Universiteit Leiden
The Netherlands

Opleiding Informatica

Explainable AI in Multi-Objective Design Optimization

Matthijs de Zeeuw

Supervisors:
M.T.M. Emmerich & K. Pereverdieva

BACHELOR THESIS

Leiden Institute of Advanced Computer Science (LIACS)
www.liacs.leidenuniv.nl

25/07/2022

Abstract

Multi-objective optimization solutions hold information, useful to designers, if it could be extracted from the optimization data. However, there is no straightforward way to apply existing eXplainable AI (XAI) methods to optimization data. The aim of this thesis is to find out how existing XAI methods can be applied to multi-objective optimization problems in design optimization. Classically XAI methods are used to augment prediction models with nontransparent 'black-box' machine learning methods. In this thesis the XAI methods are applied to black box optimization, in order to explain the output of the multi-objective optimization algorithm, which is an approximation of the Pareto front and the efficient set. This is accomplished in an indirect way. First an optimization data set is obtained. This data set is then transformed into a classification data set. The classes are given by certain regions on the Pareto front or in the objective space and the features are the decision variables. A random forest classifier is then trained on the classification data set. In the study four different XAI methods are used to explain the classifier. The tested XAI methods are SHAP, LIME, partial dependence plots, and permutation feature importance. Two simple multi-objective design optimization problems serve to demonstrate the usefulness of the approach: a two-objective welded beam design problem and a two-bar truss design problem. The SHAP method and partial dependence plots were found to be most informative when looking for decision rules for the optimization problem. Furthermore, when comparing the results of the different XAI methods it was found that there is no consensus among the methods in ranking features by importance, which requires further investigations to be fully understood.

Contents

1 Introduction

Advances in the field of AI have resulted in new and powerful algorithms that can solve complicated classification, prediction, search, and optimization problems. Because most of the more powerful AI algorithms lack transparency, the need for eXplainable AI (XAI) algorithms has become more pressing. Though various methods have been developed for classification and prediction models, there are few methods for explaining optimization data. There are many different types of optimization problems and there might be differences in the kind of information that is of interest with regards to such problems. The focus of this thesis is on multi-objective optimization in structural design problems. In design optimization, XAI methods could be used to find design rules. Design rules are relationships between variables and objectives that hold true even if the parameters of the problem changes [1]. A few methods have already been introduced. Deb and Srinivasan et al. [2] introduced "innovization" as a method for gaining a better understanding of a problem by using a multi-objective optimization algorithm and analysing the resulting data. This method was extended by Bandaru [3] to make it fully automated. Blom [1] used both, box plots, and decision trees, to learn heuristic rules for a design problem. This was done by taking an optimization data set and defining different classes. The box plots visualize the distribution of the data for each feature using a box for each class. The box plots were able to show which features were the most important for telling apart the different classes. The decision trees were trained on descriptive features that were predefined by domain experts to classify the data set by learning rules that determine to what class a solution belongs depending on the predefined features. However, so far, the common XAI techniques that have been proposed for prediction in the AI literature, have not yet been transferred to applications of multi-objective optimization. The aim of this thesis is to close this gap and find out how existing XAI methods meant for augmenting black-box classification and prediction methods can be used to explain multi-objective optimization results in black-box design optimization.

Running an XAI algorithm on multi-objective optimization data is not straightforward. XAI methods are meant to explain predictions or classifications made with machine learning models and usually require an estimator or model to be provided. However, optimization methods are not estimation models. There are two ways to work around this problem. The first option would be to somehow use the objective functions instead of the estimator. Since the objective functions determine how optimal the solutions are, explaining the objective functions would provide the required information. However, to achieve this a way must be found to pass the objective functions to the XAI method. The other approach would be to divide the optimized data set into different classes based on the type of solution (not-optimal, optimal in all objectives, etc.). A classifier is then trained on the data set. This way it is possible to get an indirect explanation of the optimization data by explaining the classifier. This last approach is easier to implement and therefore a better suited to the purposes of this project.

Research questions:

1. How can we transfer XAI from prediction to multi-objective black-box optimization?
2. To what extent do explanations of the different XAI methods agree with one another?
3. Which methods are most informative with regards to understanding the optimization data in

typical design optimization problems?

2 Definitions

2.1 XAI

XAI (eXplainable Artificial Intelligence) refers to methods that enable humans to understand the decision-making process of AI models. XAI methods can either be model-based or post-hoc. Model-based XAI refers to AI models that are built to be transparent, such as decision trees. Post-hoc XAI methods receive a trained and tested AI model and try to explain their inner workings [4]. Model-agnostic methods are a set of post-hoc XAI methods that can be used on any machine learning method, regardless of its inner workings [5].

2.2 Objective functions

Objective functions are functions that are applied over the features of a problem and evaluate the solution for a certain criterion [6]. In multi-objective optimization the optimal solutions are found by maximizing or minimizing the objective functions.

2.3 Constraints

Constraints in optimization are logical conditions that every solution to the optimization problem must satisfy. Constraints in optimization problems are usually implemented as penalty functions which penalize solutions that violate these constraints.

2.4 Pareto dominance

A solution $u \in 1..., n$ is dominated by $v \in 1, ..., n$ if and only if:

$$\forall i \in \{1, ..., n\} : u_i \leq v_i \wedge \exists i \in \{1, ..., n\} : u_i < v_i$$

[7] In words, a solution is said to dominate another solution when that solution is better than the other solution for at least one objective and not worse for any other objective [6].

2.5 Pareto front

A pareto front as the set of all dominant solutions.

2.6 Features, decision variables, and data points

Decision variables are the variables in an optimization problem for which an optimal value needs to be found. Features are the independent variables used by the model. In this case the classifier is trained on the decision variables, so features and decision variables refer to the same variables. Data points are the instances of the data set. In optimization data each solution is a data point that consists of multiple variables.

3 Methods

There is a wide range of XAI methods. Because the aim of this thesis is to give a prove of concept, only a selection of XAI techniques will be run on the test problems. As mentioned before the XAI methods will be used to explicate a classifier, trained on optimization data. For the experiments a multi-label random forest algorithm was used. Explaining a random forest requires post-hoc methods. Post-hoc methods can be divided in three classes. There are methods that try to explain a model using visualization, methods that use feature relevance, and models that use simplification [5]. Partial dependence plots, or PDPs are a way of explaining by visualization. PDPs are a good option to consider for this type of XAI because the partial dependence plot is a method that is commonly used [8] [9] and are relatively easy to implement. The most well-known technique that uses feature relevance is SHAP. For the third group LIME is an often-used method. Besides these three methods which are representative of the three groups in model-agnostic post-hoc XAI there is another method. This method, permutation feature importance, also explains using feature relevance but is simple to implement and easy to understand, which makes it an ideal method to start with.

3.1 Permutation Feature Importance

Permutation Feature Importance (PFI) is one of the easiest ways of ascertaining the relevance of features. The permutation importance of a feature X_i is determined by calculating the difference in prediction error of a model for the original data set and a data set where the relation of feature X_i with target Y is broken by randomly permuting the values of that feature [10]. Permutation Importance was first introduced by Breiman for the random forest method. [11], and was later generalized to a model-agnostic method [12]. The algorithm for calculating permutation feature importance that was implemented for this experiment is based on the paper written by Fisher, Rudin, and Dominici [10]. The algorithm works as follows. First the prediction error $e_{original}$ for the original data is calculated. Then, for each feature X_i in the data set, X_i is permuted by randomly shuffling the values. The estimator is tested on the permuted data set to determine the prediction error $e_{permuted}$. Based on $e_{original}$ and $e_permuted$ the permutation importance is calculated. The permutation importance p is given by:

$$p = \frac{e_{permuted}}{e_{original}} \tag{1}$$

These last three steps are repeated K times for each feature, resulting in K permutation importance values per feature. The resulting permutation importance values are then visualized using a box plot. The algorithm uses the mean squared error as a measure of the prediction error. It uses the mean_squared_error function from the Sklearn.metrics module which defines it as:

$$MSE(y, \hat{y}) = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} (y_i - \hat{y}_i)^2 \text{ [13]} \tag{2}$$

For the experiments the algorithm used $K = 100$.

3.2 Partial Dependence Plots

Partial Dependence Plots or PDPs depict the target response of a model over the range of possible input values for a certain feature. [14]. The target response of the model is given by the partial dependence value. Let x_s be the set of features of interest and x_c the complement of x_s, then f is given by:

$$f_{x_s} = \mathbf{E}_{x_c}[f(x_s, x_c)] = \int f(x_s, x_c) dP(x_c) \text{ [13]} \tag{3}$$

where $f(x_s, x_c)$ is the predict function and $dP(x_c)$ is the marginal distribution of x_c. Each subset of features in S has its own partial dependence function f which gives the average value of function f for the fixed subset of features of x_s when x_c is varied over $dP(x_c)$. The Sklearn.inspection module provides a function for plotting independence plots [13]. It approximates the function f by averaging over the different values of x_c for a fixed subset of x_s.

$$\hat{f}_s = \frac{1}{N} \sum_{i=1}^{N} \hat{f}(x_s, x_{c_i}) \tag{4}$$

Where $\hat{f}$ is the estimate of the true model. For multi-class classification it is necessary to plot a line for each different class, showing how the prediction probability changes when the feature x_s is varied.

3.3 SHAP

SHAP or SHapley Additive Explanations is an explaining method that uses shapley values. Shapley values are a part of game theory [15]. Shapley values were a mathematically fair way to divide the payoff of a game among the players based on how much each player contributed. A shapley value is constructed by estimating the marginal contribution of one player to a subset of the other players in the game. The shapley value is the weighted aggregate of the marginal contributions of the player for all possible subsets of players [16]. The SHAP algorithm uses these shapley values to explain machine learning methods where they reflect the contribution of a particular feature to the resulting prediction or classification. The SHAP package for python provides the TreeExplainer module [17] which was used in this experiment. The SHAP TreeExplainer method can be used to find local as well as global explanations. The TreeExplainer combines many local explanations to create global explanations of a model. In this way the explanations can remain locally faithful whilst at the same time showing larger global patterns [17]. The SHAP library offers different ways to visualize model explanations. For this experiment only summary plots which give global explanations were used. SHAP summary plots display the shapley value for each instance in the data set as a point. This is done separately for each feature. The value of the data point via a color scale. This way the impact of a feature on the model does not need to be condensed to a single value [17]. SHAP summary plots can show the importance of the separate features, the relationship between the value of the feature and its impact on the model and the distribution of feature-values, all in one plot. Force plots explain a single prediction. These plots show for each feature how much it contributed to the final decision.

3.4 LIME

LIME or Local Interpretable Model-Agnostic Explanation explicates a model by fitting an interpretable model to the model that needs to be explained. If the model that needs to be explained is complicated, it becomes very difficult to fit an interpretable model globally. The LIME method solves this by considering a single data point and fit a simple linear model that can give a local explanation of the model at that point [18]. Let the model to be explained be denoted as f and the explanation as $g \in G$, where g is an interpretable model and G is the set of all possible interpretable explanation models. The LIME method tries to find a model $g \in G$ such that it is both locally faithful to f and not too complicated. The complexity of the model is given by $\Omega(g)$ and the unfaithfulness of g to f is defined as $\mathcal{L}(f, g, \pi_x)$ where π_x is a measure of the proximity around instance x, the instance to be explained. LIME tries to find a minimum for the sun of $\Omega(g)$ and $\mathcal{L}(f, g, \pi_x)$ [19]. The LIME method first transforms x to a more interpretable representation $x^{'}$. This can be useful if the way the data is represented to the AI model is not easy to understand. Next LIME creates new data points $z^{'}$ around the instance $x^{'}$ weighted by π_x (instances nearer x get a higher weight). Each instance $z^{'}$ is then transformed back to the original representation z and labelled by using the original model $f(z)$. This creates a new data set $\mathcal{Z}$ containing the labelled points z. Using the new labelled data LIME fits a linear function g using a weighted loss function:

$$\mathcal{L}(f, g, \pi_x) = \sum_{z, z^{'} \in \mathcal{Z}} \pi_x (f(z) - g(z^{'}))^2 \tag{5}$$

This way g is trained on the more interpretable representation $z^{'}$. By weighting the loss function, the instances closer to instance x are considered more important [19]. The output of the LIME method consists of three parts: the prediction probabilities, the feature probabilities, and the feature values [20]. The prediction probabilities tables show (in the case of a multi-class classifier) the decision probability for each class. The feature probabilities show how much each feature contributed to the decision [20]. For a multi-class classifier, the feature probabilities are visualized per class. The feature values table displays the actual values of the instance that has been explained.

4 Experiments

The first step is to generate the data set that is to be explained. This is done by running an optimization algorithm on an implementation of a multi-objective optimization problem. The implementation was done in python using the DESDEO package [21]. The package also offers a solver (solve_pareto_front_representation) which returns a pareto front optimized according to the specified constraints and objective functions. The resulting data set contains both the feature values and the objective values. Because the pareto front contains no non-optimal solutions, these need to be generated separately. This is done by generating several solutions with random values for each feature within the upper and lower bounds of each variable. The objective values are obtained using the objective functions. The resulting feature values and objective values are then appended to the optimization data set.

The second step is transforming the data set into a classification data set. For a problem with two objectives, like the welded beam problem, there are four classes: the knee point area (KP) with solution optimal for all objectives, the optimal points for the first objective ($F1$), the optimal

points for the second objective ($F2$) and the non-optimal or bad solutions (BD). The classification data set is obtained by selecting a group of solutions for each class based on the objective values. Next, a classifier is trained on the classification data set. Since the optimization data sets used for the experiments are relatively small (less then 100 instances in the case of the welded beam data set), a type of classifier is needed that is not easily influenced by outliers. The classifier must also be robust against overfitting since the classifier needs to find general rules about solutions to the optimization problem. Random forest seems to be a good choice for such a problem. Random forest classifiers base the classification on the outputs of a group of decision trees. Each tree is trained on a bootstrap sample of the training data [22]. Random forest classifiers are robust to outliers and not prone to overfitting [11]. The SKLEARN package [13] provides a random forest classifier for python. The classifier has several parameters. For most parameters the default settings were used. Only for *n_estimators* (the number of trees) and *max_depth* (the maximal depth of the decision trees) a grid search was preformed to find the optimal settings. The parameter grid had three settings for each parameter. *max_depth* had a default value of 100. Beside this default values, also a smaller depth (50), and a greater depth (150) where tried. For max_depth the default setting was *none*, so a few different selections of parameter setting where tried. After 2000 there was no significant effect on the score, so the grid for the grid search used the setting 50, 100 and 150 for *n_estimators* and 100,1000 and 2000 for *max_depth*.
Based on the results of the grid search, *n_estimators* was set to 100 and *max_depth* to 1000.
Lastly, the four XAI methods where run on the test data to explain the fitted random forest classifier.

4.1 Welded beam problem

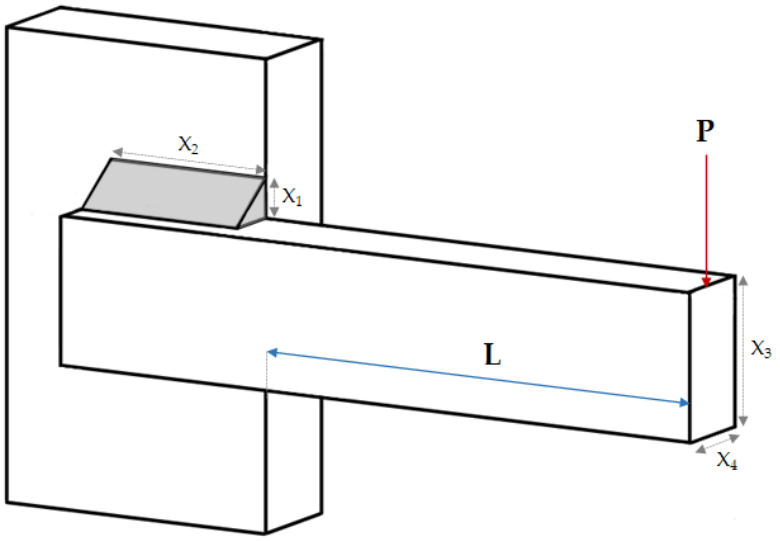

Figure 1: Visualization of the welded beam problem

Figure 4.1 shows a visualization of the welded beam problem. The welded beam problem tries

to optimize the design of a beam that is fixed on one end and has a concentrated load P at the opposite end. The parameter L gives the length of beam that sticks out. The problem is to find a beam design that is strong enough whilst minimizing the material costs. The problem has four variables: the height and breadth of the welding material (x_1, x_2) and the height and breath of the beam (x_3, x_4) and two objectives to be minimized. These two objectives, V_{weld} (the volume of the welding material) and V_{bar} (the volume of the beam) are given by:

$$V_{weld} = {x_1}^2 * x_2 \tag{6}$$

$$V_{bar} = x_3 * x_4 * (L + x_2) \tag{7}$$

which are both related to material cost.

4.2 Results

4.2.1 Feature Permutation Importance

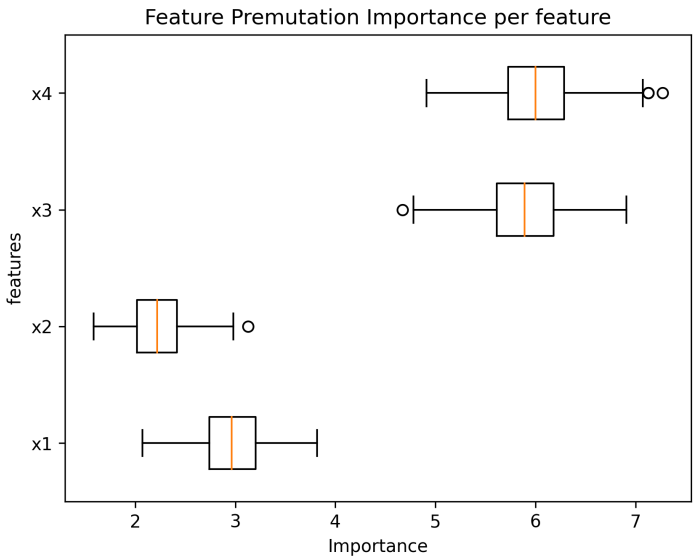

Figure 2: Box plot of the feature permutation importance for each feature of the welded beam problem.

Figure 2 shows the feature permutation importance for each feature in the welded beam data set. For each feature the permutation importance has been averaged over 100 repetitions. Because the permutation importance is calculated by dividing the error of the permuted data set by the original error (see definition 1), an importance value of 1.0 would mean that the estimator preformed equally well for the permuted data set. Any value higher then 1.0 would mean that the estimator preformed worse. The highest mean importance value is about 6.0 for feature x_4. All features have a mean feature permutation importance of more the 2.0. However, the importance values of features x_3 and x_4 are significantly higher than the importance values of x_1 and x_2.

4.2.2 Partial Dependence Plot

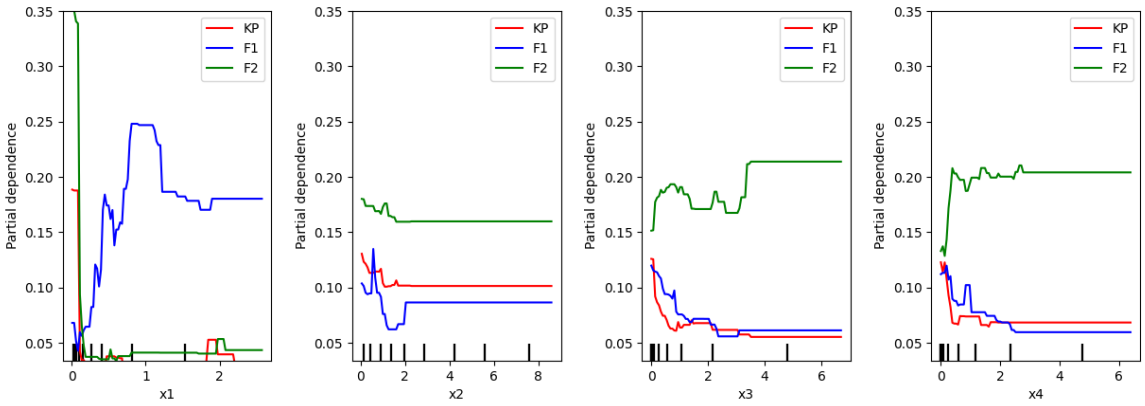

Figure 3: Shows the partial dependence for each feature, for classes KP, $F1$, $F2$.

A partial dependence plot denotes the change in the estimated prediction value for a target when a feature is varied over its marginal distribution [14]. Thus, partial dependence plots differ significantly from feature permutation importance plots which show the impact of features on the performance of the entire model. Figure 3 shows the partial dependence for classes KP, $F1$ and $F2$. For class $F2$, there seems to be and inverse relationship with regards to variables $x1$ and $x2$ and a positive dependence for x_3 and x_4. For class KP there is an inverse relationship for all four of the variables. There is a non-linear relationship between feature x_1 and class $F1$ and also between x_2 and class $F1$. For feature x_3 and X_4 the dependence seems to be inverse. When the value of x_1 becomes higher then $x_1 \approx 0.2$ it stops having a large impact on the partial dependence for classes KP and $F2$. Feature x_2 stops having an impact on the partial dependence value of any class when $x_2 > 2$. For values higher than 3 features x_3 and x_4 do not have an impact anymore.

4.2.3 SHAP

SHAP values reflect the contribution of each feature to the final prediction of an estimator. Figure 4 shows the mean of the absolute impact of each feature on the model output. For almost all classes x_1 seems to be the most important feature. Only for class BD is x_4 the feature that contributes the most not x_1. Table 1 shows the ranking of features based on relevance. The ranking is not entirely

PFI	SHAP
x_4	x_1
x_3	x_4
x_1	x_3
x_2	x_2

Table 1: Shows the ranking of features based on there relevance, for both the PFI method (figure 2 and the average absolute shap value (figure 4)

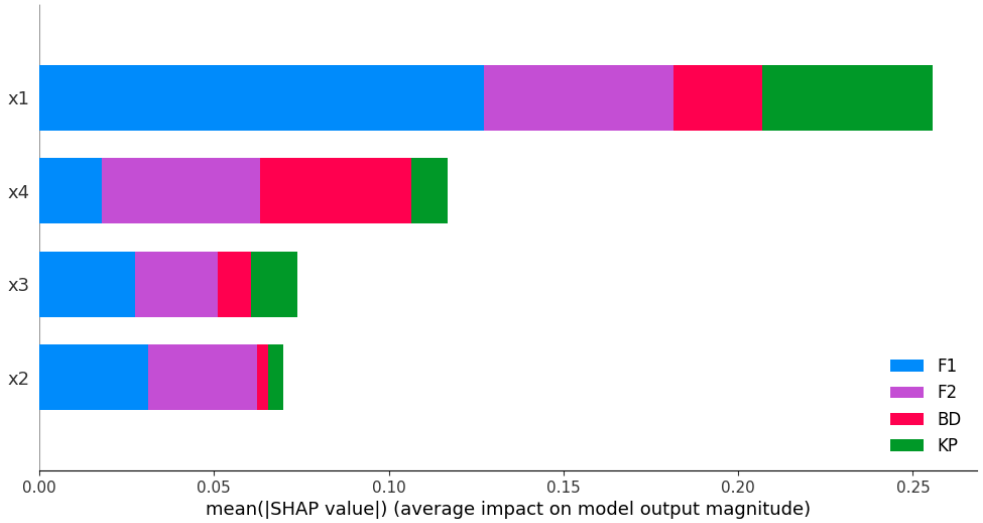

Figure 4: The mean impact on the magnitude of the model output.

the same. The most relevant feature according to the permutation importance is x_4, whereas x_1 has the highest absolute average SHAP value. Both methods consider x_4 more relevant then x_3. They also agree that x_2 is the least relevant feature. Figure 5 shows the SHAP values for each class separately. Contrary to figure 4, these summary plots show not only the magnitude, but also the direction and concentration of the effect of each feature on the model [17]. As already shown in figure 4, feature x_1 is the most important feature for predicting class KP. Figure 5(a) shows that the impact on the model output is negative for high values of x_1. A low value for x_1 seems to have often a large positive impact but not always. x_3 and x_4 also have a negative impact on the model output for high values. For x_2 high values seem to have almost no impact, whereas low values have either a slight positive or slight negative impact on the model. For class $F1$ the most important feature is also x_1 as can be seen in figure 5(b). x_1 has the opposite effect on class F1 compared to class KP (fig. 5(a)). Higher values have a positive impact on the classifier. The same is true for x_3 and x_4 though on a smaller scale. High values of x_2 have a higher, decidedly negative impact compared with its impact on class KP, though still relatively small. Figure 5(c) shows the SHAP values for class $F2$. For feature x_1 high values have almost no impact and medium values have a slight negative impact. For low values it is hard to say in what way they effect the model. For x_4 high values have a negative impact on the model and most instances with lower values have a positive impact. x_2 is the only feature that positively impact the model output for class $F2$ for all others high values have a negative impact. Figure 5(d) shows that for class BD feature x_4 is most important. For x_1 high values have a negative impact and for x_3 a positive impact. x_2 only makes a very small contribution. For high values the impact is on average slightly negative.

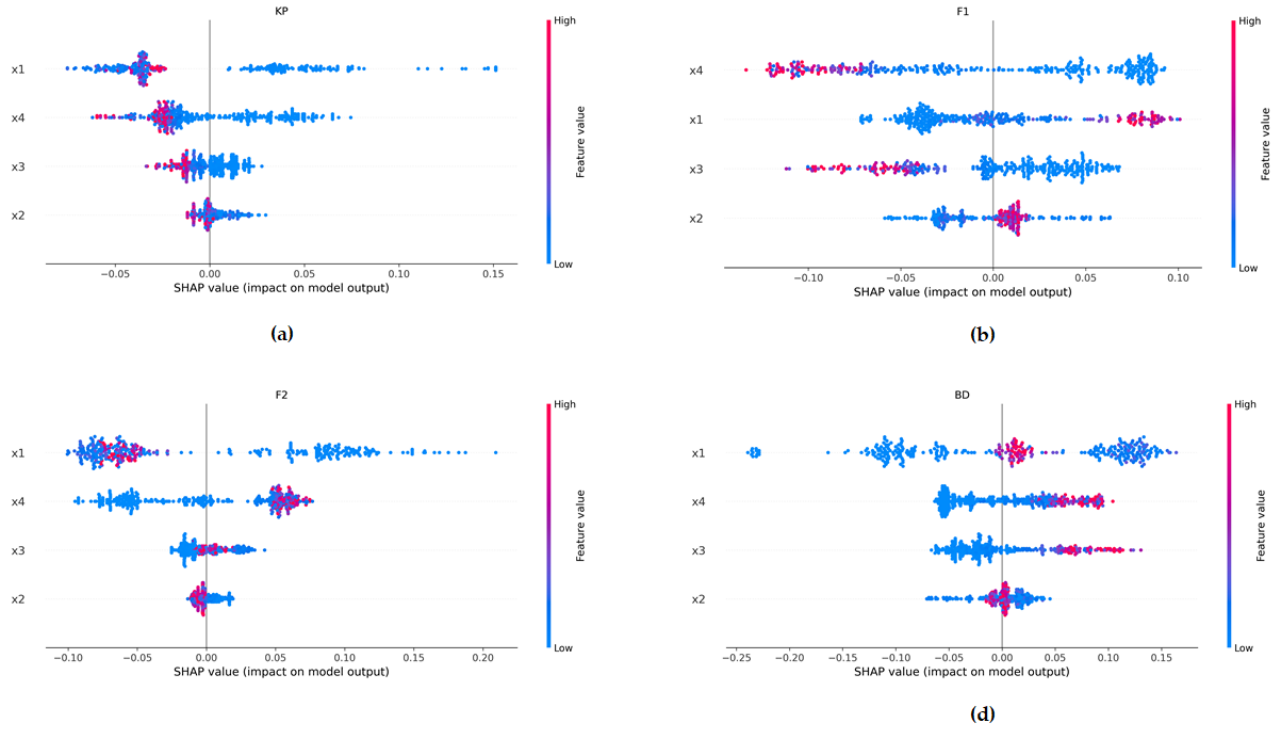

Figure 5: Shows the SHAP values of all instances for class KP.

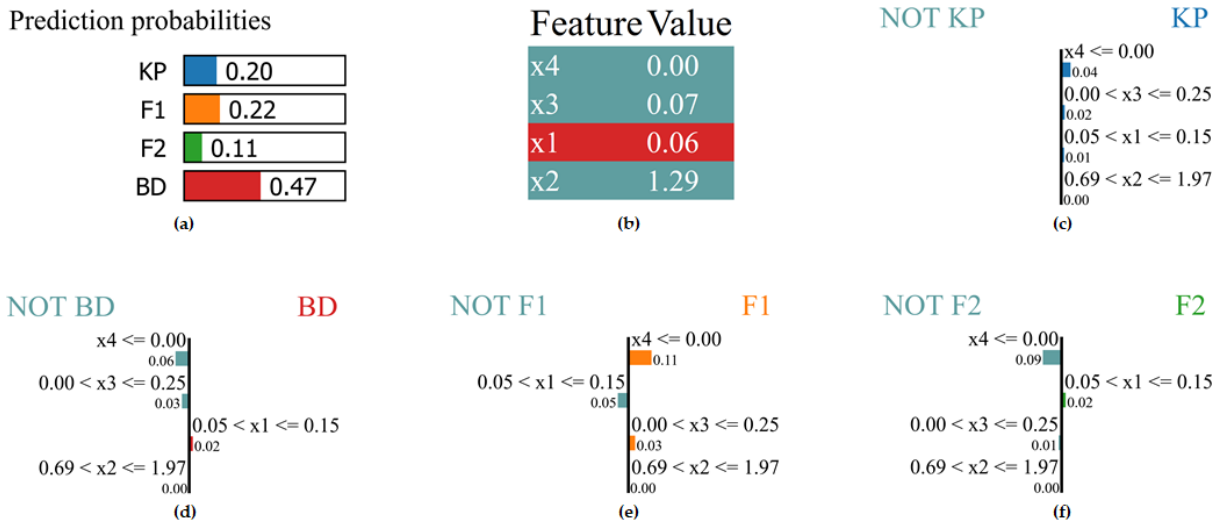

Figure 6: The LIME output for one instance in the welded beam data set. The top-left figure shows the prediction values and top-right figure the actual values of the instance. At the bottom are the feature probabilities denoting the feature importance for each class.

4.2.4 LIME

The output of the LIME method as shown in figure 6 explains an instance in the optimization data set. 6(a) shows the prediction probabilities for this instance. The model predicted class BD. The feature values (fig. 6(b)) show that the values for each feature are low. For class KP the low values all had a positive impact on the classification. This seems to correspond with the general trend shown in the SHAP summary plot for class KP (fig. 5(a)). For class $F1$ (fig. 6(e)) only x_1 has a negative impact on the classification. The SHAP summary plot (fig. 5(b)) shows that x_1 also has a negative impact on the model globally. For feature x_2 the SHAP plot shows that low values have generally a negative impact, which seems to contradict the LIME explanation. However not all instances with low values follow the global trend for x_2 meaning that sometimes a low value for x_2 can have a positive impact. For the other classes the LIME and SHAP plots seem to agree.

4.3 Two-bar truss problem

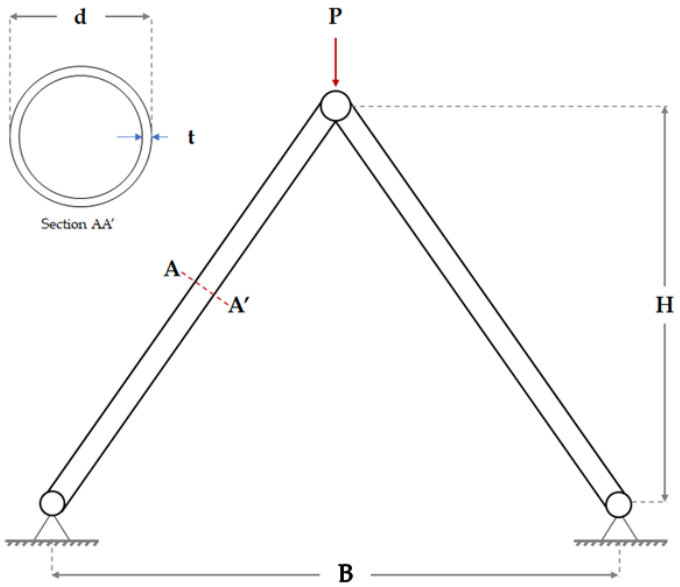

Figure 7: Visualization of the two-bar truss problem

The two-bar truss problem was originally introduced by Fox [23]. It has six variables that need to be optimized and four objectives. The variables are height (H), diameter (d), thickness (t), separation distance (B), modulus of elasticity (E), and material density (p). The objectives are weight, stress and buckling stress, calculated by the following formulas.

$$weight = p * 2 * \pi * d * t\sqrt{\left(\frac{B}{2}\right)^2 + H^2} \tag{8}$$

$$stress = \frac{p * \sqrt{\left(\frac{B}{2}\right)^2 + H^2}}{2 * t * \pi * d * H} \tag{9}$$

$$bucklingstress = \frac{\pi^2 E(d^2 + t^2)}{8[\left(\frac{B}{2}\right)^2 + H^2]} \tag{10}$$

The fourth objective, deflection was not used for this experiment. The two-bar truss problem is a bit more complicated than the welded beam problem, having six variables and three objectives. This means that the classification problem has now eight classes instead of four (KP, $F1$, $F2$, $F3$, $F12$, $F13$, $F23$).

4.4 Results

4.4.1 Feature Permutation Importance

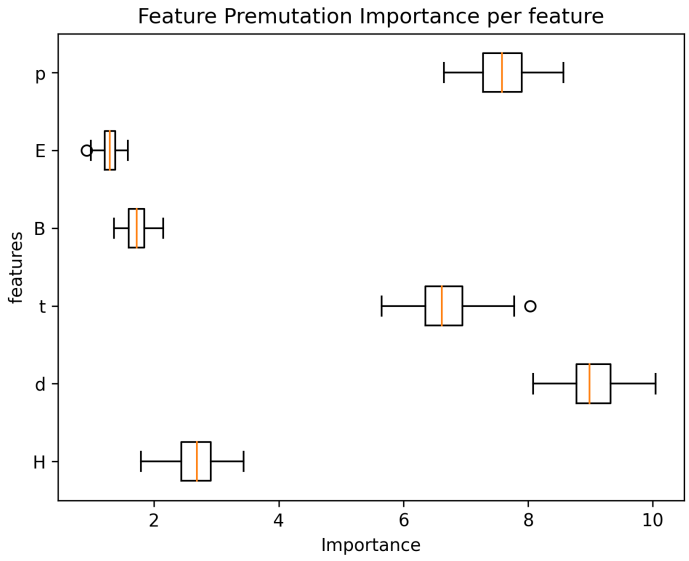

Figure 8: The permutation importance for each variable of the two-bar truss problem

When looking at figure 8 it is apparent that the three variables p, t, and d have a significantly higher importance than the other three features. Features E and B are the least important with importance values of less than two.

4.4.2 Partial Dependence Plot

Figure 9 shows that there is a monotonic inverse relationship between H and class KP. The relationship between H and $F3$ is also inverse, though not monotonic. For the other two classes, H seems to have a slight linear dependence, until $H > 5000$. There is a polynomial relationship between feature d and class $F1$. Feature t has a non-linear relationship with regards to class $F1$ and $F3$ and an inverse relationship for $F2$ and KP. There is a positive relationship between feature B and class $F1$ and $F2$. There is an inverse linear relationship between feature B and classes KP and $F3$. E has virtually no influence on any class for $E > 45000$. For p both class KP and $F3$ have an inverse polynomial relationship with that feature. The relationship between $F1$ and p is

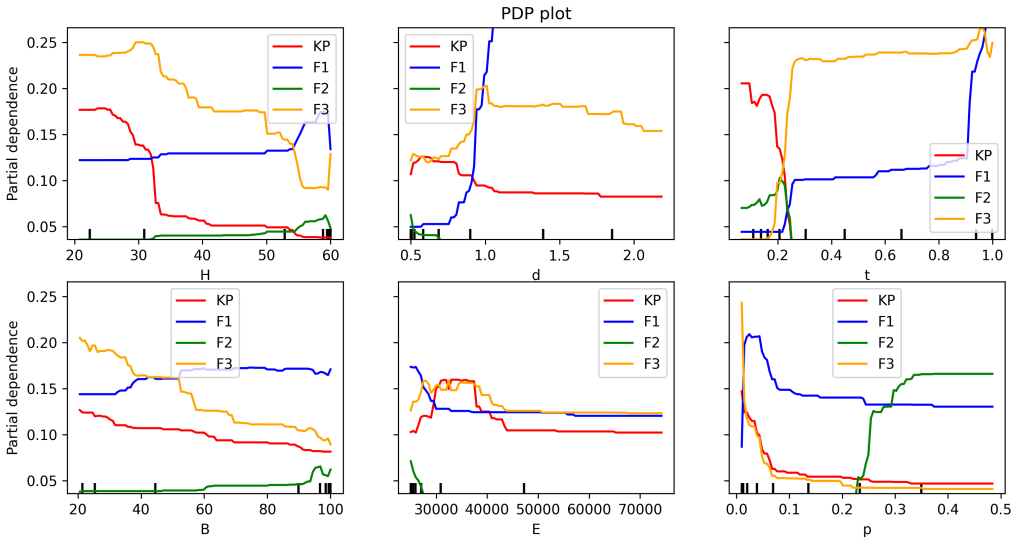

Figure 9: Partial dependence for each feature of the two-bar truss problem. Only the partial dependence of classes KP, $F1$, $F2$, and $F3$ are shown

non-linear and inverse, except for very low values of p. Between $F2$ and p there is also a non-linear relationship.

4.4.3 SHAP

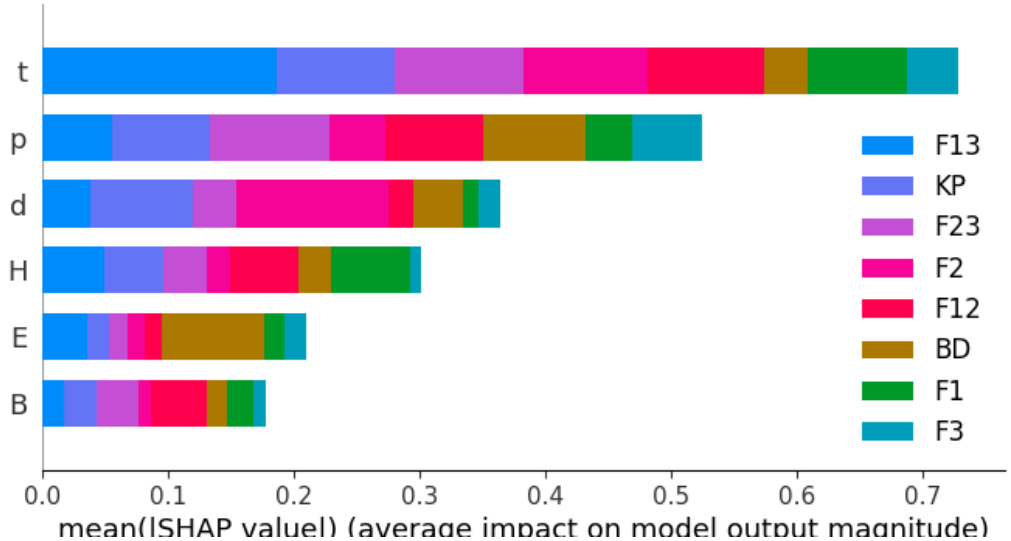

Figure 10: The mean impact on the magnitude of the model output of the welded beam problem

According to the summary plot in figure 10 t is the most important variable when considering all classes together. For class KP the feature t is still the most influential, followed by d and p. For $F1$ features t and H are the most important. $F2$ is impacted the most by features d and t, $F3$ by feature p and t. Feature B is the least influencing feature overall, followed by E. Only for class BD does feature E make a major contribution to the model output. Table 2 compares the ranking of

PFI	SHAP
d	t
p	p
t	d
H	H
B	E
E	B

Table 2: Shows the ranking of features based on there relevance, for both the PFI method (figure 8 and the average absolute shap value (figure 10)

feature based on relevance for the SHAP method (figure 10) and the PFI method (figure 8. Though the ranking is different, the three most important features are p, d, and t. PFI and SHAP agree on the ranking of p and H. Comparing the figure 11 with the global summary plot in figure 10 reveals that E has indeed only a big influence for class BD where it has a positive impact. For all other classes E has a small negative impact. The SHAP values for class KP in figure 11(a) do not all agree with the partial dependence plots in figure 9. Feature H has a positive impact according to the SHAP summary plot but has a negative influence according to the partial dependence plot. Also figure 11(b) shows that all features have a negative impact on the classifier with regards to class $F1$. The partial dependence plots seem to suggest the opposite for feature t and d (fig. 9). For class $F2$ feature d has a positive impact. This is also true for E though the impact is relatively small. For most features the high and low values are not clearly separated, making it hard to say which way the feature influences the model classification. For class $F3$ (fig. 11(d)) the high and low values or more clearly separated. p, B and H have a positive influence and t, d and E a negative one. Because the partial dependence plot in figure 9 only shows the partial dependence for classes KP, $F1$, $F2$, $F3$, the other plots (fig. 11(e-h)) cannot be compared to the partial dependence. Features t and p have the biggest impact for classes $F12$, $F23$, and $F13$. For classes $F12$ and $F13$ feature H has more importance than d. For class $F23$ it is the other way around. This is interesting because H is the second most important feature with regards to class $F1$ and less important for classes $F2$ and $F3$.

4.4.4 LIME

Figure 12(b) shows the actual feature values. Comparing these values to the marginal distributions of the features (fig. 9) shows that d, p, t and E have low feature values and H and B have high feature values. When comparing the feature probabilities for classes KP, $F1$, $F2$, $F3$ in figure 12(a,d,j,k), the top three features that are in the LIME feature probability graphs correspond to the top three features in the SHAP summary plots (fig. 11(a,b,c,d)). Because the LIME explanation is a local explanation it is not to be expected that the feature probabilities correspond to the global trends shown in the SHAP plots in figure 11. However, because all instances are represented as

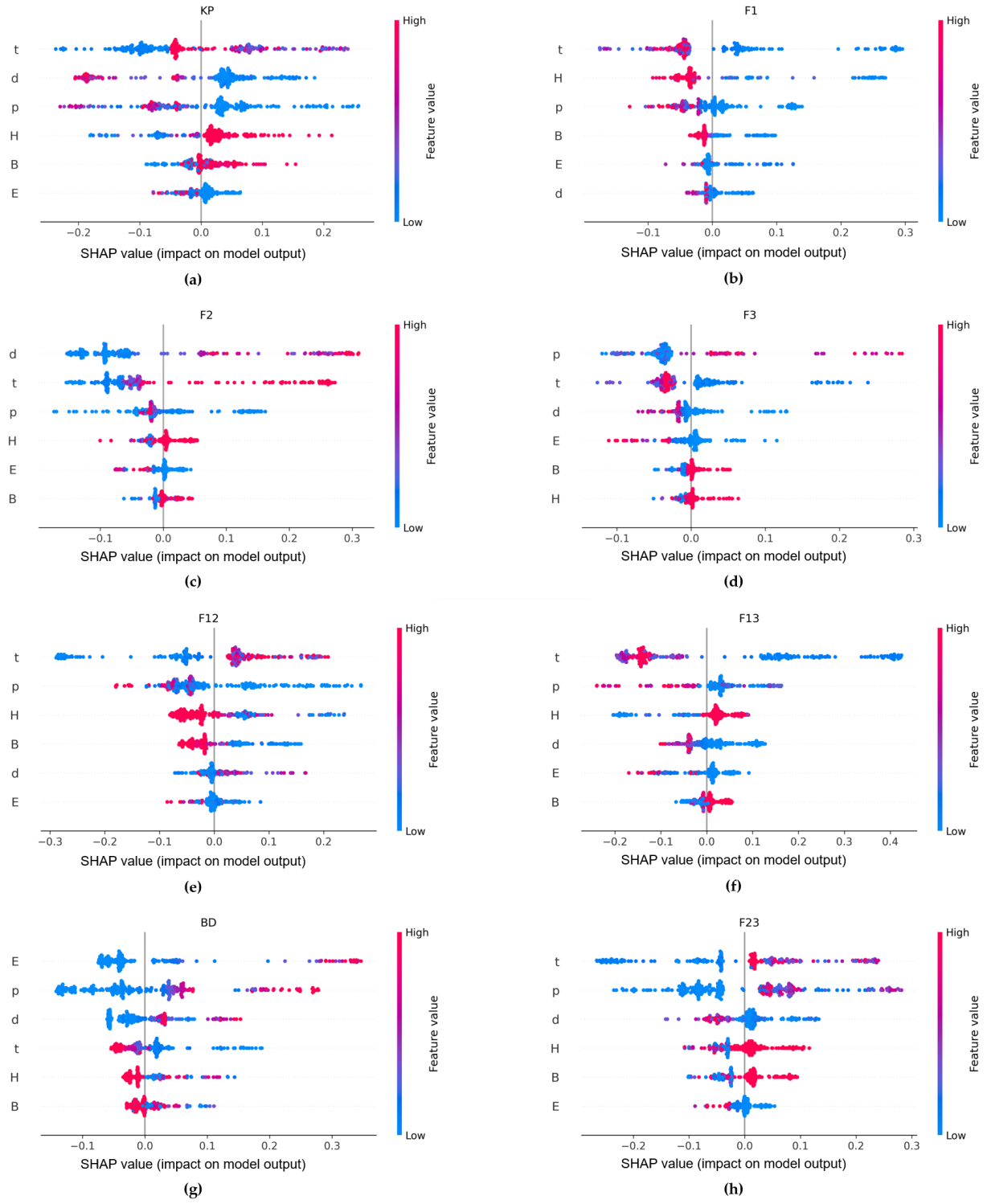

Figure 11: The SHAP values of all instances for each feature of the two-bar truss problem, visualized for each separate class

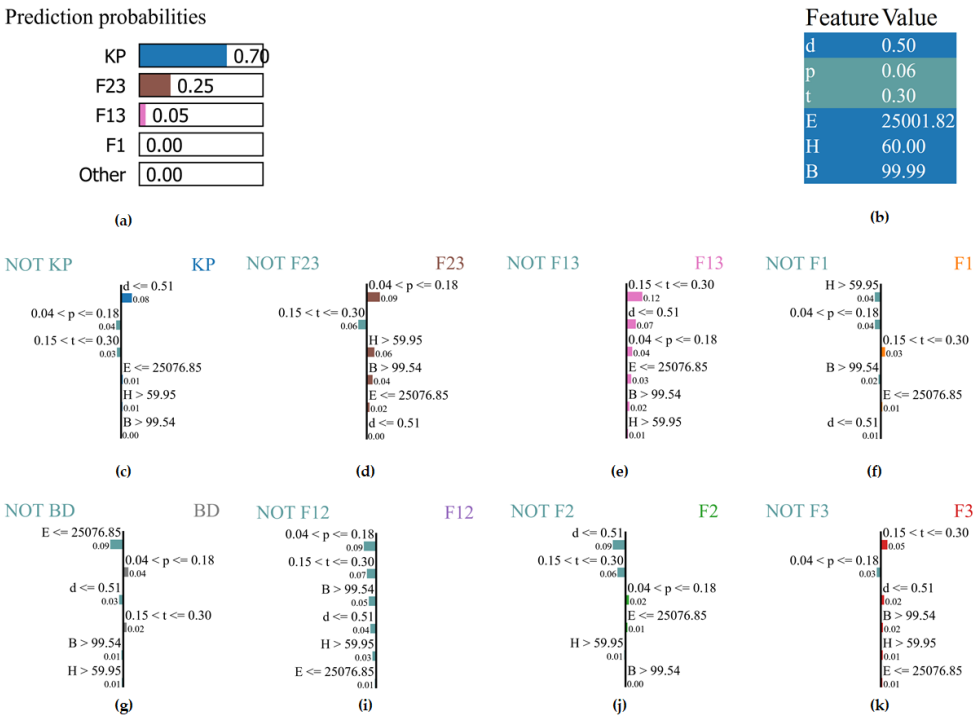

Figure 12: The LIME output for one instance in the two-bar truss data set. The top-left figure shows the prediction values and top-right figure the actual values of the instance. At the bottom are the feature probabilities denoting the feature importance for each class.

dots with a color denoting whether they have a high or a low value, it is possible to see whether any feature probabilities seem to contradict the global SHAP explanation. This does not seem to be the case, though the local LIME explanation does not always correspond to the global trend, shown by the SHAP summary plot(for example, compare figure 11(f) and figure 12(e)).

5 Conclusions and Further Research

As mentioned before the purpose of XAI for multi-objective design optimization is not only to provide a motivation for the solutions the optimization method comes up with. XAI can also be used to discover decision rules. These rules can then be used to create near optimal solutions without having to go through an optimization process again.
On the question of how to transfer XAI from prediction to multi-objective black-box optimization, the indirect approach that was taken seems to provide relevant explanations about the two optimization problems.
By transforming the data into a classification data set and training a classifier on it, there is no need to modify existing XAI methods or to find a way to pass something other than a predictor or classifier to the XAI method. Also, this approach means that any model-agnostic XAI method can be used to explicate the classifier.
The disadvantage of this indirect approach is that no classifier is totally accurate. Because of this there is a chance that not all decision rules derived from the XAI results are actual decision rules for the design problem in general.
For most methods it is not easy to say whether the methods do agree with one another. PFI, PDP, SHAP and LIME methods do not explain the classifier model in the same way. Because of this a one-to-one comparison of all methods is not possible. Still, it is possible to ascertain whether different explanations contradict each other. This is the case for the PFI plots (fig.2,8) and the SHAP global summary plots (fig. 4, 10). Both feature permutation importance and the average absolute SHAP value show which features have the most impact on the model. For both experiments the two methods have contradicting results. This might be due to the fact that each method measures importance in a different way. The PFI method defines importance as the difference in prediction error, whereas the global summary plot of the SHAP method measures the contribution each feature makes to the prediction of the model using the SHAP values.
For both experiments the LIME and SHAP outputs do not contradict each other. However only a small part of the output of the LIME and SHAP algorithms can be compared, since the LIME output is local and the SHAP output is global. It is only possible to check for each feature and class in the SHAP plots (fig. 5, 11), whether instances can be found that have a similar impact on the model as the single instance seems to have according to the feature probabilities in the LIME output.
Not all methods are equally informative with regards to understanding the optimization data in typical design optimization problems. Permutation feature importance values only show the relative importance of a feature and not in what way it influences the model. SHAP plots are more informative, showing not only the type of influence, but also the distribution of the impact for all instances in the data set. The LIME method finds local explanations. This is not useful when trying to find decision rules that hold true for the design problem in general. In this regard, it is less informative then PFI or SHAP. However, the method is very informative when the aim is to

explain why a certain solution is optimal. Whether PDPs are more or less informative the SHAP or PFI is hard to say because PDPs do not explain based on feature relevance. Which XAI methods are most informative dependents on the kind of information that is wanted.
Since the aim of this thesis was to give a prove of concept many areas are left unexplored. There are, for example, various other XAI methods that were not investigated. Also, not all design optimization data have features that can easily be interpreted by humans. In such cases it might be necessary to generate meta-features that are more reflective of the properties of the solution instead of showing the features used by the model. Lastly, research needs to be done to find out how reliable this method is.

References

[1] K. Blom, *Multi-objective mixed-integer evolutionary algorithms for building spatial design*. PhD thesis, Leiden University, 2019.

[2] K. Deb and A. Srinivasan, "Innovization: Innovating design principles through optimization," in *Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pp. 1629–1636, 2006.

[3] S. Bandaru and K. Deb, "Automated innovization for simultaneous discovery of multiple rules in bi-objective problems," in *International Conference on Evolutionary Multi-Criterion Optimization*, pp. 1–15, Springer, 2011.

[4] M. Moradi and M. Samwald, "Post-hoc explanation of black-box classifiers using confident itemsets," *Expert Systems with Applications*, vol. 165, p. 113941, 2021.

[5] A. B. Arrieta, N. Díaz-Rodríguez, J. Del Ser, A. Bennetot, S. Tabik, A. Barbado, S. García, S. Gil-López, D. Molina, R. Benjamins, *et al.*, "Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai," *Information fusion*, vol. 58, pp. 82–115, 2020.

[6] N. Riquelme, C. Von Lücken, and B. Baran, "Performance metrics in multi-objective optimization," in *2015 Latin American computing conference (CLEI)*, pp. 1–11, IEEE, 2015.

[7] M. Emmerich and A. H. Deutz, "A tutorial on multiobjective optimization: fundamentals and evolutionary methods," *Natural computing*, vol. 17, no. 3, pp. 585–609, 2018.

[8] D. P. Green and H. L. Kern, "Modeling heterogeneous treatment effects in large-scale experiments using bayesian additive regression trees," in *The annual summer meeting of the society of political methodology*, pp. 100–110, 2010.

[9] J. Elith, J. R. Leathwick, and T. Hastie, "A working guide to boosted regression trees," *Journal of animal ecology*, vol. 77, no. 4, pp. 802–813, 2008.

[10] S. Nembrini, I. R. König, and M. N. Wright, "The revival of the gini importance?," *Bioinformatics*, vol. 34, no. 21, pp. 3711–3718, 2018.

[11] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[12] A. Fisher, C. Rudin, and F. Dominici, "All models are wrong, but many are useful: Learning a variable's importance by studying an entire class of prediction models simultaneously.," *J. Mach. Learn. Res.*, vol. 20, no. 177, pp. 1–81, 2019.

[13] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[14] A. Goldstein, A. Kapelner, J. Bleich, and E. Pitkin, "Peeking inside the black box: Visualizing statistical learning with plots of individual conditional expectation," *journal of Computational and Graphical Statistics*, vol. 24, no. 1, pp. 44–65, 2015.

[15] A. Messalas, Y. Kanellopoulos, and C. Makris, "Model-agnostic interpretability with shapley values," in *2019 10th International Conference on Information, Intelligence, Systems and Applications (IISA)*, pp. 1–7, IEEE, 2019.

[16] L. Merrick and A. Taly, "The explanation game: Explaining machine learning models using shapley values," in *International Cross-Domain Conference for Machine Learning and Knowledge Extraction*, pp. 17–38, Springer, 2020.

[17] S. M. Lundberg, G. Erion, H. Chen, A. DeGrave, J. M. Prutkin, B. Nair, R. Katz, J. Himmelfarb, N. Bansal, and S.-I. Lee, "From local explanations to global understanding with explainable ai for trees," *Nature Machine Intelligence*, vol. 2, no. 1, pp. 2522–5839, 2020.

[18] D. Garreau and U. Luxburg, "Explaining the explainer: A first theoretical analysis of lime," in *International Conference on Artificial Intelligence and Statistics*, pp. 1287–1296, PMLR, 2020.

[19] M. T. Ribeiro, S. Singh, and C. Guestrin, "" why should i trust you?" explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 1135–1144, 2016.

[20] J. Dieber and S. Kirrane, "Why model why? assessing the strengths and limitations of lime," *arXiv preprint arXiv:2012.00093*, 2020.

[21] G. Misitano, B. S. Saini, B. Afsar, B. Shavazipour, and K. Miettinen, "Desdeo: The modular and open source framework for interactive multiobjective optimization," *IEEE Access*, vol. 9, pp. 148277–148295, 2021.

[22] IEEE, *Application of random forest in predicting fault-prone classes*, 2008.

[23] R. L. Fox, *Optimization methods for engineering design*. Addison-Wesley Publishing Company, 1971.