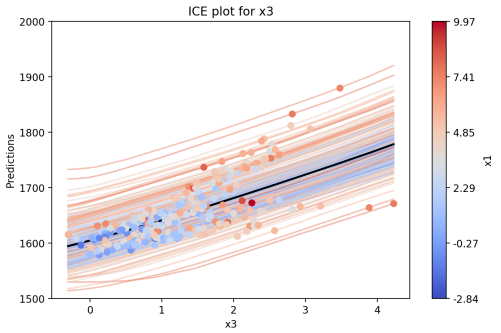


Figure 7.66.: ICE $x3$ plot of the full model.

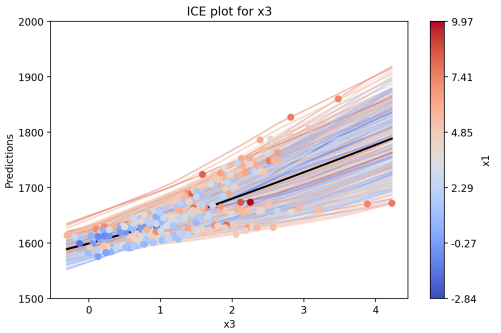


Figure 7.67.: ICE $x3$ plot of the confounder model.

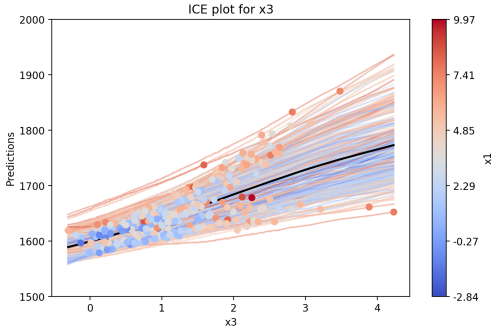


Figure 7.68.: ICE $x3$ plot of the VAE model.

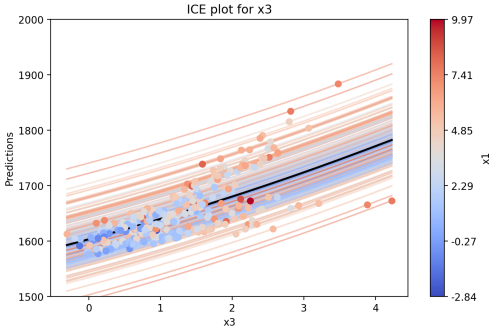


Figure 7.69.: ICE $x3$ plot of the true model.

Given that x_4 is encapsulated within the sine function in the equation for y, its influence on the magnitude of y is relatively minimal. The ground truth ICE plot (Figure 7.71) reveals the shallow periodicity, but the full model and the confounding model both approximated this with just straight lines. However, the VAE model has learned at least a slight periodic trend in areas where we have a lot of data points (Figure 7.72).

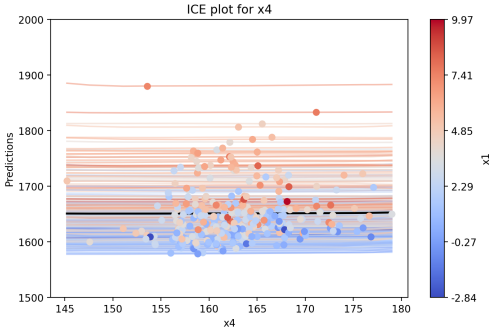


Figure 7.70.: ICE $x4$ plot of the full model.

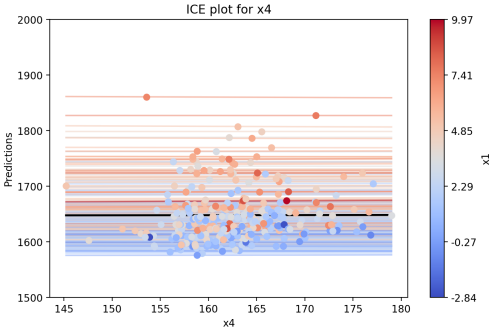


Figure 7.71.: ICE $x4$ plot of the confounder model.

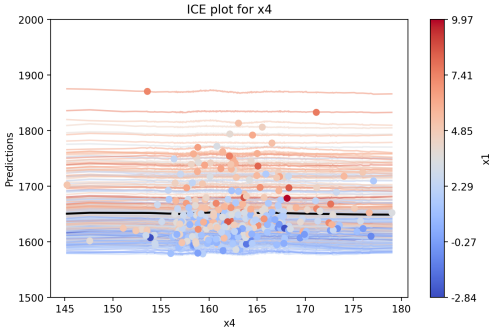


Figure 7.72.: ICE $x4$ plot of the VAE model.

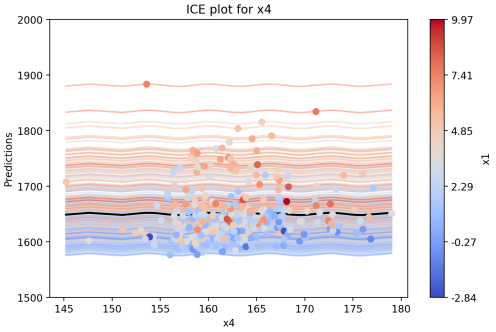


Figure 7.73.: ICE $x4$ plot of the true model.

The last variable in this data set is $x5$ which resembles a binary variable. With probability 0.5, we either add 30 or not. This can be seen in all ICE plots for the variable. Specifically, a value of 0 for x_5 results in a lower prediction, while a value of 1 leads to a higher prediction. It should be noted that the lines connecting the values of 0 and 1 in the graph are interpolations and not direct function evaluations.

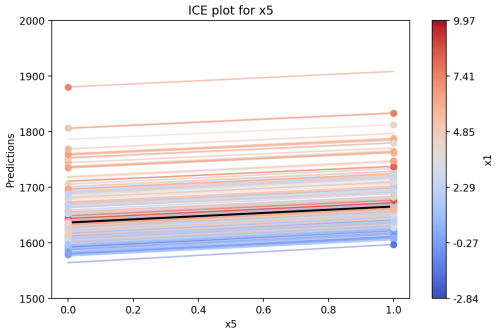


Figure 7.74.: ICE $x5$ plot of the full model.

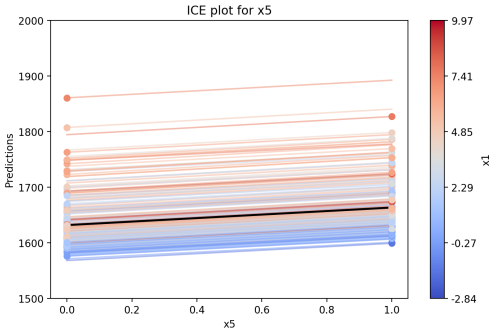


Figure 7.75.: ICE $x5$ plot of the confounder model.

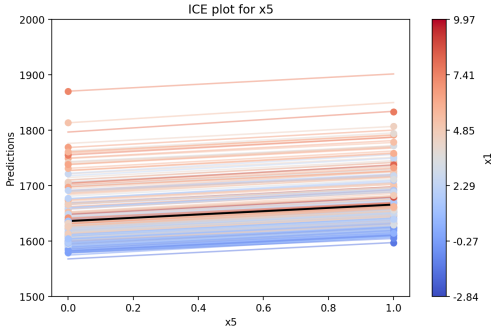


Figure 7.76.: ICE $x5$ plot of the VAE model.

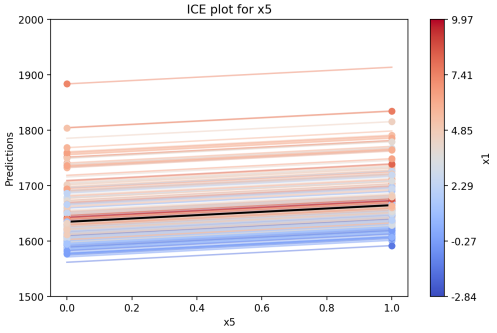


Figure 7.77.: ICE $x5$ plot of the true model.

Feature-importance / Kernel SHAP

In this analysis, we will again focus on the instances leading to the highest and lowest values of y. As before, will use the mean of all test set instances as our baseline, but with one important adjustment for $x5$. Since $x5$ is a binary variable taking only values 0 or 1, the neural network can behave in an arbitrary manner for intermediary values without incurring any penalty. Averaging $x5$ yields a value of 0.5. However, evaluating the mean instance results in a prediction even lower than the lowest instance in the test set for some models (Full, VAE).

This situation presents a problem when calculating the SHAP values against this baseline as it attributes an undeservedly significant influence to $x5$. If we plot a single ICE line for the baseline while altering $x5$, we notice that the model has learned a parabolic trajectory, instead of a straightforward linear one.

While this does not qualify as an error per se, since the model precisely predicts the relevant values at 0 and 1, it does provide an inaccurate impression of the baseline. Consequently, to solve this problem, we have adjusted the baseline value for $x5$ to 1 for all models. It is important to note that setting it to 0 would have been equally acceptable.

The baseline has the following values for each feature :

Feature	**Baseline**	**Best**	**Worst**
$x0$	1.00	2.663	0.021
$x1$	3.59	7.353	1.065
$x2$	0.001	1.893	0.029
$x3$	1.349	3.478	-0.006
$x4$	162.534	153.56	156.025
$x5$	1	0	0
y	-	1883.83	1577.08

Table 7.17.: Summary of instances

Model	Baseline Prediction
Full Model	1666.17
Confounder Model	1646.95
VAE Model	1670.36
True Model	1650.71

Table 7.18.: Baseline Predictions of Models

SHAP waterfall plots We again begin with the waterfall plots (Figure 7.78 - 7.81) for the instance with the highest y value in the samples test set. We see that the full model produces a very similar plot as the true model. Interestingly we see that the true model and the full model list $x2$ as the most important variable (Figures 7.81 and 7.78) which adds the most to the prediction, while the other two models which do not have access to $x1$ list $x3$ as the biggest contributor to the prediction. This could be due to the fact that $x1$ and $x3$ are also much more correlated (0.54) than $x1$ and $x2$ (-0.00). This means the confounder model and the VAE model make up with $x3$ for $x1$ which makes $x3$ more important. This switches the feature importance in these plots. We also observe that since the baseline for all models of the variable $x5$ was 1 we subtract roughly 30 from the prediction.

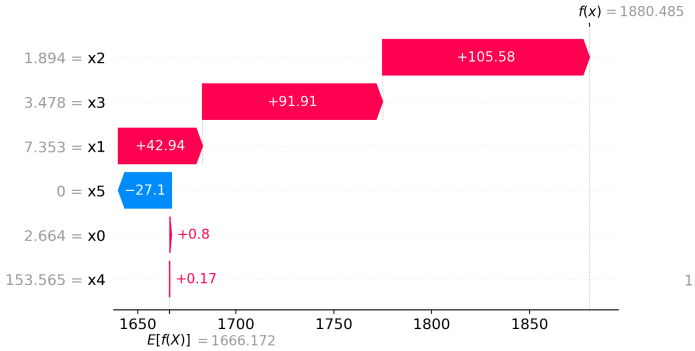


Figure 7.78.: SHAP Waterfall Plot of the full model.

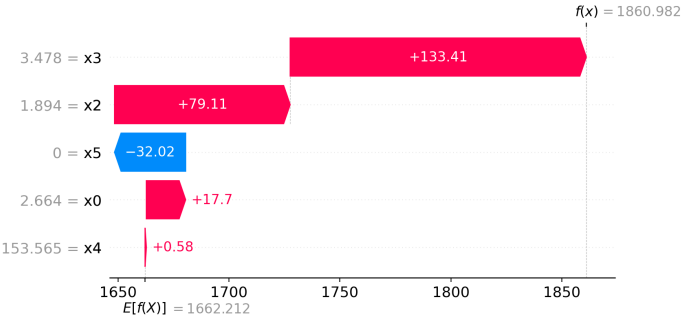


Figure 7.79.: SHAP Waterfall Plot of the confounder model.

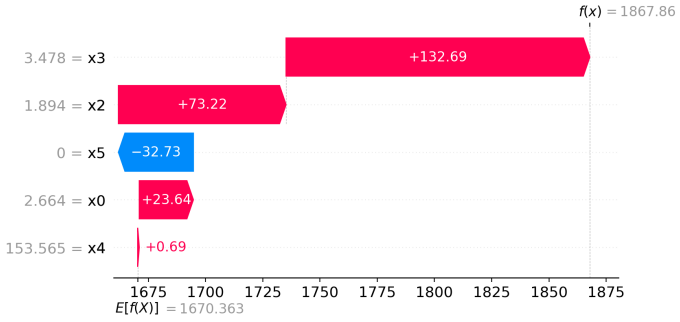


Figure 7.80.: SHAP Waterfall Plot of the VAE model.

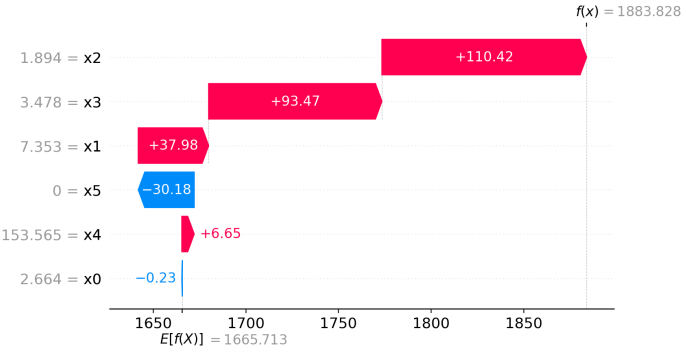


Figure 7.81.: SHAP Waterfall Plot of the true model.

The waterfall plots for the worst instance are all very similar. This time we see that for all models the variable $x3$ hast the most negative impact.

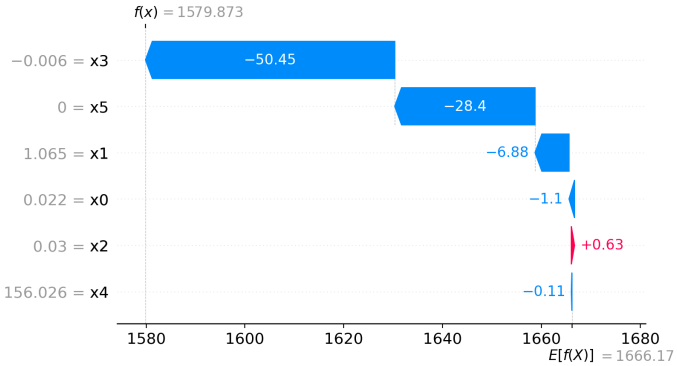


Figure 7.82.: SHAP Waterfall Plot of the full model.

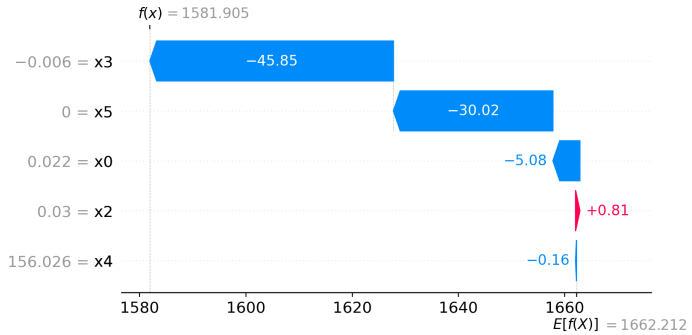


Figure 7.83.: SHAP Waterfall Plot of the confounder model.

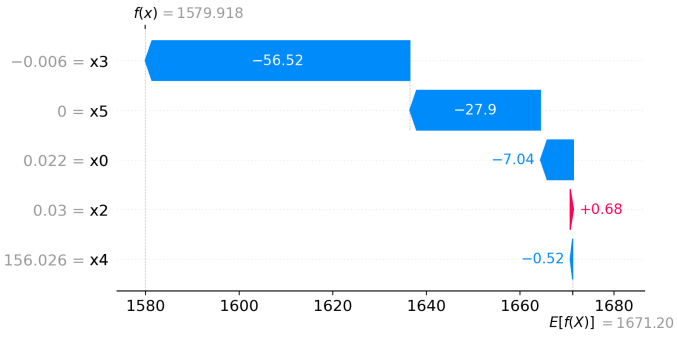


Figure 7.84.: SHAP Waterfall Plot of the VAE model.

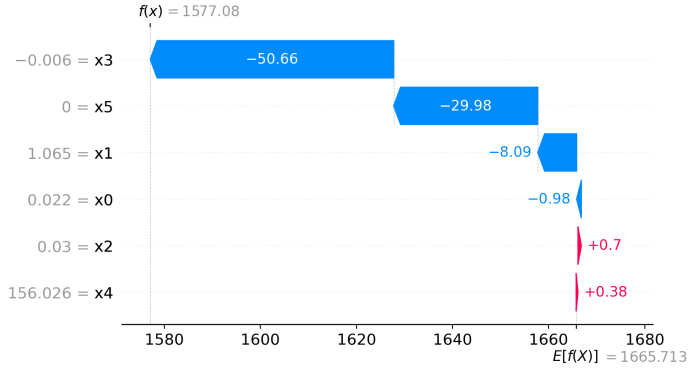


Figure 7.85.: SHAP Waterfall Plot of the true model.

Feature importance Figure 7.86 shows the feature importance of all models. The full model's importance attributed to x_1 appears to shift to x_0 in both the VAE and confounding models. Additionally, x_2 is deemed less significant in the confounding and VAE models compared to the full model. We also see that the full model has nearly identical importance as the underlying true model.

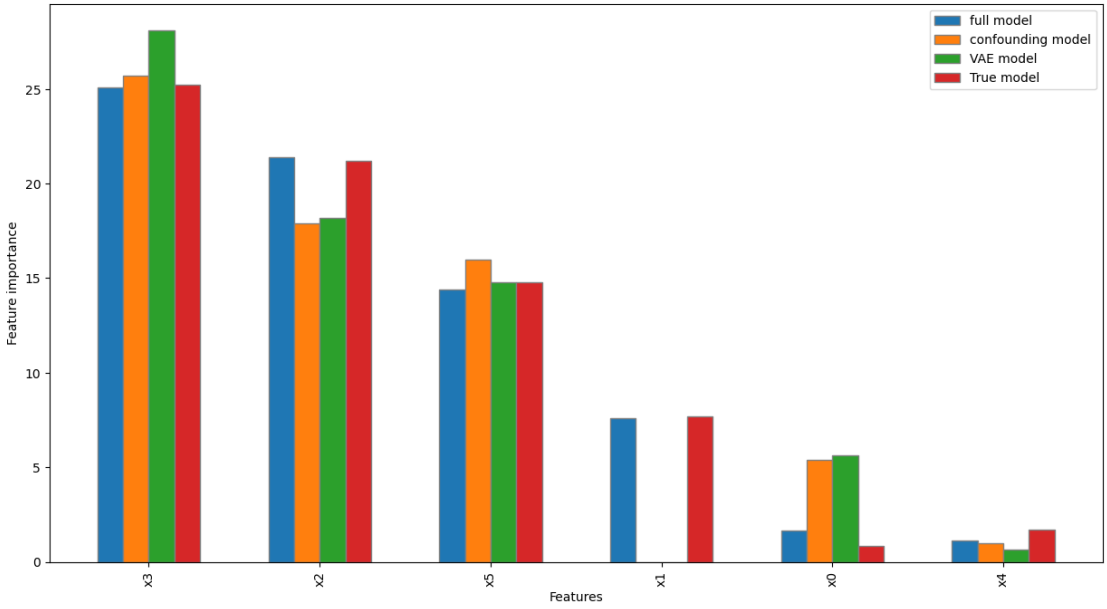


Figure 7.86.: Feature importance plot of all models.

SHAP summary plots In the concluding section of this chapter, we present the SHAP summary plots for models constructed using the toy dataset. Since we set the baseline of the variable $x5$ to 1 we see that SHAP values of instances that also have $x5 = 1$ are centered around 0. In contrast, SHAP values of instances where $x5 = 0$ are centered around -30. This shows that the baseline has some effect on the interpretation since in the true underlying equation we add 30 if $x5 = 1$ and do not subtract 30 if $x5 = 0$ but of course, the effect is equivalent. We also observe that the VAE model captured a more isolated impact of $x5$ which can be seen in the compact clustering of SHAP values instead of the more spread-out estimation of other models. Other than that we see that for both $x2$ and $x3$ a higher value will lead to a higher SHAP value in all models. We observe that the full model shows no clear structure for the variable $x0$ with respect to the relationship of SHAP value and feature value. However, the confounder model and the VAE model show a much clearer relationship and a higher variation which most probably is an artifact of the absence of variable $x1$.

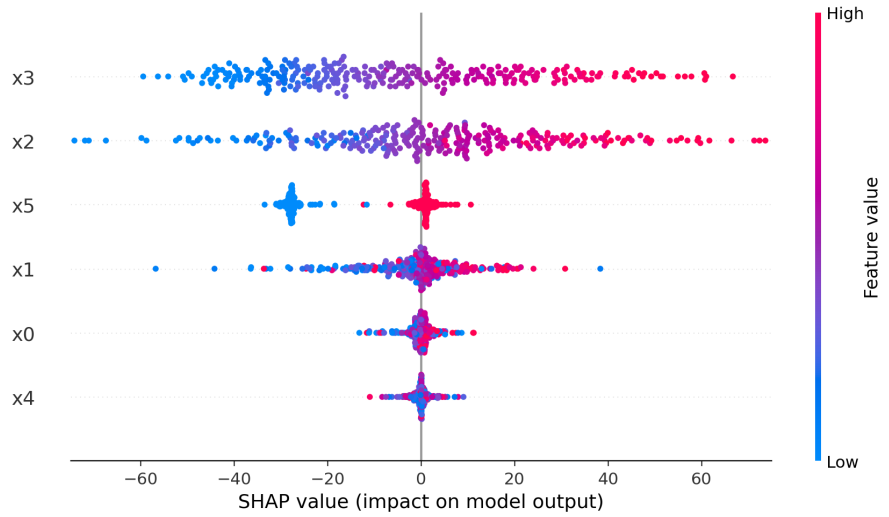


Figure 7.87.: SHAP summary plot of the full model.

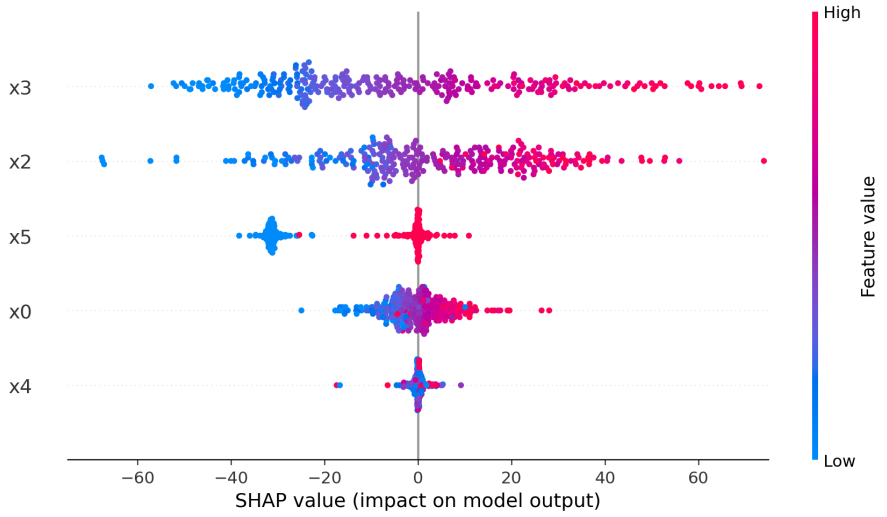


Figure 7.88.: SHAP summary plot of the confounding model.

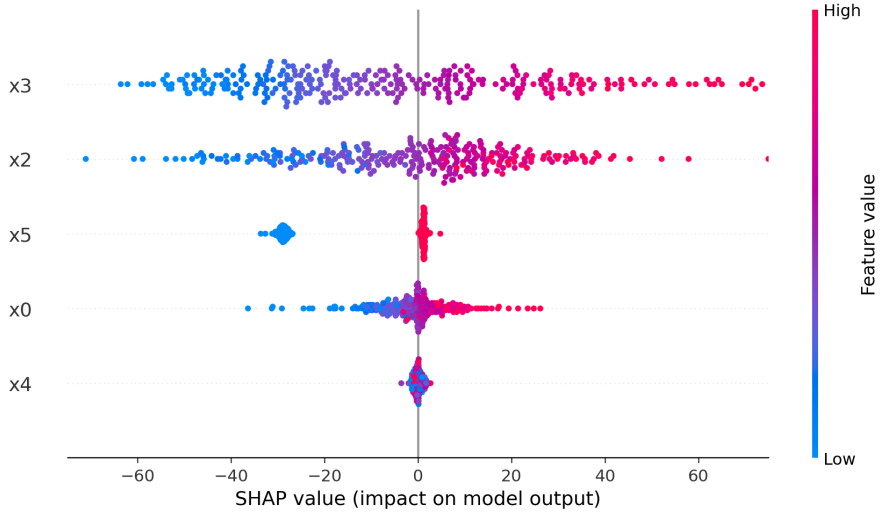


Figure 7.89.: SHAP summary plot of the VAE model.

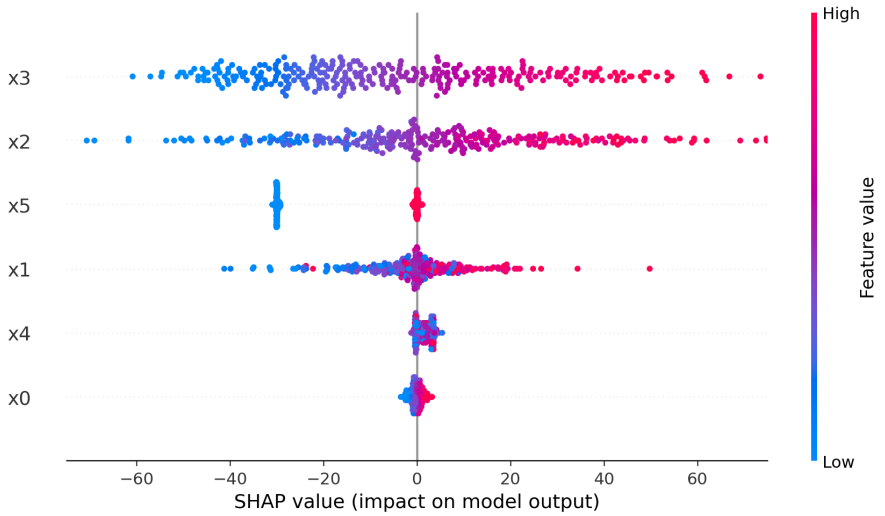


Figure 7.90.: SHAP summary plot of the true model.

Conclusion

Since all models achieve a very high R^2 score and the difference between the performance of the models is negligible we do not see big differences in our interpretation of these models. However, a consistent pattern we identified now across various datasets was again present which is a shift in feature importance when models lacked access to a specific feature. In the absence of $x1$, both the confounding model and the VAE model displayed increased reliance on variables $x0$ and $x3$. This shift was particularly evident in the SHAP waterfall plot when evaluating the instance with the apex target value, as depicted in Figures 7.78 - 7.81. Furthermore, we see that the variable $x2$ loses importance which shows that hidden confounding can not only increase the importance of a highly correlated variable but also decrease the the importance of other variables. This is an important observation when we interpret the importance and draw conclusions based on these values.

Furthermore, a notable observation was the VAE model's enhanced capability to discern the impact of variable $x5$. In comparison to the confounding model, the VAE exhibited a closer alignment with the true model in terms of capturing the essence of $x5$ (All instances have the same SHAP value instead of some small variations).

8. Discussion

8.1. Key findings

In our analysis, clear interpretational differences were evident across various models. At a macroscopic level, we saw that the interpretations of the VAE model indicate a more independent and isolated estimation of feature impact, meaning it has learned a more intuitive view of each feature's contribution to the prediction compared to the confounding model. This makes the interpretation of the model easier.

Specifically, for the bike rental and admission datasets, the VAE model consistently displayed more intuitive relationships between features. The ICE plot for the bike rental dataset's temperature (Figure 7.6) illustrated that an increase in temperature beyond a certain point corresponds with a decrease in bike rentals. This aligns with the notion that extreme heat reduces bike rental frequency, making this learned relationship in the VAE model more realistic than the relationship in the confounding model and more insightful than the full model, which primarily hinges on the season rather than the actual temperature.

Similarly, the ICE plots for the admission dataset underscored the VAE model's nuanced learning. For students with very low CGPAs, it indicated that a single high test score would not drastically elevate their admission probability. In contrast, the confounding model over-optimistically posited that just elevating the GRE score could align a student's admission probability with top-tier applicants (Figure 7.27). This contrast highlights the nuanced architecture of the VAE model, which appears to more effectively capture the contributions of individual features, thereby enhancing the model's interpretability and fidelity to the data. This quality of the VAE model was further echoed in the SHAP summary plots. The admission dataset's SHAP summary plot for the VAE model (Figure 7.53), for instance, presented consistent feature clusters, signifying a clearer, more intertwined estimation of feature influence.

A broader trend observed across datasets was the heightened reliance on closely correlated features when a particular feature was omitted. This behavior is anticipated: Correlated variables inherently carry overlapping information, so the model's increased dependency on one in the absence of another is predictable. This interplay was vividly demonstrated with the bike rental dataset's year and day_since_2011 (Figure 7.19). These tightly interwoven variables, despite their high correlation, saw varied utilization across models. The full model predominantly favored the variable year, while the VAE model and the confounding models were more inclined towards the variable day_since_2011 for capturing temporal nuances.

These results are in line with the current understanding that a VAE can disentangle

complex relationships of variables and learn a better representation of the data than the original representation [HMP+17].

These findings suggest that a simplistic examination of, for example, a model's feature importance could present challenges. Specifically, if one were to train a single model, the variable year might appear to be unimportant because the model gives greater weight to the variable day_since_2011. In reality, different models may capture the same underlying phenomenon, such as a time-dependent factor, but use different variables to do so. This is particularly relevant in the context of multicollinear datasets. While it may be evident in this case that these variables are highly correlated, there could be instances where such a relationship is not immediately obvious. To mitigate this issue, one could employ techniques like clustering the feature importances based on correlation or other metrics. This would allow for a more nuanced evaluation of the significance of variable groups or underlying concepts, such as time. Either way, we would always suggest taking the correlation matrix into consideration when interpreting models, especially when analyzing feature importance. The correlation matrices can be found in the appendix section A.3.

8.2. Answer of research questions

Research Question 1 The first research question aimed to investigate the impact of confounding variables on the interpretability of neural networks. The detailed formulation can be found in chapter 1.

Answer Research Question 1: We observed that excluding a variable from the feature set, which subsequently serves as a hidden confounding, amplifies the significance of correlated features. In our analysis, it was evident that the importance of the GRE score has increased substantially in the admissions dataset, as can be compared in Figure 7.50. Likewise, the ICE plot for the GRE score (Figure 7.27) reveals a much stronger relationship with the target variable. A similar pattern was observed in the case of the Seasons and temperature variables within the bike rental dataset, as well as the variables $x0$ and $x1$ in the toy dataset.

Research Question 2 The second research question sought to explore whether a Variational Autoencoder, which incorporates the target variable y could enhance the interpretability. The detailed formulation can be found in chapter 1.

Answer Research Question 2: An analysis of the ICE plot for the temperature variable in the VAE model (Figure 7.6) of the bike rental dataset provided a more grounded and intuitive interpretation compared to the confounding model. The VAE model discerned that high temperatures lead to a reduction in bike rentals, independent of the season, rather than implying this decrease occurs only during summer (Figure 7.4). Similarly, in the Admission dataset, we noticed that a high score on just one test does not substantially elevate the probability of admission for low CGPA students (Figure 7.28). An increase in likelihood necessitates improvements across multiple test scores (GRE, TOEFL, etc.), a relationship that arguably mirrors a more realistic scenario between the

variables (Figure 7.54). Additionally, we observed more consistent shapley values for some attributes compared to the confounding models (Figures 7.52,7.53).

This implies that the VAE model was able to isolate the effect of certain variables and thus provided a clearer insight into the underlying relationships. The consistency in Shapley values in the VAE models as compared to the confounding models suggests that the VAE approach could lead to more stable, and reliable interpretations.

8.3. PDR-Framework

As elaborated in the background section 3.1, the PDR framework encompasses three core desiderata: predictive accuracy, descriptive accuracy, and the relevancy of the interpretation. Our findings consistently revealed that the VAE model exhibited superior predictive accuracy compared to the confounding model across all datasets. Thus we can conclude that the improvement of the interpretability can be partially attributed to this fact. In certain instances, the clarity of the interpretations derived from the VAE model even surpassed those obtained from the full model. Of course, this is subjective since descriptive accuracy is generally hard to measure. Nevertheless, this could indicate that the VAE model learned relationships that were better or easier captured by the interpretability methods. This suggests that since the predictive accuracy is comparable, the model also improved a bit on descriptive accuracy. In our study, we aimed for a global understanding of the model's behavior. Thus, we opted out of more granular methods, like neuron-based layer attribution, favoring broader interpretability methods. This decision is pivotal: the choice of interpretability method can greatly influence the relevancy of the results. A noteworthy takeaway looking through the PDR lens is the intertwined relationship between predictive and descriptive accuracy. Murdoch et al. [MSK$^+$19] contend that for interpretations to be trusted, both high predictive and descriptive accuracies are pivotal. Their assertion resonates with our findings, underscoring the integral role these facets play in facilitating authentic, meaningful model insights.

8.4. Limitations and weaknesses

While we cannot rule out the possibility that the enhanced clarity in the VAE model's interpretations stems from its inherent expressiveness, it's also important to note that an effort was made to tune the models. Exploring a broader hyperparameter space in the future might offer further insights. We also note that we saw notable differences between the full and the confounding model and improvements of the VAE model only on small datasets which makes it unclear if this scales to large datasets. Given that the models on the larger toy dataset exhibited consistently high accuracy metrics which made the interpretations nearly identical, it may be a good idea to evaluate the VAE architecture on more extensive real-world datasets. Due to time constraints, we were not able to investigate this further. It is very important to add that all interpretations are done by the author which is not the gold standard when it comes to interpreting the results. Due to the complexity of a comprehensive user study, coupled with the considerable amount

of work it entails and the constraints of this thesis's scope and timeframe, we were unable to conduct one.

9. Summary

In this master's thesis, we explore the impact of confounding variables on the interpretability of neural networks and propose an enhanced model rooted in variational autoencoding to improve interpretability in the presence of latent confounders. Experiments on three distinct datasets demonstrate that omitting a single variable from the dataset - acting as a latent confounder - can drastically change the interpretation of conventional feed-forward neural networks (FFNNs). These distorted relationships were illustrated using various interpretability techniques, including Individual Conditional Expectation (ICE) plots and Kernel SHAP. Our findings reveal that the confounder influences the importance of highly correlated features and overestimates the effect of variables on the target. Furthermore, the proposed model (VAE model) yields more intuitive and consistent interpretations than standard FFNNs when affected by latent confounding variables. Building on this, given additional resources, we strongly recommend conducting a user study. When paired with an alternative modeling approach that uses the target variable y as an input to the VAE, this suggestion opens up a promising direction for future research. Additionally, it might be interesting to try to improve the representation of the feature importance plot which might also take the correlation of the variables into consideration. As we saw in the bike rental dataset the variables year and day_since_2011 are highly correlated but the importance is very different for the full and confounding model. But both are capturing a temporal trend which might make sense to cluster these importances together to have a better interpretable plot.

Bibliography

[AAA19] Mohan S Acharya, Asfia Armaan, and Aneeta S Antony. A comparison of regression models for prediction of graduate admissions. In *2019 International Conference on Computational Intelligence in Data Science (ICCIDS)*, pages 1–5, 2019.

[ALPM+13] Filippo Amato, Alberto López, Eladia María Peña-Méndez, Petr Vaňhara, Aleš Hampl, and Josef Havel. Artificial neural networks in medical diagnosis. *Journal of Applied Biomedicine*, 11(2):47–58, 2013.

[BMR+20] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. *CoRR*, abs/2005.14165, 2020.

[BSG+10] M Alan Brookhart, Til Stürmer, Robert J Glynn, Jeremy Rassen, and Sebastian Schneeweiss. Confounding control in healthcare database research. *Med. Care*, 48(6):S114–S120, June 2010.

[BYC+17] Mariusz Bojarski, Philip Yeres, Anna Choromanska, Krzysztof Choromanski, Bernhard Firner, Lawrence D. Jackel, and Urs Muller. Explaining how a deep neural network trained with end-to-end learning steers a car. *CoRR*, abs/1704.07911, 2017.

[CHH+09] Jerome Cornfield, William Haenszel, E. Cuyler Hammond, Abraham M. Lilienfeld, Michael B. Shimkin, and Ernst L. Wynder. Smoking and lung cancer: recent evidence and a discussion of some questions. *International journal of epidemiology*, 38(5):1175–1191, 2009.

[dHJL19] Pim de Haan, Dinesh Jayaraman, and Sergey Levine. Causal confusion in imitation learning. *CoRR*, abs/1905.11979, 2019.

[DVK17] Finale Doshi-Velez and Been Kim. Towards a rigorous science of interpretable machine learning, 2017.

[Fri01] Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. *The Annals of statistics*, 29(5):1189–1232, 2001.

[FTG14] Hadi Fanaee-T and Joao Gama. Event labeling combining ensemble detectors and background knowledge. *Progress in Artificial Intelligence*, 2(2):113–127, June 2014.

[GKBP15] Alex Goldstein, Adam Kapelner, Justin Bleich, and Emil Pitkin. Peeking inside the black box: Visualizing statistical learning with plots of individual conditional expectation. *Journal of computational and graphical statistics*, 24(1):44–65, 2015.

[Ham74] Frank R. Hampel. The influence curve and its role in robust estimation. *Journal of the American Statistical Association*, 69(346):383–393, 1974.

[HKN+21] Tim Head, Manoj Kumar, Holger Nahrstaedt, Gilles Louppe, and Iaroslav Shcherbatyi. scikit-optimize/scikit-optimize. Oct 2021.

[HMKG17] Tor Haldorsen, Jan Ivar Martinsen, Kristina Kjærheim, and Tom K Grimsrud. Adjustment for tobacco smoking and alcohol consumption by simultaneous analysis of several types of cancer. *Cancer Causes Control*, 28(2):155–165, February 2017.

[HMP+17] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-VAE: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations*, 2017.

[Jae72] Louis A Jaeckel. The infinitesimal jackknife. *Unpublished memorandum, Bell Telephone Laboratories, Murray Hill, NJ*, 1972.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, page 1097–1105, Red Hook, NY, USA, 2012. Curran Associates Inc.

[KW22] Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2022.

[KZEW04] David Kriebel, Ariana Zeka, Ellen A Eisen, and David H Wegman. Quantitative evaluation of the effects of uncontrolled confounding by alcohol and tobacco in occupational cancer studies. *International Journal of Epidemiology*, 33(5):1040–1045, 05 2004.

[LJR+18] Liam Li, Kevin G. Jamieson, Afshin Rostamizadeh, Ekaterina Gonina, Moritz Hardt, Benjamin Recht, and Ameet Talwalkar. Massively parallel hyperparameter tuning. *CoRR*, abs/1810.05934, 2018.

[LL17] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 4765–4774. Curran Associates, Inc., 2017.

[LSM+18] C Louizos, U Shalit, J Mooij, D Sontag, R Zemel, M Welling, U von Luxburg, I Guyon, S Bengio, H Wallach, R Fergus, S.V.N Vishwanathan, and R Garnett. Causal effect inference with deep latent-variable models. *31st Conference on Advances in Neural Information Processing Systems (NIPS 2017)*, 10:6447–6457, 2018.

[LZHH14] Znaonui Liang, Gang Zhang, Jimmy Xiangji Huang, and Qmming Vivian Hu. Deep learning for healthcare decision making with emrs. In *2014 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 556–559, 2014.

[MCPZ19] David Madras, Elliot Creager, Toniann Pitassi, and Richard Zemel. Fairness through causal awareness: Learning causal latent-variable models for biased data. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, FAT* '19, page 349–358, New York, NY, USA, 2019. Association for Computing Machinery.

[MGD23] Cristian Meo, Anirudh Goyal, and Justin Dauwels. Tc-vae: Uncovering out-of-distribution data generative factors, 2023.

[MKS+13] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602, 2013.

[MMOR+21] Francisco J. Martinez-Murcia, Andrés Ortiz, Javier Ramírez, Juan M. Górriz, and Ricardo Cruz. Deep residual transfer learning for automatic diagnosis and grading of diabetic retinopathy. *Neurocomputing*, 452:424–434, 2021.

[MNW+17] Philipp Moritz, Robert Nishihara, Stephanie Wang, Alexey Tumanov, Richard Liaw, Eric Liang, William Paul, Michael I. Jordan, and Ion Stoica. Ray: A distributed framework for emerging AI applications. *CoRR*, abs/1712.05889, 2017.

[Mol22] Christoph Molnar. *Interpretable Machine Learning*. 2 edition, 2022.

[MSK+19] W. James Murdoch, Chandan Singh, Karl Kumbier, Reza Abbasi-Asl, and Bin Yu. Definitions, methods, and applications in interpretable machine learning. *Proceedings of the National Academy of Sciences*, 116(44):22071–22080, oct 2019.

[Ope23] OpenAI. Gpt-4 technical report, 2023.

[PBS18] Reid Pryzant, Sugato Basu, and Kazoo Sone. Interpretable neural architectures for attributing an ad's performance to its writing style. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 125–135, Brussels, Belgium, November 2018. Association for Computational Linguistics.

[Pea09] Judea Pearl. *Causality*. Cambridge University Press, 2 edition, 2009.

[PGM+19] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[RTB16] Rajesh Ranganath, Dustin Tran, and David Blei. Hierarchical variational models. In *International conference on machine learning*, pages 324–333. PMLR, 2016.

[Sha53] Lloyd S Shapley. A value for n-person games. In Harold W. Kuhn and Albert W. Tucker, editors, *Contributions to the Theory of Games II*, pages 307–317. Princeton University Press, Princeton, 1953.

[SMS20] Numair Sani, Daniel Malinsky, and Ilya Shpitser. Explaining the behavior of black-box prediction algorithms with causal learning. *CoRR*, abs/2006.02482, 2020.

[TLS+22] Qiaoying Teng, Zhe Liu, Yuqing Song, Kai Han, and Yang Lu. A survey on the interpretability of deep learning in medical diagnosis. *Multimed. Syst.*, 28(6):2335–2355, June 2022.

[Van04] Jan P. Vandenbroucke. *The history of confounding*, pages 313–325. Birkhäuser Basel, Basel, 2004.

[Wen18] Lilian Weng. From autoencoder to beta-vae. *lilianweng.github.io*, 2018.

[WMR17] Sandra Wachter, Brent D. Mittelstadt, and Chris Russell. Counterfactual explanations without opening the black box: Automated decisions and the GDPR. *CoRR*, abs/1711.00399, 2017.

[ZBL+18] John R. Zech, Marcus A. Badgeley, Manway Liu, Anthony B. Costa, Joseph J. Titano, and Eric Karl Oermann. Variable generalization performance of a deep learning model to detect pneumonia in chest radiographs: A cross-sectional study. *PLOS Medicine*, 15(11):e1002683, nov 2018.

[ZGR+22] Jiaming Zeng, Michael F Gensheimer, Daniel L Rubin, Susan Athey, and Ross D Shachter. Uncovering interpretable potential confounders in electronic medical records. *Nature communications*, 13(1):1014–1014, 2022.

[ZTLT20] Yu Zhang, Peter Tiňo, Ales Leonardis, and Ke Tang. A survey on neural network interpretability. *CoRR*, abs/2012.14261, 2020.

A. Appendix

A.1. Summary statistics of datasets

	count	**mean**	**std**	**min**	**25%**	**50%**	**75%**	**max**
yr	731.0	2011.5	0.5	2011.0	2011.0	2012.0	2012.0	2012.0
temp	731.0	15.3	8.6	-5.2	7.8	15.4	22.8	32.5
hum	731.0	62.8	14.2	0.0	52.0	62.7	73.0	97.3
windspeed	731.0	12.8	5.2	1.5	9.0	12.1	15.6	34.0
cnt	731.0	4504.3	1937.2	22.0	3152.0	4548.0	5956.0	8714.0
days_since_2011	731.0	365.0	211.2	0.0	182.5	365.0	547.5	730.0
season_FALL	731.0	0.2	0.4	0.0	0.0	0.0	0.0	1.0
season_SPRING	731.0	0.3	0.4	0.0	0.0	0.0	1.0	1.0
season_SUMMER	731.0	0.3	0.4	0.0	0.0	0.0	1.0	1.0
season_WINTER	731.0	0.2	0.4	0.0	0.0	0.0	0.0	1.0
holiday	731.0	0.0	0.2	0.0	0.0	0.0	0.0	1.0
workingday	731.0	0.7	0.5	0.0	0.0	1.0	1.0	1.0
weathersit_GOOD	731.0	0.6	0.5	0.0	0.0	1.0	1.0	1.0
weathersit_MISTY	731.0	0.3	0.5	0.0	0.0	0.0	1.0	1.0
weathersit_RAIN /SNOW/STORM	731.0	0.0	0.2	0.0	0.0	0.0	0.0	1.0

Table A.1.: Summary Statistics of the Bike Rental Dataset

	count	**mean**	**std**	**min**	**25%**	**50%**	**75%**	**max**
GRE Score	500.0	316.47	11.29	290.00	308.00	317.00	325.00	340.00
TOEFL Score	500.0	107.19	6.08	92.00	103.00	107.00	112.00	120.00
University Rating	500.0	3.11	1.14	1.00	2.00	3.00	4.00	5.00
SOP	500.0	3.37	0.99	1.00	2.50	3.50	4.00	5.00
LOR	500.0	3.48	0.92	1.00	3.00	3.50	4.00	5.00
CGPA	500.0	8.57	0.60	6.80	8.12	8.56	9.04	9.92
Research	500.0	0.56	0.49	0.00	0.00	1.00	1.00	1.00
Chance of Admit	500.0	0.72	0.14	0.34	0.63	0.72	0.82	0.97

Table A.2.: Summary Statistics of the Graduate Admissions Dataset

	count	mean	std	min	25%	50%	75%	max
X_0	200000.00	1.00	1.00	-3.60	0.33	1.00	1.67	5.57
X_1	200000.00	3.60	2.15	-6.02	2.15	3.59	5.04	13.02
X_2	200000.00	-0.00	0.80	-3.56	-0.54	-0.00	0.53	3.81
X_3	200000.00	1.34	0.76	-0.62	0.77	1.27	1.84	5.49
X_4	200000.00	162.48	5.48	142.17	158.66	162.05	165.94	188.98
X_5	200000.00	0.50	0.50	0.00	0.00	0.00	1.00	1.00
Y	200000.00	1652.75	50.76	1554.97	1619.80	1641.44	1668.85	2133.27

Table A.3.: Summary Statistics for the Synthetic Dataset

A.2. Models

A.2.1. Bike rental: Models, plots, and parameters

Full model

Configuration	Value
Hidden layers	3
Hidden units	[128, 128, 64]
Dropout	False
Dropout array	[0.5, 0.322, 0.647]
Activation	ReLU
Normalization	False
Learning rate	0.005
Batch size	8
Number of epochs	400

Table A.4.: Configuration full model

Figure A.1.

Confounder model

Configuration	Value
Batch size	8
Learning rate	0.0022
Number of epochs	400
Hidden layers	3
Hidden units	[32, 32, 128]
Dropout	False
Dropout array	[0.382, 0.1466, 0.539]
Activation	ReLU
Normalization	False

Table A.5.: Configuration confounder model

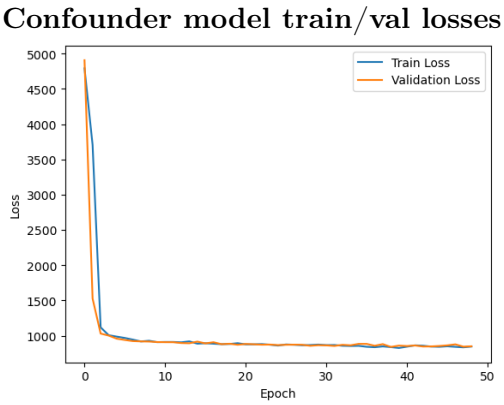


Figure A.2.

VAE model

Configuration	Value
x_dim	10
z_dim	4
z'_dim	1
$Enc_z(x)$	[128, 128]
$Enc_{z'}(z)$	[5, 5, 5]
$Dec(z, z')$	[128, 128, 10]
$\psi(z, z')$	[28, 28, 1]
Beta Values	
b1	10000
b2	1
b3	1
b4	1000
Learning rate	0.001
Batch size	32
Number of epochs	700

Table A.6.: Configuration VAE model

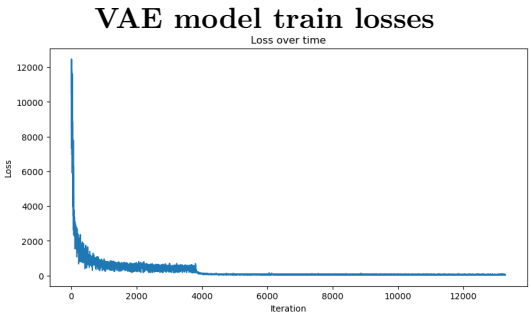


Figure A.3.

Truth-prediction plots

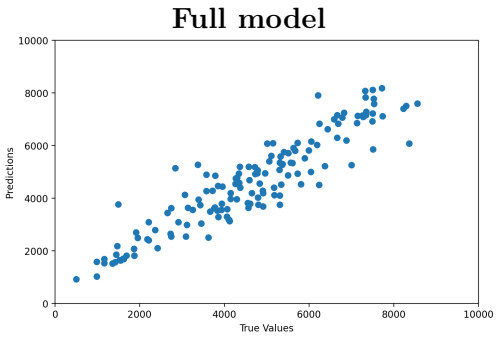


Figure A.4.: This image shows a truth-prediction plot for the full model.

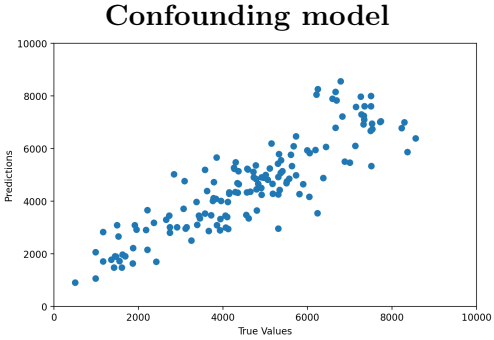


Figure A.5.: This image shows a truth-prediction plot for the confounding model.

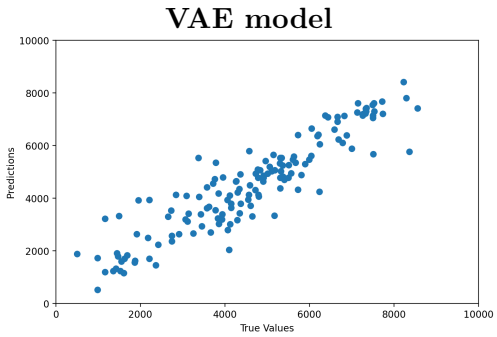


Figure A.6.: This image shows a truth-prediction plot for the VAE model.

A.2.2. Admission: Models, plots, and parameters

Full model

Configuration	Value
Batch size	8
Learning rate	5×10^{-5}
Number of epochs	300
Hidden layers	3
Hidden units	[128, 16, 32]
Dropout	False
Dropout array	[0.5262, 0.1716, 0.2839]
Activation	ReLU
Normalization	True
Patience	20

Table A.7.: Configuration for the full model

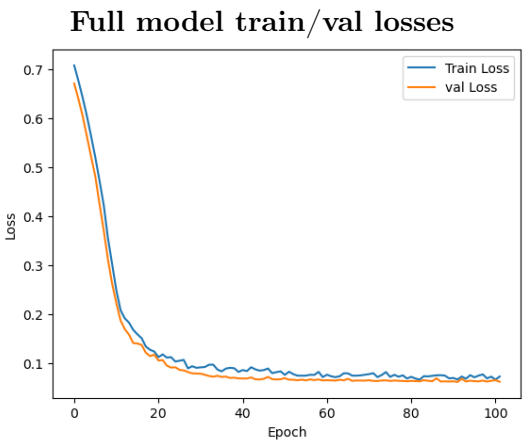


Figure A.7.

Confounder model

Configuration	Value
Batch size	16
Learning rate	0.000157
Number of epochs	500
Hidden layers	1
Hidden units	[128]
Dropout	False
Dropout array	[0.5262, 0.1716, 0.2839]
Activation	ReLU
Normalization	True
Patience	30

Table A.8.: Configuration for the confounding model

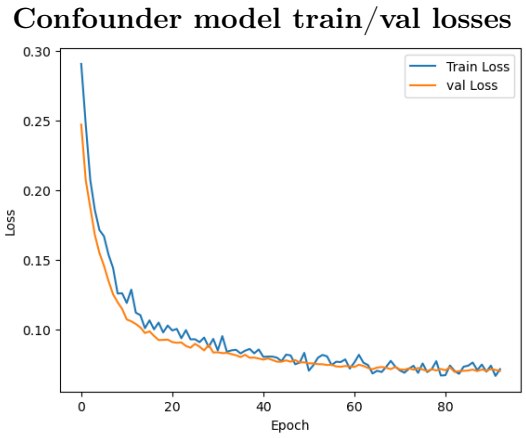


Figure A.8.

VAE model

Configuration	Value
x_dim	6
z_dim	3
z'_dim	1
$Enc_z(x)$	[10, 10, 10]
$Enc_{zz}(z)$	[10, 10]
$Dec(z, zz)$	[10, 10, 10, 6]
$\psi(z, zz)$	[12, 12, 1]
Beta Values	
b1	10000
b2	1
b3	1
b4	10000
Learning rate	0.001
Batch size	32
Number of epochs	600

Table A.9.: Configuration for VAE model

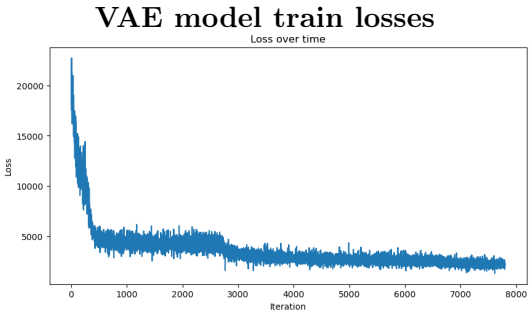


Figure A.9.

Truth-prediction plots

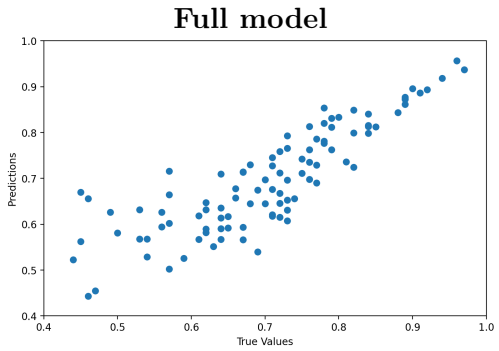


Figure A.10.: This image shows a truth-prediction plot for the full model for the admission dataset.

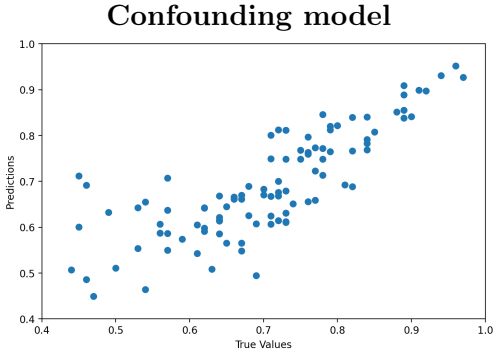


Figure A.11.: This image shows a truth-prediction plot for the confounding model for the admission dataset.

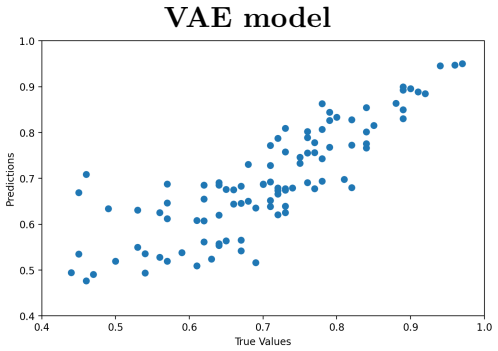


Figure A.12.: This image shows a truth-prediction plot for the VAE model for the admission dataset.

A.2.3. Toy: Models, plots, and parameters

Full model

Configuration	Value
Batch size	25
Learning rate	0.0004
Number of epochs	100
Hidden layers	3
Hidden units	[128, 64, 32]
Dropout	False
Dropout array	[0.382, 0.146, 0.539]
Activation	ReLU
Normalization	False
Patience	20

Table A.10.: Configuration confounder model

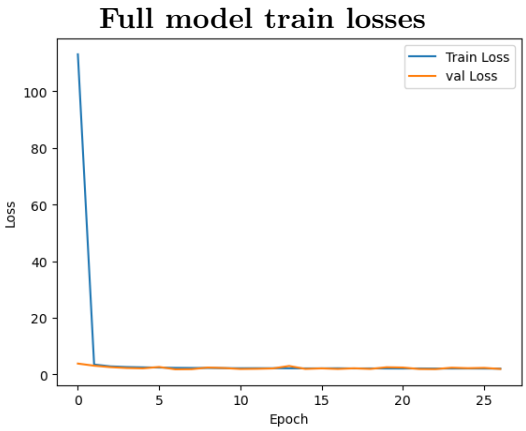


Figure A.13.

Confounder model

Configuration	Value
Batch size	53
Learning rate	0.000384
Number of epochs	100
Hidden layers	3
Hidden units	[64, 32, 32]
Dropout	False
Dropout array	[0.375, 0.642, 0.690]
Activation	ReLU
Normalization	True
Patience	20

Table A.11.: Configuration confounder model

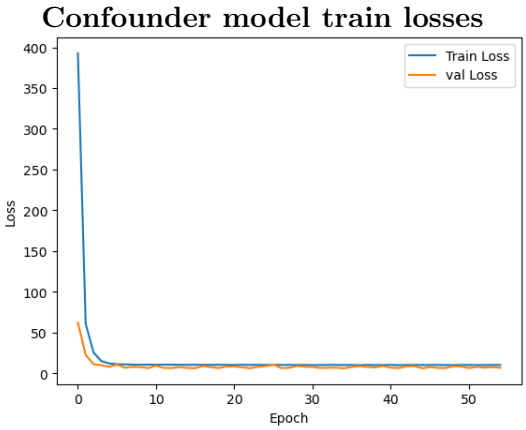


Figure A.14.

VAE model

Configuration	Value
x_dim	5
z_dim	3
z'_dim	1
$Enc_z(x)$	[64, 64, 64]
$Enc_{zz}(z)$	[10, 10]
$Dec(z, zz)$	[64, 64, 64, 5]
$\psi(z, zz)$	[32, 32, 32, 1]
Beta Values	
b1	10000
b2	1
b3	1
b4	10000
Learning rate	0.001
Batch size	1024
Number of epochs	500

Table A.12.: Configuration VAE model

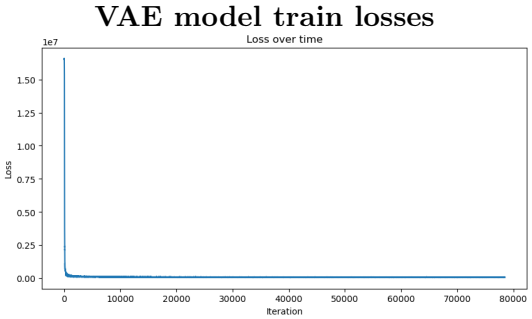


Figure A.15.

Truth-prediction plots

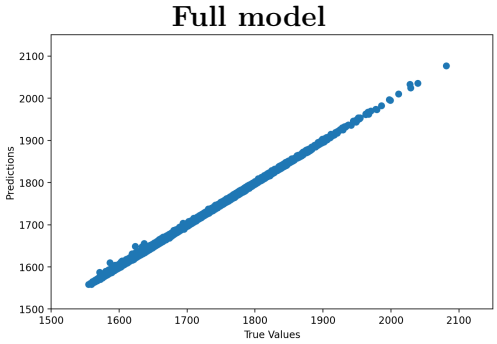


Figure A.16.: This image shows a truth-prediction plot for the full model for the toy dataset.

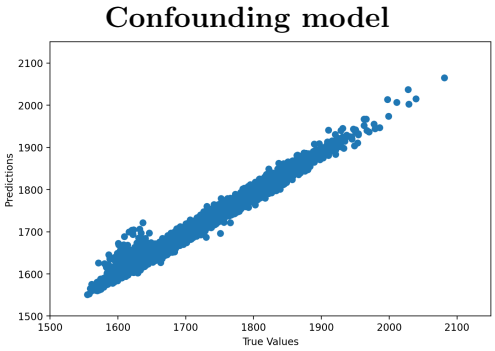


Figure A.17.: This image shows a truth-prediction plot for the confounding model for the toy dataset.

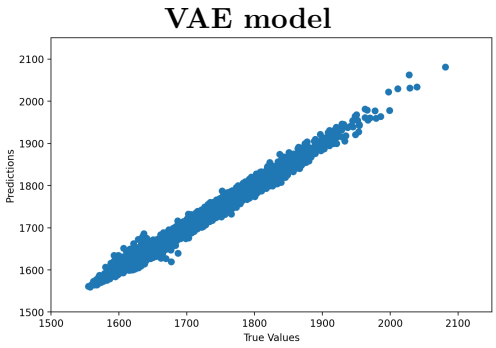


Figure A.18.: This image shows a truth-prediction plot for the VAE model for the toy dataset.

A.3. Correlation structure

The plots in this section show the correlation matrix of the datasets. We also added the latent variables of the VAE models to see how the latent space correlates with other variables. We only used the test set to calculate the correlation. $z1 - zn$ are the variables in the latent space z while $zz1 - zzn$ are the variables of the latent space z'.

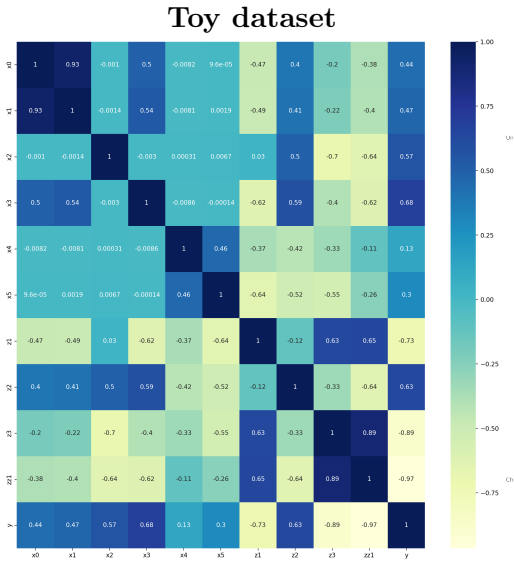


Figure A.19.

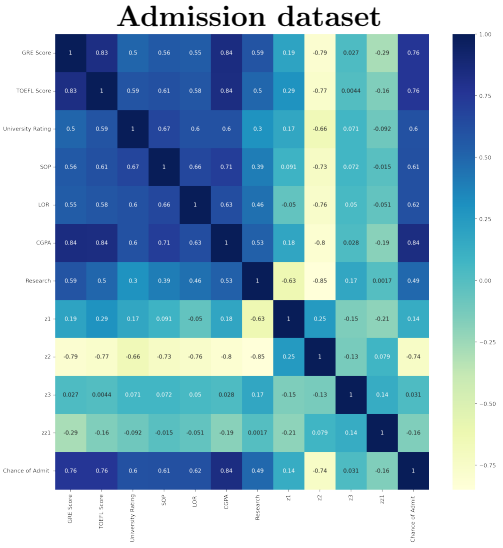


Figure A.20.

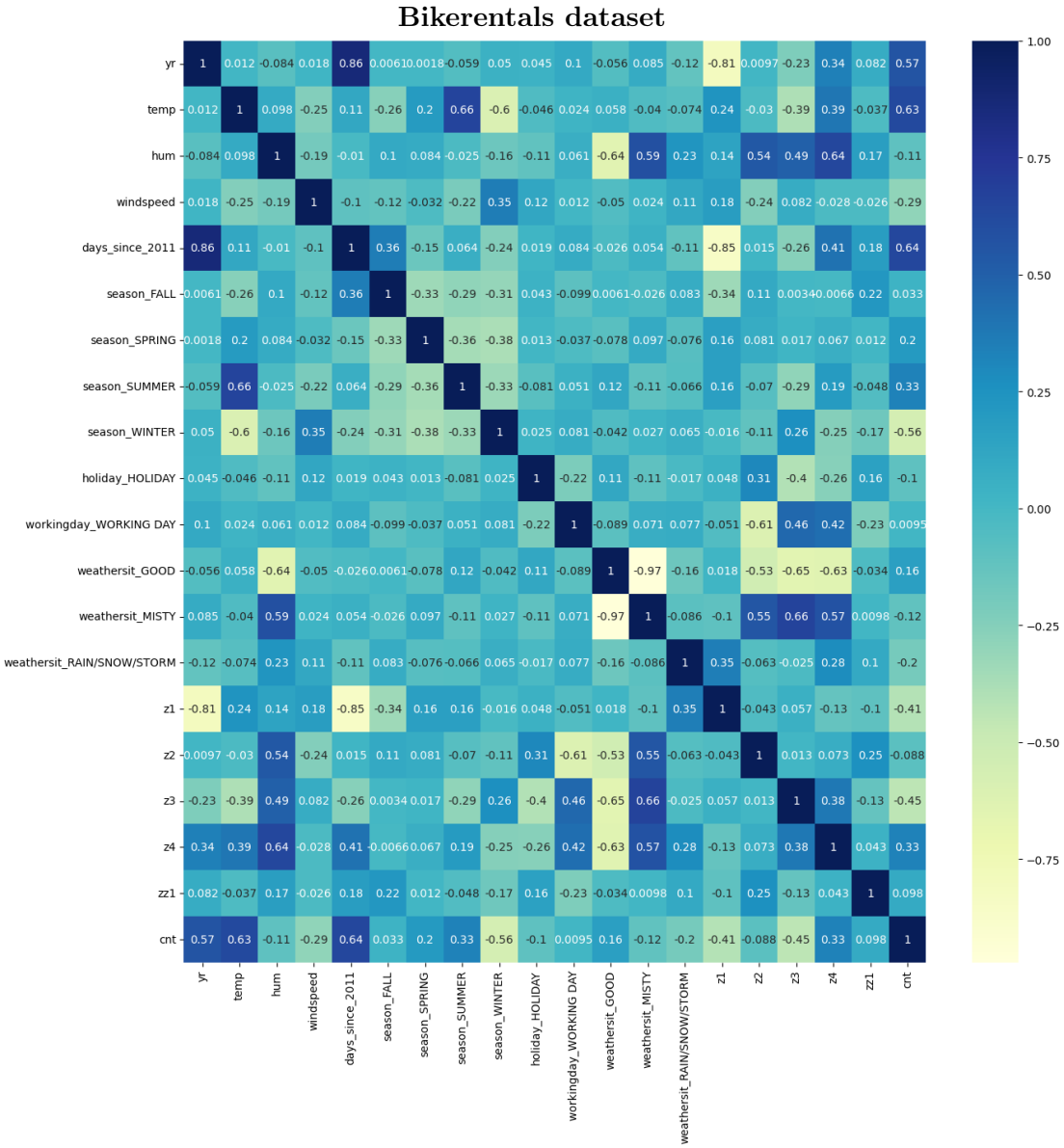

Bikerentals dataset
yr 1 0.012 -0.084 0.018 0.86 0.0061 0.0018 -0.059 0.05 0.045 0.1 -0.056 0.085 -0.12 -0.81 0.0097 -0.23 0.34 0.082 0.57
temp 0.012 1 0.098 -0.25 0.11 -0.26 0.2 0.66 -0.6 -0.046 0.024 0.058 -0.04 -0.074 0.24 -0.03 -0.39 0.39 -0.037 0.63
hum -0.084 0.098 1 -0.19 -0.01 0.1 0.084 -0.025 -0.16 -0.11 0.061 -0.64 0.59 0.23 0.14 0.54 0.49 0.64 0.17 -0.11
windspeed 0.018 -0.25 -0.19 1 -0.1 -0.12 -0.032 -0.22 0.35 0.12 0.012 -0.05 0.024 0.11 0.18 -0.24 0.082 -0.028 -0.026 -0.29
days_since_2011 0.86 0.11 -0.01 -0.1 1 0.36 -0.15 0.064 -0.24 0.019 0.084 -0.026 0.054 -0.11 -0.85 0.015 -0.26 0.41 0.18 0.64
season_FALL -0.0061 -0.26 0.1 -0.12 0.36 1 -0.33 -0.29 -0.31 0.043 -0.099 0.0061 -0.026 0.083 -0.34 0.11 0.0034 -0.0066 0.22 0.033
season_SPRING -0.0018 0.2 0.084 -0.032 -0.15 -0.33 1 -0.36 -0.38 0.013 -0.037 -0.078 0.097 -0.076 0.16 0.081 0.017 0.067 0.012 0.2
season_SUMMER -0.059 0.66 -0.025 -0.22 0.064 -0.29 -0.36 1 -0.33 -0.081 0.051 0.12 -0.11 -0.066 0.16 -0.07 -0.29 0.19 -0.048 0.33
season_WINTER 0.05 -0.6 -0.16 0.35 -0.24 -0.31 -0.38 -0.33 1 0.025 0.081 -0.042 0.027 0.065 -0.016 -0.11 0.26 -0.25 -0.17 -0.56
holiday_HOLIDAY 0.045 -0.046 -0.11 0.12 0.019 0.043 0.013 -0.081 0.025 1 -0.22 0.11 -0.11 -0.017 0.048 0.31 -0.4 -0.26 0.16 -0.1
workingday_WORKING DAY 0.1 0.024 0.061 0.012 0.084 -0.099 -0.037 0.051 0.081 -0.22 1 -0.089 0.071 0.077 -0.051 -0.61 0.46 0.42 -0.23 0.0095
weathersit_GOOD -0.056 0.058 -0.64 -0.05 -0.026 0.0061 -0.078 0.12 -0.042 0.11 -0.089 1 -0.97 -0.16 0.018 -0.53 -0.65 -0.63 -0.034 0.16
weathersit_MISTY 0.085 -0.04 0.59 0.024 0.054 -0.026 0.097 -0.11 0.027 -0.11 0.071 -0.97 1 -0.086 -0.1 0.55 0.66 0.57 0.0098 -0.12
weathersit_RAIN/SNOW/STORM -0.12 -0.074 0.23 0.11 -0.11 0.083 -0.076 -0.066 0.065 -0.017 0.077 -0.16 -0.086 1 0.35 -0.063 -0.025 0.28 0.1 -0.2
z1 -0.81 0.24 0.14 0.18 -0.85 -0.34 0.16 0.16 -0.016 0.048 -0.051 0.018 -0.1 0.35 1 -0.043 0.057 -0.13 -0.1 -0.41
z2 -0.0097 -0.03 0.54 -0.24 0.015 0.11 0.081 -0.07 -0.11 0.31 -0.61 -0.53 0.55 -0.063 -0.043 1 0.013 0.073 0.25 -0.088
z3 -0.23 -0.39 0.49 0.082 -0.26 0.0034 0.017 -0.29 0.26 -0.4 0.46 -0.65 0.66 -0.025 0.057 0.013 1 0.38 -0.13 -0.45
z4 0.34 0.39 0.64 -0.028 0.41 -0.0066 0.067 0.19 -0.25 -0.26 0.42 -0.63 0.57 0.28 -0.13 0.073 0.38 1 0.043 0.33
zz1 0.082 -0.037 0.17 -0.026 0.18 0.22 0.012 -0.048 -0.17 0.16 -0.23 -0.034 0.0098 0.1 -0.1 0.25 -0.13 0.043 1 0.098
cnt 0.57 0.63 -0.11 -0.29 0.64 0.033 0.2 0.33 -0.56 -0.1 0.0095 0.16 -0.12 -0.2 -0.41 -0.088 -0.45 0.33 0.098 1
yr
temp
hum
windspeed
days_since_2011
season_FALL
season_SPRING
season_SUMMER
season_WINTER
holiday_HOLIDAY
workingday_WORKING DAY
weathersit_GOOD
weathersit_MISTY
weathersit_RAIN/SNOW/STORM
z1
z2
z3
z4
zz1
cnt
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75

Figure A.21.

A.4. A note on feature importance

A crucial aspect of interpreting machine learning models involves identifying critical features that significantly contribute to model performance. Evaluating feature importance becomes challenging when data features are correlated, a phenomenon known as multicollinearity. Multicollinearity can give rise to several issues, such as inflated feature importance and unstable feature selection. Many real-world datasets exhibit multicollinearity, complicating the interpretation of feature importance. We will demonstrate that even for a simple linear model with perfect correlation (e.g., $X_2 = aX_1 + b$), feature importances can be highly unstable, resulting in vastly different importance values for the same features both theoretically and numerically.

Definition 1 (Linear Regression Feature Importance) *Let β_i represent the i-th weight of the linear regression model. We define the importance of the input X_i as follows:*

$$importance(X_i) = \beta_i$$

Now suppose we have features X_1 and $X_2 = aX_1 + b$ both real-valued. Let $y = \beta_1 X_1 + \beta_2 X_2 + \beta_3$. This would result in a feature importance of β_1 for X_1 and β_2 for X_2. But you can also rewrite y as :

$$\begin{aligned} y &= \beta_1 X_1 + \beta_2 X_2 + \beta_3 \\ &= \beta_1 X_1 + \beta_2 a X_1 + b + \beta_3 \\ &= (\beta_1 + \beta_2 a) X_1 + \beta_3 + \beta_2 b \end{aligned}$$

This time we would have feature importance of $(\beta_1 + \beta_2 a)$ for X_1 and 0 for X_2. It is also possible to distribute the weights differently like this :

$$\begin{aligned} y &= \beta_1 X_1 + \beta_2 X_2 + \beta_3 \\ &= \beta_1 X_1 + \beta_2 a X_1 + b + \beta_3 \\ &= (\beta_1 + \frac{\beta_2}{2} a) X_1 + \frac{\beta_2}{2} X_2 + \beta_3 + \frac{\beta_2}{2} b \end{aligned}$$

This would mean we would have feature importance of $(\beta_1 + \frac{\beta_2}{2} a)$ for X_1 and $\frac{\beta_2}{2}$ for X_2.

In fact we could choose any importance α for X_1 and γ for X_2 if $\alpha + \gamma = \beta_1 + \beta_2$. This demonstrates that theoretically if we have redundant data or features which can be calculated from one another that the magnitude of the importance can be flawed. This is also observed in numerical experiments. For this experiment, we have trained 1000 linear regression models where $X_2 = X_1$ and $y = 3X_1 + 5X_2 + 15$. The results are plotted in figure A.22.

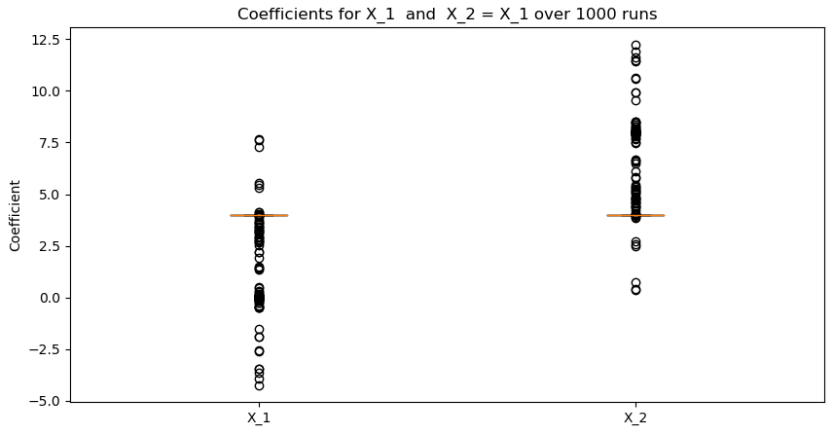


Figure A.22.: In this boxplot, the regression coefficients of X_1 and X_2 are displayed. It is evident that there is a significant variation in the coefficients for some runs, indicating that the importance is shifting from one variable to the other. However, the sum of the importance of both variables remains consistent across all runs.

In real-world scenarios, we often encounter non-identical yet highly correlated data, which leads to comparable effects. Surprisingly, when applying linear regression to data with a high correlation of 0.9, using the same configuration as before, we obtain accurate feature importance with minimal variation. When nonlinearity is introduced to the construction of the response variable, y, the performance worsens, but it remains acceptable. Nevertheless, when using a Random Forest Regressor, the same issue arises with highly correlated data.

In this experiment, we have four variables, each with a mean of 1 and the following correlation matrix:

$$\text{correlation matrix} = \begin{bmatrix} 1 & 0.9 & 0.8 & 0.7 \\ 0.9 & 1 & 0.7 & 0.8 \\ 0.8 & 0.7 & 1 & 0.9 \\ 0.7 & 0.8 & 0.9 & 1 \end{bmatrix}$$

The response variable, y, is defined as $y = 3X_1 + 5X_2 + 2X_3 + 2X_4$. We used 500 data points and 100 runs for this experiment. Since we cannot use regression coefficients, we calculate SHAP values, which also indicate feature importance. The results are illustrated in Figure A.23.

We observe that, in some cases, the importance of X_1 and X_2 can be traded off, as the SHAP value of one decreases while the other increases. The same is true for X_3 and X_4, which are also highly correlated.

As the complexity of models and data increases, it is anticipated that this phenomenon of shifting importance between highly correlated variables will persist, if not pose an even greater challenge when calculating feature importances.

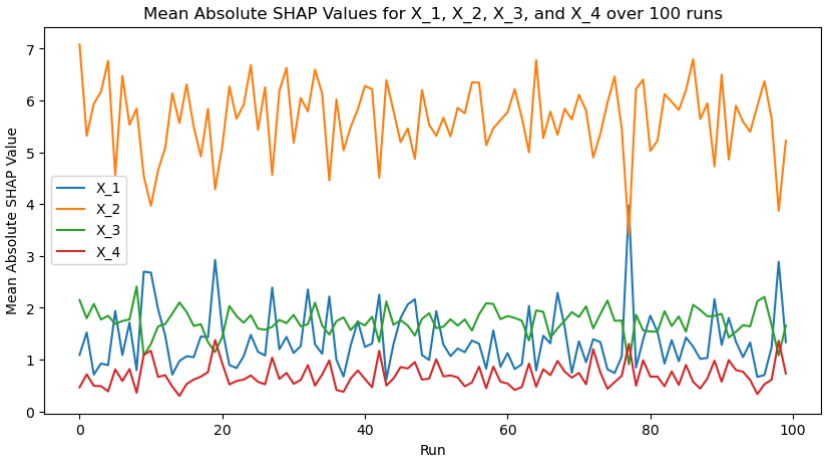


Figure A.23.: The mean SHAP values for each variable are observed over 100 runs. It becomes evident that when X_2 has a low value, X_1 tends to have a high value. This inverse relationship between X_1 and X_2 is particularly noticeable around the 80th run.