Warsaw University of Technology

FACULTY OF
MATHEMATICS AND INFORMATION SCIENCE

# Master's diploma thesis

in the field of study Mathematics
and specialisation Mathematical Statistics and Data Analysis

## Local variable importance via oscillations of Ceteris Paribus profiles

Anna Kozak

student record book number 262856

thesis supervisor
dr hab. inż. Przemysław Biecek, prof. PW

WARSAW 2020

.....

supervisor's signature

.....

author's signature

**Abstract**

Local variable importance via oscillations of Ceteris Paribus profiles

Recently, machine learning has been helping to make decisions in many aspects of our lives. By using complex models we can increase the precision of their predictions, but at the same time, we can reduce their interpretability. This results in a lack of understanding of the results obtained, so there is now a strong need to explain the decisions made by the non-interpretable models called black boxes. There are several tools for exploring and explaining the predictive models, but they are still not enough.

In this thesis, we present a new method of explaining models, for the measure of local importance of variables. It is based on Ceteris Paribus profiles. I present a new measure with its implementation, possible variants, and examples of its use.

The implementation of the introduced method is included in the R package `vivo`.

**Keywords:** machine learning, R, modeling, Ceteris Paribus profiles, local variable importance, explainable artificial intelligence

**Streszczenie**

Ocena lokalnej istotności na bazie oscylacji profili Ceteris Paribus

W ostatnim czasie uczenie maszynowe pomaga w podejmowaniu decyzji w wielu aspektach naszego życia. Używając skomplikowanych modeli zwiększamy precyzję ich predykcji, ale jednocześnie zmniejszamy ich interpretowalność. Prowadzi to do braku zrozumienia otrzymanych wyników, dlatego obecnie kładzie się duży nacisk na wyjaśnianie decyzji podjętych przez nieinterpretowalne modele zwane jako czarne skrzynki. Powstało kilka narzędzi do eksploracji oraz wyjaśnień modeli predykcyjnych, ale nadal nie są one wystarczające.

Głównym celem tej pracy jest zaproponowanie nowej metody do oceny lokalnej ważności zmiennych. Ten cel zrealizowano za pomocą miary opartej na profilach Ceteris Paribus. Proponuję nową miarę wraz z jej implementacją oraz możliwymi wariantami. Dodatkowo przedstawiam przykłady jej użycia.

Metody opisane w tej pracy zostały zaimplementowane w języku R, w pakiecie `vivo`.

**Słowa kluczowe:** uczenie maszynowe, R, modelowanie, profile Ceteris Paribus, lokalna istotność zmiennych, wyjaśnialne uczenie maszynowe

Warsaw, .....

Declaration

I hereby declare that the thesis entitled „Local variable importance via oscillations of Ceteris Paribus profiles", submitted for the Master degree, supervised by dr hab. inż. Przemysław Biecek, prof. PW, is entirely my original work apart from the recognized reference.

.....

## Acknowledgements

Firstly, I would like to express my gratitude to my supervisor Professor Przemysław Biecek for his patient guidance, and encouragement to complete this master thesis.

I would also be glad to acknowledge all members of the $MI^2$ Data Lab for providing a pleasant workplace and a friendly atmosphere, in particular, I wish to thank Alicja Gosiewska for her encouragement and motivation.

Finally, I would like to express my very profound gratitude to my parents and siblings for providing me with continuous support throughout my years of study.

*Na koniec chciałbym wyrazić moją głęboką wdzięczność moim rodzicom i rodzeństwu za zapewnienie mi niezawodnego wsparcia przez całe lata studiów.*

# Contents

# Introduction

Machine Learning is used more and more in virtually any aspect of our life. We train models to predict the future in banking, telecommunication, insurance, industry, and many other areas. The models give us predictions, however, very often we do not know how they are calculated. Can we trust these predictions? Why should we use the results of models which we do not fully understand? In 2008, researchers from Google claimed that they can predict the occurrence of influenza on the basis of google searches. The idea was that when people have flu, they are looking for information related to flu in Google. Thus providing almost instantaneous signals of the general prevalence of influenza. Initially, it was a great success, but after a few years, Google Flu Trends failed. One source of problems is that people who make searches in Google for flu may know a little about how to recognize it. Searching for flu or its symptoms can be a study of the symptoms of a flu-like illness [Lazer et al., 2014]. Another example of an inaccurate model and its consequences is IBM Watson Health, more specifically Watson for Oncology product. Internal documents from IBM Watson Health indicate that the training and effectiveness of the Watson for Oncology system was flawed due to the small number of cases and the inclusion of artificial cases [Densford, 2018]. One of the main problems is that Watson's based on statistics, but doctors don't work that way when they make a daily diagnosis. The next example is about Apple Card. It turns out that the credit card offered different credit limits for men and women. The limit on the card for men was 20 times higher than for women. US regulator started an investigation about Apple's sexist credit card [Duffy, 2019]. The first example is COMPAS (Correctional Offender Management Profiling for Alternative Sanctions) software uses an algorithm to assess potential recidivism risk, but in 2016, the technology reporter Julia Angwin and colleagues at ProPublica analyzed COMPAS assessments for more than 7,000 arrestees in Broward County, Florida, and published an investigation claiming that the algorithm was biased against African Americans [Yong, 2018].

In recent years approach to predictive modeling has changed. Much more attention is now paid to explanations that show what influences the decision of the model. This is enforced by the EU General Data Protection Regulation (GDPR) [EU, 2018]. Explainable artificial intelligence (XAI) is one of the recent approaches that aim to deliver results that can be understood by a

human expert. It tries to explain ''black–box" models to data scientists, i.e. neural networks, random forest, or xgboost.

We want to explain the predictions about the machine learning model. To achieve this we need explanatory methods, which are algorithms that generate explanations. Explanation is a way that combines model predictions with values that describe an individual in a human-understandable way. The explanations can be divided into global and local explanations. Global explanations are those that explain which features are important, how important they are, and how they interact with each other. Local explanations, on the other hand, are those that show a change of decision or contributions of variables for a single observation. One of the most popular algorithms to explain machine learning is the LIME algorithm introduced by Ribeiro [Ribeiro et al., 2016]. This technique explains the prediction of any classifier by fitting a weighted linear model on the observations similar to the observation of interest. Another popular algorithm is Shapey values [Lundberg and Lee, 2017]. This local technique is based on coalitional game theory method, variables are treated as players which can be in different coalitions. The contribution of a variable is an average over its all coalitions.

More and more packages are being created in R and Python languages to explain complex models. For example, in R, the `DALEX` [Biecek, 2018] package (moDel Agnostic Language for Exploration and eXplanation) helps to understand how complex models are working. It delivers functions for local and global model exploration. In addition, the `DALEXtra` [Maksymiuk et al., 2019] package is an extension of `DALEX` package. This package provides an easy-to-use interface for models created using Python libraries such as `scikitlearn` [Mueller, 2019], `keras` [Chollet, 2019]. The `DALEX` and `DALEXtra` packages are a part of `DrWhy.AI` universe which contains tools that can be used to make our work efficient through the whole model lifecycle. Also in Python, there are packages containing tools to interpret black–box models. The `ELI5` [Korobov and Lopuhin, 2017] package helps debug machine learning classificators and explain their predictions. Unfortunately, support is limited to tree-based models and other parametric linear models. The Python `Skater` [Kramer and Choudhary, 2018] library is designed for the black–boxes model in a global and local approach. `Skater` originally started as a LIME [Ribeiro et al., 2016] fork, but then evolved into an independent framework. It contains feature importances, local interpretations (LIME), and more. Yet another Python package for XAI is `dalex` [Kretowicz et al., 2020]. This is the Python version of `DALEX`. It contains `DALEX` functionalities and other additional ones. Of course, `dalex` is also part of `DrWhy.AI` universe.

Let us take a look at the tools for visualizing explanations. There are many ways to visualize feature variable dependence in supervised machine learning models. For example, there

are PDP (Partial Dependence Plots) [Friedman et al., 2009], ICE (Individual Conditional Expectation plots) [Goldstein et al., 2014], ALE (Accumulated Local Effects plots) [Apley, 2017] and CP (Ceteris Paribus profiles) [Biecek, 2019]. One of the most popular tools for constructing variable importance plots is `vip` [Greenwell et al., 2019] package for R. The `vip` includes Partial Dependence Plots and individual conditional expectation curves which, help to visualize feature impact. Another example is `ingredients` [Biecek et al., 2019] package for R. This is a collection of tools for assessment of feature importance and feature effects. Package includes Ceteris Paribus profiles, Partial Dependency Plots, Conditional Dependency Plots (also called M Plots), etc. Furthermore, there is the `iml` [Molnar et al., 2018] package created by Christoph Molnar. The `iml` includes feature importance, ALE, PDP, ICE, local model, and Shapley values. In addition, in Python we have libraries such as `pyCeterisParibus` [Kuźba, 2019] include implementation of Ceteris Paribus Profiles. Also `PDPbox` [Jiangchun, 2018] to visualize the effect of some variables on model prediction for any supervised learning algorithm. Additionally, the `PyCEbox` [Rochford, 2017] package, which is an implementation of ICE plots. The development of tools allows for introducing a new solution, instead of basing on static charts, interactive explanations can be created. The `modelStudio` [Baniecki and Biecek, 2019] package automates explanation of machine learning predictive models. This package generates advanced interactive and animated model explanations such as Partial Dependency Plots, Ceteris Paribus, Feature Importance, and others.

In Figure A we see a division of visual explanations into global and local.

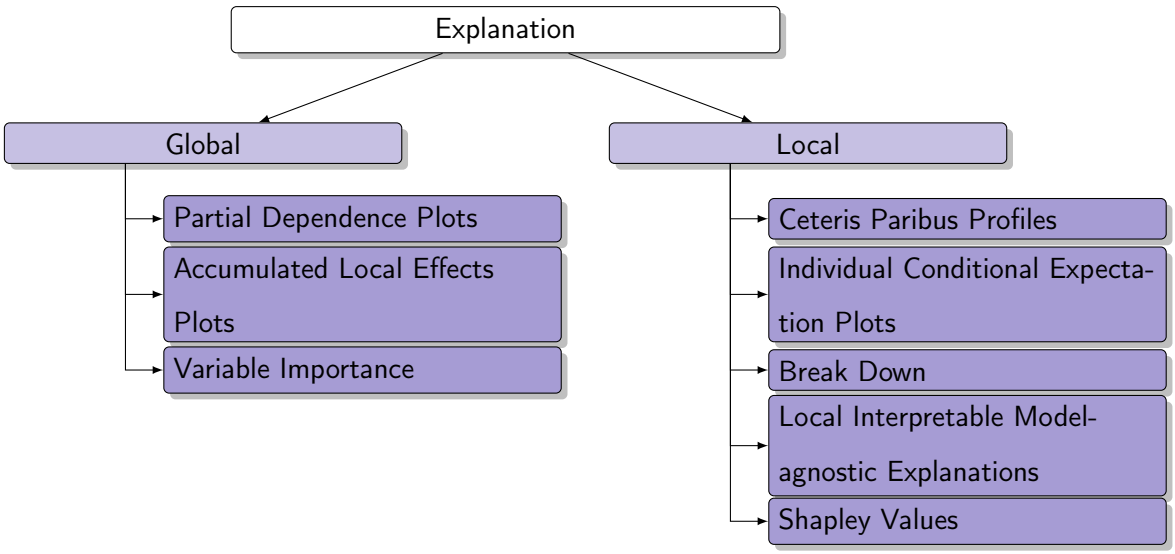



Figure A: Taxonomy of different tools for model explanation.

Based on the explanations it is possible to construct measures of the importance of variables. In 2018, Greenwell et al. proposed a measure of the importance of variables based on the Partial Dependence Plot (PDP). In other words, it can be concluded that any variable for which PDP is „flatness" is probably less important than other variables. In general, we define a function which is some selected measure of „flatness" PDP. The proposed method by the authors for $x$ variable is described as the standard deviation for continuous variables and the range statistic divided by four for factors [Greenwell et al., 2018]. This method seems to be highly useful, but it is still in the experimental phase. The method is implemented in the R package `vip` [Greenwell et al., 2019].

In this thesis, we introduce the `vivo` package for R which is a new method and a new tool for calculation of local variable importance based on Ceteris Paribus profiles. When a model has many features and plotting all one-dimensional summary statistics is troublesome, `vivo` indicates which variables are worth paying attention to. The `vivo` is also part of `DrWhy.AI` universe.

The thesis is organized into four chapters. Chapter 1 introduces terminology, methodology, definitions, and approaches to constructing a new measure. Chapter 2 contains sample use of the presented tool and comparison with LIME [Ribeiro et al., 2016], BreakDown [Staniak and Biecek, 2018] and Shapley values [Lundberg and Lee, 2017]. Chapter 3 is about package architecture, contains the structure of the package and descriptions of function. Chapter 4 summarises the results of this thesis.

# 1. Methodology

In this chapter, we introduce terminology related to machine learning modeling. Also definitions and properties of the Ceteris Paribus profiles. We introduce the approach to the construction of the new measure. Additionally, we present the ideas of other local explanations methods such as LIME, Break Down, and Shapley values.

## 1.1. Terminology

Let's introduce the basic definitions related to machine learning. These definitions are based on [Molnar, 2019].

**Definition 1.1.** An *algorithm* is a procedure for solving a mathematical problem in a finite number of steps that frequently involves repetition of an operation.

**Definition 1.2.** A *machine learning model* is the learned program that maps inputs to predictions. Other names in depending on the task are "classifier" or "regression model". In formulas, the trained machine learning model is called $f(x)$.

**Definition 1.3.** A *black–box model* is a system that does not reveal its internal mechanisms. In machine learning, the „black–box" describes a model for which we do not know the parameters or do not understand them when we look at them.

**Definition 1.4.** The *glass–box model* is a model whose behavior and prediction are understandable to humans. This is the opposite of the definition of black–box models.

**Definition 1.5.** A *training dataset* is a table with the data from which the model learns. The dataset contains the features and the target to predict.

**Definition 1.6.** An *instance* is a row in the dataset. Other names are (data) point, example, observation. An instance consists of the feature values $x_i$ and, if known, the target outcome $y_i$.

**Definition 1.7.** The *features* are the inputs used for prediction or classification. A feature is a column in the dataset. The matrix with all features is called $X$. A single instance is denoted as $x_i$. The vector of a single feature for all instances is $x^j$ and the value for the feature $j$ and instance $i$ is $x_i^j$.

**Definition 1.8.** The *target* is the information the machine learns to predict. In regression model is usually a value, in the classification model is a class.

**Definition 1.9.** The *prediction* is what the machine learning model "guesses" what the target value should be based n the given features.

## 1.2. Data set

In this chapter, we will discuss the approach to constructing a measure for local variable importance. To help us understand and make our intuition easier they will be presented on `apartments` dataset from the `DALEX` package. The sample of the few rows from apartments data set is presented in Table 1.1. This is a data set with 1000 observations and 6 variables describing prices of apartments in Warsaw, such as:

- *m2.price*, apartments price per meter-squared (in EUR), a numerical variable range 1607-6595;
- *construction.year*, the year of construction of the block of flats in which the apartment is located, a numerical variable range 1920-2010;
- *surface*, apartment's total surface in squared meters, a numerical vairable range 20-150;
- *floor*, the floor at which the apartment is located (ground floor taken to be the first floor), a numeric integer variable with values from 1 to 10;
- *no.rooms*, the total number of rooms, a numerical variable with values from 1 to 6;
- *distric*, a factor with 10 levels indicating the districr of Warsaw where the apartment is located;

Figure 1.1 shows the distributions of variables for individual variables (histograms and bar plots).

| m2.price | construction.year | surface | floor | no.rooms | district |
|---|---|---|---|---|---|
| 5897 | 1953 | 25 | 3 | 1 | Srodmiescie |
| 1818 | 1992 | 143 | 9 | 5 | Bielany |
| 3643 | 1937 | 56 | 1 | 2 | Praga |
| 3517 | 1995 | 93 | 7 | 3 | Ochota |
| 3013 | 1992 | 144 | 6 | 5 | Mokotow |
| 5795 | 1926 | 61 | 6 | 2 | Srodmiescie |

Table 1.1: First 6 rows from `apartments` data set.

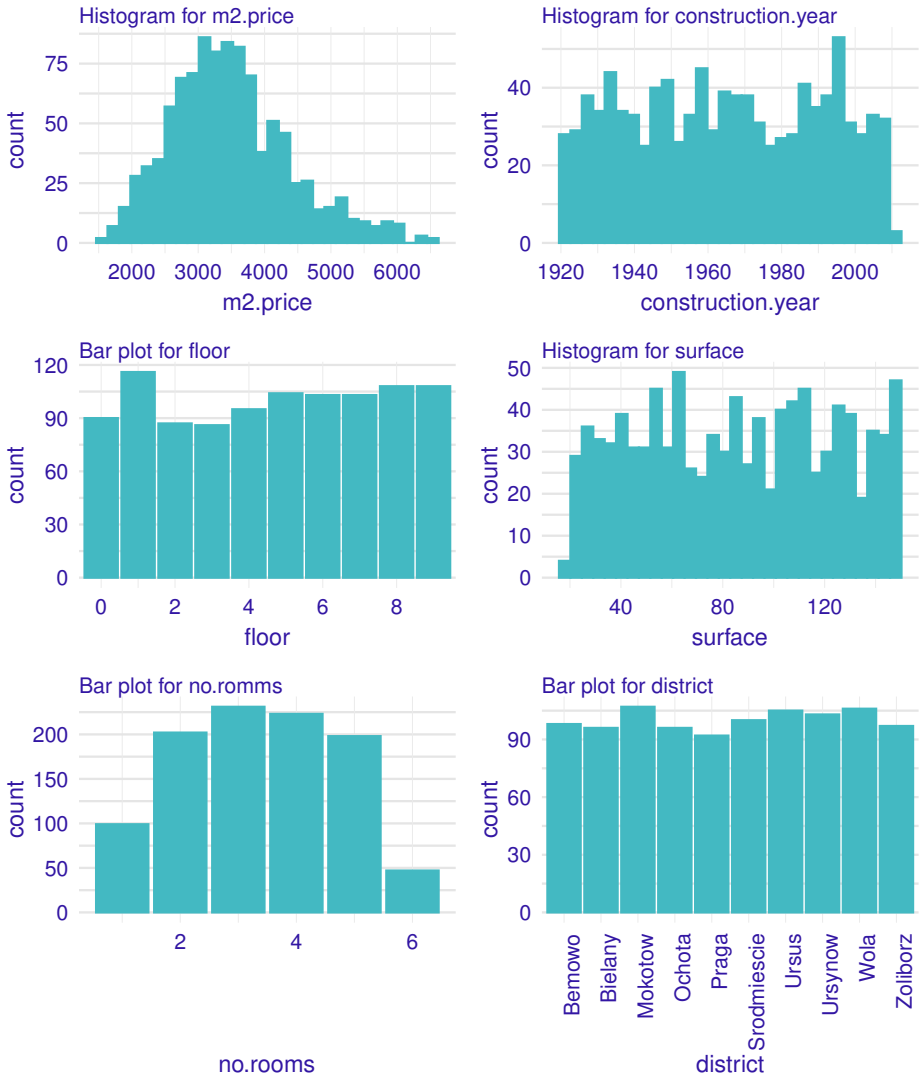



Figure 1.1: Distributions of variables. Histograms for continuous variables and bar plots for categorical variables.

## 1.3. Model

We are introducing a random forest regression model. Random forest is known for its good predictability, is able to capture interactions of low order variables, and is quite stable. We used `randomForest()` function with default parameters from the R `randomForest` [Liaw and Wiener, 2002] package to adjust it.

```
library("randomForest")
apartments_rf_model <- randomForest(m2.price ~ construction.year + surface +
floor + no.rooms, data = apartments)
```

The explanatory variable in this model is the price per square meter, while the explanatory variables are the year of construction, surface area, number of floors, and number of rooms.

## 1.4. Ceteris Paribus profiles

Ceteris Paribus is a latin phrase meaning „other things held constant" or „all else unchanged". Ceteris Paribus profiles are designed to show model response around a single point in the feature space. They show how the model response depends on changes in a single input variable, keeping all other variables unchanged. They work for any Machine Learning model and allow for model comparisons to better understand how a model is working.

Let $f(x) : \mathcal{R}^d \to \mathcal{R}$ denote a predictive model. Function take $d$ dimensional vector and calculates a numerical score.

**Definition 1.10.** Ceteris Paribus profiles for model $f$, variable $j$ and point $x_*$ are defined as

$$h_{x_*}^{f,j}(z) := f(x_*^{j|=z}),$$

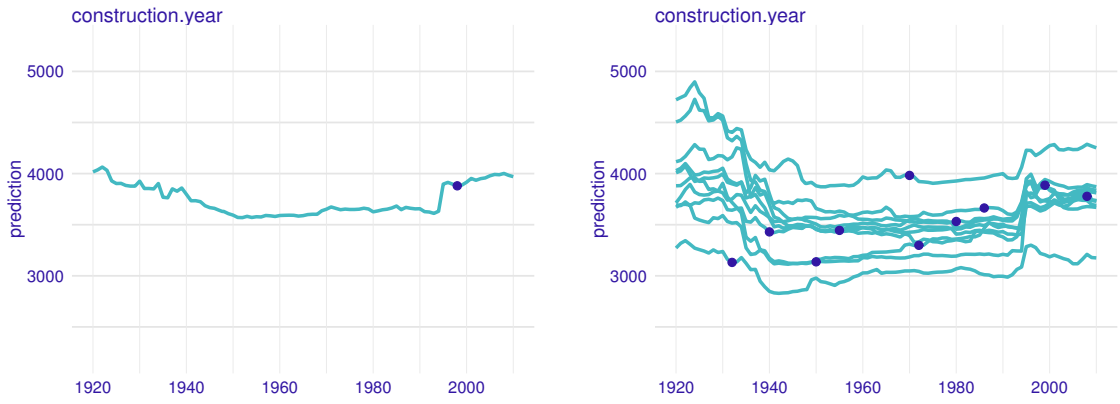
where $x_* \in \mathcal{R}^d$ refers to a point in the feature space. Moreover $x_*^{j|=z}$ denote a data point $x_*$ with all coordinates equal to $x_*$ except coordinate $j$ equal to value $z$.

Below is an example table 1.2 for observation $x_*$, which takes the value 1998 for the year of construction, 88 for the surface, 2 for the number of floors, and 3 for the number of rooms. Variable $j$ is construcion.year, so only it takes different values and the other variables have values for observation $x_*$. The Ceteris Paribus profile for the observation $x_*$ is the drawing of a curve which for the x-axis takes the values of the variable *construcion.year*, and for the y-axis the prediction ($\hat{y}$).

| construction.year | surface | floor | no.rooms | $\hat{y}$ |
|---|---|---|---|---|
| 1920 | 88 | 2 | 3 | 3997.225 |
| 1921 | 88 | 2 | 3 | 4033.060 |
| 1922 | 88 | 2 | 3 | 4040.927 |
| 1923 | 88 | 2 | 3 | 4014.828 |
| 1924 | 88 | 2 | 3 | 3907.281 |
| 1925 | 88 | 2 | 3 | 3881.083 |

Table 1.2: Example of Ceteris Paribus profile calculation for observation $x_*$ for *construction.year* variable. Observation with the following variables construction.year = 1998, surface = 88, floor = 2, no.rooms = 3.

Figure 1.3 shows the Ceteris Paribus profile for a single observation variable described above. The purple dot is value of prediction for $x_*$, $j$ is feature *construction.year* and $z$ takes value from the range 1920 to 2010. Line indicate Ceteris Paribus profiles for $x_*$, as defined when only the value of the *construction.year* changes, we can see a decrease in the prediction value (apartment price per meter-squared) if the apartment was built earlier. However, if it was built later, the price could increase.




(a) Ceteris Paribus profile for a single observation

(b) Ceteris Paribus profiles for 10 random observations

Figure 1.2: Ceteris Paribus profiles. The violet dot indicates the prediction value for observation, the line indicates the Ceteris Paribus profile. On the x-axis we have values of the variable *construction.year*, and on the y-axis we have the prediction value for observations when only the value of the variable *construction.year* changes.

To evaluate how the prediction for selected observations behaves, we can draw several Ceteris Paribus profiles. See an example in Figure 1.2b. Each profile corresponds to one observation

from the subset. The plot corresponds to 10 randomly selected observations from the `apartments` dataset. The dots indicate a value of prediction of selected observations. Aggregating the profiles by the average, we obtain the average response of the model for the variable. The profile thus obtained is called Partial Dependence Plot (PDP). It was introduced by Friedmann in 2000 [Friedman, 2000].

**Definition 1.11.** Partial Dependence Plot for a model $f$ and a variable $x^j$ is defined as

$$g_{PD}^{f,j}(z) = E[f(x^j = z, X^{-j})] = E[f(x^{j|=z})].$$

This value can be estimated by an average from Ceteris Paribus profiles

$$\hat{g}_{PD}^{f,j}(z) = \frac{1}{n}\sum_{i=1}^{N} f(x_i^j = z, x_i^{-j}) = \frac{1}{n}\sum_{i=1}^{N} f(x_i^{j|=z}),$$

where $N$ is the number of observations for which we have CP profiles.

Figure 1.3 shows a Ceteris Paribus for 50 observation and feature j equals *construction.year*. The purple line represent Partial Dependency Plot - aggregation of 50 Ceteris Paribus profiles.

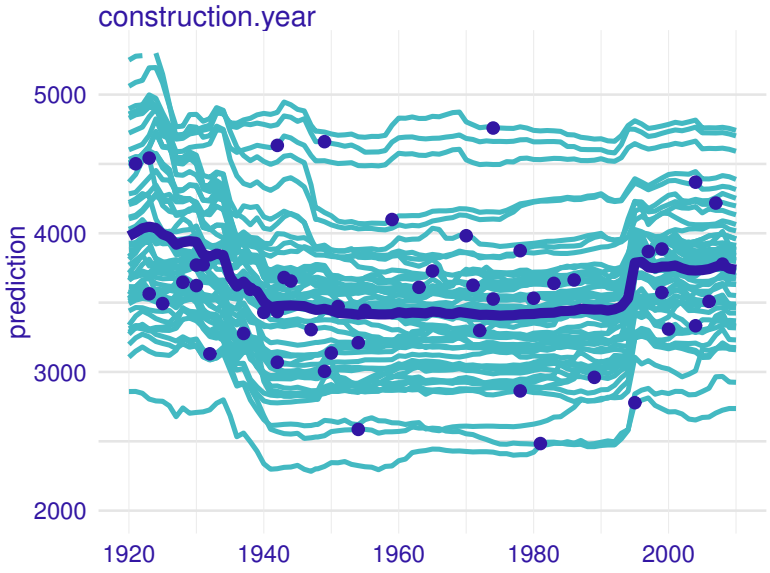



Figure 1.3: Ceteris Paribus profile. On the x-axis we have the value of the *construction.year* variable, and on the y-axis, we have the prediction value for observations when only the value of the *construction.year* variable changes. The purple dot indicates the predictive value for observation, the cyan line indicates the Ceteris Paribus profile, the purple line indicates the PDP.

## 1.5. Approach to construction oscillation measure based on Ceteris Paribus profiles

In Section 1.4 we introduced Ceteris Paribus profiles. Now, we're approaching our measure, called oscillations. But before that, let's consider the situation. Imagine that we have a data set that contains about 2000 features and 6 million records. We want to evaluate for each observation what influenced the decision. Which variables and values contributed to this prediction. To examine all variables for all observations we would have to view 2 000 $\times$ 6 000 000 plots. This is quite a challenge. The solution to this problem is an instance level importance measure.

Let us see Ceteris Paribus profiles in the Figure 1.4. To construct an importance measure we can use the CP profile, we can calculate the area under the curve. In particular, the larger the deviation along the corresponding Ceteris Paribus profile, the larger influence of an explanatory variable on prediction at a particular instance. For a variable that exercises little or no influence on model prediction, the profile will be flat or will barely change.

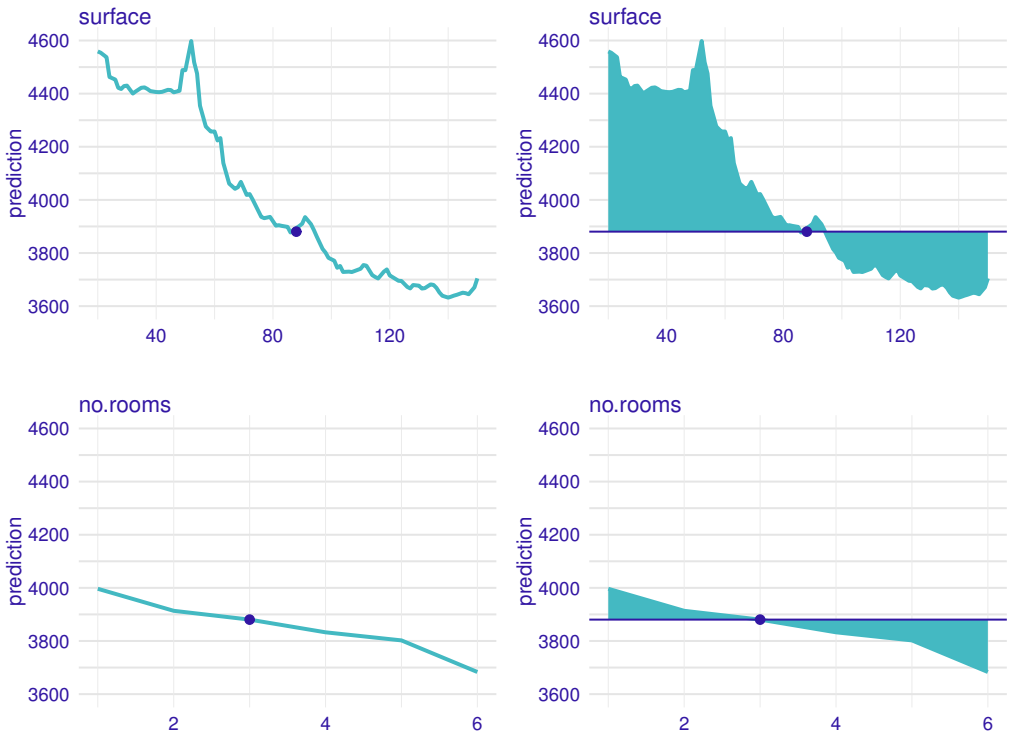



Figure 1.4: Ceteris Paribus profiles and oscillations. Purple dots indicate the predictive value for observation, the cyan line indicates the Ceteris Paribus profile, the horizontal purple line indicates the level. A cyan-colored surface is a measure of oscillation.

Figure 1.4 shows an oscillations for variable *surface* and *no.rooms* in point $x_*$. The colored

area coresponds to our measure. The larger the area, the more important is the variable. It is likely that a *surface* variable has a greather impact on prediction than the *no.rooms*.

How can we define this measure? As we know, the measure is the colored area under Ceteris Paribus Plot. In this paper we present two ways to calculate this measure.

The first one is based on calculating the integral as a field under the plot. Let denote $q^j(z)$ as probability function of the distribution of the $j-$th explantory variable. The measure of the variable importance for model at point $x_*$, computed based on the variable's Ceteris Paribus profiles, is defined as follows:

$$vip^j_{CP}(x_*) = \int_{\mathcal{R}} \mid h^j_{x_*}(z) - f(x_*) \mid q^j(z)dz = E_{X^j}\Big[ \mid h^j_{x_*}(X^j) - f(x_*) \mid \Big].$$

The distribution of j-th explanatory variable is unknown. The natural way of estimating $vip^j_{CP}(x_*)$ is

**Definition 1.12.** The oscillation of Ceteris Paribus is measured (I) as

$$\widehat{vip^j_{CP}}(x_*) = \sum_{i=1}^{n} \mid h^j_{x_*}(z_i) - f(x_*) \mid p^j(z_i), \tag{1.1}$$

where $p^j$ denote empirical density of $j$ variable.

Another possible estimate is a root from average squares of difference between Ceteris Paribus and prediction value for observation $x_*$.

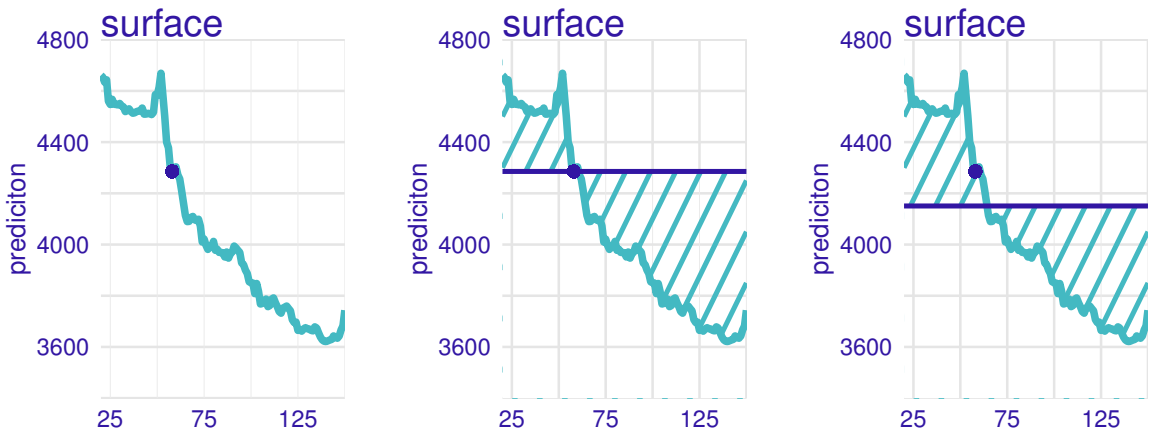
**Definition 1.13.** The oscillation measure (II) estimate as root from average squares is

$$\widetilde{vip^j_{CP}}(x_*) = \sqrt{\sum_{i=1}^{n} (h^j_{x_*}(z_i) - f(x_*))^2 p^j(z_i)}, \tag{1.2}$$

where $p^j$ denote weight based on empirical density of $j$ variable.

## 1.6. Possible variations for the measure of oscillations

We have described two possible ways how to calculate the measure. Now, let think about possible other variations. The first of the ideas is how to choose the origin level. Look at Figure 1.5. Panel 1.5a present Ceteris Paribus profile for observation. On panel 1.5b we have Ceteris Paribus profile and line represent prediction on point $x_*$ ($f(x_*)$). Panel 1.5c shows Ceteris Paribus profile and line which indicates average of the value CP profile in points $z_i$




(a) CP profile

(b) value of CP profile in point $x_*$

(c) average of CP profile values in points $z_i$

Figure 1.5: The violet dot indicates the predictive value for the observation, the cyan-colored line is the profile ceteris paribus. The purple horizontal line indicates the origin level. In panel 1.5b this line is plotted in the prediction value for observation $x_*$, in panel 1.5c it is the average value of CP profile. The dashed area is a measure of importance.

($\overline{h^j_{x_*}} = \frac{1}{n}\sum_{i=1}^{n} h^j_{x_*}(z_i)$). As we can see, the area under the CP profile changes as the reference level changes, so its choice may be important.

Following the considerations on the choice of a baseline, definitions (1.1) and (1.2) take the level into account as a prediction value for observation $x_*$, and by doing analogy we can define a measure by considering $\overline{h^j_{x_*}}$.

**Definition 1.14.** The oscillation measure (III) defines as absolute deviation and origin level as the average of the value Ceteris Paribus profiles ($\overline{h^j_{x_*}}$) is

$$\widehat{vip^j_{CP}}(x_*) = \sum_{i=1}^{n} \mid h^j_{x_*}(z_i) - \overline{h^j_{x_*}} \mid p^j(z_i), \tag{1.3}$$

where $p^j$ denote weight based on empirical density of $j$ variable.

**Definition 1.15.** The oscillation measure (IV) estimate as root from average squares and origin level as the average of the value Ceteris paribus profiles ($\overline{h^j_{x_*}}$) is

$$\widetilde{vip^j_{CP}}(x_*) = \sqrt{\sum_{i=1}^{n} (h^j_{x_*}(z_i) - \overline{h^j_{x_*}})^2 p^j(z_i)}, \tag{1.4}$$

where $p^j$ denote weight based on empirical density of $j$ variable.

Yet another variant of oscillation measure is to take into account variable density. For example,

below the CP profile for a variable *surface* and empirical density plot for that variable Figure 1.6.

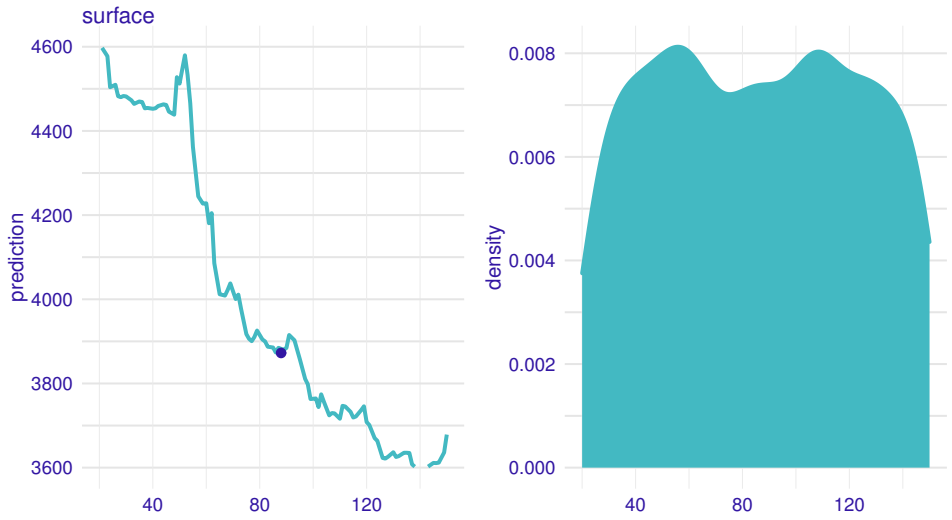



Figure 1.6: The figure shows the Ceteris Paribus profile and an empirical density plot for the *surface* variable. The left plot represent Ceteris Paribus profile for *surface* variable. The violet dot stands for observation, the cyan line indicates profile. On the x-axis we have a value of *surface* variable, on y-axis value of predictions. The right plot represents empirical density for *surface* variable with gaussian kernel.

As we can see, the variable has a distribution that is not uniform. Higher density for values smaller and larger than average. Based on the density of the variable, we can determine the weights that we can use to calculate the measure. If a variable had a uniform distribution, the weight would be the same for each variable value. We can assume such a variant in the construction of the measure and then we can rewrite the definitions (1.1) and (1.2) as follows:

**Definition 1.16.** The oscillation measure (V) as absolute deviation and no weights is

$$\widehat{vip^j_{CP}}(x_*) = \frac{1}{n}\sum_{i=1}^{n} \mid h^j_{x_*}(z_i) - f(x_*) \mid . \tag{1.5}$$

Similarly, the definition of the oscillations measure estimated by root from average squares.

**Definition 1.17.** The oscillation measure (VI) as root from average squares and no weights is estimated as

$$\widetilde{vip^j_{CP}}(x_*) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(h^j_{x_*}(z_i) - f(x_*))^2}. \tag{1.6}$$

In summary, we have introduced a local measure of the importance of variables. We have presented possible variants of the measure of oscillations and what can affect its estimation. We have three parameters responsible for the possible forms of this measure, these are:

- calculation method: we can estimate the measure of oscillation in two ways, i.e. we can use absolute deviation or square root, introduced in definition (1.12) and (1.13).

- the reference level: we introduced the discussion at the beginning of subsection 1.6, we consider two possibilities, i.e. the value of prediction in point $x_*$ or the mean of CP, introduced in definitions (1.14) and (1.15)

- density: introducing weights based on the empirical density of variables, introduced in definitions (1.16) and (1.17).

Each of these parameters can take two values. So we have $2^3 = 8$ possibilities. Let us define the two missing definitions of the measure of oscillations.

**Definition 1.18.** The oscillation measure (VII) as absolute deviation, the origin level as the average of the value Ceteris paribus profiles ($\overline{h^j_{x_*}}$), and no weights is

$$\widehat{vip^j_{CP}}(x_*) = \frac{1}{n}\sum_{i=1}^{n} \mid h^j_{x_*}(z_i) - \overline{h^j_{x_*}} \mid . \tag{1.7}$$

**Definition 1.19.** The oscillation measure (VIII) as root from average squares, the origin level as the average of the value Ceteris paribus profiles ($\overline{h^j_{x_*}}$), and no weights is estimated as

$$\widetilde{vip^j_{CP}}(x_*) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(h^j_{x_*}(z_i) - \overline{h^j_{x_*}})^2}. \tag{1.8}$$

In the table we have a summary of all possible variants of the measure.

| | (I) | (II) | (III) | (IV) | (V) | (VI) | (VII) | (VIII) |
|---|---|---|---|---|---|---|---|---|
| absolute deviance | ✓ | ✗ | ✓ | ✗ | ✓ | ✗ | ✓ | ✗ |
| point | ✓ | ✓ | ✗ | ✗ | ✓ | ✓ | ✗ | ✗ |
| density | ✓ | ✓ | ✓ | ✓ | ✗ | ✗ | ✗ | ✗ |

Table 1.3: All variants of measure. The names refer to parameters that can be set in two ways, the numbers at the top of the table (I) – (VIII) refer to the previous definitions.

## 1.7. Comparison of the proposed measure with LIME, BreakDown and Shapley values

In subsection 1.5. we introduced a method for assessment of local variable importance based on Ceteris Paribus profiles. Now, we look at three other method which calculate local variable importance.

First of them is LIME (Local Interpretable Model-agnostic Explanations), an algorithm that can explain the predictions of any classifier or regressor in a faithful way, by approximating it locally with an interpretable model.

The second method is Break Down Plot [Staniak and Biecek, 2018]. Plots show how variables move the model prediction from population average to the model prognosis for a single observation.

The last but not least method is Shapley values [Lundberg and Lee, 2017]. It is based on the idea of averaging the value of a variable's contribution overall, or a large number of, possible orderings.

Let us now show how these methods are constructed, based on the book Explanatory Model Analysis [Biecek and Burzykowski, 2019].

### 1.7.1. LIME (Local Interpretable Model-agnostic Explanations)

The LIME method was originally proposed by Ribeiro in 2016 [Ribeiro et al., 2016]. The key idea of this method is to bring the black box model locally closer using a simpler glass box model which is easier to interpret. Figure 1.7 show a idea behind LIME. The violet and light gray areas correspond to decision regions for a binary classification model. The big black dot corresponds to the instance of interest $x_*$. Other dots indicates the generated new data. The dashed line corresponds to a simple linear model fitted for the artificial data. It approximates the black-box model around the instance of interest. The simple linear model„explains" the local behavior of the black-box model.

The objective is to find a local model $M$ that approximates a black box model $f$ around the point of interest $x_*$. We can write this as

$$M(x_*) = \arg\min_{g \in G} L(f, g, \Pi_{x_*}) + \Omega(g).$$

We are looking for a local model $g$ from the class $G$ of interpretable models for a instance $x_*$. The model shall be simple so we add penalty for model complexity measured as $\Omega(g)$. The white-box model $g$ shall approximate well the black-box model $f$ locally, where $\Pi_{x_*}$ denote neighborhood

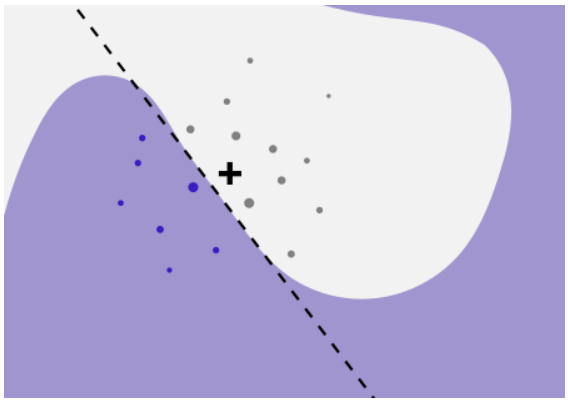


Figure 1.7: Intuition of LIME method. The purple and light grey areas correspond to the decision making regions for the binary classification model. The black cross corresponds to the $x_*$ observation being considered. The dots indicate new data generated. The dashed line corresponds to a simple linear model matching the artificial data.

of $x_*$. The $L$ stands for some goodness of fit measure. The functions $f$ and $g$ may work on different data. The black-box $f(x) : \mathcal{X} \to \mathcal{R}$ works on original feature space $\mathcal{X}$ when the glass-box function $g : \mathcal{X}' \to \mathcal{R}$ usually works on an interpretable feature space $\mathcal{X}''$. The algorithm may be used to find an interpretable surrogate model that selects K most important interpretable features.

### 1.7.2. Break Down

The basic idea is to calculate the contribution of an explanatory prediction of $f(x)$ as changes in the expected dependence of the model response on other variables. This means that we start with the average response of the model, successively adding the variables. Of course, the order in which the variables are arranged also influences the contribution values. If our model is additional, the arrangement of individual variables and values will be the same. If we have a non-additive model with $p$ variables, we have $p!$ layouts, it is complicated by calculation.

Let's denote $v(j, x_*)$ as a measure of the importance of the $j-$th variable for observation $x_*$, i.e. the $j-$th variable's contribution to the observation prediction $x_*$.

We would like the sum of individual contributions of the variables $v(j, x_*)$ to be equal to the prediction for the $x_*$ observation, so $f(x_*) = v_0 + \sum_{j=1}^{p} v(j, x_*)$, where $v_0$ is the average model response. We can also write this equation as:

$$E_X[f(X)|X^1 = x_*^1, \ldots, X^p = x_*^p] = E_X[f(X)] + \sum_{j=1}^{p} v(j, x_*),$$

then a natural proposal for $v(j, x_*)$ is

$$v(j, x_*) = E_X[f(X)|X^1 = x_*^1, \ldots, X^j = x_*^j] - E_X[f(X)|X^1 = x_*^1, \ldots, X^{j-1} = x_*^{j-1}].$$

In other words, the contribution of the jth variable is the difference between the expected value of the model prediction provided that $j$ of the first variables takes observation values $x_*$ and the expected value provided that $j-1$ of the variables takes observation values equal to $x_*$.

Note that the definition assumes that $v(j, x_*)$ depends on the order of explanatory variables.

Let's move on to the general case. Let's mark $J$ as a subset of set $K$. $K$ is a set of indexes such that $K = \{1, 2, \ldots, p\}$. $J$ is a subset of the $K$ indexes set, which is $J = \{j_1, j_2, \ldots, j_K\}$, where $j_k \in \{1, 2, \ldots, p\}$ for each $k$. In addition, let's define $L$ as a subset of $M$, where $M$ is the complement to the set of indexes for the $K$ set. Therefore, $L = \{l_1, l_2, \ldots, l_M\}$, where $l_m \in \{1, 2, \ldots, p\}$ for each $l$. The intersection of subset J with subset L is an empty set ($J \cap L = \emptyset$).

Thus,

$$\Delta^{L|J}(x_*) \equiv E_X[f(X)|X^{l_1} = x_*^{l_1}, \ldots, X^{l_M} = x_*^{l_M}, X^{j_1} = x_*^{j_1}, \ldots, X^{j_K} = x_*^{j_K}]$$

$$-E_X[f(X)|X^{j_1} = x_*^{j_1}, \ldots, X^{j_K} = x_*^{j_K}].$$

In other words, $\Delta^{L|J}(x_*)$ is the change between the expected prediction value provided that the indexed variables in the $J \cup L$ set take observation values $x_*$ and the expected value provided that the indexed variables in the J set take observation values $x_*$.

The $l-$th impact can be written as

$$\Delta^{l|J}(x_*) \equiv \Delta^{\{l\}|J}(x_*) = E_X[f(X)|X^{j_1} = x_*^{j_1}, \ldots, X^{j_K} = x_*^{j_K}, X^l = x_*^l]$$

$$-E_X[f(X)|X^{j_1} = x_*^{j_1}, \ldots, X^{j_K} = x_*^{j_K}].$$

Thus, $\Delta^{l|J}$ is the change between the prediction when determining the values of explanatory variables for indexes in the $J \cup \{l\}$ set equal to values of $x_*$ and the prediction when determining the values of explanatory variables for indexes in the $J$ set equal to values of $x_*$.

If $J = \emptyset$, then

$$\Delta^{l|\emptyset}(x_*) = E_X[f(X)|X^l = x_*^l] - E_X[f(X)] = E_X[f(X)|X^l = x_*^l] - v_0.$$

It follows that

$$v(j, x_*) = \Delta^{j|\{1,\ldots,j-1\}}(x_*) = \Delta^{\{1,\ldots,j\}|\emptyset}(x_*) - \Delta^{\{1,\ldots,j-1\}|\emptyset}(x_*).$$

To choose an ordering according to which the variables with the largest contributions are selected first, one can apply a two-step procedure. In the first step, the explanatory variables are ordered. In the second step, the conditioning is applied according to the chosen order of variables.

In the first step, the ordering is chosen based on the decreasing value of the scores equal to $|\ \Delta^{k|\emptyset}\ |$. The variable contributions can be positive or negative, for this we need an absolute value. In the second step, the variable-importance measure for the $j$-th variable is calculated as

$$v(j, x_*) = \Delta^{j|J},$$

where

$$J = \{k :|\ \Delta^{k|\emptyset}\ |<|\ \Delta^{j|\emptyset}\ |\},$$

that is, $J$ is the set of indices of explanatory variables that have scores $|\ \Delta^{k|\emptyset}\ |$ smaller than the corresponding score for variable $j$.

Below is an example of a Break Down plot (Figure 1.8) for observations from apartments set.

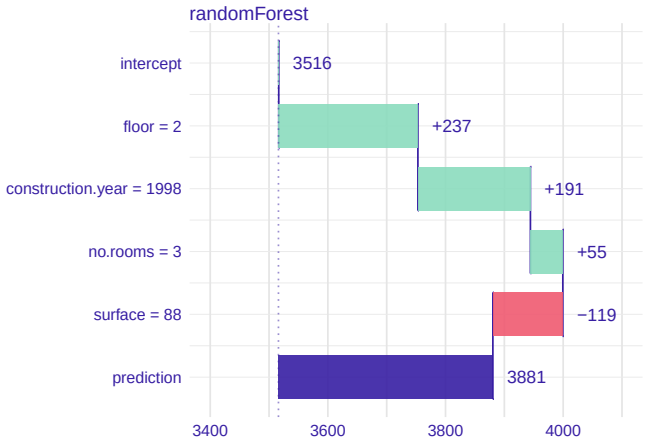



Figure 1.8: Break Down plot for observation from `apartments` set and random forest model. From the top, a vertical line represents the average response of the model, the green and red bars correspond to the contribution of the variable to the prediction. The green ones take positive values, i.e. increase the prediction value, while the red ones take negative values, i.e. decrease the prediction value. The violet bar corresponds to the prediction value for the observation. The numerical values next to the bars inform about the impact. On the x-axis we have model prediction value, on the y-axis, we have variables and their values for the observation.

### 1.7.3. Shapley values

The Shapley value method is based on Break Down predictions into parts. This is a slightly different approach than in the Break Down method. It is based on the idea of averaging the input value of a given variable overall or a large number of possible orders.

Let us consider the permutation of the set of indexes $J = \{1, 2, ..., p\}$ corresponding to $p$ of the explanatory variables in the model $f()$. Let us mark by $\pi(J, j)$ the set of indexes which are

the set $J$ before $j-$th variable. If $j-$th variable is first then $\pi(J, j) = \emptyset$. The Shapley value for observation $x_*$ is defined as follows:

$$\varphi(x_*, j) = \frac{1}{p!} \sum_J \Delta^{j|\pi(J,j)}(x_*),$$

where the sum is counted after all $p!$ possible permutations, and the importance of the variable is defined as in the Break Down method ($\Delta^{j|\pi(J,j)}(x_*)$ ). Essentially, $\varphi(x_*, j)$ is the average importance of the variables in all possible orders of explanatory variables.

An example of a Shapley value in Figure 1.9.

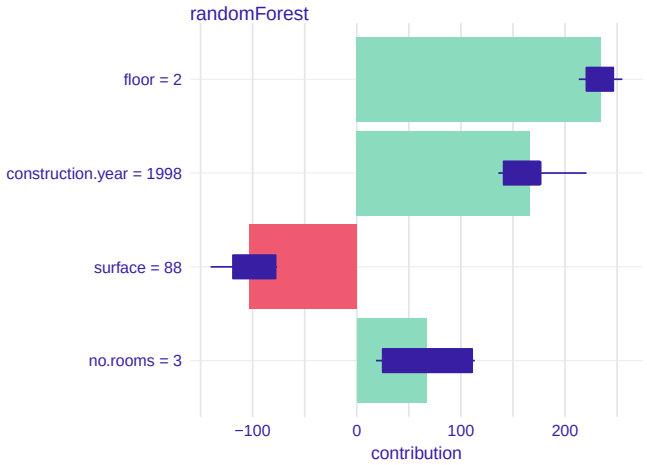



Figure 1.9: Shapley values plot for observation from `apartments` set and random forest model. The green and red bars correspond to the contribution of the variable to the prediction. The green ones take positive values, i.e. increase the prediction value, while the red ones take negative values, i.e. decrease the prediction value. Purple boxplots show the distribution of the attribution of a variable from every possible combination of variable layouts. On the x-axis we have model prediction value, on the y-axis, we have variables and their values for the observation.

# 2. Use cases

In this section, we describe the examples which show methodology from the first section. We construct an example of Ceteris Paribus profiles, then we calculate local variable importance. Finally, we show a comparison of the proposed measure with LIME, Break Down, and Shapley Values. Below we present data sets for which we will show examples. The first example is Friedman's regression problem, while the second is a set of data on property sales. We describe how the models were created on the data sets. We attach the code needed to replicate the analysis.

## 2.1. Fiedman's regression problem

Let consider the regression problem Friedman which was described in 1991 [Friedman, 1991]. We have 10 independent variables uniformly distributed on the interval $[0, 1]$, but only 5 out of these 10 are in the predictive function. Outputs are created according to the formula

$$y = 10sin(\pi x_1 x_2) + 20(x_3 - 0.5)^2 + 10x_4 + 5x_5 + \varepsilon, \tag{2.1}$$

where $\varepsilon$ in $N(0, 1)$.

We generate the data using the `mlbench.friedman1()` function from the `mlbench` [Leisch and Dimitriadou, 2012] R package. It allows us to generate a sample of a given size. We use 10000 observations, below the code we generate the data with.

```
library(mlbench)
friedman_data <- mlbench.friedman1(n = 10000)
```

Examples of several observations are included in the Table 2.1.

For data from the Friedman problem, we use the prediction function with the formula defined in (2.1).

| | X1 | X2 | X3 | X4 | X5 | X6 | X7 | X8 | X9 | X10 | $\varepsilon$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0.77 | 0.30 | 0.99 | 0.66 | 0.21 | 0.41 | 0.35 | 0.46 | 0.71 | 0.61 | 0.25 |
| 2 | 0.64 | 0.16 | 0.02 | 0.85 | 0.95 | 0.99 | 0.56 | 0.98 | 0.63 | 0.87 | 0.07 |
| 3 | 0.81 | 0.62 | 0.15 | 0.08 | 0.76 | 0.47 | 0.63 | 0.96 | 0.57 | 0.62 | 1.11 |
| 4 | 0.30 | 0.17 | 0.68 | 0.22 | 0.63 | 0.81 | 0.36 | 0.00 | 0.96 | 0.75 | -0.44 |
| 5 | 0.78 | 0.36 | 0.32 | 0.83 | 0.83 | 0.23 | 0.46 | 0.15 | 0.26 | 0.84 | -0.87 |
| 6 | 0.22 | 0.44 | 0.49 | 0.87 | 0.94 | 0.83 | 0.40 | 0.30 | 0.66 | 0.83 | 0.20 |

Table 2.1: Table with 6 sample observations of the set for the regression Friedman problem.

```
library(DALEX)
explain_friedman <- explain(model = NULL,
                            data = friedman_data$x,
                            y = friedman_data$y,
                            predict_function = function(model, data){
                                      10*sin(pi*data[,1]*data[,2]) +
                                      20*(data[,3]-0.5)^2 + 10*data[,4] +
                                      5*data[,5]
                                    }
                            )
```

### 2.1.1. Ceteris Paribus profiles

In subsection 2.1.1 we have introduced a data set and model on which we will continue to work. Now we will look at the Ceteris Paribus profiles. For calculation and drawing we will use R package `ingredients`.

Profiles are created in the following way, to the function `ceteris_paribus()` from the `ingredients` R package we give the `DALEX::explainer()` object, which we created earlier, and we give the observation for which we calculate ceteris paribus profiles, in this case, it is the first observation from the set.

```
library(ingredients)
profile <- ceteris_paribus(explain_friedman,
                           new_observation = data_friedman[1,]
                           )
```

To draw calculated profiles we use `plot()` function. Adding to the plot a value for a selected observation by `show_observation()` function.

```
plot(profile) + show_observations(profile)
```

In the Figure 2.1 we have Ceteris Paribus profiles. According to the previous definition, the prediction is only related to the first five variables, for them, we have noticed changes in the profiles, the other five variables have these flat profiles.

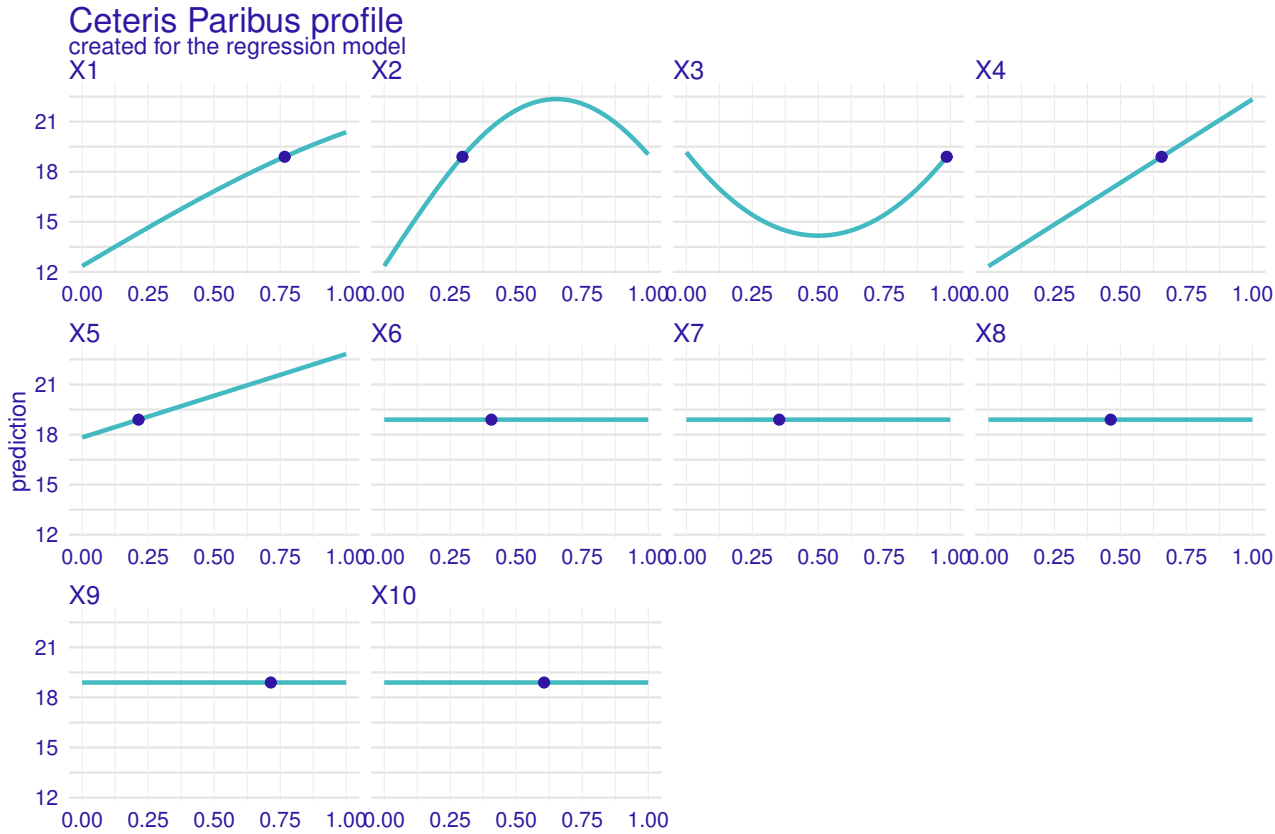



Figure 2.1: Ceteris Paribus profiles for observation from the Friedman's regressive problem set. The purple dot is the predictive value for the selected observation, the cyan line indicates the Ceteris Paribus profile. On the x-axis, we have the value of the variable under consideration, while on the y-axis we have the prediction value. Each plot corresponds to one variable in the model.

Now let's look at the profiles for 100 randomly selected observations. In Figure 2.2 we can see that, as before, the variables X6, X7, X8, X9, and X10 have flat profiles. For variables X3, X4, and X5 the profiles are arranged in the same form. For variables X1 and X2, profiles take different forms. This is due to the interaction of the variable X1 with X2.

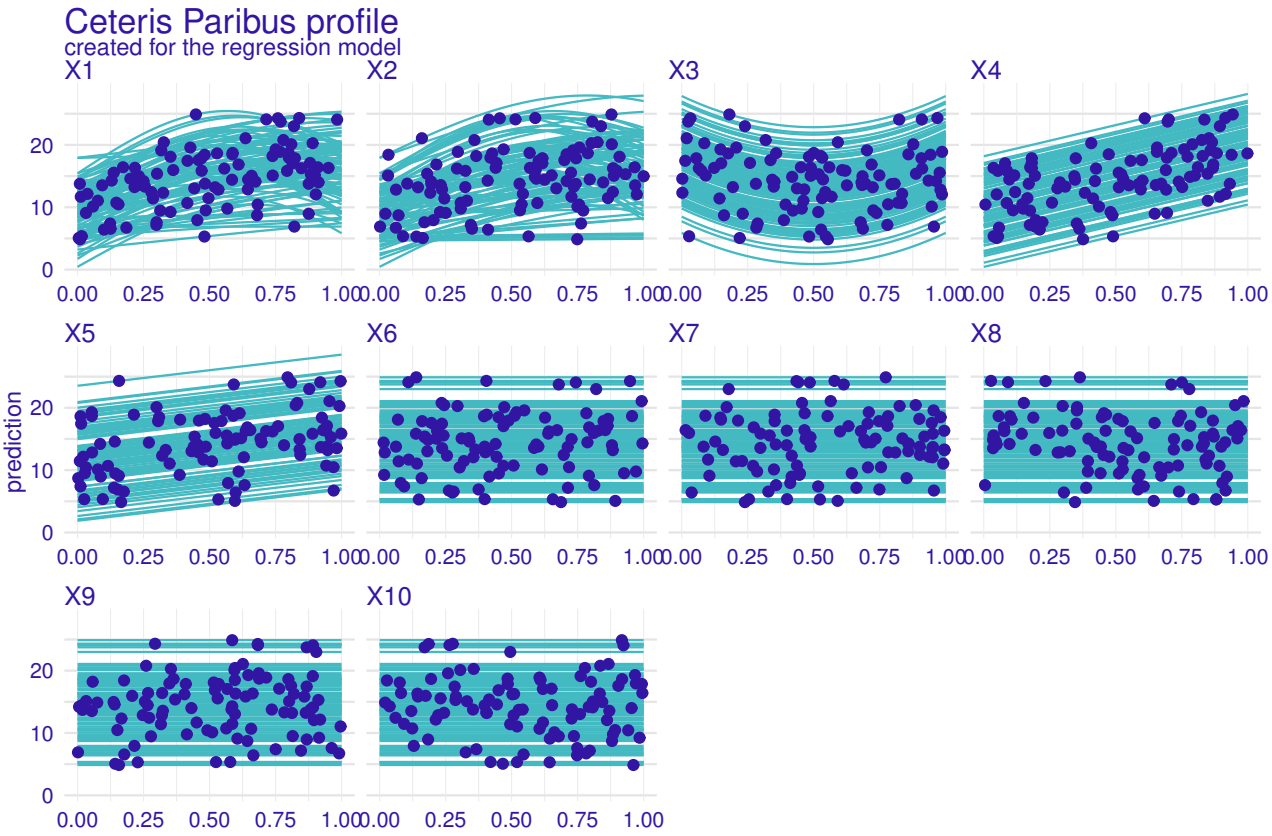



Figure 2.2: Ceteris Paribus profiles for 100 observations from the Friedman's regressive problem set. The purple dot is the predictive value for the observations, the cyan lines indicates the Ceteris Paribus profiles. On the x-axis, we have the value of the variable under consideration, while on the y-axis we have the prediction value. Each panel corresponds to one variable in the model.

### 2.1.2. Example of use new measure

In this subsection, we show the application of the measure of oscillations to evaluate the importance of variables in the previously mentioned example. Starting from Friedman's regression problem, according to formula (2.1) data are generated from a uniform distribution into [0,1] and only 5 out of 10 variables are used to model predictions. So what we should expect is that the first 5 variables will be important for the prediction value, and the rest will not. We could already see it on the Ceteris Paribus profile (Figure 2.1). The last 5 of them are completely flat. The first variant of the measure that we will use to evaluate will be given in the definition 1.12. This means that all parameters take the `TRUE` value.

```
library(vivo)
measure <- local_variable_importance(profile = profile,
```

```
                                    data = data_friedman,
                                    absolute_deviation = TRUE,
                                    point = TRUE,
                                    density = TRUE
                                    )
```

The `measure` object contains the calculated measure of the oscillation for the observation that was in Figure 2.1. Below the values in the Table 2.2.

| variable_name | X1 | X2 | X3 | X4 | X5 | X6 | X7 | X8 | X9 | X10 |
|---|---|---|---|---|---|---|---|---|---|---|
| measure | 2.61 | 2.61 | 3.11 | 2.79 | 1.69 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |

Table 2.2: Oscillation measure for selected observation.

Using the `plot()` function, we can draw the obtained measure.

```
plot(measure)
```

In Figure 2.3, we have a plot of the measure. As we could see from the Ceteris Paribus plots, the first five variables have values above zero, while the other five are zeroes, i.e. they do not affect the prediction for this observation.

We have eight variants of the measure based on oscillations. Now let's compare the obtained above with the variant when we do not take into consider the density of variables. In our case, the variables have a uniform distribution, so we should not notice major differences. To compare these two measures we have to calculate the wanted measure, and then draw with a `plot()` function.

```
measure_without_density <- local_variable_importance(profile = profile,
                                                     data = data_friedman,
                                                     absolute_deviation = TRUE,
                                                     point = TRUE,
                                                     density = FALSE
                                                     )


plot(measure,
    measure_without_density,
    color = "_label_method_",
    variables = c("X1", "X2", "X3", "X4", "X5"
    )
```

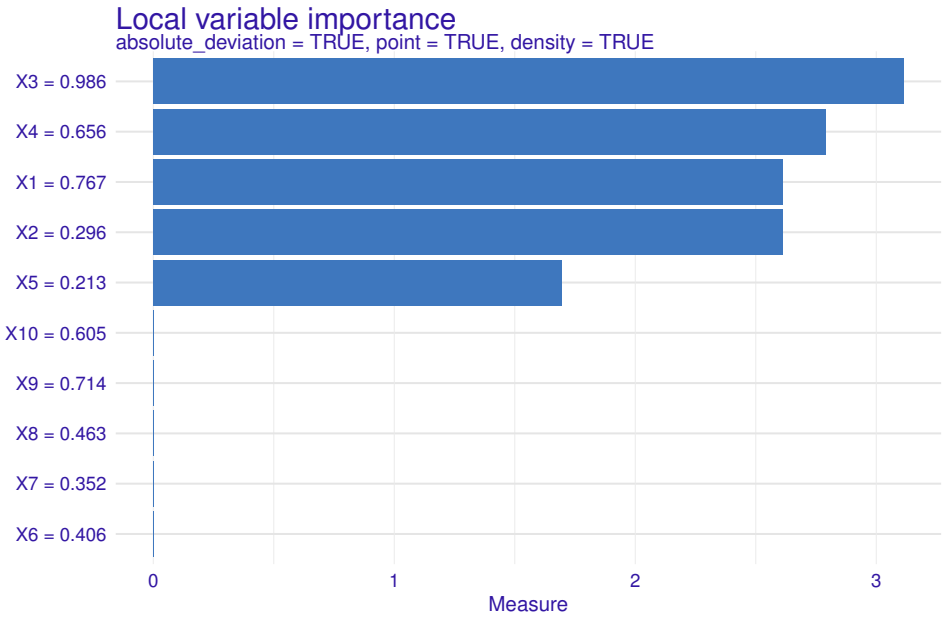



Figure 2.3: The plot shows the measure of oscillation. The bars represent the value of the measure. On the x-axis, we have the range of the measure of the oscillation, on the y-axis we have the variables in the model together with their values for the observation $x_*$. In the subtitle, we have information about the parameters used to calculate the model.

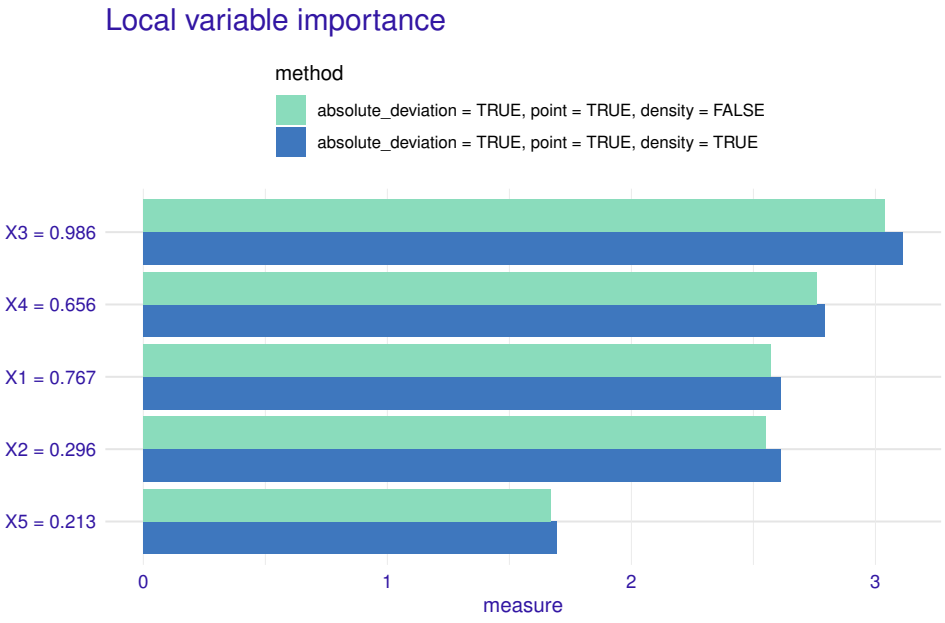



Figure 2.4: The plot shows the measure of oscillation calculated in two ways. The bars represent the value of the measure. On the x-axis, we have the range of the measure of the oscillation, on the y-axis, we have the variables in the model, color indicates the type of measure.

As we can see in Figure 2.4, calculating the measure taking into account the density or not,

the measure of importance of variables is almost the same. For each variable, the measure of density is slightly higher.

## 2.2. House Sales

The dataset contains house sale prices for King County, which includes Seattle. It includes homes sold between May 2014 and May 2015. The data is publicly available, on Kaggle and OpenML. Data contains 19 house features plus the price and the id columns, along with 21613 observations. The following Table 2.3 describes the variables contained in this dataset.

In the data, there are houses that have been sold twice or even three times. All values of variables for these houses were the same, the only difference was their price. For each pair (three) of houses, we decided to aggregate to one observation calculating the average price. An additional element of data set transformations is the conversion of units from square feet to square meters. Moreover, the data contain information on the year of construction and the year of renovation of the property. For these variables, a calculation of the property age was applied, i.e. the difference between the year of sale and the year of construction. In the case of the year of renovation, however, the number of years after the renovation, i.e. the year of the sale minus the year of renovation, in case of no renovation, the date of construction is chosen. To enrich the data set, external data are also used, i.e. data on public transport and cultural places.

We worked on this data set during the Interpretable Machine Learning class, my team prepared a chapter of the XAI book based on real estate sales data. We describe data preparation, model building, and explanation. More about this work can be found on this page `https://pbiecek.github.io/xai_stories/story-house-sale-prices.html`.

We are considering here a regressive model, because the variable explained in the model is the price logarithm. We will use one of the models built. The model is built in `mlr` [Bischl et al., 2016] R package with `xgboost` [Chen and Guestrin, 2016] R package.

### 2.2.1. Ceteris Paribus profiles

Now let's consider the property sales data set. We will calculate Ceteris Paribus profiles.

```
profile_house <- ceteris_paribus(explain_xgb, test[1,])
plot(profile_house) + show_observations(profile_house)
```

For the variables *bedrooms*, *bathrooms*, *floor*, *dist_stop*, *ncult*, *age*, *since_renovated*, and *m2_lot15* the profiles are close to flat, so the model prediction would not change when the

| Variable | Description |
|---|---|
| id | unique ID for each home sold |
| date | date of the home sale |
| price | price of each home sold |
| bedrooms | number of bedrooms |
| bathrooms | number of bathrooms, where .5 accounts for a room with a toilet but no shower |
| sqft_living | square footage of the apartments interior living space |
| sqft_lot | square footage of the land space |
| floors | number of floors |
| waterfront | apartment was overlooking the waterfront or not |
| view | how good the view of the property was |
| condition | condition of the apartment |
| grade | level of construction and design |
| sqft_above | the square footage of the interior housing space that is above ground level |
| sqft_basement | the square footage of the interior housing space that is below ground level |
| yr_built | the year the house was initially built |
| yr_renovated | the year of the house's last renovation |
| zipcode | zipcode area |
| lat | lattitude |
| long | longitude |
| sqft_living15 | the square footage of interior housing living space for the nearest 15 neighbors |
| sqft_lot15 | the square footage of the land lots of the nearest 15 neighbors |

Table 2.3: Description of variables in the House Sales dataset.

values of these variables change. In the case of the *view* and *grade* variable, the model prediction increases as the values of these variables increase, i.e. properties with a better view and better construction and design quality have a higher price. For the variable *lat* and *long* describing the geographical location of the property, the profiles show that if the property was located further north or slightly to the west, the price could increase. The variables *m2_living* and *m2_lot*, indicate a sharp increase in price as the value of this variable increases. In other words, if it was a property with a larger area, the price would increase. Similarly, we see an increase in the variables *m2_above* and *m2_lining15*. The only variable that describes space and reaches a lower prediction with an increase in value is *m2_basement*. A possible reason for this is that the area where the property is located in wet and there are frequent flooding of basements, so the

smaller the room, the less chance of flooding.

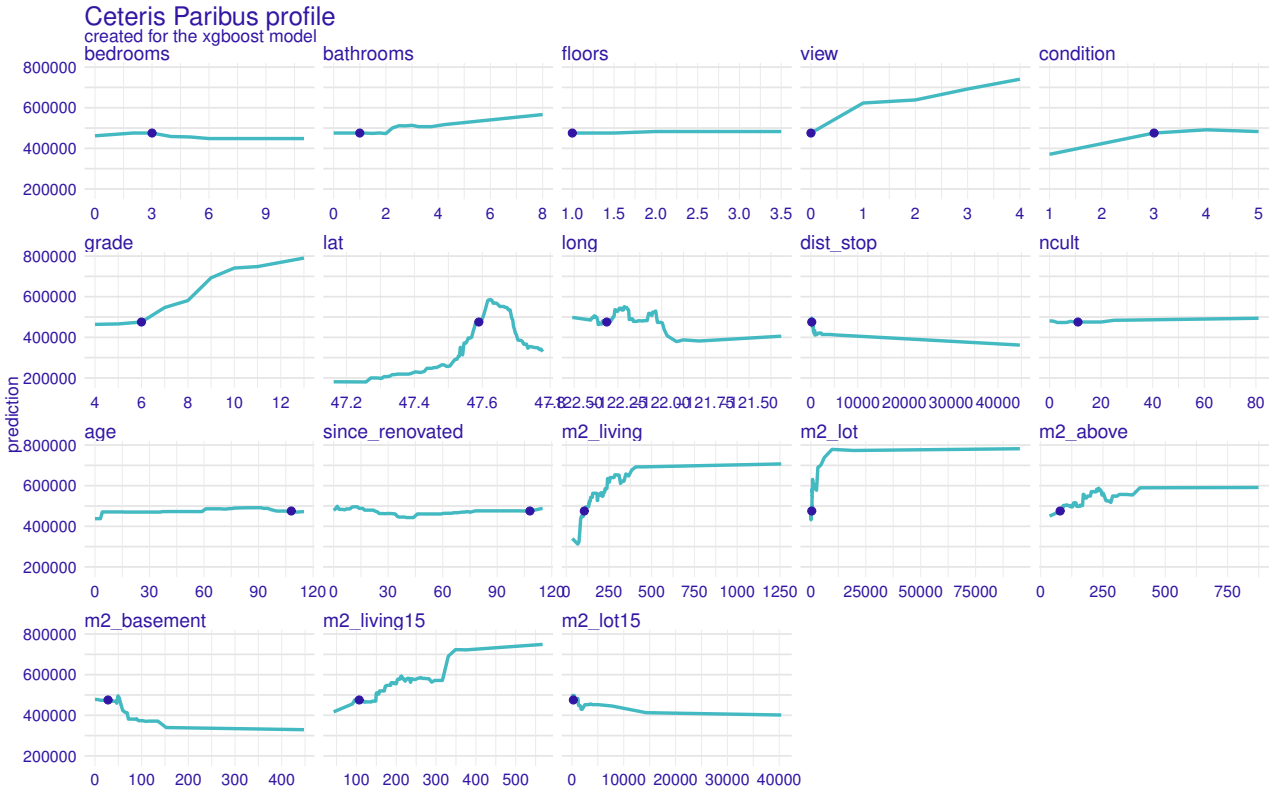
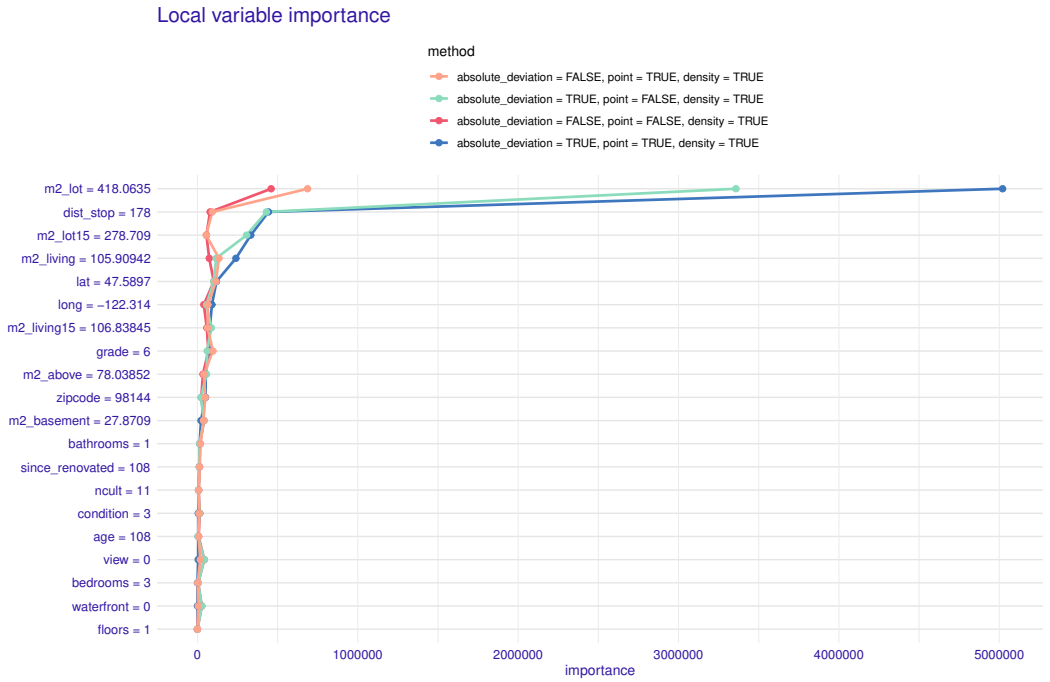



Figure 2.5: Ceteris paribus profiles for variables from the house sales data set. In the plot, a purple dot indicates the observation for which we draw profiles. The cyan line indicates the Ceteris Paribus profile. On the x-axis we have the values of the variables, and on the y-axis, we have the prediction value.

### 2.2.2. Example of use an oscillation measure

Now we will see what results we get for the previously selected observation. We will do the comparison in two steps, we have eight variants of the measure based on Ceteris Paribus profiles. We divide the density parameter into two groups, because our variables have different distributions as opposed to the regression Friedman problem example. Below are two plots showing the calculated local importance of the variables. In Figure 2.6 panel A) we have measures for which the density parameter is equal to TRUE. As we can see, for the first variables (i.e. *m2_lot*, *dist_stop*, *m2_lot15*, *m2_living*) the measures show different values. For these variables, the baseline and calculation methods are crucial. For the other variables, the measures take similar values, they agree on their importance. All variables agree that the variable *m2_lot* is the most important and for this observation, it significantly influences the prediction.

A)




B)

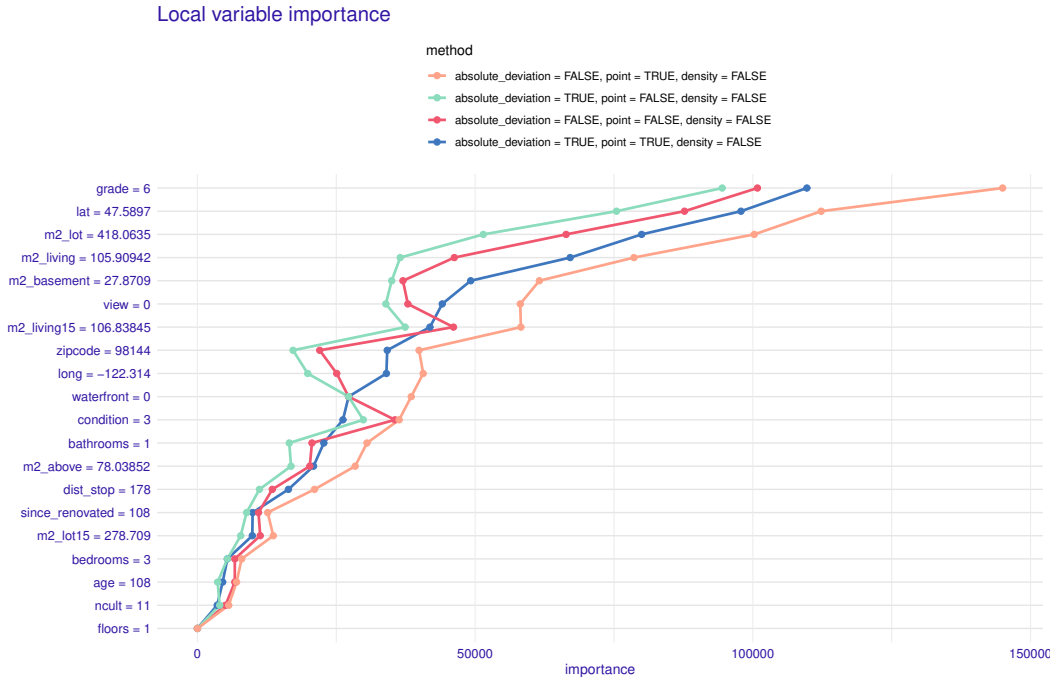



Figure 2.6: Local variable importance in all variants divided into two panels by density parameter. On panel A) variants where the density parameter is equal TRUE, on panel B) variants where density parameter is equal FALSE. On the x-axis we have a value of the measure, on the y-axis, we have variables with value for selected observation. Each color corresponds to one of the variants of the measure.

On the other hand, in Figure 2.6 panel B) we have a comparison of measures for which the density parameter is equal to FALSE. That is, we do not take into the weights based on empirical density. In this case, all methods show the order for the first four variables. These are *grade*, *lat*, *m2_lot*, and *m2_living*. In contrast to the previous chart, we can see greater differences between the measures for the other variables.

### 2.2.3. Comparison of the proposed measure with LIME, BreakDown and Shapley values

In this subsection, we will show the local importance of the variables calculated by the methods described in Chapter 1. All explanations are on the previously selected property.

#### LIME

Below we present the explanation of the LIME package that we have obtained. In the LIME method, low-dimensional explanations are most often used, so we choose 8 variables indicated by the method as the most important. The code to generate the explanation.

```
library(lime)
explain_mod <- lime(test, mod_xgboost, n_permutations = 2000)
xgb_lime <- explain(test[1,], explain_mod, n_features = 8)
plot_features(expl)
```

The most important variables according to the LIME method are *lat* and *m2_living*. *Grade* and *m2_lot* are next.

#### BreakDown

To generate the Break Down chart we will use the following code.

```
library(iBreakDown)
bd_xgb  <- break_down(explain_xgb, test[1, ])
plot(bd_xgb)
```

From the figure, we can see that the *m2_living* variable has the greatest influence on the prediction, and another variable is *lat* and *grade*. The *lat* variable has a positive impact on the prediction.

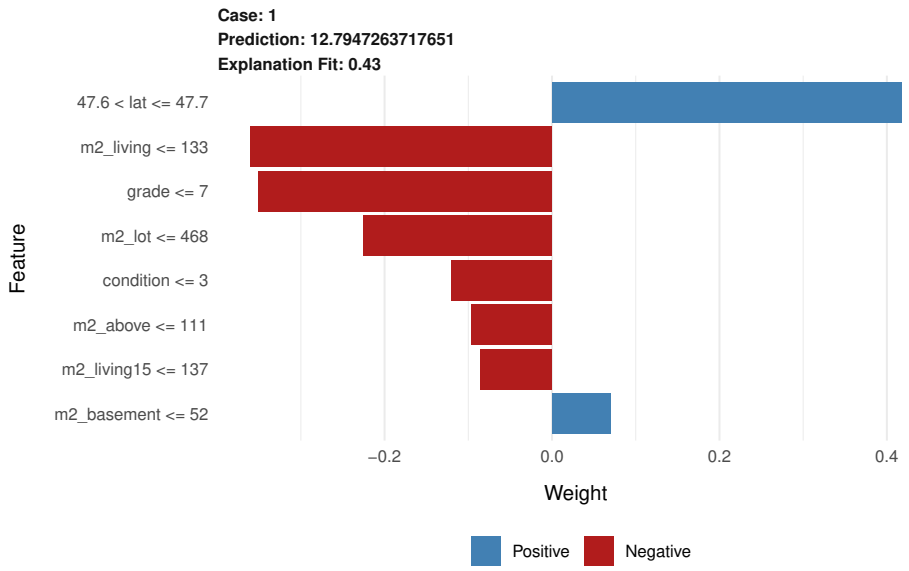



Figure 2.7: Local variable importance by LIME method. On the x-axis, we have the measure of importance, on the y-axis we have the variables. The lengths of the bars correspond to the value of the measure and the color to the positive and negative impact on the prediction value.

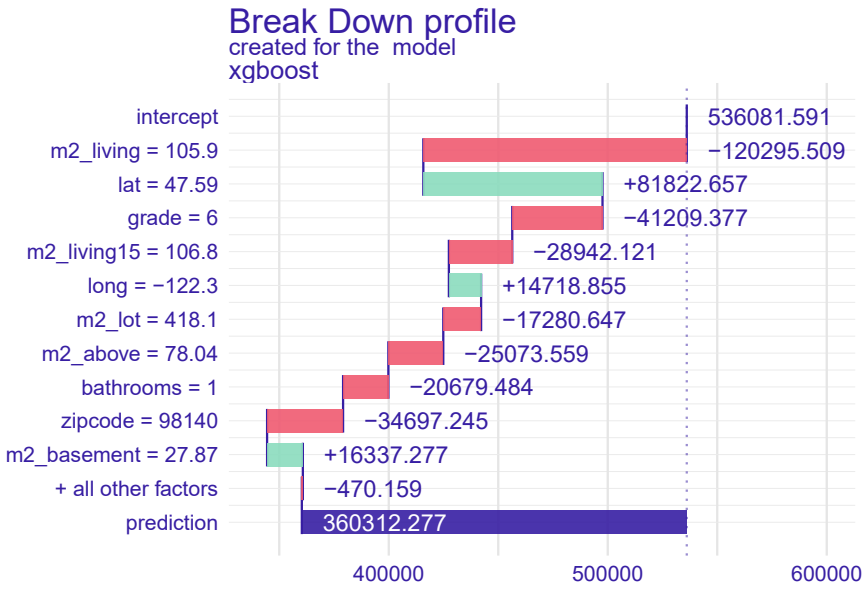



Figure 2.8: Local variable importance by Break Down. On the x-axis, we have a measure of importance, on the y-axis we have variables with values for the selected observation. The lengths of the bars correspond to the values of the measure, and the color is positive (green) and negative (red) for the prediction value.

### Shapley values

The last method, but no least important is Shapey values. We also use the `iBreakdown` R package [Gosiewska and Biecek, 2019] to creating an explanation of Shapey values.

```
shap_xgb <- shap(explain_xgb, test[1,])
plot(shap_xgb)
```

As with Break Down, the most important variables are *lat*, *m2_living* and *grade*.

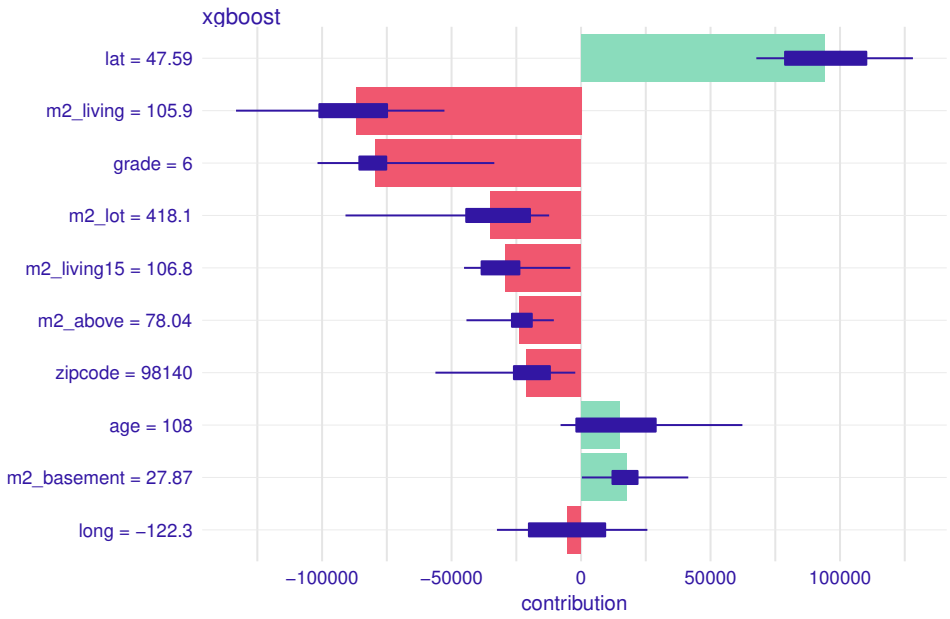



Figure 2.9: Local variable importance by Shap. On the x-axis, we have a measure of importance, on the y-axis we have variables with values for the selected observation. The lengths of the bars correspond to the values of the measure, and the color is positive (green) and negative (red) for the prediction value..

To summarize, for this randomly selected observation, all the methods have agreed to determine the variables that most influence the value of the prediction. There are certain observations for which the explanations received will differ. It is then worth looking at each of them to be able to understand what influenced such a decision. It is worth emphasizing that in explanation machine learning there is no division into better and worse methods, each works in its own way and it is worth paying attention to all of them.

# 3. Software


In this chapter, we look at the structure of the R package `vivo`. It contains functions calculating the local importance of variables introduced in the first chapter. The `vivo` is a part of `DrWhy.AI` collection of tools for Visual Exploration, Explanation and Debugging of Predictive Models. The package is on CRAN in version 0.2.0.

## 3.1. Structure of the package

The `vivo` package has six functions.

- `calculate_variable_split()`,
- `calculate_weight()`,
- `local_variable_importance()`,
- `plot.local_importance()`,
- `global_variable_importance()`,
- `plot.global_importance()`.

We describe the first four functions because they are related to the topic of the thesis. The other two functions are an extension of the global importance of variables. The `calculate_weight()` and `variable_split()` help to calculate a weight based on empirical density on raw data. The main `local_variable_importance()` function calculates the measure of local importance. The `plot.local_importance()` function visualizes the calculated measure. The `local_variable_importance()` takes the `ceteris_paribus_explainer` object created by the `predict_parts()` function from the `DALEX` package or `ceteris_paribus()` from `ingredients` package. The diagram below shows the dependencies between the functions.

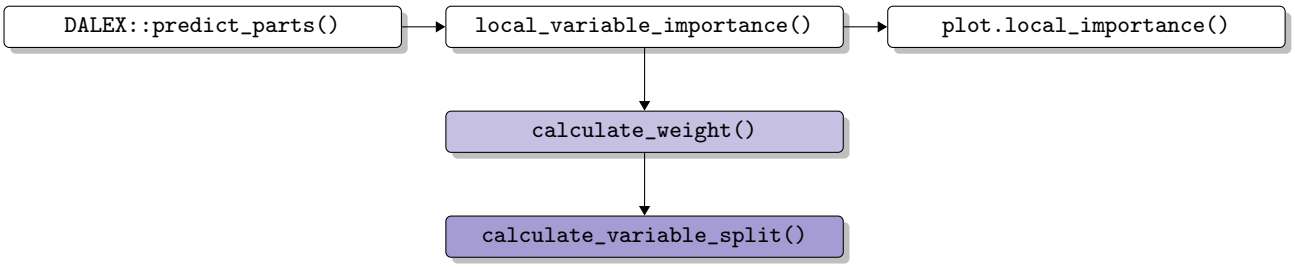



Figure 3.1: Dependencies between the functions of the `vivo` package.

The `vivo` package imports four libraries such as `DALEX`, `ingredients` and `ggplot2`. Below is a visualisation of the dependencies created with R `deepdep` package [Rafacz et al., 2020].

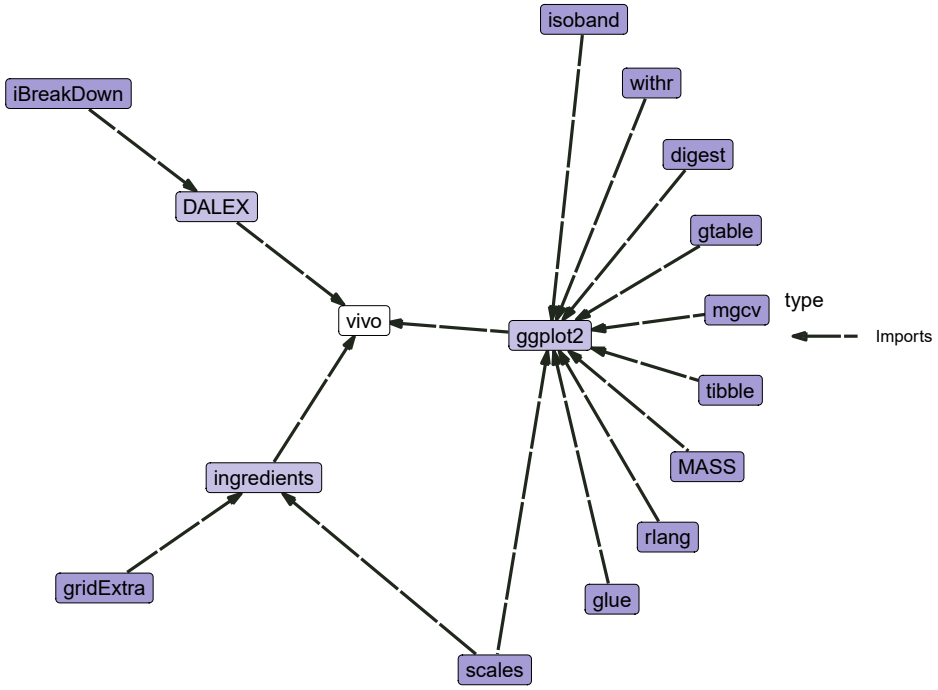



Figure 3.2: Visualization of dependencies for the `vivo` package.

### 3.1.1. `calculate_variable_split` function

This function calculate candidate splits for each selected variable. For numerical variables splits are calculated as percentiles (in general uniform quantiles of the length `grid_points`). For all other variables splits are calculated as unique values.

```
calculate_variable_split(data,
                         variables = colnames(data),
```

```
                grid_points = 101
                )
```

Arguments, which function takes:

| Argument | Description |
|---|---|
| `data` | validation dataset. Is used to determine distribution of observations. |
| `variables` | names of variables for which splits shall be calculated |
| `grid_points` | number of points used for response path |

### 3.1.2. `calculate_weight` function

This function calculate an empirical density of raw data based on variable split from Ceteris Paribus profiles. Then calculated weight for values generated by `DALEX::predict_parts()`, `DALEX::individual_profile()` or `ingredients::ceteris_paribus()`.

```
calculate_weight(profiles,
                 data,
                 variable_split
                 )
```

Arguments, which function takes:

| Argument | Description |
|---|---|
| `profiles` | data.frame generated by `DALEX::predict_parts()`, `DALEX::individual_profile()` or `ingredients::ceteris_paribus()` |
| `data` | data.frame with raw data to model |
| `variable_split` | list generated by `vivo::calculate_variable_split()` |

### 3.1.3. `local_variable_importance` function

This function calculate local importance measure based on Ceteris Paribus profiles in eight variants.

```
local_variable_importance(profiles,
                          data,
                          absolute_deviation = TRUE,
                          point = TRUE,
                          density = TRUE,
                          grid_points = 101)
```

Arguments, which function takes:

| Argument | Description |
|---|---|
| `profiles` | data.frame generated by `DALEX::predict_parts()`, `DALEX::individual_profile()` or `ingredients::ceteris_paribus()` |
| `data` | data.frame with raw data to model |
| `absolute_deviation` | logical parameter, if `absolute_deviation` = TRUE then measue is calculated as absolute deviation, else is calculated as a root from average squares |
| `point` | logical parameter, if `point` = TRUE then measure is calculated as a distance from $f(x)$, else measure is calculated as a distance from average CP |
| `density` | logical parameter, if `density` = TRUE then measure is weighted based on the density of variable, else is not weighted |
| `grid_points` | maximum number of points for profile calculations, the default values is 101, the same as in `ingredients::ceteris_paribus`, if you use a different on, you should also change here |

### 3.1.4. `plot(<local_importance>)` function

Function plot.local_importance plots local importance measure based on Ceteris Paribus profiles.

```
plot(x,
    ...,
    variables = NULL,
    color = NULL,
    type = NULL,
    title = "Local variable importance"
)
```

| Argument | Description |
|---|---|
| `x` | object returned from `local_variable_importance()` function |
| `...` | other parameters |
| `variables` | if not `NULL` then only variables will be presented |
| `color` | a character. How to aggregated measure? Either "_label_method_" or "_label_model_" |
| `type` | a character. How variables shall be plotted? Either "bars" (default) or "lines" |
| `title` | the plot's title, by default "Local variable importance" |

# 4. Summary

In this thesis, I present a new measure of local variable importance in models based on Ceteris Paribus profiles. This measure can be applied to any model in a model agnostic fashion, especially to the so-called black-box models. It allows to determine the most influential variables for the selected observation. In the first chapter, I present the methodology and introduce the new measure, the next step is to formulate possible variants of the measure. Additionally, I present the idea of other measures of local importance of variables. The final effect is the R package `vivo`. All functions included in chapter two are implemented in it. Documentation and examples are available at the link `https://modeloriented.github.io/vivo/`. The stable version of the package is on CRAN, the developer version is on GitHub (`https://github.com/ModelOriented/vivo`). The third chapter is devoted to examples, we show you how to determine the CP profiles and the measure entered. The examples contain codes that can be reproduced. During the work, the package was extended to the global importance of variables based on Patrial Dependence Profiles.

In addition, the results of this thesis were presented at the conference WhyR?2019 as a lightning talk. The title of the presentation is as follows„vivo: Is it Victoria In Variable impOrtance detection?". Also during the MLinPL2019 conference, I presented a poster, entitled „vivo: local variable importance via oscillations". A poster from this conference is attached to the next page.

There is a potential area for future work, we can introduce a measure for two-dimensional Ceteris Paribus profiles or variable interactions.

# vivo: local variable importance via oscillations

Anna Kozak[1], Przemysław Biecek[1]
[1]Faculty of Mathematics and Information Science, Warsaw University of Technology

## Interpretability

When a model has many features and plotting all one-dimensional summary statistics is troublesome, **vivo** indicates which variables are worth paying attention to. The **vivo** is an R package which calculates instance level feature importance (measure of local sensitivity). The feature importance is based on Ceteris Paribus profiles and can be calculated in a few variants.

## Ceteris Paribus Profiles

Ceteris Paribus is a latin phrase meaning *"other things held constant"* or *"all else unchanged"*. Ceteris Paribus Plots show how the model response depends on changes in a single input variable, keeping all other variables unchanged. They work for any Machine Learning model and allow for model comparisons to better understand how a **black box model** works.

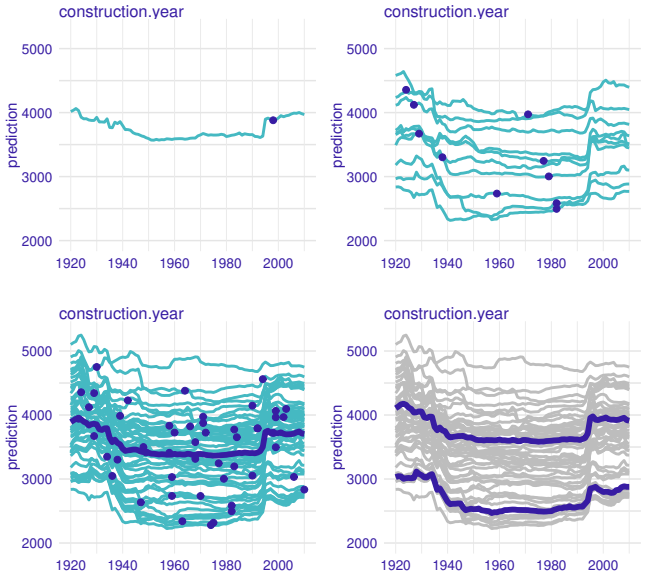



Figure 1: The first plot shows the Ceteris Paribus profile for a single observation. The plot on the right shows the profiles for a few observations. In the lower left corner we have profiles for observations and a line showing their aggregation - a partial dependency plot. The last plot shows the aggregation of profiles using clustering.

## Methodology

The our measure of local variable importance is based on the oscillations of the Ceteris Paribus profiles. In particular, the larger the deviation along the corresponding Ceteris Paribus profile, the larger influence of an explanatory variable on prediction at a particular instance. For a variable that exercises little or no influence on model prediction, the profile will be flat or will barely change.

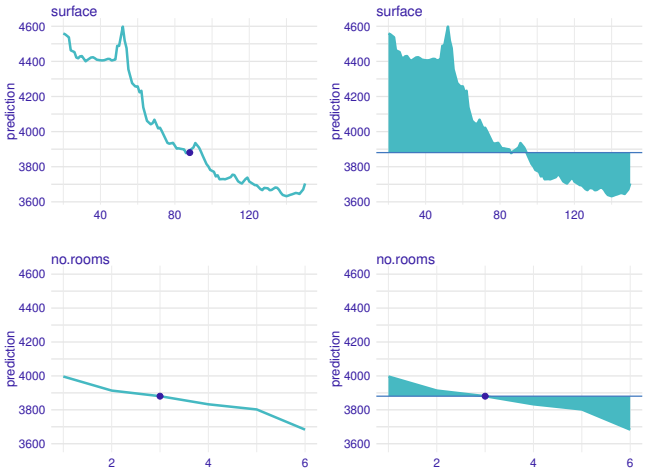
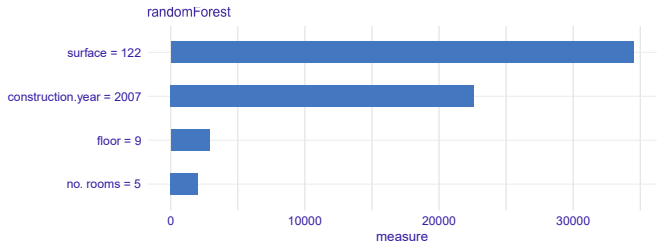


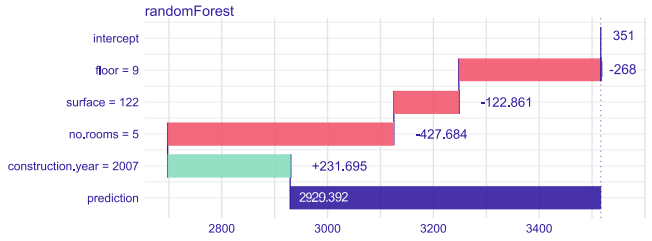

Figure 2: The value of the colored area is our measure. The larger the area, the more important is the variable.

## Comparison of the proposed measure with LIME, iBreakDown and SHAP

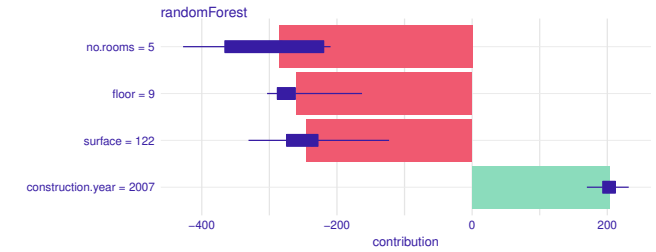
Below is a comparison of methods of local importance of variables based on the black box model - random forest. The vivo, iBreakDown and LIME measures indicate other variables as significant. This only confirms the essence of using various tools when explaining **black box models**.



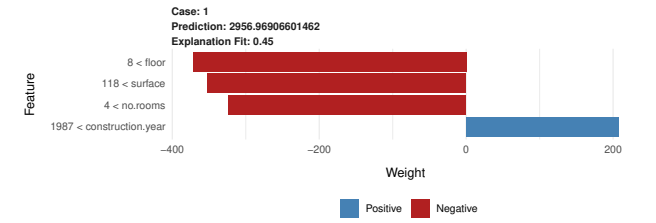

(a) vivo measure




(b) iBreakDown measure




(c) SHAP values




(d) LIME measure

Figure 3: Comparison of methods

## Details

Oscillations of Ceteris Paribus profiles are easy to interpret and understand. By using the average of oscillations, it is possible to select the most important variables for an instance prediction. This method can easily be extended to two or more variables.

## References

[1] Przemysław Biecek. *ingredients: Effects and Importances of Model Ingredients*, 2019. URL https://cran.r-project.org/web/packages/ingredients/index.html.

[2] Przemysław Biecek and Tomasz Burzykowski. *Predictive Models: Explore, Explain, and Debug*. 2019. URL https://pbiecek.github.io/PM_VEE/.

[3] Alicja Gosiewska and Przemyslaw Biecek. iBreakDown: Uncertainty of Model Explanations for Non-additive Predictive Models, 2019. URL https://arxiv.org/abs/1903.11420v1.

[4] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 4765–4774. Curran Associates, Inc., 2017. URL http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf.

[5] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. *Why Should I Trust You?: Explaining the Predictions of Any Classifier*, page 1135–1144. ACM Press, 2016. ISBN 978-1-4503-4232-2. doi: 10.1145/2939672.2939778. URL http://dl.acm.org/citation.cfm?doid=2939672.2939778.

# Bibliography

[Apley, 2017] Apley, D. W. (2017). Aleplot: Accumulated local effects (ale) plots and partial dependence (pd) plots.

[Baniecki and Biecek, 2019] Baniecki, H. and Biecek, P. (2019). modelStudio: Interactive studio with explanations for ML predictive models. *The Journal of Open Source Software*.

[Biecek, 2018] Biecek, P. (2018). Dalex: Explainers for complex predictive models in r. *Journal of Machine Learning Research*, 19.

[Biecek, 2019] Biecek, P. (2019). *ceterisParibus: Ceteris Paribus Profiles*. R package version 0.3.1.

[Biecek et al., 2019] Biecek, P., Baniecki, H., and Izdebski, A. (2019). *ingredients: Effects and Importances of Model Ingredients*. R package version 0.5.0.

[Biecek and Burzykowski, 2019] Biecek, P. and Burzykowski, T. (2019). *Explanatory Model Analysis*.

[Bischl et al., 2016] Bischl, B., Lang, M., Kotthoff, L., Schiffner, J., Richter, J., Studerus, E., Casalicchio, G., and Jones, Z. M. (2016). mlr: Machine learning in r. *Journal of Machine Learning Research*, 17(170):1–5.

[Chen and Guestrin, 2016] Chen, T. and Guestrin, C. (2016). Xgboost: A scalable tree boosting system. *CoRR*, abs/1603.02754.

[Chollet, 2019] Chollet, F. (2019). *Deep Learning for humans*. Python package version 2.3.1.

[Densford, 2018] Densford, F. (2018). Report: Ibm watson delivered 'unsafe and inaccurate' cancer recommendations.

[Duffy, 2019] Duffy, C. (2019). Apple co-founder steve wozniak says apple card discriminated against his wife.

[EU, 2018] EU (2018). General data protection regulation.

[Friedman et al., 2009] Friedman, J., Hastie, T., and Tibshirani, R. (2009). *The Elements of Statistical Learning*.

[Friedman, 1991] Friedman, J. H. (1991). Multivariate adaptive regression splines. 19(1):1–67.

[Friedman, 2000] Friedman, J. H. (2000). Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29:1189–1232.

[Goldstein et al., 2014] Goldstein, A., Kapelner, A., Bleich, J., and Pitkin, E. (2014). Peeking inside the black box: Visualizing statistical learning with plots of individual conditional expectation.

[Gosiewska and Biecek, 2019] Gosiewska, A. and Biecek, P. (2019). iBreakDown: Uncertainty of Model Explanations for Non-additive Predictive Models.

[Greenwell et al., 2019] Greenwell, B., Boehmke, B., and Gray, B. (2019). *vip: Variable Importance Plots*. R package version 0.1.3.

[Greenwell et al., 2018] Greenwell, B. M., Boehmke, B. C., and McCarthy, A. J. (2018). A simple and effective model-based variable importance measure.

[Jiangchun, 2018] Jiangchun, L. (2018). *Python Partial Dependence Plot Toolbox*.

[Korobov and Lopuhin, 2017] Korobov, M. and Lopuhin, K. (2017). Eli5: Python package which helps to debug machine learning classifiers and explain their predictions.

[Kramer and Choudhary, 2018] Kramer, A. and Choudhary, P. (2018). *Model Interpretation Library*. Python package version 1.1.2.

[Kretowicz et al., 2020] Kretowicz, W., Baniecki, H., and Biecek, P. (2020). *moDel Agnostic Language for Exploration and eXplanation*.

[Kuźba, 2019] Kuźba, M. (2019). *Ceteris Paribus python package*. Python package version 0.5.1.

[Lazer et al., 2014] Lazer, D., Kennedy, R., King, G., and Vespignani, A. (2014). The parable of google flu: Traps in big data analysis. *Science(New York, N.Y.)*, 343:1203–5.

[Leisch and Dimitriadou, 2012] Leisch, F. and Dimitriadou, E. (2012). *mlbench: Machine Learning Benchmark Problems*. R package version 2.1-1.

[Liaw and Wiener, 2002] Liaw, A. and Wiener, M. (2002). Classification and regression by randomforest. *R News*, 2(3):18–22.

[Lundberg and Lee, 2017] Lundberg, S. M. and Lee, S.-I. (2017). A Unified Approach to Interpreting Model Predictions. pages 4765–4774.

[Maksymiuk et al., 2019] Maksymiuk, S., Biecek, P., Pekala, K., and Kozak, A. (2019). *Extension for 'DALEX' Package*. R package version 0.2.0.

[Molnar, 2019] Molnar, C. (2019). *Interpretable Machine Learning*. https://christophm.github.io/interpretable-ml-book/.

[Molnar et al., 2018] Molnar, C., Bischl, B., and Casalicchio, G. (2018). iml: An R package for Interpretable Machine Learning. *JOSS*, 3(26):786.

[Mueller, 2019] Mueller, A. (2019). *A set of python modules for machine learning and data mining*. Python package version 0.21.3.

[Rafacz et al., 2020] Rafacz, D., Baniecki, H., and Bakala, S. M. M. (2020). *Visualise and Explore the Deep Dependencies of R Packages*. R package version 0.2.1.

[Ribeiro et al., 2016] Ribeiro, M. T., Singh, S., and Guestrin, C. (2016). *Why Should I Trust You?: Explaining the Predictions of Any Classifier*, page 1135–1144. ACM Press.

[Rochford, 2017] Rochford, A. (2017). *The Python Individual Conditional Expectation Toolbox*.

[Staniak and Biecek, 2018] Staniak, M. and Biecek, P. (2018). Explanations of model predictions with live and breakdown packages.

[Yong, 2018] Yong, E. (2018). A popular algorithm is no better at predicting crimes than random people.

# List of Figures

# List of Tables

# List of Annexes

## Content of the included CD

The attached CD contains the vivo package for R (version 0.2.0) and the code included in the paper.