Warsaw University of Technology

FACULTY OF
MATHEMATICS AND INFORMATION SCIENCE

Bachelor's diploma thesis

in the field of study Data Science

Methods for extraction of interactions from predictive models

Paweł Fijałkowski

student record book number 305706

Mateusz Krzyziński

student record book number 305739

Artur Żółkowski

student record book number 305770

thesis supervisor

dr hab. inż. Przemysław Biecek, prof. uczelni

WARSAW 2023

Abstract

Methods for extraction of interactions from predictive models

Predictive models are frequently used to analyze and forecast outcomes based on input data. However, the non-additive relationships between features in these models can be challenging to interpret, making it difficult to understand the underlying mechanisms driving the predictions. Moreover, they also cause problems and misleading interpretations of the models' explanations obtained by popular explainable artificial intelligence (xAI) methods.

In this thesis, we thoroughly examine the methods for the extraction of feature interactions from predictive models. Extracting interactions between features can significantly increase the interpretability of the model and, in some cases, may also result in improved model performance. We provide a comprehensive review and a Python implementation of the most well-known feature extraction interaction methods, both model-agnostic and model-specific. We also evaluate the effectiveness of selected methods through experiments on synthetic datasets. The benchmark created applies to regression and the often neglected classification tasks. To illustrate the practical applications of the described methods and our implementation, we present an example of a real-world use case. We describe the process of applying the developed techniques to explain models based on stylometric features created by the NASK National Research Institute.

Our work contributes to ongoing efforts to improve the interpretability and transparency of predictive models – aspects that are essential for their responsible use in various applications. It is also a valuable resource for researchers and practitioners seeking to improve the explainability of their models as the created `artemis` Python package is publicly available as open-source software.

Keywords: machine learning, feature interactions, explainable artificial intelligence

Streszczenie

Metody ekstrakcji interakcji z modeli predykcyjnych

Modele predykcyjne są często wykorzystywane do analizy i prognozowania wyników na podstawie danych wejściowych. Jednak nieaddytywne zależności pomiędzy zmiennymi w tych modelach mogą być trudne do zinterpretowania, co utrudnia zrozumienie mechanizmów leżących u podstaw predykcji. Co więcej, powodują one również problemy i mylące interpretacje wyjaśnień modeli uzyskanych za pomocą popularnych metod wyjaśnialnej sztucznej inteligencji (xAI).

W niniejszej rozprawie dokładnie badamy metody ekstrakcji interakcji zmiennych z modeli predykcyjnych. Ekstrakcja interakcji pomiędzy zmiennymi może znacząco zwiększyć interpretowalność modelu, a w niektórych przypadkach może również skutkować poprawą jakości modelu. Przedstawiamy kompleksowy przegląd oraz implementację w języku Python najbardziej znanych metod ekstrakcji interakcji zmiennych, zarówno tych agnostycznych dla modelu, jak i specyficznych dla modelu (wykorzystujących jego strukturę). Oceniamy również skuteczność wybranych metod poprzez eksperymenty na syntetycznych zbiorach danych. Stworzony benchmark dotyczy zadań regresji oraz często zaniedbywanych zadań klasyfikacyjnych. Aby zilustrować praktyczne zastosowanie opisanych metod i naszej implementacji, przedstawiamy przykładowy przypadek użycia w świecie rzeczywistym. Opisujemy proces zastosowania opracowanych technik do wyjaśniania modeli opartych o zmienne stylometryczne stworzonych przez NASK Państwowy Instytut Badawczy.

Nasza praca wnosi wkład w bieżące wysiłki na rzecz poprawy interpretowalności i przejrzystości modeli predykcyjnych – aspektów niezbędnych do ich odpowiedzialnego wykorzystania w różnych zastosowaniach. Jest to również cenne źródło informacji dla badaczy i praktyków poszukujących możliwości poprawy wyjaśnialności swoich modeli, ponieważ stworzony dla języka Python pakiet `artemis` jest publicznie dostępny jako oprogramowanie open-source.

Słowa kluczowe: uczenie maszynowe, interakcje zmiennych, wyjaśnialna sztuczna inteligencja

Contents

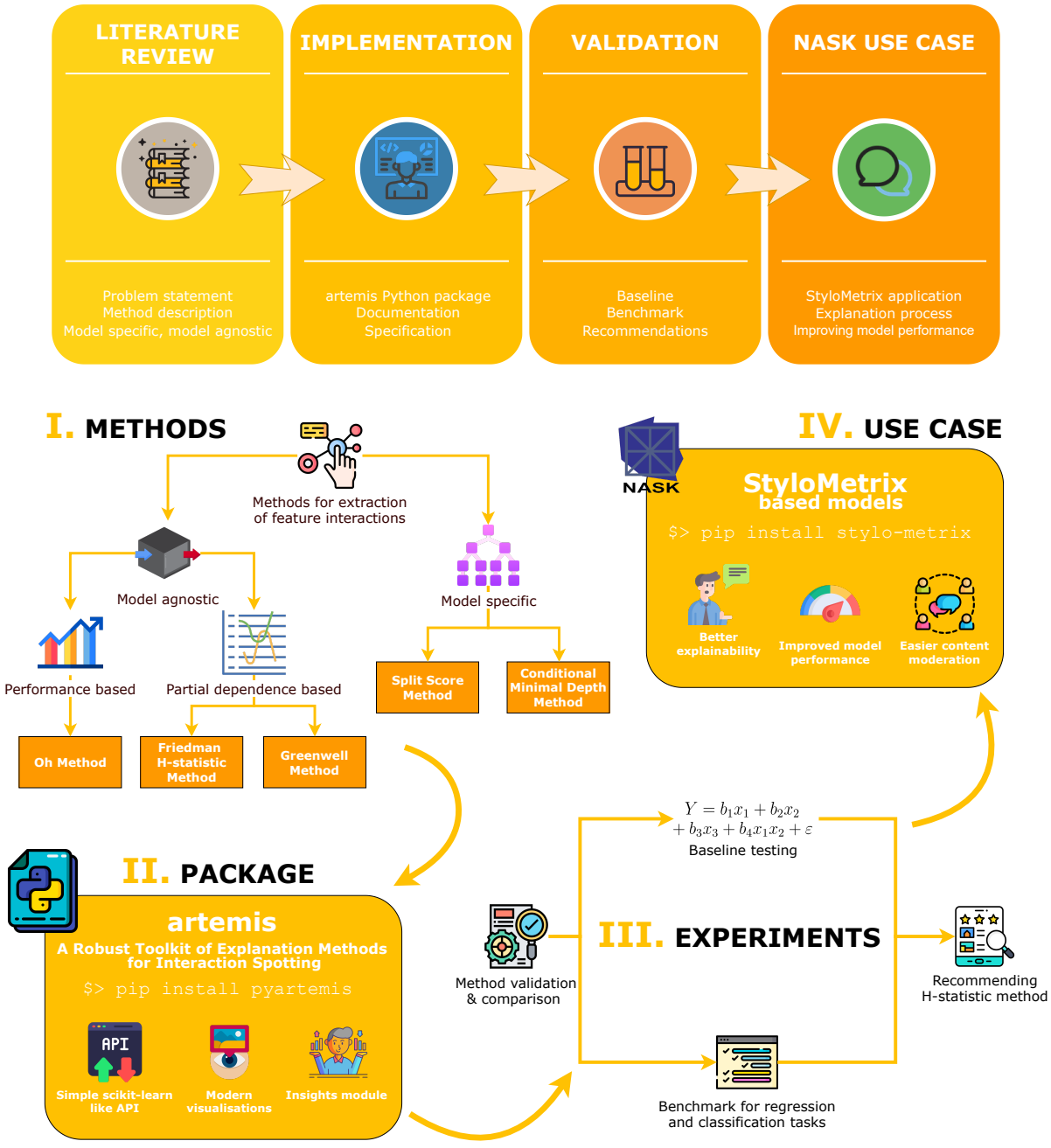

Figure A: Graphical abstract of the presented thesis.

1. Introduction

Predictive models created using machine learning and deep learning techniques are increasingly used in practice, from financial applications [74] to medical diagnostics [71, 75]. Their widespread use is due to their predictive capabilities, as they can outperform humans in many tasks [52, 68]. However, their high performance is often related to complexity and lack of transparency. Such complex modern predictive models are called black-boxes [1]. It becomes increasingly difficult for stakeholders and even machine learning practitioners to understand how they work [32, 61].

Therefore, in recent years, a new direction of research has been advocated related to the creation of explainable artificial intelligence (xAI) methods that can extract useful insights from complex predictive models, help to build trust in them, and enable more effective decision-making [31, 47]. The academic community has proposed many explanation methods and approaches that address the post-hoc interpretability of models. Among these are model-agnostic methods, that can be applied to almost any model, e.g., accumulated local effects (ALE) [4], SHapley Additive exPlanations (SHAP) [49], local interpretable model-agnostic explanations (LIME) [65]. There are also model-specific methods that make use of the model's structure, e.g. Layer-wise relevance propagation (LRP) [6], TreeSHAP [50], and Gradient-weighted class activation mapping (Grad-CAM) [67].

From the user's point of view, it is crucial to be able to use ready-made implementations of methods described in the literature. Some methods are available in open-source packages for the Python [2, 5, 8] and R [12, 54]. For a broader review of explanation methods and the topic of explainability, please refer to reviews in the form of surveys [9, 38] or books [11, 53].

1.1. Motivation

The primary motivation for this thesis is the problems encountered in applying the aforementioned methods to a specific class of classification models based on stylometric features of text data. Models of this type are created by NASK National Research Institute based on Sty-

loMetrix [59]. It is a tool for creating text representations in the form of vectors that quantifies linguistic features of a text. Such features[1] included in the model are considered interpretable, while it is still necessary to see how the models take them into account for predictions.

Although xAI methods and tools appear to be a solution to the problem of interpretability of black-box models, they still face unresolved issues and challenges arising in many dimensions, which were also faced in this use case. Namely, there are concerns related to the socio-technical aspects of using xAI in government [14] or problems relevant from the business and industry perspective [73]. Moreover, there are more general problems and challenges related not so much to the use of the methods but to their operation and conception itself [21, 56].

One of the problems detailed in [56] is **the confusing and misleading interpretations of explanations resulting from interactions between features in the model**. The occurrence of interactions in the model itself can be seen as a positive phenomenon, as it is related to the ability to depict complex relationships between features and the target[2]. However, many explanation methods assume an additive structure of the model (and therefore explanation) and have inherent deficiencies connected with their inability to account for interactions. For example, methods such as partial dependence plots (PDP) [29] and accumulated local effects (ALE) [4], which combine local explanations for individual predictions into a global explanation for the entire model, omit dependencies and interactions between features through the aggregations performed. In addition, methods that decompose predictions, like local interpretable model-agnostic explanations (LIME) [65] and SHapley Additive exPlanations (SHAP) [49], also operate on an additive scheme, assigning attributions to single features, without separating main effects from interaction effects. Similarly, most methods for testing the feature importance, like permutation feature importance (PFI) [26], only assign the importance to a single feature, which in practice is the sum of the importance of that feature and all its interactions with the other features [16].

One potential solution to this problem is not to rely on a single method of explanation, but to juxtapose them with each other, thus sequentially analyzing the model of interest [7]. Another solution is to directly incorporate interactions into explanations or even use methods that extract interactions straight from the predictive model. We decided to rely on the second proposed approach as there are a number of such methods proposed in the literature, e.g., H-statistic [30], partial dependence-based Greenwell method [33], variable interaction network method [39], conditional minimal depth method [44, 60], split score method [45], SHAP interac-

[1] In this thesis, we will use the term *features*, which is synonymous with terms such as *input variables*, *explanatory variables* and *independent variables*.

[2] In this thesis, we will use the term *target*, which is synonymous with terms such as *output variables*, *response variable* and *dependent variable*.

tions method [48], performance-based Oh method [58]. However, two key limitations should be pointed out: (1) many of these methods do not have an implementation and existing implementations have different APIs that are inconsistent with each other, so they can be difficult to use; (2) the methods developed have not been compared with each other, and there is no benchmark of this type of solution.

The present thesis was to be seen as a first step towards responding to the above limitations and a description of how the discussed methods can be applied to real-world problems. Hence, the main goals of this thesis were:

(G1) creating a review of the literature related to feature interactions – a description of selected approaches, their advantages, and disadvantages;

(G2) implementing selected methods in the form of a scientific software package for Python programming language;

(G3) creating a benchmark analysis of implemented techniques to compare their theoretical capabilities for extracting interactions from models solving classification and regression tasks using datasets from literature to check if there are notable differences;

(G4) applying the methods for extraction of interactions to a real-life problem of analyzing models based on StyloMetrix features to present how usage of such explanation techniques contributes to improving the created machine learning process.

1.2. Problem definition

The main issue addressed in this work is the extraction of feature interactions from predictive models. Although interaction is not a new concept, the definition of this term in the literature appears inconsistent and ambiguous, especially comparing different fields where predictive modeling or statistical methods are used [13]. Even in the statistics literature, there has been a long-standing discussion of interactions centred around the three main issues identified by Sir David Cox [20]: (1) the definition and importance of the interaction concept, (2) methods of interaction extraction and detection, (3) clarification of the nature of interactions and their application in modeling practice.

Definition 1.1 (Interaction). The best-constituted definition of an *interaction* is that it occurs when the effect of one feature on the target depends on the value of another feature or features, which can be understood as deviations from additivity in the model of interest [23, 35].

Several approaches can be used to identify interactions in statistical analysis. These include using visualization techniques to explore the data and statistical tests to formally assess the presence of interactions. When dealing with continuous features, any deviation from additivity is considered an interaction, and so to test the presence of the interaction between pair of features, the product of those features is often used. To illustrate, in regression models, a model with response variable Y, explanatory variables X_1, X_2, and error variable ε is additive and has no interactions if:

$$Y = \beta_1 X_1 + \beta_2 X_2 + \varepsilon,$$

while it is non-additive when at least one pair of explanatory variables is bound within non-additive operation. Example of non-additive regression model with interaction term of $\beta_{12} X_1 X_2$ is:

$$Y = \beta_1 X_1 + \beta_2 X_2 + \beta_{12} X_1 X_2 + \varepsilon.$$

Similarly, products of more than two features can represent higher-order interactions. With additional assumptions, two-way analysis of variance (ANOVA) [27] can be used to determine whether the interaction term is significant in the model defined in this way.

For models other than linear regression, other methods of testing the occurrence of interactions are necessary. They are often based on the structure of models of a specific type, and thus the definitions of interactions adopted in these methods may be different than the ones mentioned above. However, another concept often used in the study of interactions, particularly when a model-agnostic analysis is required, is the decomposition of predictions into partial dependence (PD) functions, which also allows for using a similar definition and the examination of deviations from additivity.

1.3. Related work

There are many methods and approaches related to feature interactions in predictive models. Some of them do not directly relate to the explanation of the model of interest but involve the creation of a new one. An example is an approach based on the use of the Additive Groves of Trees algorithm [69, 70], where many models are used only to explain the interactions contained in the data. The obtained results are true to the data, not true to any model [17]. Another technique is based on training a new model called interaction forest [40] that explicitly models interactions. In contrast, in this work, we limit ourselves only to methods that concern the explanations of a specific existing model in terms of interactions occurring in it, i.e., modeled by

it. To our knowledge, there is no scientific work that reviews, surveys or benchmarks methods of this type.

As mentioned above, several interaction extraction techniques are based on the partial dependence function. One of them is the H-statistic method introduced by Friedman and Popescu [30], which in the case of an analysis of two features based on calculating the difference between a two-way partial dependence function and the sum of two one-way partial dependence functions for both features. The same applies to higher-order interactions. The measure of interaction is the fraction of variance of the partial dependence function explained by the calculated differences. Another method using PD proposed by Greenwell et al. [33] is based on the concept of feature importance, understood as the variance of the partial dependence function. It measures the variance of the feature importance of one feature conditional on the other to get pairwise interaction. Another approach is to extract interactions based on prediction performance, which was introduced by Oh [58]. This technique uses permutation feature importance for decomposition similar to H-statistic.

The above methods are model-agnostic and work for any type of model. However, there are also techniques that target one type of model or a family of models with similar structural properties. One example is the method for extraction of interactions from tree-based ensemble models based on the conditional minimal depth concept described by Ishwaran et al. [44] consisting in calculating the distance between split nodes for a pair of features in one sub-tree. In the case of gradient-boosted trees models, comparing split scores (gain, cover) [18, 46] of consecutive nodes in trees is proposed for extracting interactions [45].

All of the methods mentioned above have no implementation in the Python language, hence they have been chosen to be developed as part of the package created. However, they do not represent all possible techniques. Namely, there are also SHAP interaction values [48, 50] and the method for detection feature interactions from neural network weights [72], both with existing Python implementations. Another method worth mentioning is the approach based on functional ANOVA decomposition [39], which is one of the first works on interaction extraction. In this method, the strength of the interaction is related to the loss caused by the projection of the model of interest onto the space of all additive models. The output of this method is a graphical representation of the interaction structure in the form of the so-called Variable Interaction Network (VIN). There are also techniques developed more recently, such as the Regional Effect Plots with implicit Interaction Detection (REPID) [36] with R code being available. This is an approach developed to detect the interaction of a selected feature with all the others, so that the feature effect can be presented more reliably in selected regions.

Some of the methods implemented in our package also have their implementations in R, even in the form of packages, such as `iml` [54] (H-statistic implementation), `vip` [34] (Greenwell method), `randomForestExplainer` [60] (conditional minimal depth), `EIX` [45] (split score), or `vivid` [42] (unnormalized H-statistic). The last one implements the visualization methods described by Inglis et al. [43] also used in our package. However, to our knowledge, there is no package that is a comprehensive interaction extraction toolkit implementing even more than one such method. Furthermore, most of the interaction extraction methods do not have any open-source implementation available in Python language. Hence the belief that such software has great potential and is needed in the community.

1.4. Contributions

Our work offers a novel and comprehensive contribution to the field of feature interaction analysis. The contributions of this thesis can be summarized as follows:

- We perform a literature review of feature interactions in predictive models. We describe selected methods for extraction of interactions, pointing out their advantages and disadvantages, and present methods related to the analysis of models in terms of interactions (concerning additivity and visualization of explanation results). An introductory description is given in Section 1.3, and a detailed theoretical description of the methods can be found in Chapter 2. It is the realization of goal (G1).

- We provide and describe the created solution – the Python package called `artemis`. It provides the implementation of various methods for extraction of interactions; both model-agnostic, which can be applied to a wide range of models, and model-specific for widely-used tree-based models, making the package versatile. The package also includes additional tools for further evaluating feature interactions in predictive models, such as measuring additivity and comparing results from different feature interaction extraction methods. To the best of our knowledge, it is the first Python package being a set of tools aimed at analysing predictive models in terms of feature interactions. The requirements specification of the `artemis` package is provided in Chapter 3, while the technical description is contained in Chapter 4. The package and its description represent the fulfilment of goal (G2).

- We describe the experiments conducted to benchmark the implemented methods. We compare methods mostly on synthetic data generated to include interactions. With knowledge of ground-truth interactions, we validate the quality of the methods developed and recom-

mend which ones to use. A description of the experiments is included in Chapter 5 and it constitutes the accomplishment of goal (G3).

- We present a use case application of interaction extraction methods to analyze models based on linguistic text features obtained with StyloMetrix. We describe how their use impacts the further development of solutions being developed at NASK National Research Institute. A description of this use case is provided in Chapter 6, and it indicates the achievement of goal (G4).

With these contributions, we fully realize the goals of the presented thesis.

1.5. Division of work

Most of the core tasks were carried out jointly by all team members, especially:

- research planning, literature review,
- package conceptualization and design,
- implementation of model-agnostic and model-agnostic feature interaction methods,
- implementation of feature importance methods,
- creating documentation and API reference,

Other tasks were also performed or planned jointly while they had their coordinator. The assignment of other tasks to coordinators is presented in Table 1.1.

Table 1.1: Division of work

Team member	Task
Paweł Fijałkowski	• Chapter 4 – technical documentation • Chapter 5 – experiments, benchmarks • implementation of auxiliary methods • manual testing, unit tests
Mateusz Krzyziński	• Chapter 1 – introduction • Chapter 2 – review of the literature • Chapter 6 – NASK • implementation of visualization methods • applying package to real-world data
Artur Żółkowski	• Chapter 3 – package specification • Chapter 7 – conclusions • refactoring, correcting • creating experiments on synthetic data

2. Review of methods for feature interactions analysis

In this chapter, we describe the selected methods related to the analysis and explanation of predictive models in terms of feature interactions from the theoretical side. We give their exact formulation, mathematical background, or algorithms used. The chapter is structured as follows: Section 2.1 presents methods for extracting interactions with corresponding methods for determining the importance of single features that can be analyzed together. It is a non-exhaustive review as we focus on techniques implemented in the `artemis` package. Section 2.3 describes the theoretical basis of the visualizations proposed in the literature and used in the package. Section 2.4 discusses other methods that facilitate the analysis of feature interactions and require mathematical elaboration.

2.1. Model-agnostic methods

Partial dependence function Two of the three implemented model-agnostic methods are based on the partial dependence profiles proposed by Friedman [29]. They are used to present the marginal effect of selected features on the model's response. In the case of a classification task it is the probability of the selected class returned by the model. For selected features x_S (fixed at their values) and other features $X_{\setminus S}$ used by the model $\hat{f}$, partial dependence function is defined as:

$$F_S(x_S) = \mathbb{E}_{\setminus S}\left[\hat{f}(x_S, X_{\setminus S})\right] = \int \hat{f}(x_S, X_{\setminus S}) p_{\setminus S}(X_{\setminus S}) \mathrm{d}X_{\setminus S}, \tag{2.1}$$

where $p_{\setminus S}$ is the marginal probability density of $X_{\setminus S}$.

Partial dependence function (Equation 2.1) can be estimated from data [29] as:

$$\widehat{F_S}(x_S) = \frac{1}{N}\sum_{i=1}^{N} \hat{f}(x_{iS}, X_{i\setminus S}), \tag{2.2}$$

where N is the number of observations in the dataset. The corresponding procedure for determining the PD values for a pair of features is presented in the Algorithm 1.

In addition, the method for determining the importance of features corresponding to both interaction methods based on partial-dependence functions also uses these function. It estimates

importance as fluctuations in one-dimensional PD functions (see Equation 2.7).

Algorithm 1 An algorithm for estimating partial dependence function for two features

Input: model of interest $\hat{f}$, dataset X, selected features j, k

Output: estimated values of partial dependence function F_{jk}

$\text{VAL}_{jk} \leftarrow X_j \times X_k$ ▷ Estimation can also be performed only for pairs of values occurring in the data.

for $(x_j^*, x_k^*) \in \text{VAL}_{jk}$ **do**

(1) copy the dataset X

(2) replace the original values of X_j with the value x_j^*

(3) replace the original values of X_k with the value x_k^*

(4) get predictions for the modified dataset

(5) average obtained predictions to get $\hat{F}_{jk}(x_j^*, x_k^*)$

end for

2.1.1. Friedman H-statistic

The H-statistic proposed by Friedman and Popescu [30] is a method based on decomposing higher-order partial dependence functions into lower-order ones. Specifically, for features x_j, x_k, it is calculated as the fraction of variance of higher-order PD function $F_{jk}(x_j, x_k)$ captured by the interaction, which is defined as the difference between $F_{jk}(x_j, x_k)$ and sum of lower-order PD functions $F_j(x_j) + F_k(x_k)$. Importantly, his method requires that the PD functions be centered so that they have a mean value of 0. The exact formula is:

$$H_{jk}^2 = \frac{\sum_{i=1}^{N} \left[F_{jk}(x_j^{(i)}, x_k^{(i)}) - F_j(x_j^{(i)}) - F_k(x_k^{(i)}) \right]^2}{\sum_{i=1}^{N} \left[F_{jk}(x_j^{(i)}, x_k^{(i)}) \right]^2}. \tag{2.3}$$

The H-statistic can also be used to assess the strength of the interaction of more than two features. As an example, an interesting application is testing whether a single selected feature interacts with any other features included in the model. It can be done by calculating the following measure:

$$H_j^2 = \frac{\sum_{i=1}^{N} \left[f(x^{(i)}) - F_j(x_j^{(i)}) - F_{-j}(x_{-j}^{(i)}) \right]^2}{\sum_{i=1}^{N} \left[f(x^{(i)}) \right]^2}. \tag{2.4}$$

The H-statistic is equal to 0 when there is no interaction and 1 when all variance of higher-order partial dependence (PD) is explained by the sum of lower-order PD functions, indicating that the overall effect of two features comes solely from their interaction. However, there may also be values greater than 1, which is difficult to interpret. To check the significance of a found

interaction value, Friedman and Popescu [30] propose the use of parametric bootstrap [25] to obtain a null distribution. Unfortunately, this approach is not applicable without knowledge of the distribution of the generated dataset.

Moreover, it should be noted that the H-statistic may be prone to overestimating interactions between low-importance pairs of features. This occurs when the denominator in Equation 2.3 is small, but the total effect of two features consists mainly of their interaction. To address this issue, an unnormalized H-statistic [43] can be used. It takes into account only the numerator from Equation 2.3, thus potentially reducing the number of spurious interactions indicated by the method. The formula for unnormalized H-statistic is as follows:

$$H^*_{jk} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left[F_{jk}(x_j^{(i)}, x_k^{(i)}) - F_j(x_j^{(i)}) - F_k(x_k^{(i)})\right]^2}. \tag{2.5}$$

Moreover, the same modification is applicable to the case of other sets of features, including the calculation of interactions between one feature and all others:

$$[H^*_j]^2 = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left[f(x^{(i)}) - F_j(x_j^{(i)}) - F_{-j}(x_{-j}^{(i)})\right]^2}. \tag{2.6}$$

It is worth noting that thanks to such a formulation, the obtained values are in the same range of values as the predictions of the model, which may be considered another advantage of this modification.

2.1.2. Greenwell method

The Greenwell method [33] is a technique for quantifying the relative feature interactions based on fluctuations in their partial dependence profiles. This approach is motivated by the idea that significant fluctuations in a feature's partial dependence profile indicate a strong dependence on the model's response to that feature and therefore, a high level of importance in the model.

Different fluctuation measures for partial dependence function values might be used, but the authors suggest using standard deviation for continuous features and standard deviation approximation for small samples when dealing with categorical features:

$$\mathrm{imp}(x_j) = \begin{cases} \sqrt{\frac{1}{m-1}\sum_{i=1}^{m}\left[F_j(x_j^{(i)}) - \frac{1}{m}\sum_{i=1}^{m}F_j(x_j^{(i)})\right]^2} & \text{if } x_j \text{ is continuous,} \\ \frac{\max_i\left(F_j(x_j^{(i)})\right) - \min_i\left(F_j(x_j^{(i)})\right)}{4} & \text{if } x_j \text{ is categorical.} \end{cases} \tag{2.7}$$

For pair of features x_j, x_k, the strength of the interaction is calculated based on their conditional importance when conditioning for one another. The conditional importance of x_j with respect to $x_k = z$ is defined as:

$$\text{imp}(x_j|x_k = z) = \begin{cases} \sqrt{\frac{1}{m-1}\sum_{i=1}^{m}\left[F_{jk}(x_j^{(i)}, z) - \frac{1}{m}\sum_{i=1}^{m} F_{jk}(x_j^{(i)}, z)\right]^2} & \text{if } x_j \text{ is continuous,} \\ \frac{\max_i\left(F_{jk}(x_j^{(i)},z)\right) - \min_i\left(F_{jk}(x_j^{(i)},z)\right)}{4} & \text{if } x_j \text{ is categorical.} \end{cases} \tag{2.8}$$

Values obtained in this way for each unique value z of feature x_j are then aggregated using the standard deviation $\mathbf{s}$. This process is then repeated in reverse to determine the conditional importance of x_j with respect to x_i values, and the two results are averaged to obtain the overall feature interaction value:

$$G_{jk} = \frac{1}{2}\left[\mathbf{s}(\text{imp}(x_j/x_k)) + \mathbf{s}(\text{imp}(x_k/x_j))\right]. \tag{2.9}$$

Figure 2.1 illustrates how the method works.

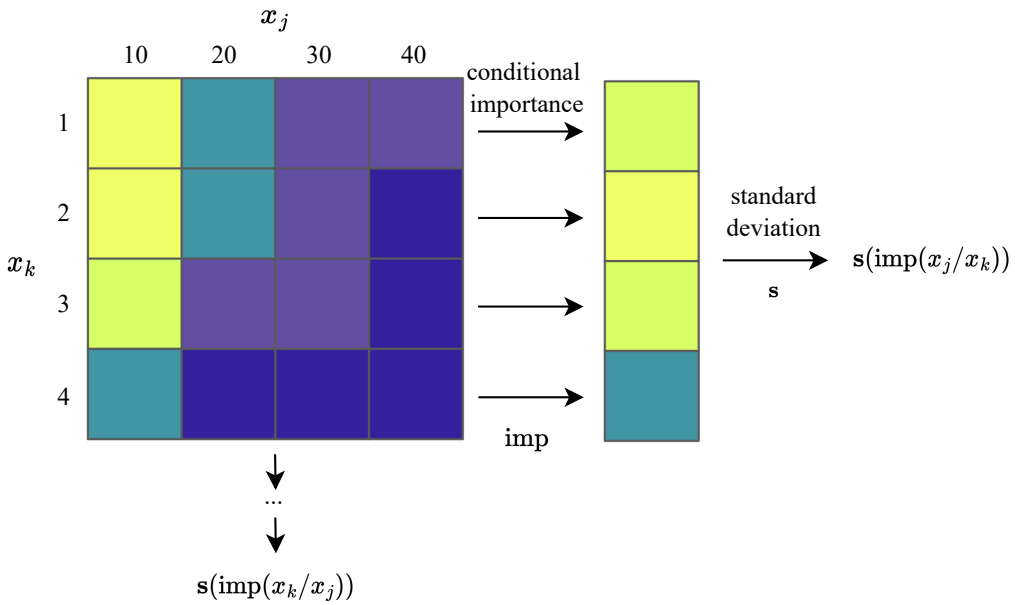

Figure 2.1: A diagram showing the operation of the Greenwell method. Interactions are estimated based on a two-dimensional partial dependence function depicted as a heat map.

While the method is relatively straightforward to understand, it does have some limitations. One potential drawback is that the calculation of partial dependence profiles requires the generation of a Cartesian product of x_i and x_j, which can lead to the inclusion of pairs of values out-of-manifold. Additionally, this process is computationally expensive, as it requires calculating profiles for all pairs of unique values in both features, resulting in quadratic complexity.

This can be problematic when working with large datasets and may require sampling techniques. Another issue with the Greenwell method is that it assumes that greater fluctuations in the average of conditional importance correspond to stronger feature interactions, but this has not been rigorously proven. Despite these limitations, the Greenwell method remains useful for extracting feature interactions from data.

2.1.3. Oh method

A method proposed by Oh [58] defines feature interactions in terms of prediction performance. It uses permutation feature importance and calculates the strength of interaction between features x_j and x_k as:

$$O_{jk} = \mathrm{PFI}(x_j) + \mathrm{PFI}(x_k) - \mathrm{PFI}(\{x_j, x_k\}). \tag{2.10}$$

Thus, this method assumes that in the absence of interaction, the permutation importance of a pair of features can be decomposed into the sum of their single-importance values ($\mathrm{PFI}(\{x_j, x_k\}) = \mathrm{PFI}(x_j) + \mathrm{PFI}(x_k)$). In the original article, the author proposes to distinguish between negative and positive interactions, while in our package we only include the absolute values of the interactions calculated in this way due to the difficulty of interpreting negative ones.

The metric must be compatible with the problem type to calculate permutation feature importance. The metrics proposed as default are accuracy for the classification task and RMSE for the regression task.

The Oh method is easy to calculate and faster than partial dependence-based techniques. However, the definition of interaction in this method is separate from others studied in the literature. It is not entirely consistent with intuition, as indicated by the results obtained in experiments (Chapter 5). Naturally, the corresponding value of the importance of a single feature is the well-known permutation feature importance.

2.2. Model-specific methods

2.2.1. Conditional Minimal Depth method

The Conditional Minimal Depth method is based on the proposition described by Ishwaran et al. [44]. It has been refined and implemented in the open-source `randomForestExplainer` R package [60].

This method is tailored for tree-based ensemble models as it uses trees' structure to define interaction. Namely, pair interaction between features x_j and x_k is described in terms of depths

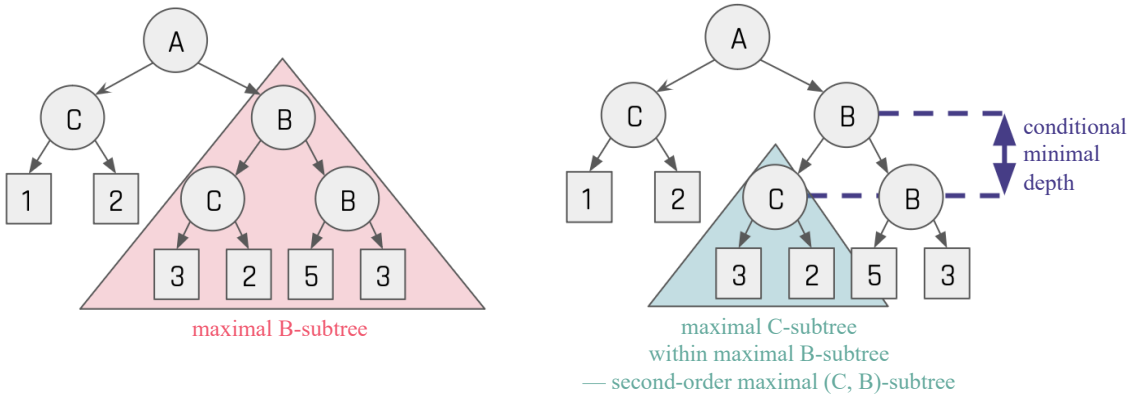

Figure 2.2: Illustration of maximal subtrees and conditional minimal depth measure. Note that there are two B-subtrees, but the smaller one is contained within the bigger one, so the latter is maximal.

of the maximal x_j-subtrees within maximal x_k-subtrees. x_k-subtree is defined as a subtree with root node being split using x_k feature and closest to the root of the whole considered tree, while maximal x_k-subtree is the one that is not a subtree of any other larger x_k-subtree. Moreover, the maximal x_j-subtree within maximal x_k-subtree is called second-order maximal (x_j, x_k)-subtree (see Figure 2.2).

The measure of interaction is the averaged value of calculated depths in all the trees in the analyzed ensemble. In this case, smaller values of the metric mean greater strength of interaction, as this means that the pair of features occur in trees close to each other. Moreover, this is an asymmetric measure – the maximal subtree when conditioned on feature x_j within the maximal subtree when conditioned on feature x_k is not equivalent to the maximal subtree when conditioned on feature x_k within the maximal subtree when conditioned on feature x_j. This highlights the importance of the feature on which the conditioning takes place in analyzing potential interactions with other features. In addition, a given second-order subtree may not occur in all trees in the ensemble, so a simultaneous analysis of the number of occurrences of a given structure is critical (the mean conditional minimal depth alone can be misleading).

The implementation provided in the `artemis` package is optimized for the number of traversed pairs of trees, significantly improving the expected average time complexity of the solution and therefore allowing for smooth near real-time usage.

The corresponding importance measure used is the minimum depth of the node split using a given feature (the depth of the maximal subtree for this feature) averaged over all trees. To be able to compare interaction values with importance values, a value of one is subtracted from the conditional minimum depths so that the resulting values, as for importance, start from 0.

2.2.2. Split Score method

The Split Score method is tailored for analyzing gradient tree boosting models as it utilizes structure and additional information collected in models built in accordance with algorithms of this kind. It has been implemented in the open-source `EIX` R package [45].

The method is based on analyzing split scores (gain, cover) used in algorithms such as XGBoost [18] and LightGBM [46]. Split scores are measures calculated for every non-terminal node of built trees. Particularly, the gain is a measure that determines the gain (profit) from splitting a given node, while the cover is calculated as the sum of the second-order derivatives on the loss function of the observations classified into a given node.

Two features are considered to interact with each other in the tree if one feature is a child of another and the split score of the child feature is greater than the split score of the parent. Thus, this method is also asymmetric.

The final measure of interaction can be obtained by aggregating split scores of child features from pairs of interacting features. The aggregation function used is averages and sums of split scores.

Since the trees in the algorithms supported by this method are created and added greedily, aggregating to a single-number interaction measure may be misleading. For this reason, it is valuable to analyze split scores for individual trees, which is also made possible by the `artemis` package.

Measures of the feature importance associated with this method are also related to the tree structure of the model. Split scores are also possible but aggregated by occurrences of individual features. There are also additional measures such as the average depth of nodes split by selected feature, average depth weighted by node gain, root frequency, and weighted root frequency.

2.3. Visualization methods

Visualization methods (and visual analytics in general) play a crucial role in explainable artificial intelligence [3]. Visualizing the results of methods for extraction of interactions is no different.

Inglis et al. [43] proposed a method tailored to visualize the interaction along with the importance of features. The described technique involves using heat map visualization with seriation. In such a display, the diagonal of the matrix contains the validities of the individual features, while the off-diagonal contains measures of interaction between pairs of features. The two series

of values have separate color scales (since they can be expressed on different numerical scales). We use color palettes that are friendly to people with color vision deficiency (i.e., sequential single-hue color scales: green for feature importance and purple for interaction values). This representation makes it possible to simultaneously analyze two types of explanations that are interrelated. Focusing on interactions makes it easier to see whether features with a large interaction value also have a large effect on the model alone (it can make it possible to spot spurious interactions). An essential aspect of this visualization is also seriation, i.e., determining the order of the features in the matrix. The goal is to group the most important and interacting features in the upper left corner of the matrix, making it easier to interpret the results. For this purpose, we calculate the following score for each feature:

$$w_j = \frac{1}{\max_i \mathrm{v}_i}\mathrm{v}_j + \frac{1}{\max_{i,k} \mathrm{s}_{ik}} \max_k \mathrm{s}_{jk}, \tag{2.11}$$

where v_j stands for importance of feature j and s_{jk} denotes interaction value for pair of features j and k. The features are then sorted according to the calculated scores, depending on whether a smaller or larger value of the selected measure indicates greater importance and interactions.

This type of visualization is the default in the `artemis` package. An example plot of this type generated by the package is shown in Figure 2.3. There are also more types of visualizations possible, including the more common ones, i.e.:

- graph visualization (Figure 2.4) – also proposed by Inglis et al. [43]. It draws interactions and importance in the form of a graph, the vertices of which are individual features, with the vertices' size depending on the features' importance. Edges in the graph correspond to interactions – their weight (thickness) is related to the interaction strength of a given pair of features. For methods with asymmetric two-way interactions, the graph drawn is a directed graph; for the symmetric case, the edges have no direction. In addition, for consistency and ease of interpretation, the same colors are used for the heat map.

- Bar charts showing interactions (Figures 2.5 and 2.6). They are used to visualize two types of interactions: two-way interactions (in each method) and the interaction of a single feature with all the others (obtained using the H-statistic method). For consistency with the package's convention, the bars are purple in color.

- Visualization for tree-based ensemble models summarizing the results from the Conditional Minimal Depth method (Figure 2.7). It is based on the chart possible to generate in the `randomForestExplainer` package. Interactions (pairs of features) are visualized as bars, whose color illustrates the number of occurrences of such structures in the trees. The height of the bar indicates the measure of the interaction – the mean conditional minimal depth

and an additional indicator (called lollipop) denotes the importance of the parent feature – the mean minimal depth.

- Visualization for gradient tree boosting models summarizing the results from the Split Score method (Figure 2.8). It is based on the chart possible to generate in the `EIX` package. This visualization illustrates the gains of consecutive nodes from the first k selected trees. The labels refer to the features used for the splits that give the highest score. There are also labels for pairs of consecutive nodes for which the child nodes give a higher score than the parent ones (the definition of interaction in this method).

In addition to the visualization methods showing results directly related to interactions, it is possible to visualize two-dimensional and one-dimensional partial dependence profiles. This functionality is available for methods based on PD functions as an additional feature to facilitate and complement the interpretability of the results. The resulting plots show the values of the PD function in the form of a contour plot (for two features) and a linear plot (for one feature), see Figure 2.9.

However, due to the large number of pairs that can be analyzed and to facilitate the comparison of two-dimensional partial dependence functions with each other, it is also possible to generate a grid of charts so that each panel contains a PD function visualization for a different pair of features. Such a visualization is called a zenplot [37] and for partial dependence profiles have been proposed by Inglis et al. [43]. An example is shown in Figure 2.10. One of its key advantages is that it saves space by plotting successive panels sequentially so that adjacent axes are associated with the same feature. This property also allows for easier comparison of values. The order of drawing panels on the so-called zigzags is determined by the Eulerian path algorithm [41].

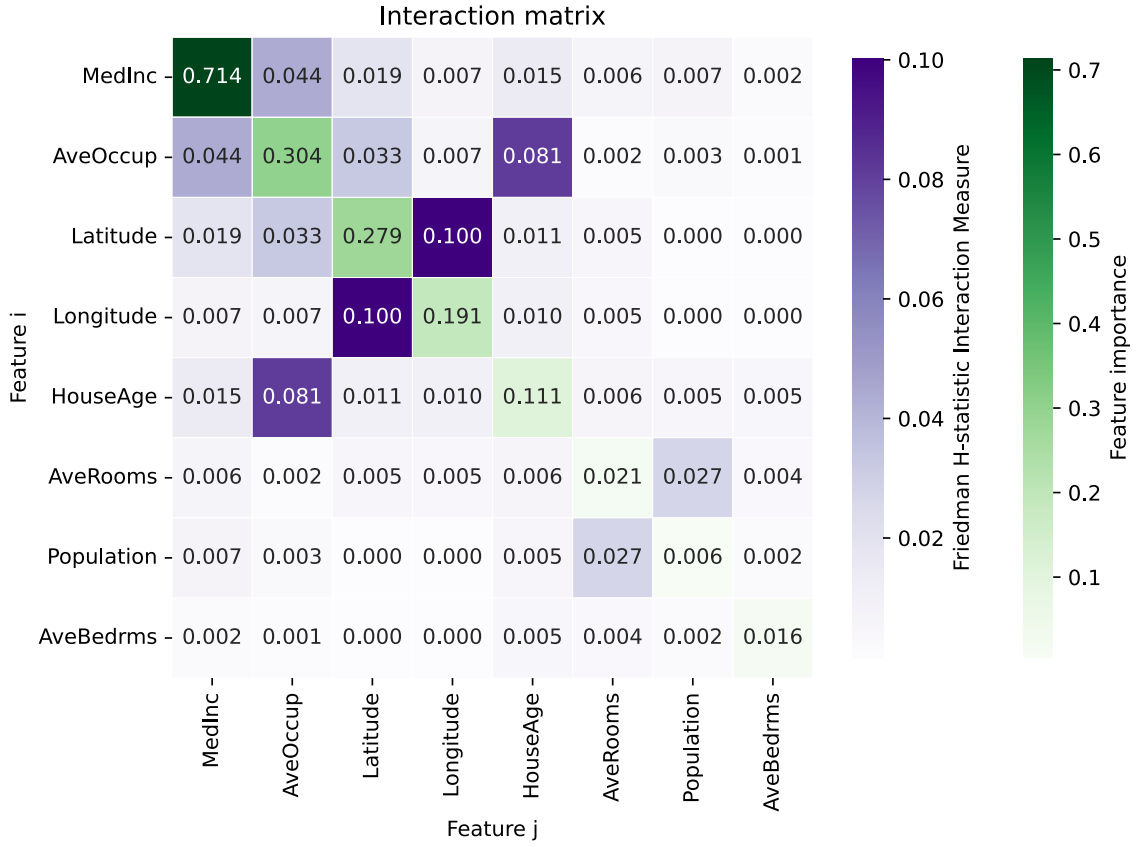

Figure 2.3: Example heat map visualization of interactions for the California housing dataset.

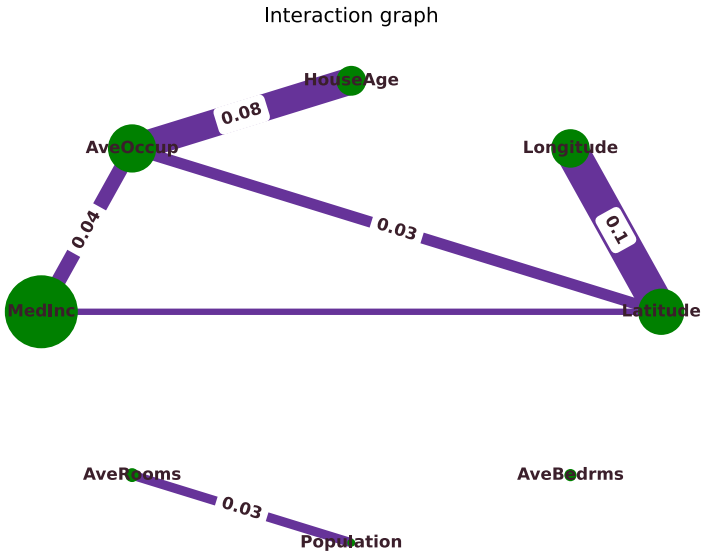

Figure 2.4: Example graph visualization of interactions for the California housing dataset.

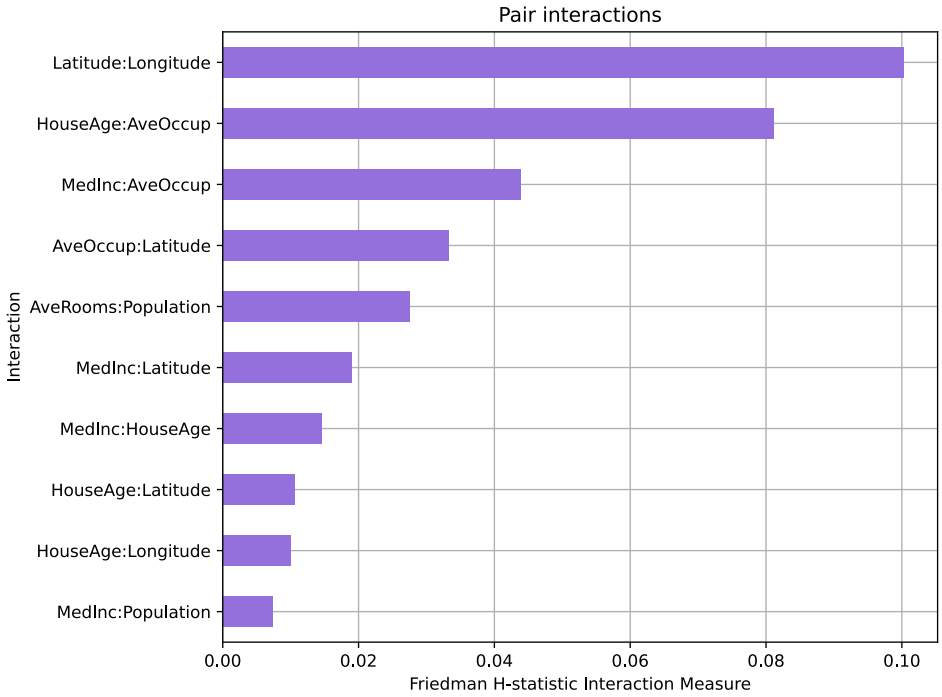

Figure 2.5: Example bar chart visualization of pair interactions for the California housing dataset.

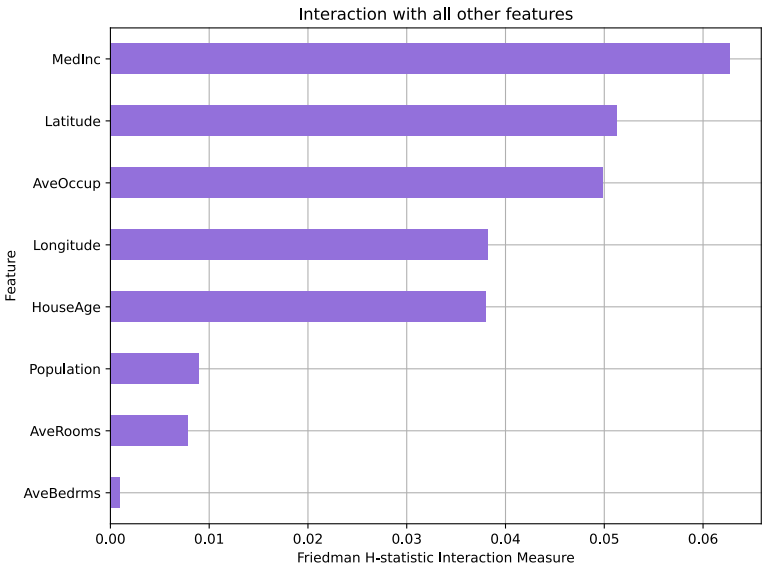

Figure 2.6: Example bar chart visualization of one vs. all interactions for the California housing dataset.

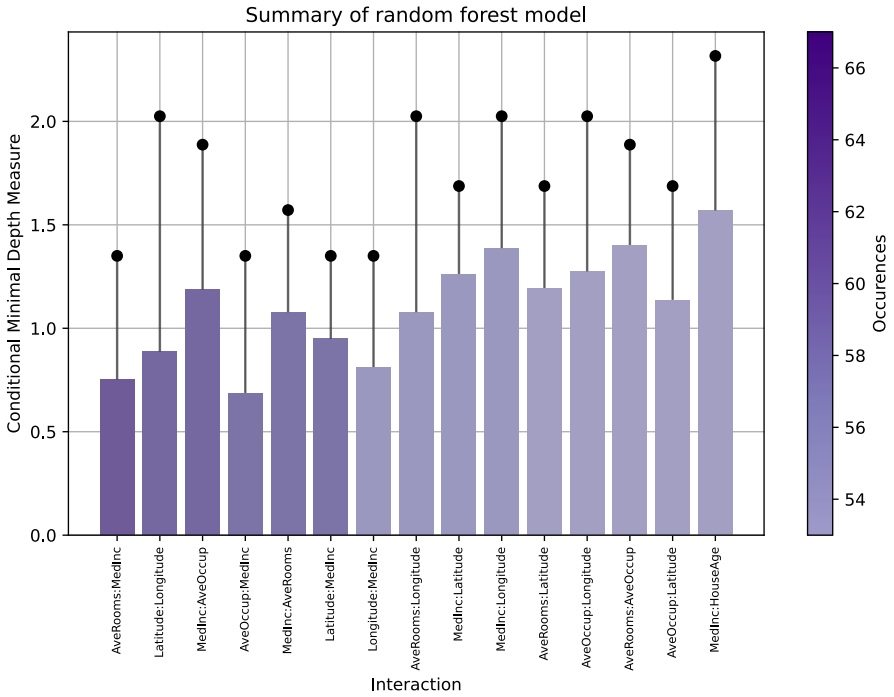

Figure 2.7: Example visualization for a tree-based ensemble model trained on the California housing dataset.

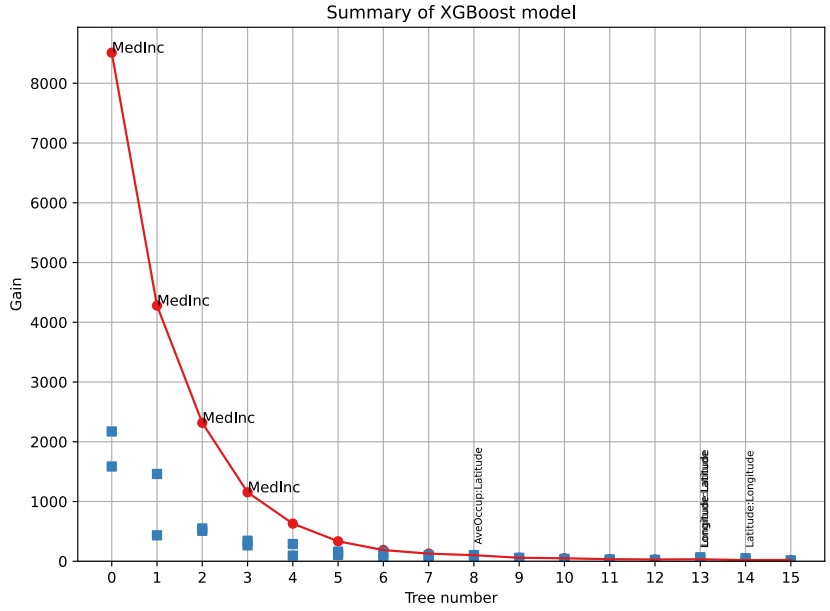

Figure 2.8: Example visualization for a gradient tree boosting model trained on the California housing dataset.

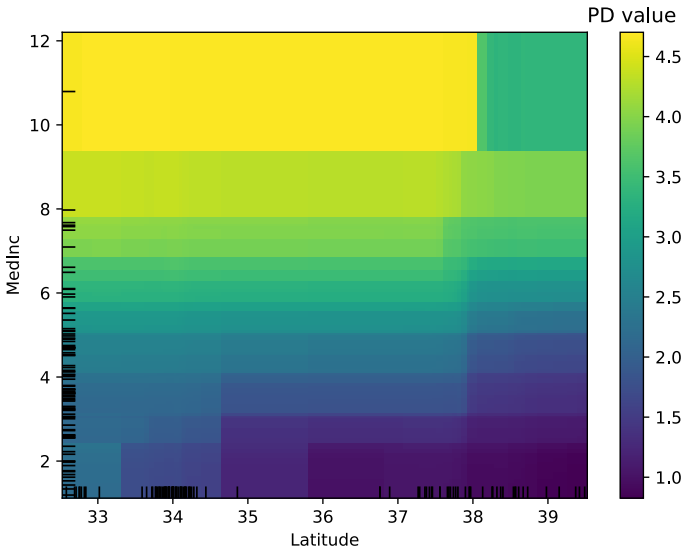

Figure 2.9: Example two-dimensional partial dependence plot for selected feature pair for the California housing dataset.

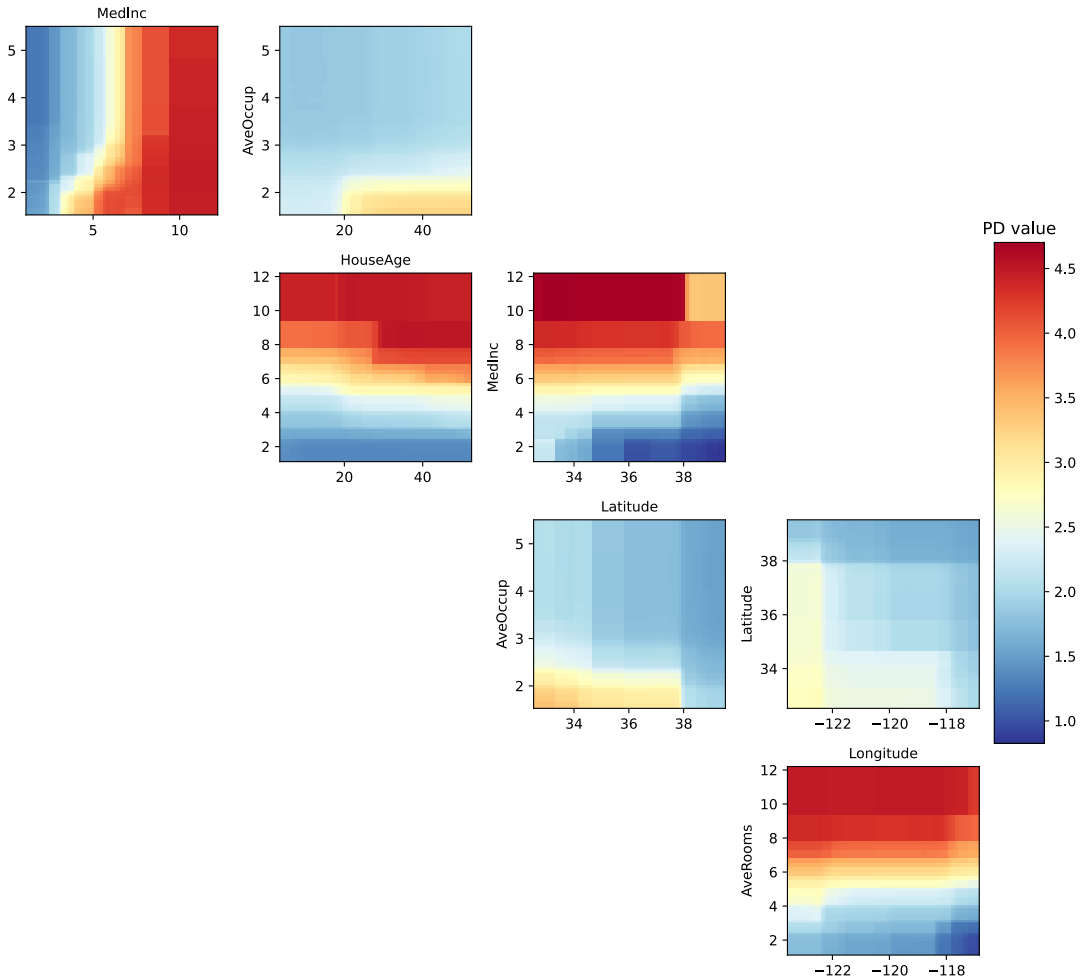

Figure 2.10: Example zenplot interaction visualization for the California housing dataset.

2.4. Additional methods

Additivity Meter

One of the functionalities of our package is the ability to measure the additivity index of the analyzed model. For this purpose, we use a concept based on the definition of Interaction Strength given by [55] but we use the partial dependence function instead of accumulated local effects (for consistency with the use of PD in interaction extraction methods).

In more detail, we calculate the sum of the main effects of PD as:

$$f_{PD}(x) = f_0 + F_1(x_1) + \ldots + F_p(x_p), \tag{2.12}$$

where f_0 is the mean baseline prediction and p is the number of features included in a model. Then, we compute the additivity index as:

$$AddInd = 1 - \frac{\sum_{i=1}^{N} \left[\hat{f}(x^{(i)}) - f_{PD}(x^{(i)})\right]^2}{\sum_{i=1}^{N} \left[\hat{f}(x^{(i)}) - f_0\right]^2}. \tag{2.13}$$

Note that additive models such as linear regression have an additivity index of 1.

Method Comparator

The Method Comparator feature in the Insights module allows `artemis` users to compare the results of two feature interaction extraction methods, both visually and numerically. This enables them to evaluate the stability of the results and identify any potential issues.

To quantify the consistency of the results from the two methods, rank correlation is calculated. The user can choose from three options: Kendall rank correlation (Equation 2.14), Spearman rank correlation (Equation 2.15), or Pearson correlation (Equation 2.16):

$$KendallCorr = \frac{2}{n(n-1)} \sum_{i<j} \operatorname{sgn}(x_i - x_j) \cdot \operatorname{sgn}(y_i - y_j), \tag{2.14}$$

$$SpearmanCorr = 1 - \frac{6 \sum_{i=1}^{n} (x_i - y_i)^2}{n(n^2 - 1)}, \tag{2.15}$$

$$PearsonCorr = \frac{\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n} (x_i - \bar{x})^2} \sqrt{\sum_{i=1}^{n} (y_i - \bar{y})^2}}. \tag{2.16}$$

This allows for a comprehensive analysis of the coherence of the results obtained using different methods.

The Method Comparator module provides a visual comparison of the results of two feature interaction extraction methods by plotting the one-vs-one interaction values on a 2D Cartesian plane. While the absolute values cannot be directly compared, the monotonicity of the resulting chart can be used to infer the agreement between the methods in terms of the order of strength of the feature interactions. An example of such a plot is shown in Figure 2.11.

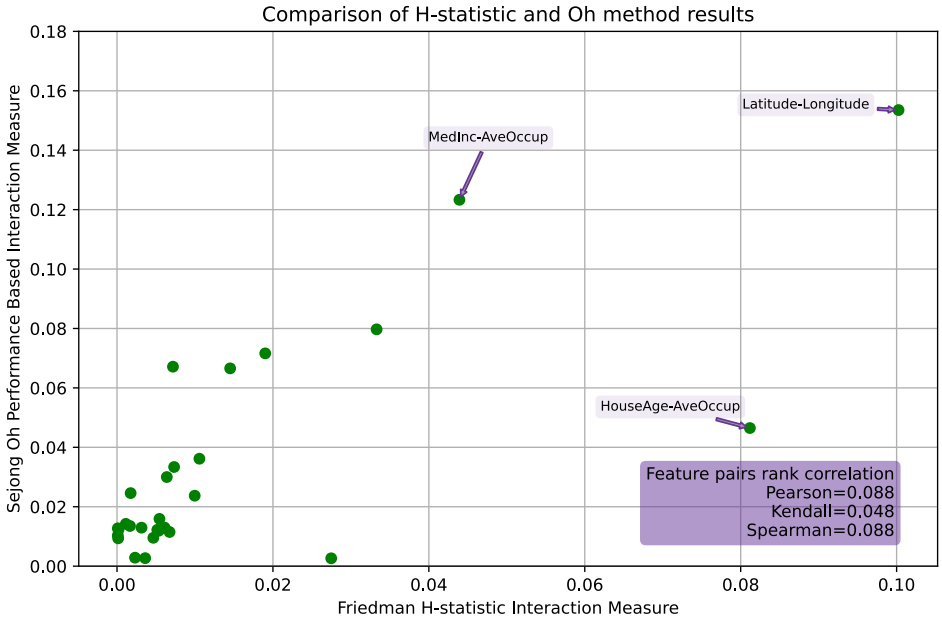

Figure 2.11: Visual method comparison for Friedman H-statistic method and Oh method. The model was trained on the California housing dataset.

3. The `artemis` package – specification

In this chapter, we describe created package called `artemis` conceptually, from a business perspective. We do this through a specification created for the developed software. Section 3.1 contains an executive summary, while further sections deal with requirements – both functional (Section 3.2) and non-functional (Section 3.3). All the requirements are met by the prepared implementation of the package, so they constitute an adequate description of its functionalities.

3.1. Executive summary

`artemis` is a `Python` package for data scientists and machine learning practitioners which exposes standardized API for extracting feature interactions from predictive models using a number of different methods described in scientific literature. The package provides both model-agnostic (no assumption about model structure), and model-specific (e.g., tree-based models) feature interaction methods, as well as other methods that can facilitate and support the analysis and exploration of the predictive model in the context of feature interactions. The available methods are suited to tabular data and classification and regression problems.

The main functionality is that users are able to scrutinize a wide range of models by examining feature interactions in them by finding the strongest ones (in terms of numerical values of implemented methods) and creating tailored visualizations.

The developed package is available as open-source software on the Python Package Index (PyPI) repository.

3.2. Functional requirements

Functional requirements of the project are summarized in the form of user stories [19] included in the Table 3.1 and corresponding acceptance criteria.

Table 3.1: User stories

ID	As a...	I want to...	so that I can...
REQ1	data scientist,	know which features interact with each other in any created model,	gain valuable insight into the operation of any model and therefore improve my problem understanding.
REQ2	data scientist,	be able to measure the strength of feature interactions in my model,	know what interactions are the most important for the created model and its prediction.
REQ3	data scientist,	be able to choose the most proper method parameters in my case,	control how explanations are calculated and the time needed to generate them.
REQ4	data scientist,	be able to compare interaction values provided by different methods,	compare their results and be sure of the truthfulness and stability of the obtained results.
REQ5	data scientist,	visualize the numerical results of the selected explanation method,	present plots to a business client or in a report.
REQ6	data scientist,	visualize the form of selected interaction of two features on a two-dimensional heat map,	better understand the nature of this interaction.
REQ7	data scientist,	be able to extract interaction only from a group of selected features,	find interactions only in the feature subset of interest.
REQ8	data scientist,	compare feature interactions with single feature importance,	scrutinize groups of features that not only interact but also are significant to the model.
REQ9	data scientist,	know how each of the available methods works,	choose the appropriate one and understand its result better.

Acceptance criteria User stories are complemented by the following acceptance criteria formatted as checklists, which also provide a more detailed description of functional requirements.

Ad. REQ1

- Each instance of the class associated with the method for extraction of interactions has an `ovo` attribute with values of the strength of pair interactions. Sorted results and provided visualizations with thresholds to enable extraction of crucial and significant interactions. Due to the computational cost and low added value, the automatic and default calculation of higher-order interactions was abandoned.
- The model-agnostic methods handle many packages like `scikit-learn` [62] automatically. In the case of more custom models, it is possible to provide a predictive function directly, which is an adaptation to the package API.

Ad. REQ2

- Each instance of the class associated with the method for extraction of interactions has an ovo attribute which, after the fit method is called (calculations are performed), stores the results – values of the strength of pair interactions.

Ad. REQ3

- The implemented methods have adjustable parameters corresponding to the details of their operation, such as the sample size taken into account for the calculation.
- Method parameters are described in detail in docstrings and documentation generated on their basis. Default parameters are selected and explicitly specified where it is possible. Their values ensure fast and stable execution of calculations.

Ad. REQ4

- There are 3 significantly different model-agnostic methods implemented with many possible parameters to customize their operation. These are: `FriedmanHStatisticMethod`, `GreenwellMethod`, `SejongOhMethod`.
- There are two significantly different model-specific methods for tree-based models implemented. They also have parameters to customize their operation. These are: `SplitScoreMethod`, `ConditionalMinimalDepthMethod`.

Ad. REQ5

- The package provides the possibility to visualize the results of each of the implemented methods. There are 4 types of visualizations available for each method of extraction of interactions: `heatmap`, `bar chart`, `graph`, and `summary plot` that combines all previous visualizations. Moreover, there are visualizations that are specific and available only for selected method: `bar_chart_ova` with one vs all interactions for `FriedmanHStatisticMethod`, `lolliplot` for method tailored for gradient-boosted trees models, and `bar_chart_conditional` for summarizing of tree-based models.

Ad. REQ6

- The package allows the visualization of the interaction in the form of a two-dimensional heat map in two ways: without the partial dependence profile-based interaction method – with the help of the `PartialDependenceVisualizer` class and with the partial dependence profile-based interaction method –with `plot_profile` method.

Ad. REQ7

- Interaction extraction methods enable the specification of which features are to be considered in them. There are parameter `features` in `fit` methods that can be used where it is applicable (due to the different characteristics of methods).

Ad. REQ8

- Every method for computing the strength of the interactions also has a corresponding method for calculating the importance of individual features (module importance methods).

Ad. REQ9

- The package has clear but polished and detailed documentation of each implemented method – a description of their mathematical and statistical background and individual parameters.

3.3. Non-functional requirements

Table 3.2 and Table 3.3 specify the non-functional requirements that should be followed in order to maximize the usefulness of the package. Non-functional requirements are described in terms of **U**sability, **R**elability, **P**erformance, **S**upportability (**URPS**).

Table 3.2: Non-functional requirements related to usability and reliability.

Requirements area	Description
Usability	• All functions, parameters, and descriptions are written in English. • All implemented functions are consistent with each other (e.g., in terms of syntax and naming convention). • Visualizations shown aesthetically display relevant feature interactions following the common principles of creating visualizations. • Colors used for visualizations are easily recognizable for users with color blindness, and the font is not smaller than 10 pt. • The user has easy access to function descriptions through offline documentation that can be displayed with the `help()` function. • The form of returned data is readable for the user (e.g., column names are understandable).
Reliability	• Interaction extraction algorithms used on identical data and with the same parameters are deterministic, i.e., give reproducible results. • There are no errors in the standard usage scenario, i.e., assuming the use of industry-standard data types/objects (`pandas` data frames, `NumPy` arrays, `scikit-learn` models). • The package does not undergo permanent failures, i.e., the occurrence of an error (e.g., caused by the wrong specification of parameters by the user) does not entail further unusability of the package.

Table 3.3: Non-functional requirements related to performance and supportability.

Requirements area	**Description**
Performance	• The library works in near real-time, allowing for smooth use in the experiments. The limit of 5 minutes of calculation time for feature interaction summary using the default parameters is not exceeded. • Each method is analysed in terms of time complexity. If a given method's execution time exceeds execution time by nature, mitigation measures (sampling) should be used.
Supportability	• To be compatible with most software on the market, the package support all versions of Python higher than 3.8. • Comprehensive documentation of the methods and their implementation is provided. • The package is easy to install, i.e., installable via the `pip` Python package-management system.

4. The `artemis` package – technical documentation

This chapter provides a technical overview of the `artemis` package. Specifically, we focus on the software engineering aspects of the developed software. The first section (Section 4.1) provides an overview of the package, while the following sections delve into more detailed descriptions. In particular, Section 4.2 covers the package's architecture and modules, and Section 4.3 describes the client interface and workflow for communication within the package.

4.1. Package overview

The proposed solution is a Python package - `artemis`, which stands for *A Robust Toolkit of Explanation Methods for Interaction Spotting*. It is available as open-source software and is stored in `PyPI`. The main purpose of the package is to provide simple `scikit-learn` style `API` [15] for calculating feature interaction methods known from the literature. Classes representing each interaction method are imported from the shallow tree package directory. Exposed methods enable the user to calculate the selected method and inquire about the results (both visually and numerically). Solution architecture is relatively simple as `artemis` has no disk-space memory state and passes data in-memory through objects on the user's machine. It can be described via package diagram (Figure 4.1) depicting dependencies between the distinguished modules.

In the following section, we will summarise the package modules depicted in Figure 4.1. These summaries will provide an overview of the purpose and function of each module.

- **Interaction methods** – the most important part of the `artemis` project, it contains the most important functionalities that constitute the added value of the solution. Model-agnostic (applicable to each model) and model-specific (applicable only to selected models with a specific structure) feature interaction methods are implemented in this module.

- **Importance methods** – created as complementary, capable of providing supplemental information to the predictive model analysis. It contains implementations of model-agnostic

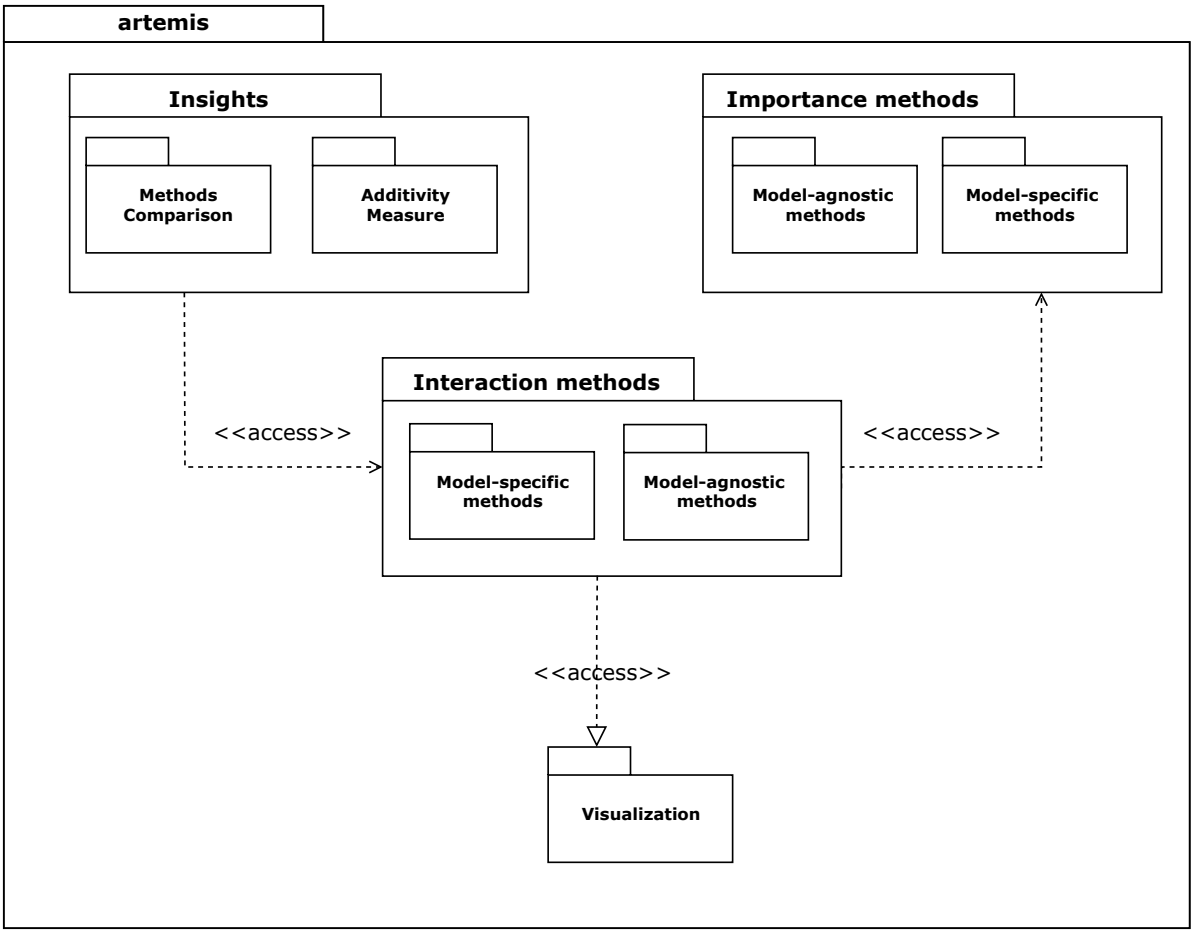

Figure 4.1: Diagram of modules for proposed solution – `artemis` Python package

and model-specific methods for detecting feature importance. Feature interaction methods and their corresponding importance methods are based on the same concepts, so the information about the relative feature importance may indicate that a given interaction is worth analyzing.

- **Visualization** – allows the user to visualize the results of the obtained explanation of the interactions, including the information about the importance of features. The visualization module enables the automatic selection of the appropriate type and graphical parameters of the plot to the results.

- **Insights** – additional analytics methods for users to explore models and data further. Allows for visual and numerical comparison of results from different feature interaction methods as well as gaining further insight into the additivity of the model.

4.2. Modules description

The structure of the created package (without division into modules) is visualized on the simplified UML diagrams presented in Figures 4.2 and 4.3. Some of the functionalities are not directly class methods but are created as helper functions in the appropriate directories, which is a common practice in Python packages. Full `artemis` documentation in the `HTML` format can be found at `GitHub Pages`. The documentation is generated from code on `master` branch and will evolve with any repository changes.

In this part of the document, we will describe the modules depicted in Figure 4.1 concerning specific classes from the diagram in Figure 4.2 and 4.3.

Interaction methods The interaction methods module includes methods for extracting interactions from predictive models, which is the main goal of the designed solution. All implemented methods (corresponding classes) will inherit from the abstract class `FeatureInteractionMethod`, which contains the key attributes and methods:

- attributes:
  - `method` – string describing method name,
  - `features_included` – a list of strings describing names of features used in the analysis (it is possible to analyze interactions only for a selected set of features),
  - `ovo` – pandas DataFrame containing interaction strength between pairs of features: their names and the numerical value of interaction,
  - `X_sampled` – pandas DataFrame being a subset of provided dataset that was used in calculations,
  - `feature_importance` – pandas DataFrame containing feature importance; names of features and numerical value of their importance,
  - `pairs` – a list of pairs of features for which interactions are calculated
- methods:
  - `fit(model, X)` – a method used to calculate feature interaction and importance for passed model and data-set. For model-specific methods that use model structure to extract feature interactions (`ConditionalMinimalDepthMethod` and `SplitScoreMethod`), data (`X` parameter) is not used and is not necessary,
  - `plot(vis_type)` – a method used to plot results in the form of the selected visualization. Requires executing `fit(model, X)`.

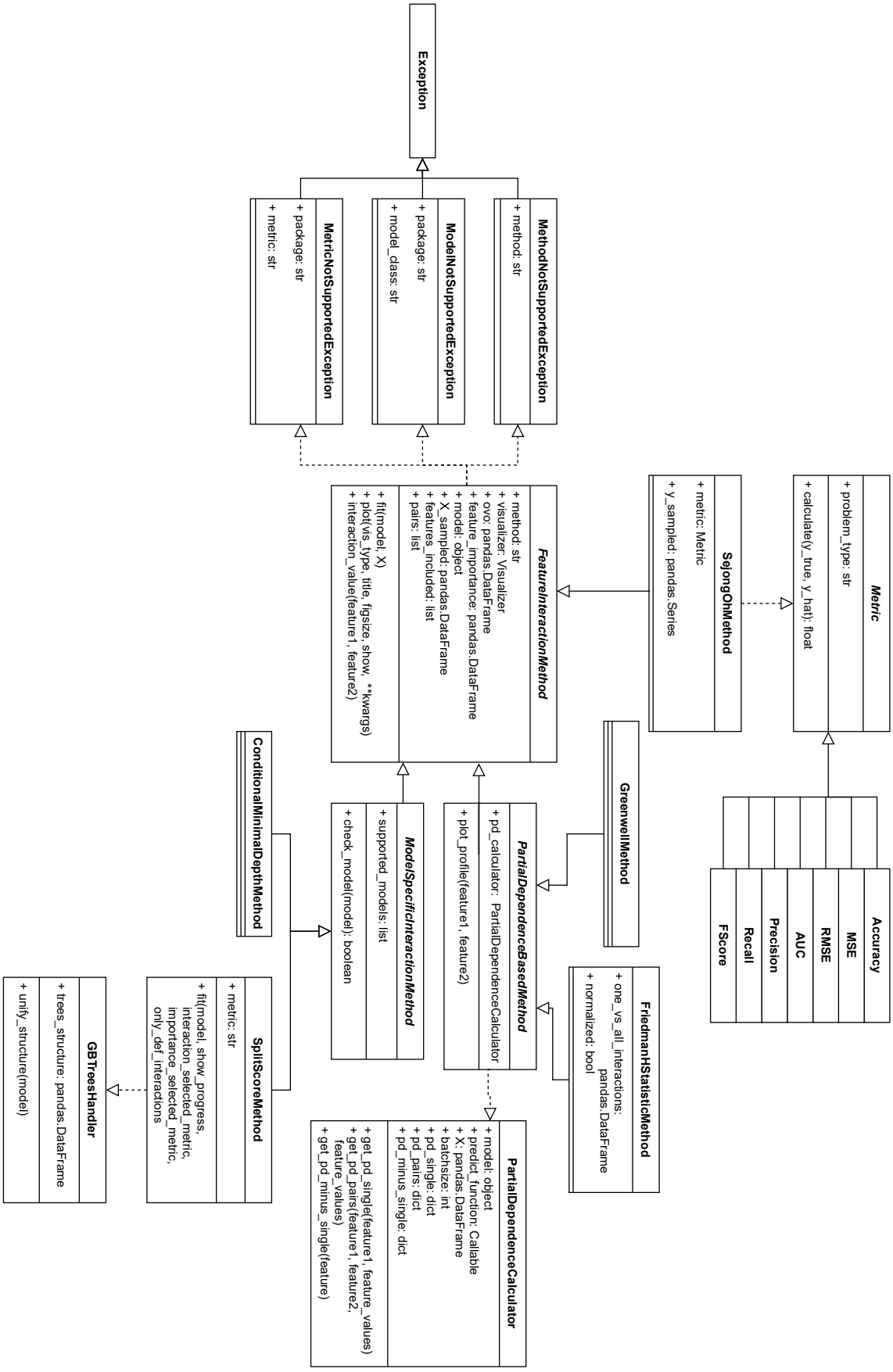

Figure 4.2: UML class diagram of `artemis` package part 1.

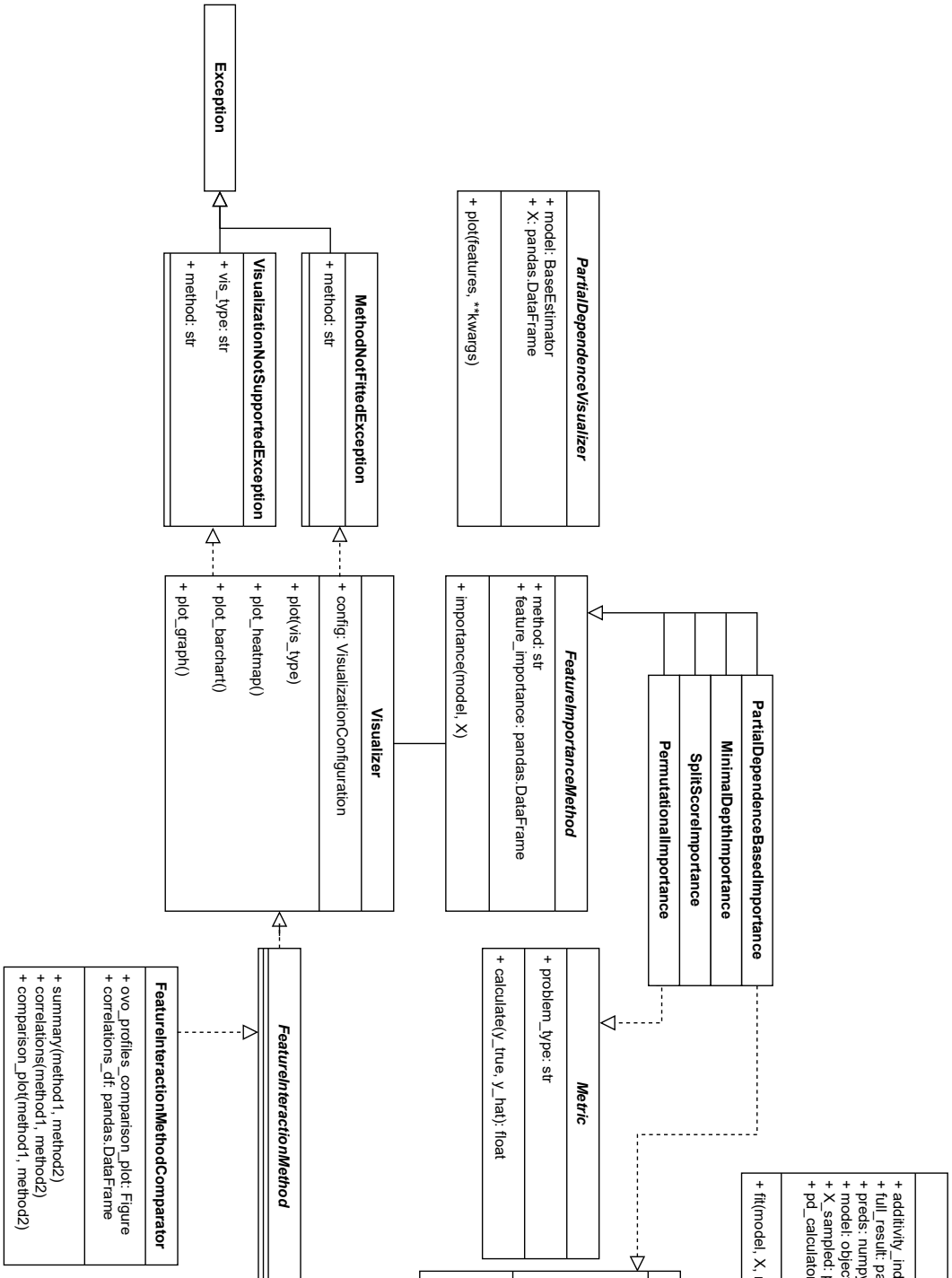

Figure 4.3: UML class diagram of `artemis` package part 2.

The methods available in the `artemis` package are divided into two groups: model-agnostic and model-specific. The model-agnostic methods, which are implemented in `SejongOhMethod`, `GreenwellMethod`, and `FriedmanHStatisticMethod` classes, are designed to be applicable to a wide range of models. The `GreenwellMethod` and `FriedmanHStatisticMethod` classes both inherit from the abstract class `PartialDependenceBasedMethod`, which implements functionality related to the calculation and use of partial dependence functions. On the other hand, the model-specific methods are explicitly designed for extracting interactions from commonly used tree-based models. These methods are implemented in the `ConditionalMinimalDepthMethod` and `SplitScoreMethod` classes. The tree-based models' structure, which reflects their creation method, allows for the extraction of more information than is possible using model-agnostic methods. A detailed mathematical description of the implemented feature interaction extraction methods can be found in Section 2.1.

Importance methods The package will complete the interaction picture with feature importance to avoid misleading interactions between features that do not significantly affect model predictions. Each feature interaction method will have a corresponding feature importance method. As they are similar in nature, `FeatureImportanceMethod` and `FeatureInteractionMethod` attributes and structure will be analogical (see Figure 4.2). `FeatureImportanceMethod` implementations will be exposed as part of the package API, but creators of the package do not recommend pairing importance and interaction methods differently than described below. The coherent nature of the pair significantly improves the explainability of the result. Default mapping between importance and interaction methods is shown in the Table 4.1.

Table 4.1: Default interaction - importance method pairings.

Importance method	**Interaction methods**
`PartialDependenceBasedImportance`	`GreenwellMethod` `FriedmanHStatisticMethod`
`MinimalDepthImportance`	`ConditionalMinimalDepthMethod`
`SplitScoreImportance`	`SplitScoreMethod`
`PermutationalImportance`	`SejongOhMethod`

Visualization The visualization module in the feature interaction library `artemis` plays a crucial role in enabling users to understand and analyze the relationships between different features in the data. The `Visualizer` class, specifically, allows users to create a range of visualizations, in-

cluding interaction graphs, one-vs-one interaction heat maps, and one-vs-all and one-vs-one interaction bar charts. These visualizations provide insight into the interactions between features and the importance of each feature, as determined by the corresponding `FeatureImportanceMethod`.

Technically, `Visualizer` class fetches visualization configurations from the `VisualisationConfigurationProvider` and creates the requested figure. If a requested visualization is not supported for a given method, the `Visualisator` will raise a `VisualisationNotSupportedException`. The visualizations used in the `artemis` package are described in Section 2.3 with a selection of examples.

Insights The insight module was introduced to provide users with additional tools for analyzing and understanding the feature interaction strength and congruency. One of the key features of this module is the ability to visually and numerically compare the results from different feature interaction methods. The `FeatureInteractionMethodComparator` plots the values from both methods on the x-y axis, allowing users to visually assess the degree of similarity between the two methods. Numerically, the rank correlation of the one-vs-one profiles (`KENDALL` by default) can be calculated to further compare the results of the different methods.

In addition, the `AdditivityMeter` allows users to calculate the additivity index for a model, which is a metric for assessing the extent to which information is carried through feature interactions. A fully additive model, with no feature interactions, will have an additivity index of 1. The mathematical foundations of these additional methods are described in more detail in Section 2.4. Overall, the insight module provides valuable tools for gaining a deeper understanding of the stability of the obtained results (comparing whether methods return congruent results) as well as measuring the relative influence of all feature interactions on the target.

4.3. Client interface and package workflow

Workflow To better understand the operation of the package and the communication between modules and classes, an additional description has been prepared, paying attention to describing typical package user workflow. The first aspect is a sequence diagram (Figure 4.4) showing the processes involved and the sequence of messages exchanged in the base package usage scenario.

The client (user) selects the interaction method of interest, initializes a new object, and receives its representation. Then executes calculations using `fit` method, providing model, data, and possibly other method-specific parameters. `FeatureInteractionMethod` calculates feature interactions values and requests `FeatureImportanceMethod` to calculate feature importance. The

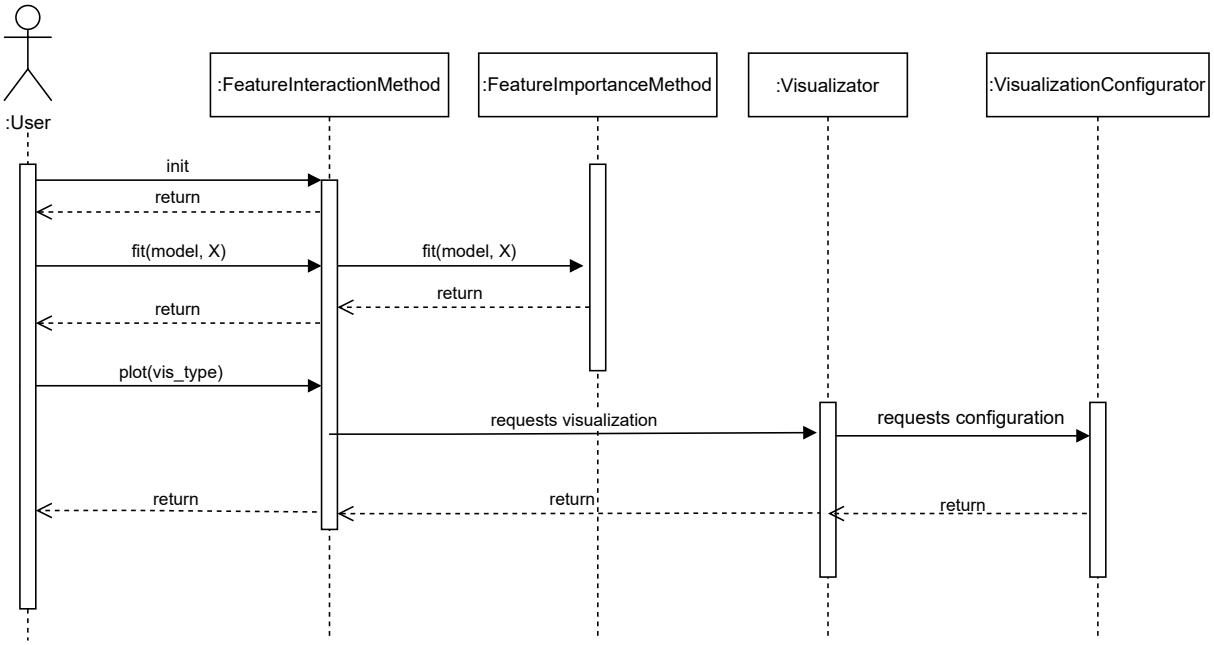

Figure 4.4: Sequence diagram for the interaction calculation procedure.

representation is modified. At this point, interaction and importance values can be accessed. Optionally, the user can trigger visualization methods to show figures of interest for further model insight. Those requests are passed to `Visualizer` object that can retrieve each visualization configuration from `VisualizationConfigurator` and plot the chart. Visualization configuration parameters can be overridden using standard Python `**kwargs` argument.

The sequential form of the workflow related to the preparation of the single explanation was illustrated in the activity diagram (Figure 4.5).

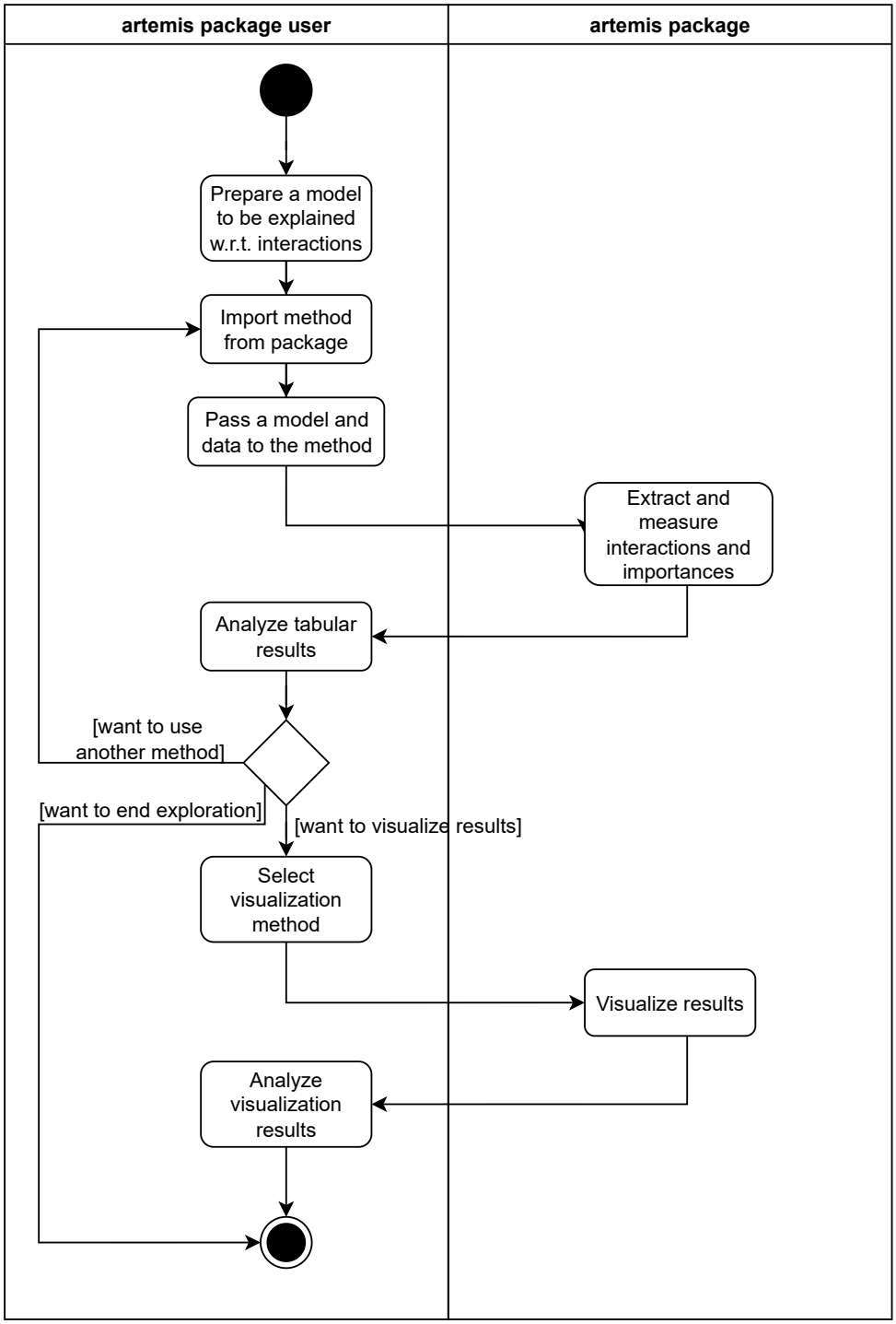

Figure 4.5: Activity diagram for the explanation procedure.

To complement the package description, a state diagram of the main object was also prepared (Figure 4.6). The main object in the package is an instance of the class inheriting from the `FeatureInteractionMethod` class. After the creation, the instance does not have information about the data or model. It contains only the configuration information for performing calculations. Calculations start after calling the `fit` method. If an error occurs during this process (e.g., the model is not fitted or it is incompatible with the selected model-specific method), the

process is aborted, and an exception is thrown (all custom exceptions are listed in 4.3). If the computation is successful, the `FeatureInteractionMethod` object enters the fitted state. If an object is in a fitted state, it object means that both feature interactions and importance are accessible. Such an object can be saved via serialization as `pickle` file or removed.

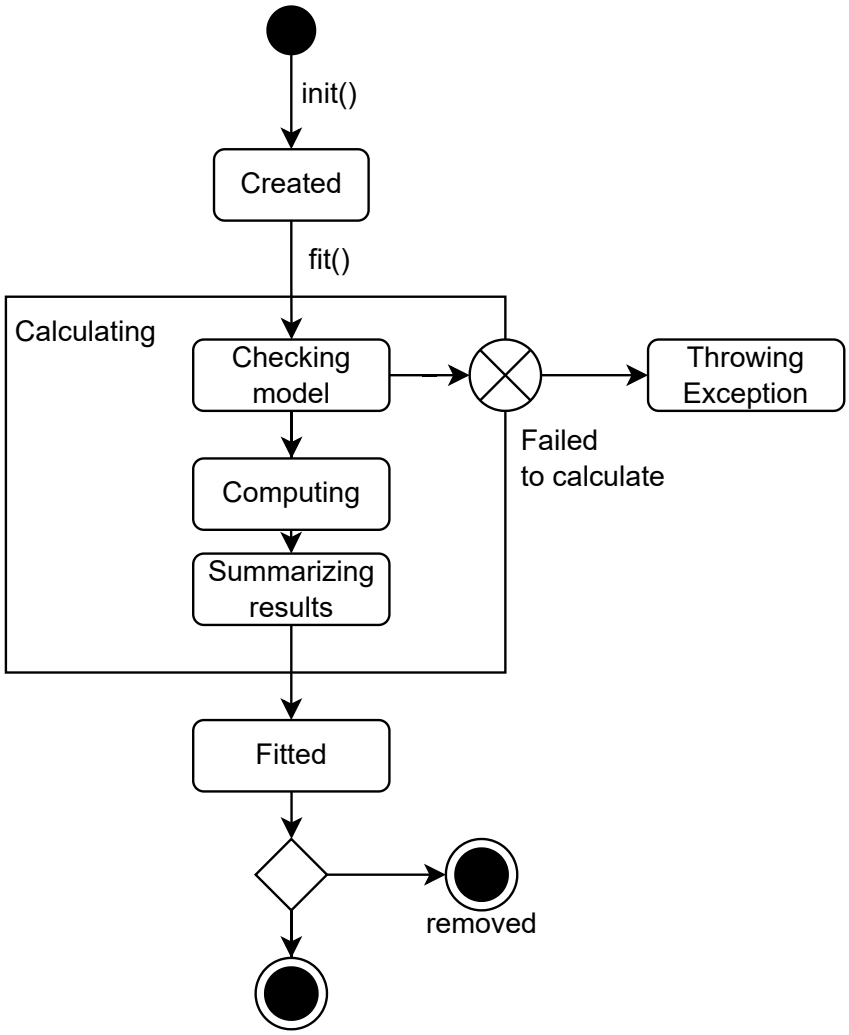

Figure 4.6: State diagram for `FeatureInteractionMethod` class object.

Effective error handling is a crucial aspect of the `artemis` package's workflow. The package provides meaningful error messages that enable the user to quickly understand and address issues, allowing them to focus on the complexities of the feature interaction methods. When the package encounters an unexpected stage during its workflow, it will raise an exception. These exceptions may be raised either by the `artemis` package itself or by its dependencies. The package also includes custom exceptions for handling specific error cases. Custom `artemis` exceptions:

1. `VisualisationNotSupportedException` – each feature interaction method supports a subset of all visualizations. Therefore, if the user requests a specific visualization (e.g. bar chart representing `one vs all` profiles) and this visualization is not supported by his method of

choice, the exception will be raised.

2. `MethodNotSupportedException` – if the user requests creating visualizations for a feature interaction method that does not exist, the exception will be raised.

3. `MethodNotFittedException` – if the user requests creating visualizations for a method that did not execute `fit`.

4. `ModelNotSupportedException` – if the user provides a model that is not supported for the feature interaction calculation.

5. `MetricNotSupportedException` – if the user provides an unsupported metric (applicable to `SejongOhMethod` and `SplitScoreMethod`)

Data types The data types returned and the API reference is clearly specified in the class diagram and the `artemis` documentation. This information is important for users to understand how to use and interpret the output of the various methods in the package. Additionally, it helps to ensure that users can effectively utilize the features and functionality of the library. Data types used in `artemis` are similar in nature to those from `scikit-learn`. It enables the user to integrate `artemis` into an already existing stack based on the most popular machine-learning library in `Python`. In addition, `artemis` works offline, so no network configuration or Internet access is necessary.

However, the key aspect to be clarified is the required input interface for model-specific feature interaction methods.

The `ConditionalMinimalDepthMethod` is applicable for the random forest implementations from the `scikit-learn` package. Supported models include:

- `RandomForestClassifier`,
- `ExtraTreesClassifier`,
- `RandomForestRegressor`,
- `ExtraTreesRegressor`.

Moreover, the method can also be applied to other models from `sklearn.ensemble` module that use decision trees (models from `sklearn.tree` module) as base estimators.

The `SplitScoreMethod` is applicable for gradient boosting decision trees implementations from `XGBoost` and `LightGBM` Python packages. These two implementations were selected due to their popularity and the possibility of extracting information about the scores of individual splits in the analyzed decision trees.

5. Experiments

This chapter presents the outcomes of experiments performed using the `artemis` package. Both synthetic data generated from known distributions and toy data with intuitive relationships and semi-realistic characteristics were utilized. The final section of this chapter offers a thorough assessment of the performance and scalability of the `artemis` package.

5.1. Synthetic datasets

In this section, the theoretical capabilities of implemented model-agnostic methods to detect feature interactions are tested on models solving regression and classification problems. In Section 5.1.1, the reference data model is a simple baseline function and in Section 5.1.2 two benchmark functions known from the literature are used. It is expected that the feature interaction methods implemented in `artemis` will be able to detect appropriate interactions.

Note that the explanations generated are supposed to be true for the model, not the data – so not all interactions need to be found by the model. However, for such simple functions, the most important interactions should coincide.

5.1.1. Baseline

In order to evaluate capabilities of feature interaction methods, we will conduct a baseline experiment on data generated according to Equation 5.1. The data includes three features: x_1, x_2, and x_3, and features x_1 and x_2 are in a non-linear interaction as demonstrated by the $b_4 x_1 x_2$ term. This study aims to compare the ability of model-agnostic methods (implemented in the `artemis` package) to extract interaction between x_1 and x_2 from models trained on data generated using a baseline function:

$$Y = b_1 x_1 + b_2 x_2 + b_3 x_3 + b_4 x_1 x_2 + \varepsilon, \tag{5.1}$$

where $x_1, x_2, x_3 \sim U(-5, 5)$, $[b_1, b_2, b_3, b_4] = [3, -7, 10, 2]$, $\varepsilon \sim N(0, 1)$.

The grid of charts (Figure 5.1) presents the interactions calculated for the linear regression,

random forest (`max_depth=25`, `max_features=2`) and neural network (`hidden_layer_sizes=(5, 2)`, `activation='relu'`) models trained on data generated from Equation 5.1. Greenwell, Oh, and H-statistic (both normalized and unnormalized) methods were used in the comparison.

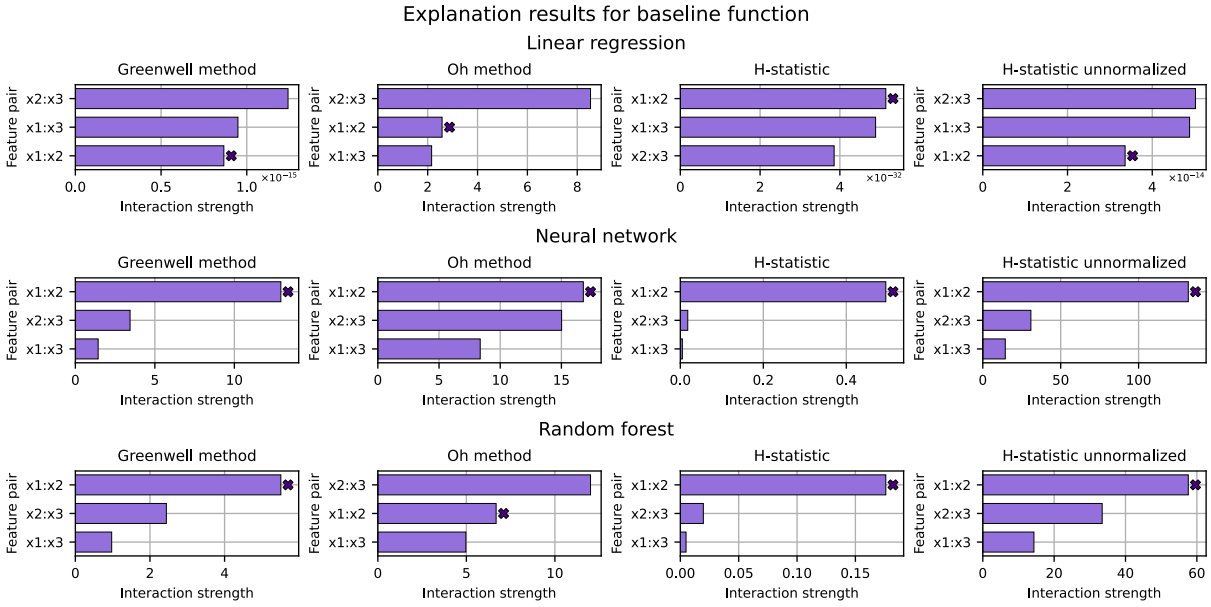

Figure 5.1: Results of model-agnostic methods applied to the linear regression, neural network, and random forest. Baseline function (5.1) with non-linear interaction between x_1 and x_2 was used. '$\times$' next to the feature pair denotes true interaction in the data.

Due to its nature and mathematical formulation, the linear regression model is unable to detect feature interactions. Therefore, as a sanity-check of interaction method correctness, we check if all methods have interaction values close to 0 (first row in Figure 5.1). As we can see, all methods except the Oh method correctly pointed to the negligible value of all interactions. This suggests the Oh method has high limitations and changes in model performance are not the best indicator of interaction in this case.

For neural network and random forest models, all of the partial dependence-based methods correctly indicated (x_1, x_2) pair as the one with the strongest non-linear feature interaction. It is worth noting that the normalized H-statistic has correctly pointed to the negligible interaction values for (x_2, x_3) and (x_1, x_3). Performance-based Oh Method has correctly selected (x_1, x_2) pair only for the well-performing neural-network model (although the rest of the interactions still have comparable values). It might suggest that the model performance might improve the Oh method's abilities to extract feature interactions.

Performance improvement Having discovered the interaction between x_1 and x_2, we decided to enrich data used for the training of models with a new feature: $x_4 = x_1 \cdot x_2$. Figure 5.2 presents

the change in mean squared error before and after adding the x_4 feature to the data for each model. We can see that adding detected interaction to the data significantly improved the performance of all the models. Of course, in this case, the target becomes a linear function of the features, so the MSE of the linear model is close to zero. It suggests that discovering interaction in the data not only improves the understanding and explainability of the model/problem but also may improve the model performance itself.

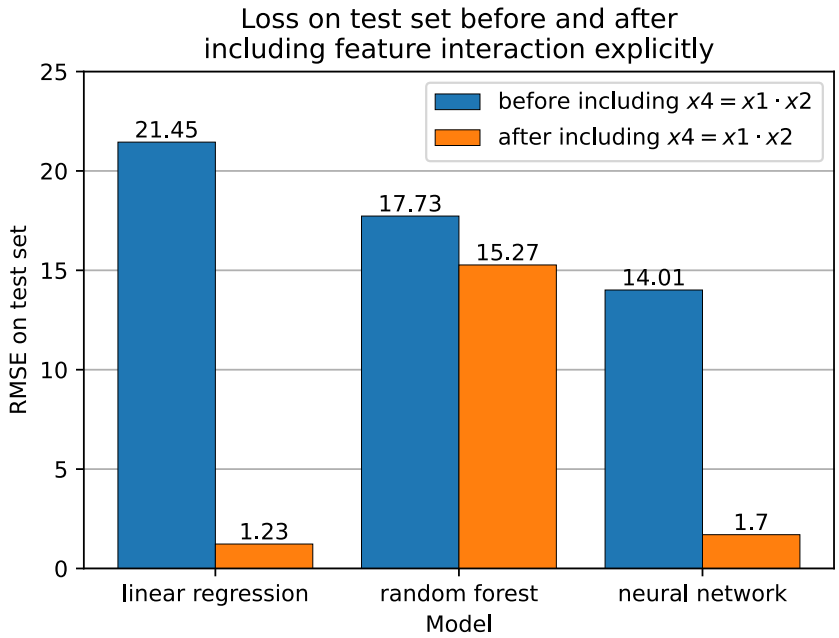

Figure 5.2: RMSE change on test dataset for each model after adding $x_4 = x_1 \cdot x_2$ feature. Discovering interaction and enriching the data might improve the model's performance.

The above example raises an important question about how in a real-life scenario should the `artemis` user know what function best represents the interaction between pair of features. In our case, data was generated from the known model, so we know that non-additivity is captured within the multiplication operation between x_1 and x_2. Currently, automatic function search is not implemented, but creating such functionality and combining it with `artemis` lays the foundation for future work. It is also the current topic in research work.

5.1.2. Benchmark

Setup In this section, we used ten more-complex functions known from the literature (see Table 5.1) to generate datasets for testing implemented methods for both the regression and the classification models. These functions are designed to contain predefined pair interactions and higher-order interactions, which vary in strength and form. All of them are based on formulas used before by Tsang et al. [72]. It is worth noting that one of the resulting datasets (related to

Table 5.1: Functions used in the benchmark sourced from [72].
For $F_2, F_3, \ldots, F_{10}$, $x_1, x_2, \ldots, x_{10} \sim U(-1, 1)$, whereas for F_1, $x_1, x_2, x_3, x_6, x_7, x_9 \sim U(0, 1)$ and $x_4, x_5, x_8, x_{10} \sim U(0.6, 1)$.

Name	Function
F_1	$\pi^{x_1 x_2} \cdot \sqrt{2x_3} - \sin^{-1}(x_4) + \log(x_3 + x_5) - \frac{x_9}{x_{10}}\sqrt{\frac{x_7}{x_8}} - x_2 x_7$
F_2	$\pi^{x_1 x_2}\sqrt{2\left\lvert x_3\right\rvert} - \sin^{-1}\left(0.5x_4\right) + \log\left(\left\lvert x_3 + x_5\right\rvert + 1\right) + \frac{x_9}{1+\left\lvert x_{10}\right\rvert}\sqrt{\frac{x_7}{1+\left\lvert x_8\right\rvert} - x_2 x_7}$
F_3	$\exp\left\lvert x_1 - x_2\right\rvert + \left\lvert x_2 x_3\right\rvert - x_3^{2\left\lvert x_4\right\rvert} + \log\left(x_4^2 + x_5^2 + x_7^2 + x_8^2\right) + x_9 + \frac{1}{1+x_{10}^2}$
F_4	$\exp\left\lvert x_1 - x_2\right\rvert + \left\lvert x_2 x_3 \right. \left\lvert - x_3^{2\left\lvert x_4\right\rvert} + \left(x_1 x_4\right)^2 + \log\left(x_4^2 + x_5^2 + x_7^2 + x_8^2\right) + x_9 + \frac{1}{1+x_{10}^2}$
F_5	$\frac{1}{1+x_1^2+x_2^2+x_3^2} + \sqrt{\exp\left(x_4 + x_5\right)} + \left\lvert x_6 + x_7\right\rvert + x_8 x_9 x_{10}$
F_6	$\exp\left(\left\lvert x_1 x_2\right\rvert + 1\right) - \exp\left(\left\lvert x_3 + x_4\right\rvert + 1\right) + \cos\left(x_5 + x_6 - x_8\right) + \sqrt{x_8^2 + x_9^2 + x_{10}^2}$
F_7	$\left(\tan^{-1}\left(x_1\right) + \tan^{-1}\left(x_2\right)\right)^2 + \max\left(x_3 x_4 + x_6, 0\right) - \frac{1}{1+\left(\prod_{i=4}^{8} x_i\right)^2} + \left(\frac{\left\lvert x_7\right\rvert}{1+\left\lvert x_9\right\rvert}\right) + \sum_{i=1}^{10} x_i$
F_8	$x_1 x_2 + 2^{x_3+x_5+x_6} + 2^{x_3+x_4+x_5+x_7} + \sin\left(x_7 \sin\left(x_8 + x_9\right)\right) + \arccos\left(0.9x_{10}\right)$
F_9	$\tanh\left(x_1 x_2 + x_3 x_4\right)\sqrt{\left\lvert x_5\right\rvert} + \exp\left(x_5 + x_6\right) + \log\left(\left(x_6 x_7 x_8\right)^2 + 1\right) + x_9 x_{10} + \frac{1}{1+\left\lvert x_{10}\right\rvert}$
F_{10}	$\sinh\left(x_1 + x_2\right) + \arccos\left(\tanh\left(x_3 + x_5 + x_7\right)\right) + \cos\left(x_4 + x_5\right) + \sec\left(x_7 x_9\right)$

function F_1) is the most popular synthetic dataset used in the literature on feature interactions in predictive models. It was first used by Hooker [39], then also by Sorokina et al. [70] and Tsang et al. [72]. We analyze the results for the function F_1 in detail, while we aggregate the others to draw more general conclusions about the performance of the implemented methods.

Experiments for the classification problem with synthetic data are lacking in the literature, even in works introducing new methods for the extraction of interactions. Therefore, we decided to test the capabilities to extract feature interactions from classification models using the same functions to generate the datasets as for the regression tasks. We employed a standard logistic function (inverse-logit, Equation 5.2) to transform the function output into probabilities and then used these probabilities to draw a target for each observation by sampling from a Bernoulli distribution with a given probability of success:

$$\text{logit}^{-1}(x) = \frac{\exp\left(x\right)}{\exp\left(x\right) + 1}. \tag{5.2}$$

In this experiment, for the F_1 function, we use the *ideal* model (i.e., model that uses the data-generating functions for prediction) to derive ground-truth values of interactions for regression problems. This allows checking which interactions are actually the most relevant. However, as in the actual use case of these methods, the form of the function is not known explicitly, we benchmark the methods on the machine learning models, in this case, using a random forest algorithm – we apply explanations of such models for all functions in the benchmark. The hyperparameters

of the random forest models are chosen arbitrarily, setting the number of decision trees at 100, the maximum tree depth at 10, and the number of features to consider when looking for the best split at $\sqrt{p}$ where p is the number of features in the model. It is important to note that the explanations for random forest models do not have to necessarily coincide with those of *ideal* model. For each of the ten functions $(F_1, \ldots, F_{10})$ and the two tasks (regression, classification), we generate 5000 observations, 90% of which are used as a training set and 10% as a test set on which explanations are generated.

Analysis of the results for F_1 function Since the F_1 function is the most common function for testing algorithms for extraction of interactions, we analyze the results for it in detail. Table 5.2 presents expected pairs of features in interaction for this dataset, with respect to their constituent in the formula from Table 5.1.

Table 5.2: Constituents from F_1 function and expected feature interaction pairs.

Constituent	**Pairs of interacting features**
$\pi^{x_1 x_2} \cdot \sqrt{2x_3}$	$(x_1, x_2), (x_1, x_3), (x_2, x_3)$
$\sin^{-1}(x_4)$	—
$\log(x_3 + x_5)$	(x_3, x_5)
$\frac{x_9}{x_{10}}\sqrt{\frac{x_7}{x_8}}$	$(x_7, x_8), (x_7, x_9), (x_7, x_{10}), (x_8, x_9), (x_8, x_{10}), (x_9, x_{10})$
$x_2 x_7$	(x_2, x_7)

The results of the explanations for the ideal model for the regression task presented in Figure 5.3 show that the implemented methods can mostly extract the pairs of features between which there are interactions. Greenwell method, H-statistic and unnormalized version of H-statistic returned non-zero interaction strength measure values only for the expected pairs of features. Only the results obtained with the Oh method deviate from the others due to how interactions are defined in this approach. On the other hand, it is worth noting that no interaction appeared with a feature not used to generate output, i.e. feature x_6. This indicates that all methods satisfy the property that if feature x_j does not contribute to any prediction of the model, all pairs of features containing x_j have an interaction strength of 0 (dummy feature property analogy from the SHAP method [49]). It should also be noted the high agreement between the interaction strength rankings obtained as the results of the Greenwell method and the unnormalized H-statistic.

The results for the random forest regression model trained on the dataset generated from the F_1 function (shown in Figure 5.4) confirm the finding about the outlying results coming from the

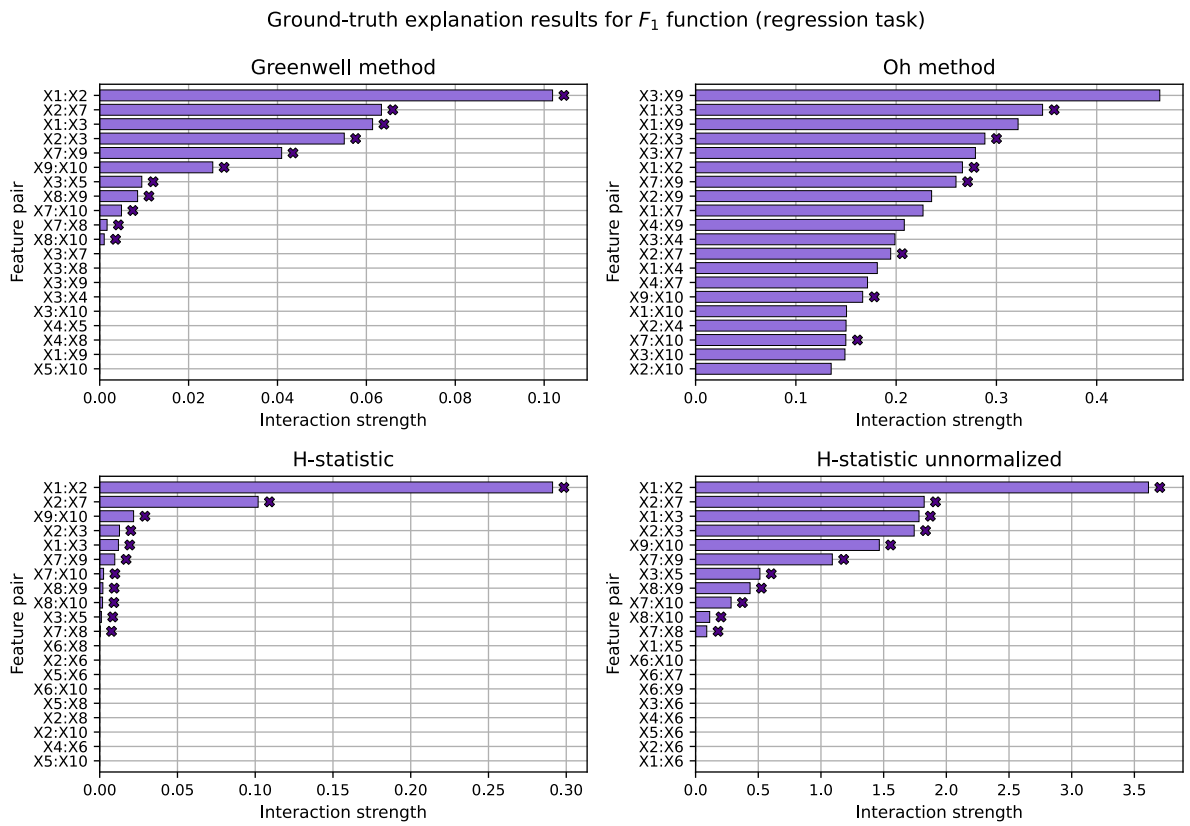

Figure 5.3: Result of model-agnostic methods applied to the *ideal* models for F_1 function. Top 20 feature pairs according to each method are presented. '×' next to the feature pair denotes true interaction in the data.

Oh method. All other methods identified the pair of features x_1 and x_2 as the most important interaction, reflecting the ground-truth interaction values. Moreover, all PD-based methods indicated the top 6 strongest interactions actual ones, i.e., those defined in the data. Among these interactions are the remaining pairs from the trio (x_1, x_2, x_3), indicating the high significance of this three-way interaction. According to the Greenwell method and the unnormalized H-statistic, the three corresponding pairs are even the most important interactions. Here again, the consistency between the results of these two methods can be seen here. The expected interaction pair that caused the most problems for the methods is the (x_8, x_{10}) feature pair, which does not appear in the top 20 interactions for any of the methods. This may be related with its low significance even in the ground-truth case, which can be further explained by the occurrence of both of these features in the denominators in the F_1 function and their distribution, which combined results in a small effect on the F_1 values. The explanations for this model and dataset show that the H-statistic is the most conservative, and the values of this measure vary the most.

As the three methods indicate the interaction between features x_1 and x_2 as the most crucial, it may be seen as a suggestion to perform feature engineering and take this pair interaction

directly into account. Indeed, in this case, after adding the product of features x_1 and x_2 as a separate feature $x_{new} = x_1 \cdot x_2$ (without any additional transformations), the performance of the model improved significantly. More specifically, the mean absolute error dropped from 1.43 to 0.19, while the mean squared error decreased from 2.57 to 0.06.

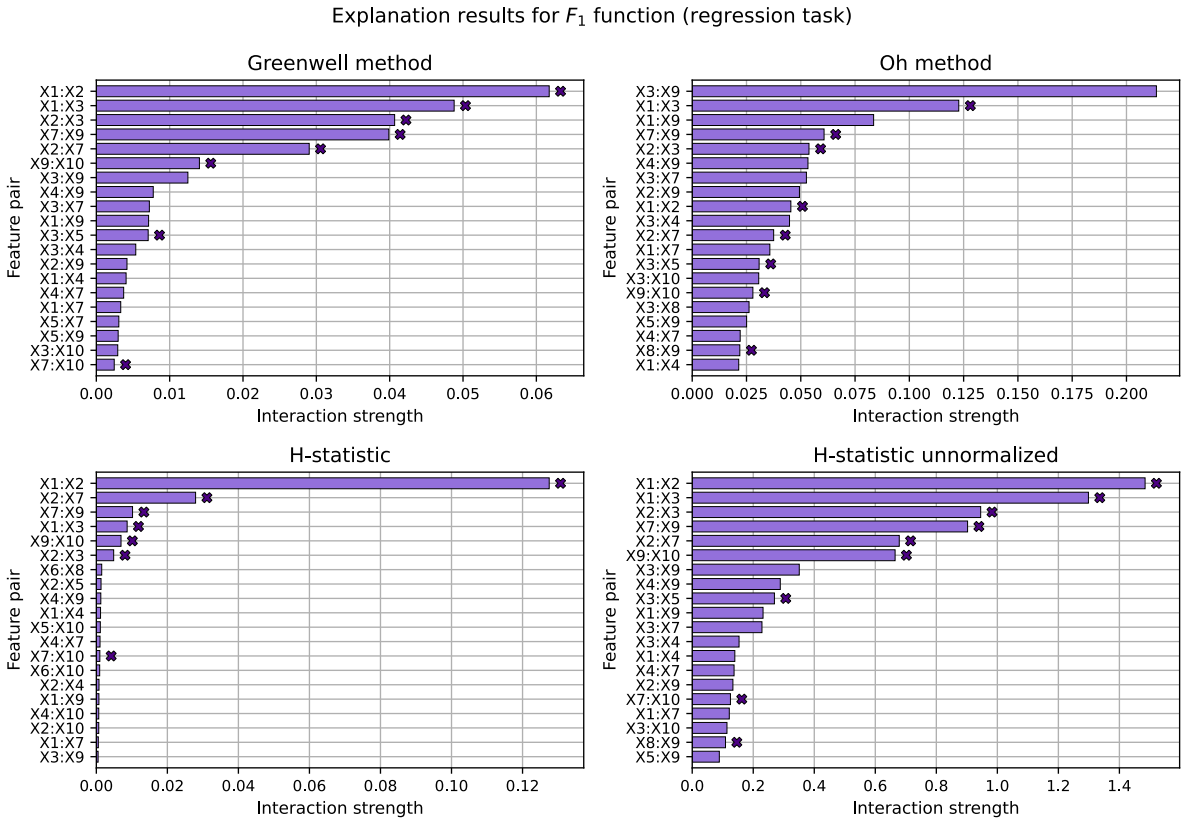

Figure 5.4: Result of model-agnostic methods applied to the random forest model trained on the dataset generated by F_1 function. The top 20 feature pairs according to each method are presented. '×' next to the feature pair denotes true interaction in the data.

The results for a classification task based on the same data-generating function with an appropriate transformation look noticeably different, which can be seen in Figure 5.5. The number of correctly indicated interactions is clearly lower. It can also be seen that the values indicated by all methods are more uniformly distributed. The pair (x_1, x_2) remains the most important pair interaction only according to the indications of the H-statistic. In contrast, it can be seen that this method have a problem with the other two-way interactions. Namely, among the top 20, it identified the fewest pairs that are true interactions in the data. Apart from the H-statistic, only the Oh method indicated a true interaction in first place. For the Greenwell method and unnormalized H-statistics, the pair of features (x_3, x_9) comes in first place. Interestingly, according to the Oh method, it is the second most important interaction (also indicated in the regression problem), but it does not appear at all in the top 20 pairs

according to the H-statistics values. This shows how much variation there can be in the results of different methods, even those based on similar concepts like partial dependence functions. It is thereby important to study more broadly how methods perform for different functions.

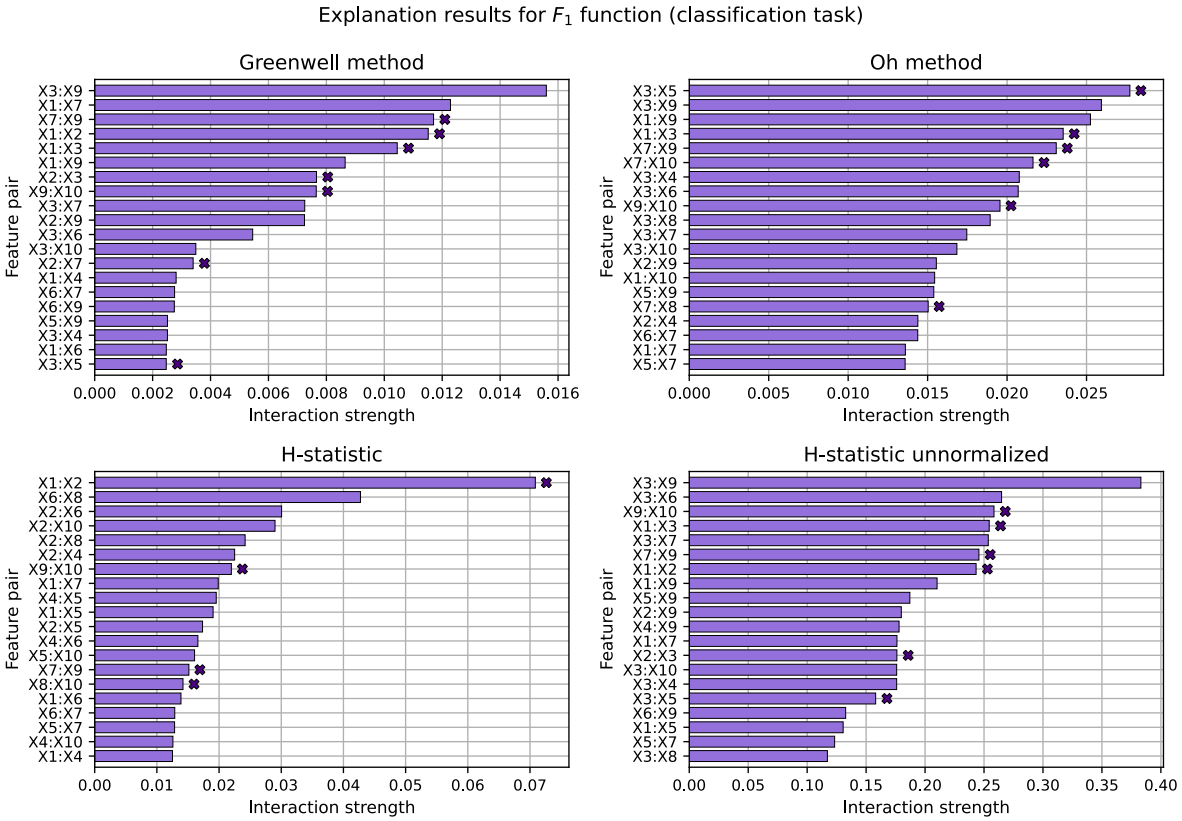

Figure 5.5: Result of model-agnostic methods applied to the random forest model trained on the dataset generated by F_1 function and transformation to a classification problem. The top 20 feature pairs according to each method are presented. '×' next to the feature pair denotes true interaction in the data.

Benchmark results For more general conclusions on the operation of the model-agnostic methods for extraction of interactions, we performed experiments on the remaining functions and analyzed them in terms of the results obtained using selected metrics. The well-known concept of metrics calculated based on the confusion matrix was used to summarize the achieved outputs of conducted experiments. For simplicity, a binary setup was adopted resulting from the use of an indicator determining whether a given pair of features is expected to interact (at least by the definition of the functions).

Testing various thresholds on interaction strength values, we created receiver operating characteristic curves for each dataset and method, which allowed calculation of the are under the curve (AUC). Figure 5.6 illustrates the distribution of AUCs for 10 datasets for regression and classification tasks. According to this metric, considering the median of the values obtained, H-statistics can be identified as the best method for both regression and classification tasks. On the other hand, a look at the distribution shows that the spread of AUC values is quite large. For the regression problem, the difference between the maximum and minimum value is similar for each method (about 0.5-0.6), while for classification problems, one can see greater fluctuations in the performance of the results obtained using the H-statistics and its unnormalized version. In both cases, the Oh method based on model performance metrics is the worst. In particular, in the case of regression tasks, the results obtained with this method are noticeably poorer than with the other methods considered.

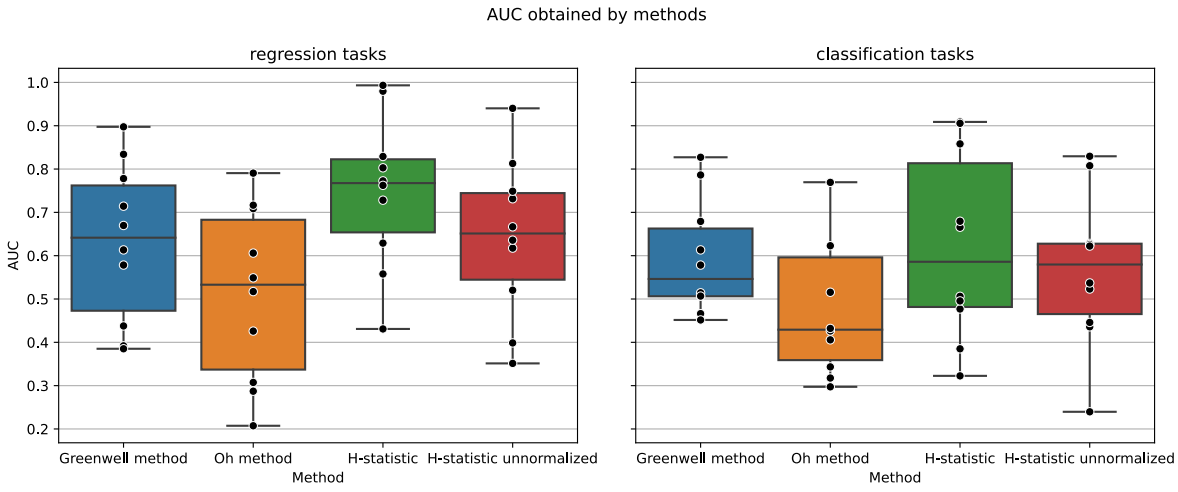

Figure 5.6: Distribution of AUC values of the two-way interaction strengths calculated by the analyzed model-agnostic methods for regression and classification tasks.

A more detailed overview is provided by analyzing the AUC results for specific datasets (see Table 5.3 for values for regression tasks and Table 5.4 for values for classification tasks). In both cases, the H-statistic is also the best in terms of the number of datasets for which its AUC is

the highest – more precisely, 5 for regression and 4 for classification. Moreover, in the regression setting, the AUC for the H-statistic is never the lowest compared to the values for the other methods in the same task. In the classification setting, such a situation occurs twice. It is worth noting that the Oh method did not prove to be the best in any of the problems analyzed.

Table 5.3: AUC of the values of the two-way interaction strengths calculated by the analyzed model-agnostic methods for regression tasks created on benchmark functions.

Function	**Greenwell**	**Oh**	**H-statistic**	**H-statistic unnormalized**
F_1	0.778	0.709	0.773	**0.813**
F_2	0.578	0.606	**0.629**	0.618
F_3	0.670	0.426	**0.762**	0.636
F_4	0.714	0.517	**0.803**	0.731
F_5	0.392	0.307	**0.980**	0.351
F_6	0.385	0.287	**0.993**	0.520
F_7	**0.438**	0.207	0.431	0.399
F_8	**0.834**	0.717	0.728	0.749
F_9	0.613	0.549	0.558	**0.667**
F_{10}	0.897	0.791	0.829	**0.940**
average	0.630	0.511	**0.749**	0.642

On the other hand, the performance of the methods can also be studied while paying more attention to which pairs of features are indicated as the most important interactions, as these are usually the ones a potential stakeholder pays attention to. Thus, Figure 5.7 and Figure 5.8 present an analysis related to the sensitivity of benchmarked methods by showing how many of the expected feature pairs were indicated in the top k pairs with the largest values of the metric these methods compute. These charts show that in most cases (related to tasks, datasets, and methods used) the pair indicated first actually reflects the interaction defined in the data. In the regression, the counter situation occurs only for the F_2 function with all methods except H-statistic and the F_1 function with the Oh method. Therefore, the H-statistic indicates the actual interaction as the most important pair every time. A similar property of this method can be observed in the case of classification tasks. In contrast, in this setting, there are more deviations from this expected property for other methods.

Table 5.4: AUC of the values of the two-way interaction strengths calculated by the analyzed model-agnostic methods for classification tasks created on benchmark functions.

Function	**Greenwell**	**Oh**	**H-statistic**	**H-statistic unnormalized**
F_1	**0.679**	0.623	0.385	0.628
F_2	**0.578**	0.426	0.477	0.523
F_3	0.506	0.432	0.506	**0.537**
F_4	0.514	0.406	**0.666**	0.626
F_5	0.466	0.297	**0.858**	0.436
F_6	0.507	0.318	**0.909**	0.446
F_7	**0.452**	0.343	0.323	0.240
F_8	0.827	0.770	**0.906**	0.829
F_9	0.613	0.516	0.496	**0.622**
F_{10}	0.786	0.769	0.679	**0.808**
average	0.593	0.490	**0.620**	0.569

In summary, the benchmark performed and the analyses conducted allow us to recommend the H-statistic as the first choice when looking for a method for extraction of interactions from predictive regression models. Additionally, when it comes to classification models, H-statistic is also worth considering, but if the results obtained are unintuitive and questionable, the Greenwell method and the unnormalized version of H-statistics are also worth checking. When using implementations from the developed `artemis` package, the time and computing power requirements for these methods are reduced, thanks to the ability to transfer previously calculated partial dependence function values.

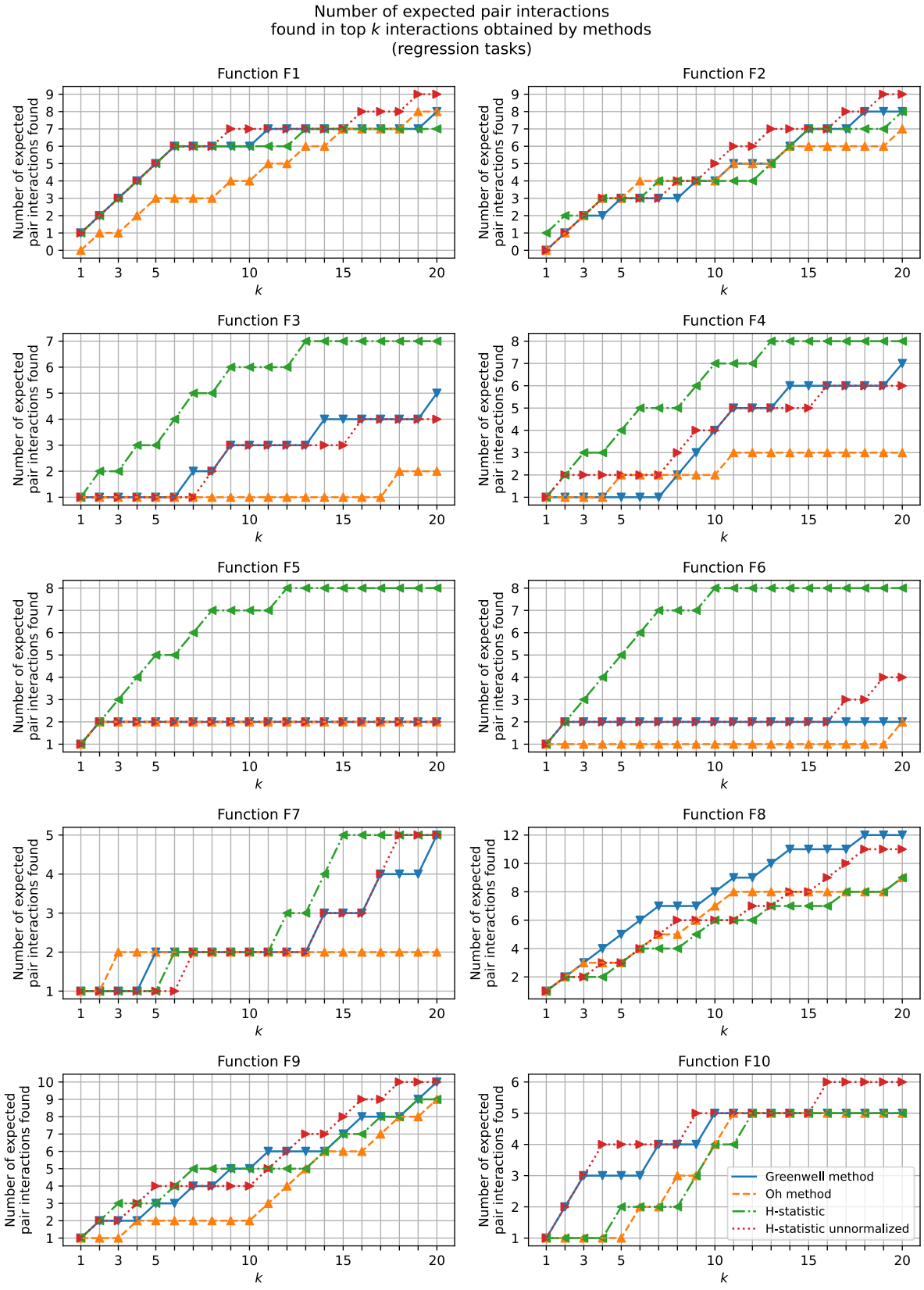

Figure 5.7: A sensitivity-based summary of the results of explanations for regression tasks created on benchmark functions. A number of expected pair interactions indicated by considered methods in top k pair interactions.

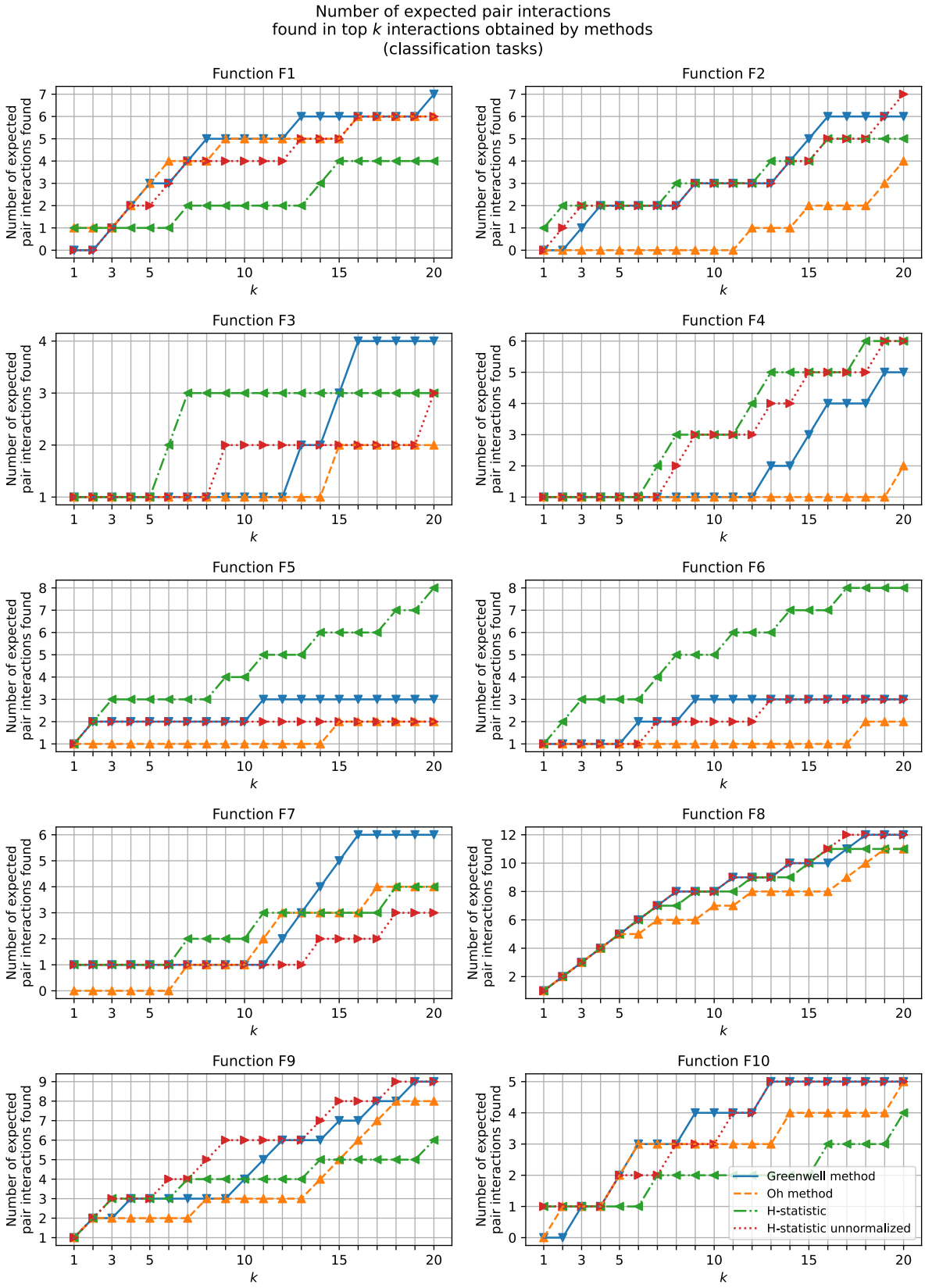

Figure 5.8: A sensitivity-based summary of the results of explanations for classification tasks created on benchmark functions. A number of expected pair interactions indicated by considered methods in top k pair interactions.

5.2. Toy datasets

In this experiment, the `artemis` package is used to analyze and interpret the machine learning model that has been trained on the `California Housing` dataset. This dataset is widely used for machine learning experiments, particularly regression tasks. It consists of information on various features of houses in California, including the location coordinates, number of bedrooms and bathrooms, and the area's median income, among others.

To verify the capabilities of the `artemis` package on this dataset, we conducted an experiment in which we trained a `XGBoost` model on the data and applied several interaction extraction methods from the `artemis` package. The results of two selected methods, the model-agnostic Friedman H-statistic, and the model-specific Split Score were plotted in Figure 5.9. Plots showed a strong interaction between the Longitude and Latitude features. These findings confirm the importance of location and interaction between coordinate features in determining house values. They are supported by Figure 5.10, presenting the partial dependence plot and a map showing the location and price of houses in California. The distinct shape of the California coast plotted

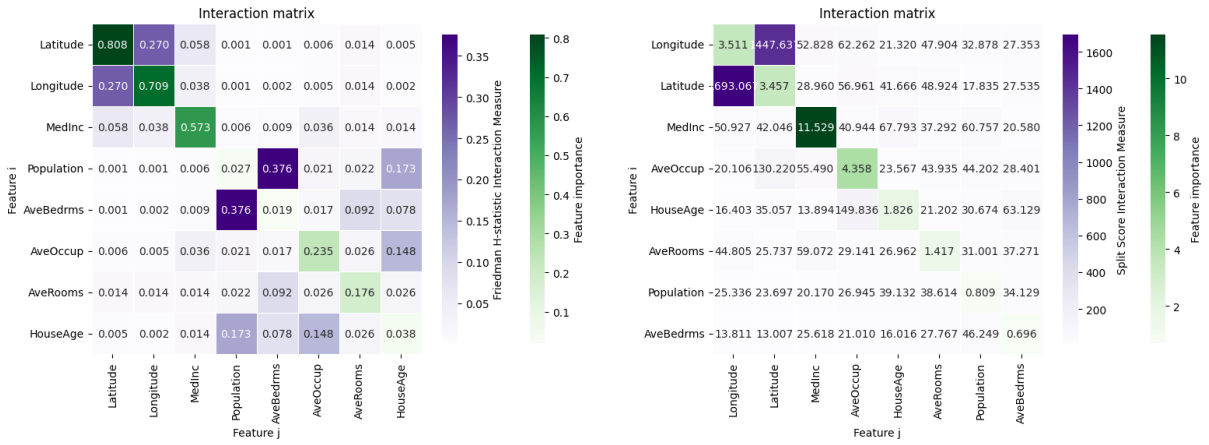

Figure 5.9: H-statistic and Split Score one vs. one interactions.

in the partial dependence plot Figure 5.10 highlights the higher prices of houses in this area, while the map shows that houses in renowned districts tend to be priced higher than those in other areas. These results suggest that real estate agents and home buyers should consider the location of a house when determining its value, as it can significantly impact the price.

Overall, this experiment demonstrates the effectiveness of the `artemis` package in extracting interactions between features and identifying essential relationships that may not be obvious. The strong interaction between the Longitude and Latitude features confirms the importance of location in the housing market and the value of using the `artemis` package to understand better the factors that influence house values.

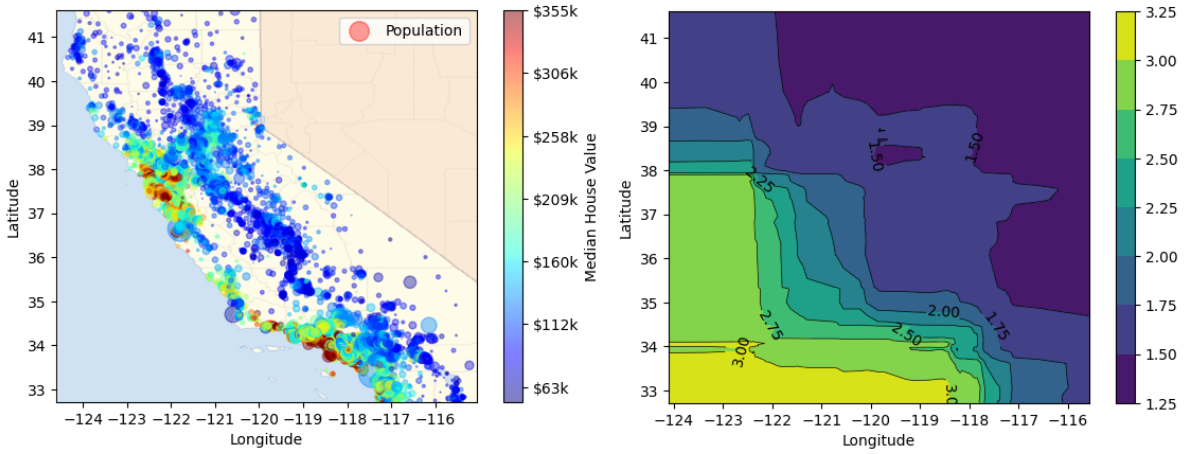

Figure 5.10: California map with house prices and two-dimensional partial dependence plot for latitude and longitude.

5.3. Assessment of the solution

In this section, we will evaluate the performance, scalability, and user-friendliness of the `artemis` package. The package has been developed based on best practices in both scientific research and software development, and any limitations are due to the inherent nature of the methods. We will analyze the package's performance in terms of execution time, as well as its scalability in handling large datasets. Additionally, we will examine the ease of use of the package through its API design and documentation.

Performance, scalability To improve the efficiency and scalability of the `artemis` feature interaction extraction Python package, we have optimized various calculations, such as partial dependence and conditional minimal depth search, and provided reasonable default settings. This allows users to work in a near real-time manner without experiencing significant delays. However, it should be noted that the performance of each method may be affected by the number of samples, number of features, or tree structure (in the case of tree-based methods). Figures 5.11 and 5.12 show the time complexity comparison of different model-agnostic methods with respect to the size of the data and the number of features, respectively. It should be noted that tree-based methods are only impacted by the tree structure, and are not affected by the size of the data (and therefore are not included in the Figures).

In terms of performance, the Friedman H-statistic is most sensitive to the number of samples, while the Sejong Oh method is most sensitive to the number of features. On the other hand, the Greenwell method performs particularly well for large datasets, both in terms of the number of features and the number of samples, and appears to be numerically stable. It is worth noting

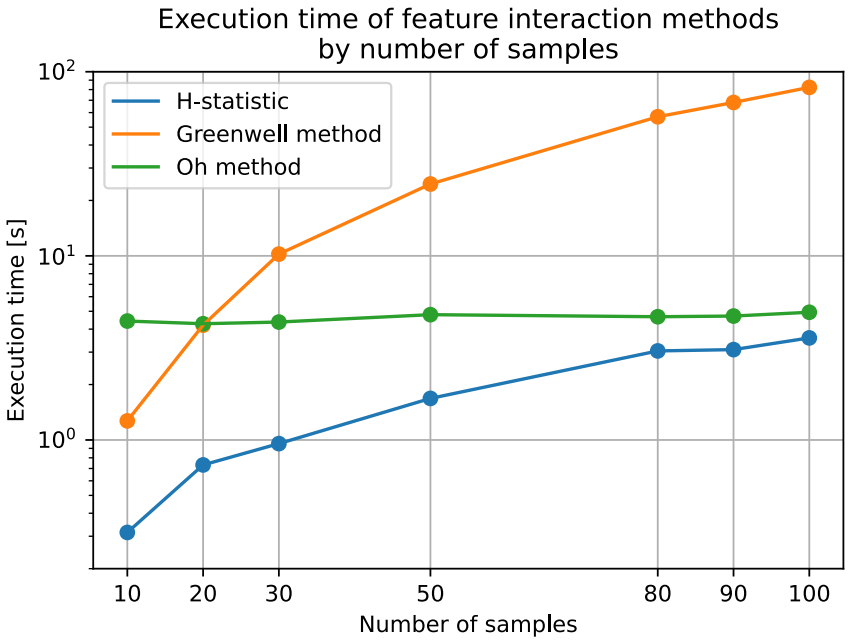

Figure 5.11: Scalability of feature interaction methods: execution time as a function of sample size.

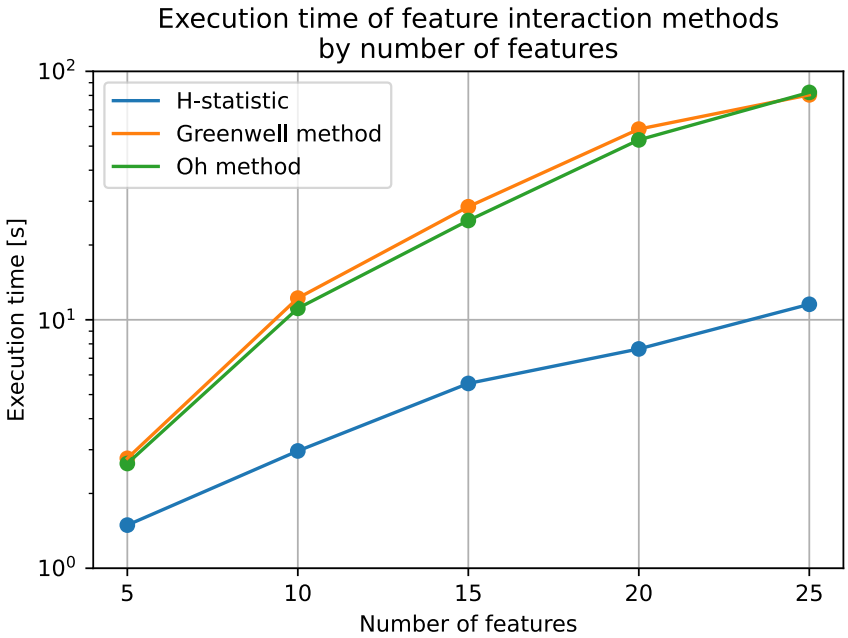

Figure 5.12: Scalability of feature interaction methods. Execution time as a function of a number of features.

that the sensitivity of each method to the number of samples and features may vary depending on the specific characteristics of the dataset.

User friendliness To facilitate the integration of the `artemis` package into the existing codebase and enable users to learn quickly, the API has been designed to be similar to that of

`scikit-learn` [15], a widely popular machine learning package in Python. An example of how to use the method is provided below:

```
from artemis.interactions_methods.model_agnostic import FriedmanHStatisticMethod
h_stat = FriedmanHStatisticMethod()
h_stat.fit(model, X, 500, show_progress=True)
# model implements ‘predict‘ function, X - data
```

This thesis and accompanying package documentation was designed to enable users to easily understand the functioning of each feature interaction method and how to use it. The goal is to facilitate the use of these methods for users who may not have a deep understanding of their underlying mechanics.

6. NASK StyloMetrix use case

This chapter discusses the application of implemented interaction extraction methods to analyze and explain predictive models trained on StyloMetrix features. We describe what the process of using such explanation methods in this real-world application looked like and what effects it had.

6.1. Background

Stylometry is a branch of natural language processing (NLP) that focuses on statistical and linguistic methods to analyze and identify unique styles of different texts or authors [66]. Stylometry usually uses a text representation composed of features extracted directly from text statistics, such as word frequencies, sentence length, and grammatical patterns. Stylometric vectors can be conceived as yet another type of text embeddings in addition to semantic-based approaches like Word2Vec [51] and GloVe [63], and more recent context-sensitive embeddings based on ELMo [64] or BERT [22]. They can capture words, paragraphs, or entire texts. However, the stylometry-based approach has one substantial advantage over such (often giving better model performance) embeddings. Namely, all of the stylometric features are easily interpretable, calculated, and constructed with the use of pre-trained parsers or taggers, not requiring large language models.

The predominant application of stylometry is authorship attribution or text genre classification tasks [57]. However, stylometric vectors can also be used for other tasks like content moderation (detecting harmful content, i.e., hate speech). It is precisely the application of the stylometric approach to modeling textual data that has been addressed at the NASK National Research Institute.

The project aims to develop a machine learning-based technology for content classification and moderation, which can improve the quality of content on websites and allow for increased revenue from improved ad sales services displayed next to the native content. The system created is being developed and tested based on text data from the Wykop website, which is a social bookmarking

site and Polish equivalent of Reddit (www.reddit.com) and Digg (www.digg.com).

Since the vast majority of texts on the Wykop website are written in Polish, it is necessary to use a stylometric tool adapted to this language. Such an open-source package is the Python library StyloMetrix developed at the NASK NLP Department lead by Okulska and Zawadzka [59]. It creates text embeddings based on linguistic features such as the occurrence of parts of speech, gradation of adjectives and adverbs, or flexion-related characteristics. The embedding size is related to the number of defined metrics. However, it is possible to define new ones and increase the dimensionality of vectors without recalculating the previously calculated values. Unlike other known stylometric libraries for the Polish language, like Stylo [24] for R, the metrics are written by human experts and not based on the TF-IDF or other word or punctuation frequency matrices.

The proposed solutions are thus based on interpretable embeddings (each metric translates directly into a specific grammar pattern). However, the used models remain complex and hard to follow for human calculation capacity. It affects the need to apply methods of explanation, including techniques for extracting the feature interactions, which are crucial from the point of view of the linguistic description of the text.

6.2. Explanation process

The process of applying the techniques of explainable artificial intelligence was included in the creation of the solution as part of the audit of the trained models. The goal was to better understand black-box models and verify whether they are properly reasoning. This can be understood as part of the evaluation step in the model audit phase according to the Model Development Process [10]. In the case of such a delicate matter as hate speech, it is crucial to monitor the process closely and not overlook important trends.

Since the project does not assume the use of only one model, various models and their committees (including models based on other embeddings) are currently being tested. Hence, the process of applying explanatory methods has been multi-threaded as well as iterative and associated with the use of human-in-the-loop methodology [76]. More precisely, explanations were initially generated for simple models, taking into account only stylometric features, and then for more and more complex models that also used other features, such as the presence of certain characters, and keywords, or also used various embeddings juxtaposed. Explanations were generated not only for individual model instances but also for their ensembles.

Due to their computational cost, explanations for feature interactions were generated for the feature selected as the most important based on previous explanation results (with importance

measured on the basis of partial dependence profile variability and permutation feature importance technique). In addition, the possibility of sharing results between different methods based on the partial dependence functions, offered by the created package was used, which significantly shortened the time of obtaining results.

6.3. Example results of explanations

The following is a selected analysis example, which is one iteration of the process described earlier. The analyzed scenario concerns a model based on data containing 89 features extracted with the StyloMetrix package and 31 additional text properties, such as whether it contains hyperlinks, the number of words, and the inclusion of specific keywords, including vulgarisms. The data refers to 27,223 texts that are real user posts from the Wykop website. Similarly, the explained model was created as part of the NASK project and not just for this work. So this is an example of a real-world application of explanations focusing on interactions.

The data includes posts classified into one of four classes. One of them is the neutral class containing posts that do not violate the site's rules and regulations. Three classes refer to posts that have been reported and marked by moderators as actually harmful. These classes are: (1) promoting hatred or violence, drastic content, (2) advertising content, spam, (3) attacking or violating my personal rights. It is worth noting that while each post is matched to one group, it may actually contain the marks of multiple offences against the standards. In addition, the inherent subjectivity of labeling influences the fact that the task is difficult. Thus, in some experiments, classes related to posts that violate the rules are combined into one large class. Moreover, in the evaluation of models, it is not only their performance that matters but also their interpretability and way of reasoning.

The analyzed model is a voting classifier composed of three sub-models created for the problem of binary classification. The ensemble includes logistic regression with L_2 penalty, a random forest classifier with 100 trees and an AdaBoost [28] classifier with 50 trees. The soft method based on the probabilities returned by included models was chosen as the voting rule. The results achieved by this voting classifier are shown in Figure 6.1. They were generated on a test set representing 25% of the available data.

Each model in the analyzed ensemble has its own predictive function so it can be analyzed separately. However, for a more holistic view of the model (and for the sake of clarity), we present explanations for the entire voting classifier, i.e., explanations based on the use of probabilities returned by the metamodel. Due to a large number of features, which affects the high compu-

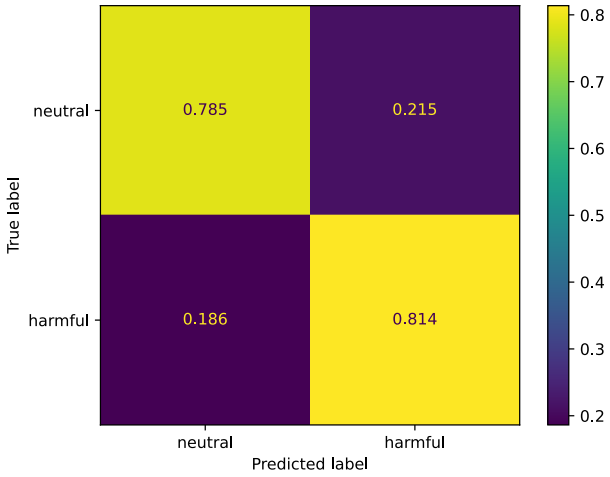

Figure 6.1: Confusion matrix with normalization by class support size for the analyzed model.

tational cost of generating explanations, but also causes the cognitive problems related to the reception of explanations, i.e., their analysis, measures of interaction strength were calculated only for selected pairs of features. Thus, we present explanations obtained using the H-statistics method for pairs formed from the 30 most important stylometric features, where the importance was examined by a PD-based method implemented in the `artemis` package. A full list of these features (in order from most to least important), along with descriptions of meaning, is included in Table 6.1.

First, it is worthwhile to see which features and their interactions are important for the model and whether its reasons are in a way that is intuitive to humans. A heat map with importance and interaction strengths values is presented in Figure 6.2.

The first thing to note is that there is a noticeable difference in the importance of individual features – the model largely uses only a small set of them. The most important of these is `IN_N_7W`, which stands for nouns in vocative case incidence and corresponds to directly addressing someone or something. The second and third most important features that also stand out are similar in interpretation but they involve pronouns – these are pronouns in vocative case incidence (`IN_PRO_7W`) and second person singular pronoun incidence (`IN_PRO_2S`), respectively. Interestingly, the feature `IN_PRO_7W` does not interact, which is due to the fact that the use of such an inflectional construction is very rare.

Interaction analysis shows that most of the evident ones occur within a group of about 10 features. The largest H-statistic value was calculated for a pair of features (`IN_N_7W`, `L_TCCT1`), where the second feature denotes the percentage of tokens covering 1% of most common types. According to a domain expert, this is due to the simplicity of the language for posts with

Table 6.1: List of top 30 most important StyloMetrix features for the analyzed model according to PD-based importance method.

No.	Feature	Description
1	IN_N_7W	Nouns in vocative case incidence
2	G_V	Verb incidence
3	L_TCCT1	Percentage of tokens covering 1% of most common types
4	IN_V_PERF	Verbs in perfective aspect incidence
5	IN_PRO_2S	Second person singular pronoun incidence
6	L_CONT_A	Content words incidence
7	IN_V_IMPERF	Verbs in imperfective aspect incidence
8	G_N	slash Noun incidence
9	IN_V_INFL	Inflected verb incidence
10	IN_V_IMP	Verbs in imperative mood incidence
11	IN_PRO_7W	Pronouns in vocative case incidence
12	IN_V_PPAS	Passive adjectival participles incidence
13	G_PRO	Pronoun incidence
14	L_TTR_LA	Type-token ratio for words lemmas
15	G_ADJ	Adjective incidence
16	L_NAME	Incidence of proper names (all words)
17	L_CONT_T	Content words types incidence
18	IN_V_COND	Verbs in conditional mood incidence
19	PS_M_DOMb	Incidence of words with less than mean dominance
20	IN_V_3S	Third person singular verb incidence
21	L_TCCT5	Percentage of tokens covering 5% of most common types
22	IN_V_PRES	Verbs in present tense incidence
23	IN_V_PAST	Verbs in past tense incidence
24	IN_V_3P	Third person plural verb incidence
25	G_ADV	Adverb incidence
26	SY_S_EX	Words in exclamative sentences incidence
27	L_PERSN	Incidence of personal names (all words)
28	L_PLACEN	Incidence of place names (all words)
29	SY_S_IN	Words in interrogative sentences incidence
30	IN_V_PACT	Active adjectival participles incidence

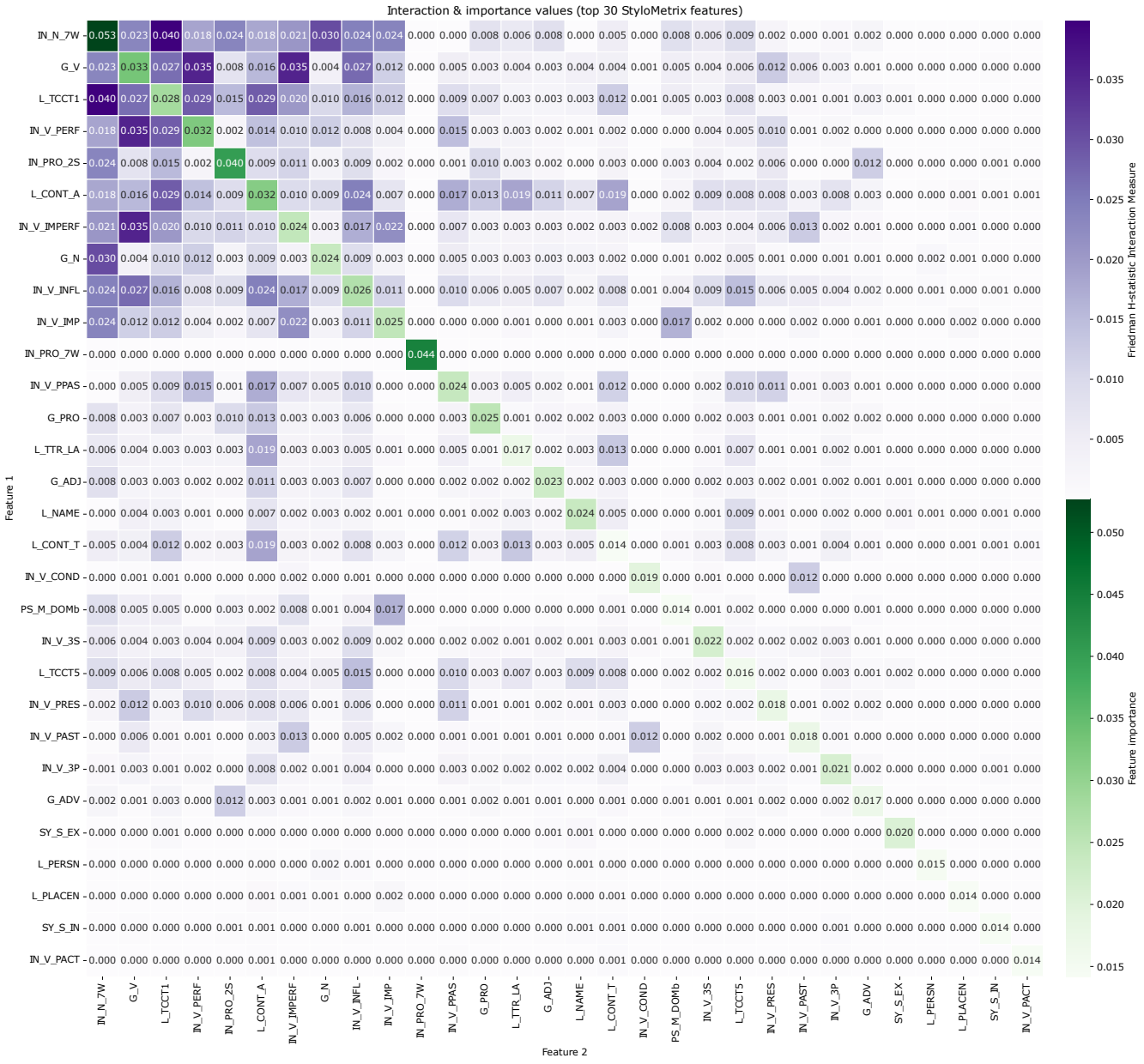

Figure 6.2: Heat map visualization with seriation for top 30 StyloMetrix features.

harmful content. The next most relevant interactions concern the occurrence of verbs and the joint analysis of whether they are in a perfect or imperfect form. So these interactions seem intuitive, and the paired features are meaningfully related.

A thorough examination of each pair can be quite difficult and time-consuming. Utilizing zenplots to analyze the interactions between the key features in the model can be a valuable tool in streamlining this process. However, it is also worth investigating what is the characteristic of interactions between the most important features, if any are present. Based on Figure 6.2, we can say that the feature `IN_PRO_7W` does not interact, so it is worth looking at the interaction between the features `IN_N_7W` and `IN_PRO_2S`, which is visualized in Figure 6.3.

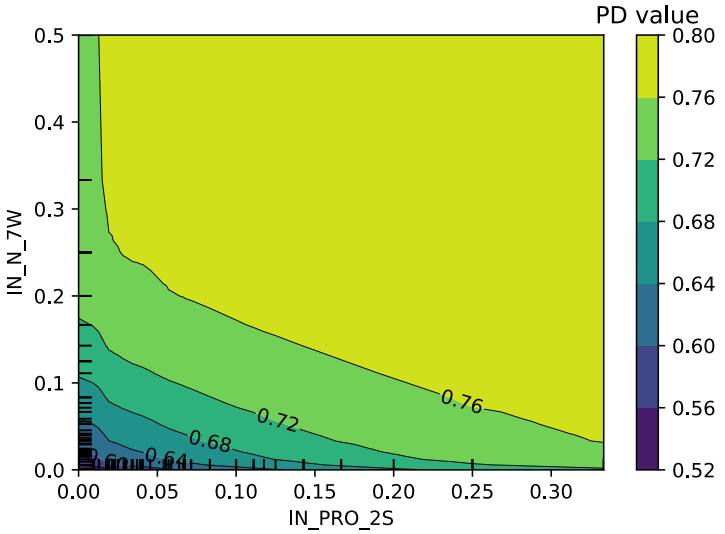

Figure 6.3: Two-dimensional partial dependence plot for features `IN_N_7W` and `IN_PRO_2S`.

This analysis reveals that while the use of a noun in the vocative case alone is not indicative of harmful language, its use in conjunction with the second person singular pronoun increases the probability of harmful language in a given post. According to a domain expert, this may be a sociological phenomenon worthy of attention and even more thorough examination.

6.4. Evaluation of explanations and the impact of the results

The results of the explanations were consulted on an ongoing basis with a domain expert with thorough knowledge of linguistics and machine learning. According to expert opinion, explanations linked to interactions provide a more holistic and complete insight into the model's reasoning than explanations based on attributions and the importance of single features (metrics) alone. Moreover, it turned out that the results of the explanations reveal that the models use expected relationships between features as well as less obvious interactions (see Section 6.3). In addition, due to their impact on interpretability, interactions between stylometric features and the occurrence of specific keywords in texts and other non-interpretable embeddings were also considered interesting and worth investigating, but that is beyond the scope of this thesis.

Repeated patterns of interaction have been taken into account in the system development process, and on their basis, subsequent StyloMetrix metrics are created. The motivation for this is that even simple feature engineering based on knowledge of extracted interactions enables better classifier performance. In the case of the example shown in Section 6.3, the addition of the three most important interactions (in terms of the value ratios of the features included) raised the

accuracy of the model from 0.802 to 0.804 and F1 score from 0.796 to 0.798. These changes are not substantial, but using new metrics built on text parsing rather than mathematical operations could work even better. Thus, new metrics are currently under development and will be used in future versions of the created models.

7. Conclusions

7.1. Summary

In this thesis, we tackle the issue of extracting interactions between features from predictive models. The theoretical part is the conducted literature review, in which selected methods proposed in the literature are described. The selection of techniques is carried out so as to present both model-agnostic and model-specific methods. For each of them, we cover its background, which allows noting that many techniques use similar concepts. We also describe interaction visualization techniques and additional accompanying visualization methods, which are crucial for the reception of xAI explanations related to interactions.

Moreover, the present work introduces the `artemis` Python package developed to improve the interpretability of machine learning models through the analysis of feature interactions. It offers implementations of described model-agnostic and model-specific methods in a user-friendly and easy-to-use API. To the best of our knowledge, it is the first Python package being a set of tools aimed at analysing predictive models in terms of feature interactions. We also provide the description of the created solution from a technical perspective.

The efficacy of the package is demonstrated through its usage in performed experiments and benchmarks (including the first benchmark for classification problems). Experiments on simple synthetic data show that the use of the package is worthwhile – adding the extracted interactions to the models results in significant improvements in their quality. Benchmarks conducted on multiple datasets demonstrate that the results obtained by different model-agnostic methods differ, and the best can be observed for the H-statistic method.

Moreover, we provide a real-life use case for described and implemented methods. We describe the process of explaining models created by NASK National Research Institute based on the use of the developed package. This case shows the potential of this software as a valuable resource for improving both the interpretability of machine learning models but also their performance.

7.2. Limitations

The first limitation of this work is that in addition to the methods described in detail and implemented in the `artemis` package, there are other techniques for extracting interactions from predictive models that have not received much attention here. However, this is a topic broad enough to be beyond the scope of the thesis.

There are also limitations that directly affect the constraints of the created package. First, many of the described methods are computationally efficient. Thus, for large datasets, the computational demands can be considerable, making it challenging to use the `artemis` package for datasets with a large number of features. This is partially mitigated by enabling the calculation of only partial results (for selected features) and sharing results between methods. However, further research on using more efficient algorithms or approximations could be valuable in improving the solution's scalability.

Another limitation is the modality of the supported data. This research is focused only on tabular data. Similarly, the package is only able to process data of such modality, meaning that it is not applicable to data in other forms such as images or text. In terms of the types of problems that can be addressed, the solution is currently limited to classification and regression tasks. While it could potentially be adapted for use in survival analysis, this would require additional development work and may not be straightforward. One potential solution for these limitations could be the development of a more general framework for feature interaction analysis that can be applied to a wider range of problem types and data modalities.

Overall, it is important to recognize the solution's limitations and carefully consider whether it is appropriate for a given task. While the `artemis` package is a useful tool for analyzing feature interactions in tabular data for classification and regression tasks, it is not a comprehensive solution for all data analysis needs. Further research and development may be required to address these limitations and expand the capabilities of the package.

7.3. Future work

Several directions for future research could enhance the capabilities of the `artemis` package. One area of focus could be the incorporation of additional feature interaction methods beyond those currently implemented. A review of the literature reveals a number of techniques that have been proposed for analyzing feature interactions. Incorporating these methods into the package is an obvious way to develop the created software.

Another opportunity for improvement would be to extend the range of supported model implementations for model-specific feature interaction methods. Currently, the package supports only a limited set of model implementations, and expanding this list could allow for the application of the package to a wider range of problems.

Finally, from a theoretical perspective, it would be beneficial to conduct a more comprehensive benchmark of the various feature interaction methods with the use of implementation from the package. This could involve evaluating the performance of the methods on a larger and more diverse set of datasets. Such an evaluation would provide further valuable insights into the relative strengths and weaknesses of the different methods.

In conclusion, there are numerous possibilities for future work to improve or generalize proposed methods and to enhance the functionality and applicability of the `artemis` package described in this thesis. By addressing these limitations and building on the current foundation, it may be possible to create more robust and versatile techniques and tools for the extraction of feature interaction from predictive models.

Bibliography

[1] Amina Adadi and Mohammed Berrada. „Peeking Inside the Black-Box: A Survey on Explainable Artificial Intelligence (XAI)". In: *IEEE Access* 6 (2018), pp. 52138–52160.

[2] Maximilian Alber et al. „iNNvestigate Neural Networks!" In: *Journal of Machine Learning Research* 20.93 (2019), pp. 1–8.

[3] Gulsum Alicioglu and Bo Sun. „A Survey of Visual Analytics for Explainable Artificial Intelligence Methods". In: *Computers & Graphics* 102 (2022), pp. 502–520.

[4] Daniel W Apley and Jingyu Zhu. „Visualizing the Effects of Predictor Variables in Black Box Supervised Learning Models". In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 82.4 (2020), pp. 1059–1086.

[5] Vijay Arya et al. *One Explanation Does Not Fit All: A Toolkit and Taxonomy of AI Explainability Techniques*. 2019.

[6] Sebastian Bach et al. „On Pixel-Wise Explanations for Non-Linear Classifier Decisions by Layer-Wise Relevance Propagation". In: *PLOS ONE* 10.7 (July 2015), pp. 1–46.

[7] Hubert Baniecki and Przemysław Biecek. „The Grammar of Interactive Explanatory Model Analysis". In: *arXiv preprint arXiv:2005.00497* (2020).

[8] Hubert Baniecki et al. „dalex: responsible machine learning with interactive explainability and fairness in Python". In: *The Journal of Machine Learning Research* 22.1 (2021), pp. 9759–9765.

[9] Alejandro Barredo Arrieta et al. „Explainable Artificial Intelligence (XAI): Concepts, Taxonomies, Opportunities and Challenges Toward Responsible AI". In: *Information Fusion* 58 (2020), pp. 82–115. ISSN: 1566-2535.

[10] Przemyslaw Biecek. „Model Development Process". In: *arXiv preprint arXiv:1907.04461* (2019).

[11] Przemyslaw Biecek and Tomasz Burzykowski. *Explanatory Model Analysis*. Chapman and Hall/CRC, New York, 2021. ISBN: 9780367135591.

[12] Przemysław Biecek. „DALEX: explainers for complex predictive models in R". In: *The Journal of Machine Learning Research* 19.1 (2018), pp. 3245–3249.

[13] Anne-Laure Boulesteix et al. „Letter to the Editor: On the Term 'Interaction' and Related Phrases in the Literature on Random Forests". In: *Briefings in Bioinformatics* 16.2 (Apr. 2014), pp. 338–345.

[14] Hans de Bruijn, Martijn Warnier, and Marijn Janssen. „The Perils and Pitfalls of Explainable AI: Strategies for Explaining Algorithmic Decision-making". In: *Government Information Quarterly* 39.2 (2022), p. 101666.

[15] Lars Buitinck et al. „API Design for Machine Learning Software: Experiences from the scikit-learn Project". In: *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*. 2013, pp. 108–122.

[16] Giuseppe Casalicchio, Christoph Molnar, and Bernd Bischl. „Visualizing the Feature Importance for Black Box Models". In: *Machine Learning and Knowledge Discovery in Databases*. Ed. by Michele Berlingerio et al. Cham: Springer International Publishing, 2019, pp. 655–670.

[17] Hugh Chen et al. „True to the Model or True to the Data?" In: *arXiv preprint arXiv:2006.16234* (2020).

[18] Tianqi Chen and Carlos Guestrin. „XGBoost: A Scalable Tree Boosting System". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '16. Association for Computing Machinery, 2016, pp. 785–794.

[19] Mike Cohn. *User Stories Applied: For Agile Software Development*. Addison Wesley Longman Publishing Co., Inc., 2004. ISBN: 0321205685.

[20] David R Cox. „Interaction". In: *International Statistical Review* (1984), pp. 1–24.

[21] Arun Das and Paul Rad. „Opportunities and Challenges in Explainable Artificial Intelligence (XAI): A Survey". In: *arXiv preprint arXiv:2006.11371* (2020).

[22] Jacob Devlin et al. „BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019*. Ed. by Jill Burstein, Christy Doran, and Thamar Solorio. Association for Computational Linguistics, 2019, pp. 4171–4186.

[23] Yadolah Dodge, David Cox, and Daniel Commenges. *The Oxford Dictionary of Statistical Terms*. Oxford University Press, 2003. ISBN: 0198509944.

[24] Maciej Eder, Jan Rybicki, and Mike Kestemont. „Stylometry with R: A Package for Computational Text Analysis". In: *The R Journal* 8.1 (2016), pp. 107–121.

[25] Bradley Efron and Robert J Tibshirani. *An Introduction to the Bootstrap*. Chapman and Hall/CRC, 1994. ISBN: 9780429246593.

[26] Aaron Fisher, Cynthia Rudin, and Francesca Dominici. „All Models are Wrong, but Many are Useful: Learning a Variable's Importance by Studying an Entire Class of Prediction Models Simultaneously". In: *Journal of Machine Learning Research* 20.177 (2019), pp. 1–81.

[27] R. A. Fisher. „The Correlation between Relatives on the Supposition of Mendelian Inheritance". In: *Transactions of the Royal Society of Edinburgh* 52.2 (1919), pp. 399–433.

[28] Yoav Freund and Robert E Schapire. „A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting". In: *Journal of Computer and System Sciences* 55.1 (1997), pp. 119–139.

[29] Jerome H Friedman. „Greedy Function Approximation: A Gradient Boosting Machine". In: *The Annals of Statistics* 29.5 (2001), pp. 1189–1232.

[30] Jerome H Friedman and Bogdan E Popescu. „Predictive Learning via Rule Ensembles". In: *The Annals of Applied Statistics* (2008), pp. 916–954.

[31] Randy Goebel et al. „Explainable AI: the new 42?" In: *Machine Learning and Knowledge Extraction*. Vol. 11015. Lecture Notes in Computer Science. Springer. 2018, pp. 295–303.

[32] Bryce Goodman and Seth Flaxman. „European Union Regulations on Algorithmic Decision-Making and a "Right to Explanation"". In: *AI Magazine* 38.3 (2017), pp. 50–57.

[33] Brandon M Greenwell, Bradley C Boehmke, and Andrew J McCarthy. „A Simple and Effective Model-Based Variable Importance Measure". In: *arXiv preprint arXiv:1805.04755* (2018).

[34] Brandon M. Greenwell and Bradley C. Boehmke. „Variable Importance Plots—An Introduction to the Vip Package". In: *The R Journal* 12.1 (2020), pp. 343–366.

[35] Trevor Hastie et al. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Vol. 2. Springer, 2009. ISBN: 0387848576.

[36] Julia Herbinger, Bernd Bischl, and Giuseppe Casalicchio. „REPID: Regional Effect Plots with Implicit Interaction Detection". In: *International Conference on Artificial Intelligence and Statistics, AISTATS 2022*. 2022, pp. 10209–10233.

[37] Marius Hofert and Wayne Oldford. „Zigzag Expanded Navigation Plots in R: The R Package Zenplots". In: *Journal of Statistical Software* 4 (2020), pp. 1–44.

[38] Andreas Holzinger et al. „Explainable AI Methods – A Brief Overview". In: *International Workshop on Extending Explainable AI Beyond Deep Models and Classifiers*. Springer. 2022, pp. 13–38.

[39] Giles Hooker. „Discovering Additive Structure in Black Box Functions". In: *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2004, pp. 575–580.

[40] Roman Hornung and Anne-Laure Boulesteix. „Interaction Forests: Identifying and Exploiting Interpretable Quantitative and Qualitative Interaction Effects". In: *Computational Statistics & Data Analysis* 171 (2022), p. 107460.

[41] Catherine B Hurley and R Wayne Oldford. „Eulerian Tour Algorithms for Data Visualization and the PairViz Package". In: *Computational Statistics* 26.4 (2011), pp. 613–633.

[42] Alan Inglis, Andrew Parnell, and Catherine Hurley. „vivid: An R Package for Variable Importance and Variable Interactions Displays for Machine Learning Models". In: *arXiv preprint arXiv:2210.11391* (2022).

[43] Alan Inglis, Andrew Parnell, and Catherine B. Hurley. „Visualizing Variable Importance and Variable Interaction Effects in Machine Learning Models". In: *Journal of Computational and Graphical Statistics* 31.3 (2022), pp. 766–778.

[44] Hemant Ishwaran et al. „High-Dimensional Variable Selection for Survival Data". In: *Journal of the American Statistical Association* 105.489 (2010), pp. 205–217.

[45] Ewelina Karbowiak and Przemyslaw Biecek. *EIX: Explain Interactions in 'XGBoost'*. R package version 1.1. 2022. URL: `https://github.com/ModelOriented/EIX`.

[46] Guolin Ke et al. „LightGBM: A Highly Efficient Gradient Boosting Decision Tree". In: *Advances in Neural Information Processing Systems (NIPS 2017)*. Ed. by I. Guyon et al. Vol. 30. 2017, pp. 3146–3154.

[47] Zachary C Lipton. „The Mythos of Model Interpretability: In Machine Learning, the Concept of Interpretability is Both Important and Slippery". In: *Queue* 16.3 (2018), pp. 31–57.

[48] Scott M Lundberg, Gabriel G Erion, and Su-In Lee. „Consistent Individualized Feature Attribution for Tree Ensembles". In: *arXiv preprint arXiv:1802.03888* (2018).

[49] Scott M Lundberg and Su-In Lee. „A Unified Approach to Interpreting Model Predictions". In: *Advances in Neural Information Processing Systems (NIPS 2017)*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc., 2017, pp. 4765–4774.

[50] Scott M Lundberg et al. „From Local Explanations to Global Understanding with Explainable AI for Trees". In: *Nature Machine Intelligence* 2.1 (2020), pp. 56–67.

[51] Tomás Mikolov et al. „Efficient Estimation of Word Representations in Vector Space". In: *1st International Conference on Learning Representations, ICLR 2013*. Ed. by Yoshua Bengio and Yann LeCun. 2013.

[52] Azalia Mirhoseini et al. „A Graph Placement Methodology for Fast Chip Design". In: *Nature* 594.7862 (2021), pp. 207–212.

[53] Christoph Molnar. *Interpretable Machine Learning. A Guide for Making Black Box Models Explainable*. 2nd ed. 2022. URL: `https://christophm.github.io/interpretable-ml-book`.

[54] Christoph Molnar, Giuseppe Casalicchio, and Bernd Bischl. „Iml: An R Package for Interpretable Machine Learning". In: *Journal of Open Source Software* 3.26 (2018), p. 786.

[55] Christoph Molnar, Giuseppe Casalicchio, and Bernd Bischl. „Quantifying Model Complexity via Functional Decomposition for Better Post-hoc Interpretability". In: *Machine Learning and Knowledge Discovery in Databases (ECML PKDD 2019)*. Ed. by Peggy Cellier and Kurt Driessens. Springer. 2020, pp. 193–204.

[56] Christoph Molnar et al. „General Pitfalls of Model-Agnostic Interpretation Methods for Machine Learning Models". In: *xxAI - Beyond Explainable AI - International Workshop, Held in Conjunction with ICML 2020, July 18, 2020, Vienna, Austria, Revised and Extended Papers*. Ed. by Andreas Holzinger et al. Vol. 13200. Lecture Notes in Computer Science. Springer, 2020, pp. 39–68.

[57] Tempestt Neal et al. „Surveying Stylometry Techniques and Applications". In: *ACM Computing Surveys* 50.6 (Nov. 2017). ISSN: 0360-0300.

[58] Sejong Oh. „Feature Interaction in Terms of Prediction Performance". In: *Applied Sciences* 9.23 (2019).

[59] Inez Okulska and Anna Zawadzka. „Styles with Benefits. The StyloMetrix Vectors for Stylistic and Semantic Text Classification of Small-Scale Datasets and Different Sample Length". In: *Proceedings of the 3rd Polish Conference on Artificial Intelligence* (2022), pp. 176–180.

[60] Aleksandra Paluszynska, Przemyslaw Biecek, and Yue Jiang. *randomForestExplainer: Explaining and Visualizing Random Forests in Terms of Variable Importance*. R package version 0.10.1. 2020. URL: `https://CRAN.R-project.org/package=randomForestExplainer`.

[61] Judea Pearl. „The Limitations of Opaque Learning Machines". In: *Possible Minds: Twenty-five Ways of Looking at AI*. Ed. by John Brockman. Penguin Press, 2019. Chap. 2, pp. 13–19. ISBN: 0525557997.

[62] F. Pedregosa et al. „scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[63] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. „GloVe: Global Vectors for Word Representation". In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing, EMNLP 2014*. Ed. by Alessandro Moschitti, Bo Pang, and Walter Daelemans. ACL, 2014, pp. 1532–1543.

[64] Matthew E. Peters et al. „Deep Contextualized Word Representations". In: *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018*. Ed. by Marilyn A. Walker, Heng Ji, and Amanda Stent. Association for Computational Linguistics, 2018, pp. 2227–2237.

[65] Marco Tulio Ribeiro, Sameer Singh, and Carlos 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*. Association for Computing Machinery, 2016, pp. 1135–1144.

[66] Jacques Savoy. *Machine Learning Methods for Stylometry: Authorship Attribution and Author Profiling*. Springer, 2020. ISBN: 978-3-030-53359-5.

[67] Ramprasaath R Selvaraju et al. „Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization". In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 618–626.

[68] David Silver et al. „Mastering the Game of Go with Deep Neural Networks and Tree Search". In: *Nature* 529.7587 (2016), pp. 484–489.

[69] Daria Sorokina, Rich Caruana, and Mirek Riedewald. „Additive Groves of Regression Trees". In: *Machine Learning: ECML 2007*. Ed. by Joost N. Kok et al. Vol. 4701. Lecture Notes in Computer Science. Springer, 2007, pp. 323–334.

[70] Daria Sorokina et al. „Detecting Statistical Interactions with Additive Groves of Trees". In: *Proceedings of the 25th International Conference on Machine Learning*. ICML '08. Association for Computing Machinery, 2008, pp. 1000–1007.

[71] Eric J Topol. „High-performance Medicine: the Convergence of Human and Artificial Intelligence". In: *Nature Medicine* 25.1 (2019), pp. 44–56.

[72] Michael Tsang, Dehua Cheng, and Yan Liu. „Detecting Statistical Interactions from Neural Network Weights". In: *6th International Conference on Learning Representations, ICLR 2018*. 2018.

[73] Sahil Verma et al. „Pitfalls of Explainable ML: An Industry Perspective". In: *arXiv preprint arXiv:2106.07758* (2021).

[74] Larry D Wall. „Some Financial Regulatory Implications of Artificial Intelligence". In: *Journal of Economics and Business* 100 (2018), pp. 55–63.

[75] Kun-Hsing Yu, Andrew L Beam, and Isaac S Kohane. „Artificial Intelligence in Healthcare". In: *Nature Biomedical Engineering* 2.10 (2018), pp. 719–731.

[76] Fabio Massimo Zanzotto. „Human-in-the-loop Artificial Intelligence". In: *Journal of Artificial Intelligence Research* 64 (2019), pp. 243–252.

List of symbols and abbreviations

xAI	explainable artificial intelligence
PD, PDP	partial dependence, partial dependence profile
PFI	permutation feature importance
NASK	Naukowa i Akademicka Sieć Komputerowa

List of Figures

List of Tables