FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Assignment of bachelor's thesis

Title:	A Heuristic for Automatic Distribution Selection in Open Source H2O AutoML
Student:	Vojtěch Müller
Supervisor:	Ing. Tomáš Frýda
Study program:	Informatics
Branch / specialization:	Knowledge Engineering
Department:	Department of Applied Mathematics
Validity:	until the end of summer semester 2022/2023

Instructions

Describe the Automated Machine Learning (AutoML). Focus on algorithms supporting attribute "distribution".

1. Explain why a proper selection of this attribute is essential for prediction performance.
2. Select at least one appropriate heuristic to choose distribution and implement it into the H2O AutoML.
3. Benchmark your implementation with suitable datasets, and discuss the difference in prediction performance.

Electronically approved by Ing. Karel Klouda, Ph.D. on 10 July 2022 in Prague.

Bachelor's thesis

A HEURISTIC FOR AUTOMATIC DISTRIBUTION SELECTION IN OPEN SOURCE H2O AUTOML

Vojtěch Müller

Faculty of Information Technology
Department of Applied Mathematics
Supervisor: Ing. Tomáš Frýda
May 11, 2023

Czech Technical University in Prague
Faculty of Information Technology

Citation of this thesis: Müller Vojtěch. *A Heuristic for Automatic Distribution Selection in Open Source H2O AutoML.* Bachelor's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2023.

Contents

List of Figures

List of Tables

List of code listings

Chtěl bych poděkovat především vedoucímu této práce – Ing. Tomáši Frýdovi, za velmi aktivní přístup k vedení práce, za časté konzultace, vysvětlování, rychlé odepisování a především za jeho trpělivost. Také děkuji svým rodičům, prarodičům a dalším blízkým za podporu.

Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis. I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular that the Czech Technical University in Prague has the right to conclude a license agreement on the utilization of this thesis as a school work under the provisions of Article 60 (1) of the Act.

In Prague on May 11, 2023 .....

Abstract

This thesis aims to create a heuristic for the automatic selection of a hyperparameter that represents a statistical distribution of the target variable in the *H2O AutoML framework*. Various approaches were tested, providing different results, for instance, using artificially generated data, using datasets from *OpenML platform*, or different benchmark methods.

The proposed heuristic improves the prediction performance in four followed criteria by about 5 %. This one is implemented to the *H2O AutoML*. The thesis also examines unsuccessful attempts to provide a solid baseline for future improvements.

Keywords H2O AutoML, automated distribution selection, automated hyperparameter selection, automated machine learning, data generation

Abstrakt

Tato práce se zabývá vytvořením heuristiky pro automatický výběr hyperparametru reprezentujícího statistickou distribuci vysvětlované proměnné pro *H2O AutoML framework*. Bylo vyzkoušeno několik přístupů s různými výsledky, jako například, použití uměle vygenerovaných dat, dat získaných z platformy *OpenML* nebo různé metody provádění benchmarků.

Podařilo se vytvořit heuristiku, která ve čtyřech sledovaných metrikách zlepšuje predikční schopnost v průměru o zhruba 5 % a následně ji implementovat do *H2O AutoML*. Kromě úspěšných přístupů práce také zkoumá neúspěšné přístupy, které mohou pomoci pro budoucí zlepšování.

Klíčová slova H2O AutoML, automatický výběr distribuce, automatický výběr hyperparametrů, automatické strojové učení, generování dat

Glossary

AUC
Area Under Curve. 11

AutoML
Automated Machine Learning. 9–11, 13, 16, 17

CD diagram
Critical difference diagram. 11

DRF
Distributed Random Forest. 13–15

ERD
Mean residual deviance. 4, 27, 30–32, 36, 37

GBM
Gradient Boosting Machine. 14, 15, 24, 31

GLM
Generilized Linear Model. 14–16, 19–21, 24

IQR
Interquartile range. 24

MAE
Mean Absolute Error. 3, 15, 16, 27, 30–33, 37

MSE
Mean Squared Error. 3, 4, 15, 16

PDF
Probability density function. 4–7

PMF
Probability mass function. 4–6

RMSE
Root Mean Squared Error. 4, 11, 27, 30–32, 37

RMSLE
Root Mean Squared Logarithmic Error. 4, 17, 18, 27, 31, 32, 36, 37

XGBoost
Extreme Gradient Boosting. 14–17, 24

Introduction

Machine learning has become a crucial computer science component in recent years. It is widely used in all areas, from business to research. Almost every company has some application where *machine learning* is used to help achieve better results. With the increased usage, there is a natural effort to make things easier.

Machine learning tasks usually share some common steps – that are being done in almost every task – for instance data preprocessing, model selection, model evaluation... Those tasks also take a significant amount of time, and at the project's end, some may prove futile.

What if those time-consuming tasks could be automated, so the first result that would give more data insights would be obtained almost immediately? Here comes the *automated machine learning* that removes all the need to handle those tasks.

One of the available *automated machine learning* frameworks is *H2O AutoML*. This framework uses various models and gradually builds the best one. One thing in common is that the learning strategy differs based on the selected *data distribution*.

It is up to the user which distribution will be chosen. But *automated machine learning* should be done with the minimum effort – is there a way how would *H2O AutoML* framework pick the most suitable distribution by itself?

This thesis aims to create and prove a *heuristic* that would decide which distribution should be used, based on the given data – automize the selection of one of the hyperparameters. The heuristic should be more powerful than the default behavior (*Gaussian* distribution for all regression tasks). That could improve the overall results and outcomes for users who lack knowledge of statistical distributions or those who do not know the most suitable distribution for their data.

The heuristic itself is proposed *only for the regression tasks* since the distribution selection for *classification tasks* is already automated.[1]

I chose this topic for my interest in machine learning and statistics. Since most of the *machine learning* models have parameters that influence the results, it would be interesting to find a relation between input data and the best hyperparameter value. And a *distribution* parameter could provide this relationship since it is directly connected with the input data.

The thesis consists of the following chapters:

Statistical distributions and metrics This chapter describes basic and well-known concepts used later in the thesis. It is organized into two sections – in the first one, evaluation metrics are described. The ones that were used to create and evaluate the heuristic itself. In the second part, statistical distributions used in this thesis are briefly described, accompanied by the summary table for later reference.

[1] *H2O AutoML* supports only two classification distributions - one for the *binary classification*, other for the *multinomial classfication* – so the automated selection is simple.

Automated machine learning This chapter describes the motivation behind automated machine learning, its advantages, and disadvantages, followed by a concise description of the selected existing frameworks. Next is a brief description of the *AutoML benchmarks*, and lastly, platform *OpenML* and basic concepts of *meta-learning* are described to lay the theoretical foundation for the practical part.

H2O AutoML This chapter describes the *H2O AutoML framework*. Firstly, a quick overview of the whole framework is provided, followed by an overview of the available models within. Next, the *distribution hyperparameter*, the core of this thesis, is described – why it is essential for the prediction and how it influences model behavior. And at the end of the chapter, the whole automated machine learning process (as it is in *H2O AutoML*) is described together with the practical example of usage.

Data generation This chapter describes the generation of artificial data, how it is generated, which properties are used to create noise between the predictors and predicted variables, and other additions to make the artificial data more diverse.

Heuristic creation This chapter describes the creation of the heuristic. Firstly, requirements for the heuristic are described, followed by the description of the chosen *meta-features* from which the heuristic is built. The creation can be described by three main attempts; the last one is the only success. At the end of the chapter is described how the IF...ELSE rules for the heuristic are created.

Implementation to the H2O AutoML This chapter describes the prototype implementation of the heuristic to *H2O AutoML* using the Python interface of the framework. This is followed by the functionality test that was also run in the chapter *H2O AutoML* – but without heuristic.

Chapter 1

Statistical distributions and metrics

This chapter describes basic and well-known concepts used later in the thesis. It is organized into two sections – in the first one (1.1), evaluation metrics are described. The ones that were used to create and evaluate the heuristic itself. In the second part (1.2), statistical distributions used in this thesis are briefly described, accompanied by the summary table for later reference.

1.1 Evaluation metrics

Evaluation metrics tell how good is the performance of the model. Each metric covers a different purpose; the choice is on the user, which criterion is more important. At the heuristic creation (chapter 5), a big part of needs is covered – thanks to the metrics listed below.

The metrics and equations are taken from H2O's documentation [1]. In the formulas, n is the number of observations, y is the vector *true values*, and $\hat{y}$ is the vector of *predicted values*.

1.1.1 Mean Absolute Error (MAE)

Mean Absolute Error (MAE) is a metric whose units are the same as in the predicted target. Since the error is not squared, it is quite robust to the *outliers*.

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{1.1}$$

1.1.2 Mean Squared Error (MSE)

Mean Squared Error (MSE) has not the same units as the target variable – the values are squared. Since the difference between the predicted and true values is powered, it is more affected by *outliers*.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{1.2}$$

1.1.3 Root Mean Squared Error (RMSE)

Root Mean Squared Error (RMSE) is extended MSE. It is also more affected by *outliers* but has the same units as the target variable.

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{1.3}$$

1.1.4 Root Mean Squared Logarithmic Error (RMSLE)

Root Mean Squared Logarithmic Error (RMSLE) extends the RMSE formula with the *logarithmic scale*. This helps to balance the contribution of small and large errors.

$$\text{RMSLE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\log(y_i + 1) - \log(\hat{y}_i + 1))^2} \tag{1.4}$$

In the formula, 1 is added to avoid the logarithm of zero. If the values are negative, *H2O AutoML* produces *null value*.

1.1.5 Mean Residual Deviance

Mean residual deviance (ERD) is the only used metric whose exact form depends on the target's distribution. This is because the deviance is based on the *likelihood function*, which differs in each distribution. In general, in all its forms in *H2O AutoML*, it measures the goodness of fit. This metric is also known as *deviance* in other frameworks.

The base idea is to calculate the difference between the *log-likelihood* of the *saturated model* (number of parameters equals the number of observations, so the model is flawless for the provided data) and *created model*.

ERD is defined as:

$$ERD = -2\log\frac{L_{\text{saturated}}}{L_{\text{created}}} \tag{1.5}$$

Where L is the *log-likelihood* of the model.

1.2 Statistical distributions

This thesis aims to create a heuristic to tell the best value for the hyperparameter "distribution" in *H2O AutoML*. This hyperparameter represents the statistical distribution of the explained variable. It is more elaborated in the section 3.1.2.

For the *regression tasks*, *H2O AutoML* uses *continuous* distributions as well as *discrete* distributions. *Continuous* distributions can be described by Probability density function (PDF) – the area beneath the curve represents the probability of observation in the given range. Analogically, there are Probability mass function (PMF) for the *discrete* distributions. The functionality is similar but in a discrete space.

The following sections describe the distributions available in *H2O AutoML* or the distributions used for the heuristic creation. Please note that the distribution parameter names (like Poisson's λ) use current conventions, not necessarily the original names. On the right side are examples of the corresponding PMF or PDF.

1.2.1 Gaussian

Gaussian distribution [2] (also known as *Normal distribution*) is the most common distribution that can represent many phenomena and is also the default distribution in *H2O AutoML* (for regression tasks) if other is not specified. It is a *continuous* distribution and has two parameters, *mean* ($\mu \in \mathbb{R}$) and *variance* ($\sigma^2 \in \mathbb{R}^+$). PDF is defined as:

$$f(x; \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(\frac{-(x-\mu)^2}{2\sigma^2}\right) \tag{1.6}$$

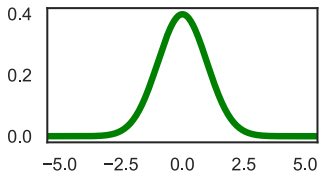


Figure 1.1 PDF of the *Gaussian* distribution with $\mu = 0$ and $\sigma^2 = 1$.

1.2.2 Poisson

A *random variable* X from the *Poisson distribution* [3], denoted as $X \sim Poisson$, is a *discrete random variable* supporting only positive numbers and zero. It can be distinguished, for example, as the probability of the number of occurrences in the given interval. Although it is a *discrete random variable H2O AutoML* treats the distribution as *continuous* and can even be used on the non-discrete data. The hallmark of this distribution is that *mean* is equal to *variance*.
It has one parameter $\lambda \in \mathbb{R}^+$ representing an *expected value* and a *variance* of X. The PMF is defined as:

$$f(X = k; \lambda) = \frac{\lambda^k}{k!} \cdot e^{-\lambda}, \quad k \in \mathbb{N}_0 \tag{1.7}$$

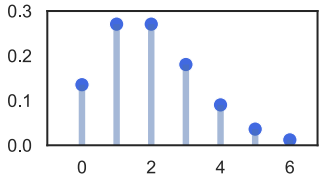


Figure 1.2 PMF of the *Poisson* distribution with $\lambda = 2$.

1.2.3 Gamma

Gamma distribution [4] is a two-parameter *continuous distribution* supporting real positive numbers ($\mathbb{R}^+$). If the random variable X is from the *Gamma distribution*, it is denoted as $X \sim Gamma$.
Distribution has two possibilities for how it can be parameterized. There is a *bijective function* between them. This thesis uses only one of them – parameters *shape* ($\alpha \in \mathbb{R}^+$) and *scale* ($\beta \in \mathbb{R}^+$). *Shape* represents number of modeled events while *scale* represents the mean time between them. The PDF is defined as:

$$f(x; \alpha, \beta) = \frac{\beta^\alpha}{\Gamma(\alpha)} x^{\alpha-1} e^{-\beta x}, \quad x > 0 \tag{1.8}$$

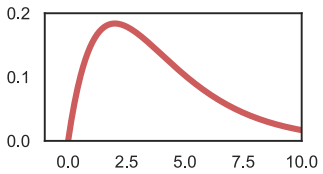


Figure 1.3 PMF of the *gamma* distribution with $\alpha = 2$ and $\beta = 2$.

1.2.4 Negative binomial

Negative binomial distribution [5] is similar to *Poisson distribution*. It is also *discrete* – but excludes the restriction that the variance equals the mean. In the following text, the random variable X from the *Negative binomial distribution* is denoted as $X \sim \text{NB}$.
It has two parameters, $r \in \mathbb{R}^+$ represents the number of successes to be achieved, and $p \in [0, 1]$ tells the probability of each success. k in the formula represents observations.
PMF is defined as:

$$f(X = k; r, p) = \binom{k + r - 1}{k} p^r (1 - p)^k \tag{1.9}$$

Where k is the number of failures before r-th success.

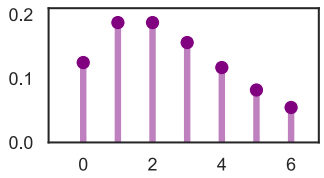


Figure 1.4 PMF of the *negative binomial* distribution with $k = 10$, $r = 3$ and $p = 1/2$.

1.2.5 Laplace

Laplace distribution [6] ($X \sim \text{Laplace}$) is similar to the *Gaussian distribution* with the difference of having heavier tails, so it is more robust to the outliers. The domain is $\mathbb{R}$. As *Gaussian*, it takes two parameters, *location* ($\mu \in \mathbb{R}$) and *scale* ($b \in \mathbb{R}^+$). PDF is defined as:

$$f(x; \mu, b) = \frac{1}{2b} \exp\left(-\frac{|x - \mu|}{b}\right) \tag{1.10}$$

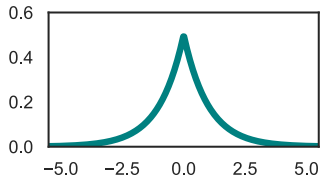


Figure 1.5 PDF of the *Laplace* distribution with $\mu = 0$ and $b = 1$.

1.2.6 Tweedie

Tweedie ($X \sim \text{Tweedie}$) [7] is a *family of distributions* that varies based on the value of the *variance power* parameter (p). Regarding the H2O's implementation [8], *power* parameter is defined only within the interval ($1 < p < 2$). Another two parameters are *location* ($\mu \in \mathbb{R}^+$) and *scale* ($\phi \in \mathbb{R}^+$). PDF is defined as:

$$f(x; \mu, \phi) = a_p(x, \phi) \exp\left(\frac{-d(x, \mu)}{2\phi}\right) \tag{1.11}$$

In the formula, the function $d(x, \mu)$ is a *unit deviance*, and function a_p is the *power function* that has different forms based on the *power* parameter.
Some values of the *variance power* parameter result in other distributions:

- $p = 0$ *Gaussian distribution*
- $p = 1$ *Poisson distribution*
- $p = 2$ *Gamma distribution*
- $p = 3$ *Inverse Gaussian distribution*

As was stated, *H2O AutoML* offers *variance power* parameter only in the range $1 < p < 2$.

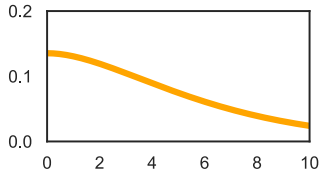


Figure 1.6 PDF of the *Tweedie* distribution with *tweedie_power* $= 1.5$, $\mu = 4$ and $\phi = 2$.

1.2.7 Cauchy

Cauchy distribution ($X \sim \text{Cauchy}$) [9] is a *continous* distribution with one significant property – it has no mean nor variance. This distribution is unavailable in the *H2O AutoML*, but this thesis uses it to create artificial data. It has two parameters, *location* ($\alpha \in \mathbb{R}$) and *scale* ($\theta \in \mathbb{R}^+$). PDF is defined as:

$$f(x; \alpha, \theta) = \frac{1}{\pi\theta} \cdot \left[1 + \left(\frac{x - \alpha}{\theta}\right)^2\right]^{-1} \tag{1.12}$$

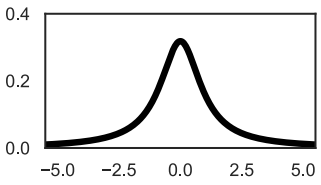


Figure 1.7 PDF of the *Cauchy* distribution with $\mu = 0$ and $\sigma^2 = 1$.

1.2.8 Distributions summary

For better readability, table 1.1 summarizes all the aforementioned distributions and their parameters.

Table 1.1 Distributions overview

Distribution	Domain	Type	Parameters	
Gaussian	$\mathbb{R}$	Continuous	$\mu \in \mathbb{R}$ (location),	$\sigma^2 \in \mathbb{R}^+$ (scale)
Poisson	$\mathbb{N}$	Discrete	$\lambda \in \mathbb{R}^+$ (rate)	
Gamma	$\mathbb{R}^+$	Continuous	$\alpha \in \mathbb{R}^+$ (shape),	$\beta \in \mathbb{R}^+$ (rate)
Neg. bin.	$\mathbb{N}$	Discrete	$r \in \mathbb{N}_0$ (rate),	$p \in [0, 1]$ (probability)
Laplace	$\mathbb{R}$	Continuous	$\mu \in \mathbb{R}$ (location),	$b \in \mathbb{R}^+$
Tweedie	$\mathbb{R}^+$	Continuous	$p \in (1, 2)$ (power)	$\mu \in \mathbb{R}^+$ (location), $\phi \in \mathbb{R}^+$ (scale)
Cauchy	$\mathbb{R}$	Continuous	$\alpha \in \mathbb{R}$ (location),	$\theta \in \mathbb{R}^+$ (scale)

Chapter 2

Automated machine learning

This chapter describes the motivation behind Automated Machine Learning (AutoML), its advantages and disadvantages, followed by a concise description of the selected existing frameworks. Next is a brief description of the AutoML benchmarks, and lastly, platform OpenML and basic concepts of meta-learning are described to lay the theoretical foundation for the practical part.

2.1 Motivation

Machine learning is nowadays used in many various tasks. Images, texts, audio, video, and almost all kinds of data can be processed to obtain more information and insights into the given problem domain. One can detect anomalies in the data, predict values, predict who or what is in the picture, and many other applications. Some tasks are more complex in preprocessing and computability, while others are simpler and often very similar in a workflow.

There are various attempts to describe the workflow, for example, by introducing the standards for the data mining process. One of the introduced standards is CRISP-DM [10]. This process consists of defined steps that should be followed in the whole data mining process. Those steps are:

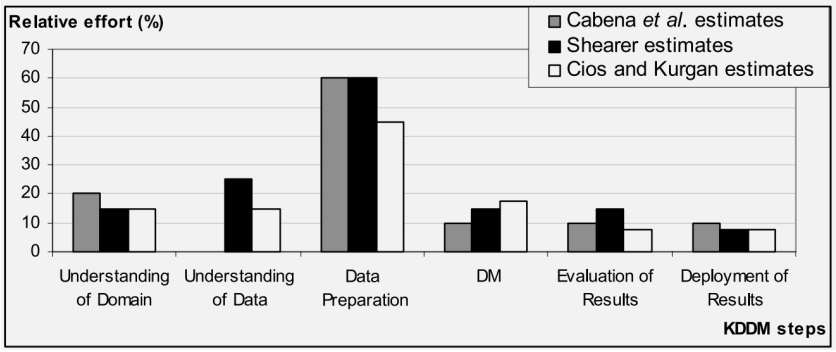
1. Business Understanding
2. Data Understanding
3. Data Preparation
4. Modelling
5. Evaluation
6. Deployment

Research was done to measure the time spent on each task in the CRISP-DM process [11]. The results can be seen in figure 2.1.

Figure 2.1 shows that most of the time is spent on data preparation. When getting new data, it may take a while to see the first results and insights regarding the preprocessing and proper model training. An automated solution would be handy because first results and insights would be obtainable almost as soon as one gets the data, without needing more profound expertise and domain knowledge which are almost inevitable requirements at data preprocessing.

Another case may occur when a researcher (in any field of study) wants to know if there is some relation or other insights. Researchers in other areas of study (for instance, chemistry) may have

Figure 2.1 Relative time spent on parts of the *KDDM process*. The results are from the three different surveys. [11]



some programming and machine learning knowledge. However, they may still be missing crucial information on how to build a proper machine learning model. In the simplest case, AutoML requires only input data almost in any condition. Those frameworks may make machine learning more accessible for other study fields, removing the barrier of insufficient knowledge.

This approach can be full-fledged for the more straightforward datasets while saving a non-negligible amount of time and resources. Generally, it is a good baseline where to start. The first decent results are obtained almost immediately without manual data preprocessing and model tuning.

Since everything is automated, this does not give the best results, but it can provide a solid baseline without further work. Part of the *H2O AutoML framework* is the explainability module, which can significantly help with data and model understanding and the relations between features.

2.2 Available AutoML frameworks

AutoML has increased attention in recent years, that naturally leads to an increased number of frameworks. Since the number of available frameworks is steadily growing, only a few will be described, based on the AutoML Benchmark [12] that is being regularly done – the most recent one in 2022. More about AutoML benchmarks in section 2.3.

H2O AutoML is easily distributable over multiple computation devices and it is *big data* friendly. It automatically handles *data preprocessing* (*one-hot encoding, normalization,...*), *model selection*, *hyperparameter optimization*, and *model evaluation*. The integrated explainability module offers easily-obtainable access to the data and model insights. More info is in chapter 3. [8]

Auto-Sklearn is built on the widely used machine learning framework *scikit-learn* [13]. It does similar steps as *H2O AutoML*. It uses *Bayesian optimization* and *meta-learning*. Unlike *H2O AutoML*, it is not built for *distributed systems* and *big data*. [14]

FLAML is a relatively new framework that combines *iterative boosting algorithms* from various libraries with a small addition from the *scikit-learn* [13]. It emphasizes computation efficiency. [15]

AutoGluon can be used for tabular data but can also be used for images and texts (apart from the aforementioned frameworks). Unlike other models, it does not perform *hyperparameter tuning* nor *search for the best model*. It emphasizes more on *model selection* (that is being done by *stacking*) rather than data preprocessing. [16]

2.3 AutoML frameworks benchmark

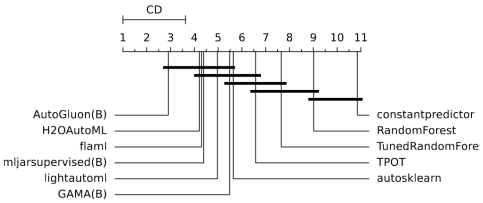
With the increasing number of AutoML systems, there is a natural need to compare them. This section describes one of the AutoML benchmarks [12] (done in 2022), comparing the selected top-performing ones. Conditions were the same for all participating frameworks. Benchmarks were done on *classification* (*binary and multi*), and *regression tasks* on the carefully chosen datasets from the *OpenML platfrom* [17] (described in section 2.4). Each model got the same time, and then the performance was evaluated.

The Area Under Curve (AUC) for *binary classification*, *log-loss* for *multinomial classification*, and RMSE for *regression tasks*.

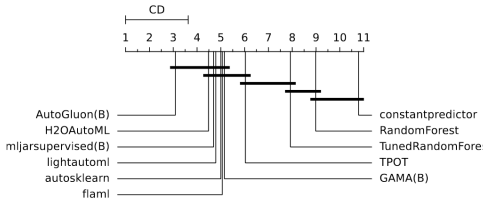
To show the results, Critical difference diagram (CD diagram) [18] are used – the visualization method based on rank and statistical tests.

The results for the *multi-class classification* and *regression* can be found in figure 2.2.

Figure 2.2 Results of AutoML frameworks benchmark [12] using CD diagram [18]. The x-axis represents the rank – the lower the rank, the better. Thick black horizontal lines connect the frameworks whose differences in results are not statistically significant between themselves. The uppermost line with the label "CD" determines the ranks whose results are not statistically significant from the correct results.



(e) Regression, 1 hour



(f) Regression, 4 hours

It can be seen that the AutoGluon framework leads in both categories. As a matter of fact, it leads in all six categories (the other categories are supervised classifications, see [12]). In *regression tasks*, *H2O AutoML* is in the second place. And the heuristic that is being proposed in this thesis aims to improve the regression score.

2.4 OpenML

As mentioned in section 2.3, data for the AutoML benchmarks come from the *OpenML platform* [17] that also this thesis uses. It is an open platform for sharing any machine learning related materials. Not only datasets but the whole pipelines, benchmarks, and experiments can be found here. Since it is an open platform, anyone can contribute and use the datasets for free.

To use datasets from *OpenML*, one can utilize its package, available in *Python* [19]. The *API credentials* are needed to start using it.

Available items can be filtered by their type. For example, datasets can be categorized by the type of task that uses them. There are *eight* task categories, basic types like *supervised regression, supervised classification, clustering*, but there are also more specific tasks, like *survival analysis* or *learning curve construction*...

Datasets are accompanied by its *meta-data*, through which they can also be filtered, from the common properties like *dataset shape*, *number of missing values*, or *dataset name* to more specific regarding the task type, like the *number of classes in the explained variable* (classification).

Since it is an open platform where anyone can contribute, not all datasets share the same qualities. Currently (2023), there are 2600 available datasets (through the *python package*) for *supervised regression tasks*. When creating a heuristic, it sometimes happened that explained variable in the dataset, labeled as *regression task*, had only two values, coincidentally $\{0, 1\}$,

which makes it a *binary classification*. This problem with others, like *insufficient size*, *missing metadata*, or *missing target variable*, can be easily filtered out during fetching.

The last problem encountered with the *OpenML platform* was *data source bias*. Multiple datasets have the same source (for example, standardized chemical experiments in one laboratory – same structure, but different values). Those datasets usually share part of the name and are easily identifiable. One of those groups currently has 1102 regression datasets. Most of those datasets in the group share the features. If those datasets were used in the *meta learning*, their shared properties would heavily influence *meta-model*. Its generalization ability would be negatively affected, resulting in overfitting. For this reason, they are filtered out.

2.5 Meta-learning

In [20], *meta-learning* is defined as: "*the study of principled methods that exploit* metaknowledge *to obtain efficient models and solutions by adapting machine learning processes.*" In this definition, *metaknowledge* is any knowledge gained from the prior tasks useful for future modeling.

Meta-learning can be interpreted as deriving knowledge from the prior tasks to improve the effectiveness and results of future ones. It can be seen as a sort of learning. However, instead of learning, e.g., how to recognize cats in a picture with the neural network, one *learns how to learn* how to improve the recognition based on prior experience and input data characteristics.

This thesis deals with the *meta-learning* from the input data characteristics point of view. The heuristic itself (described in chapter 5) is built on the information obtained from the various datasets and their properties, not considering the model characteristics part of this study field.

Can the optimal hyperparameter value be derived only from the input data? Moreover, if so – why would one bother with hyperparameter tuning, which takes a lot of computation and time?

It may not work for most hyperparameters since most are not directly linked with the input data. This thesis aims to determine the *distribution* hyperparameter that tells which data distribution corresponds the most with the explained variable. So the value represents the data characteristics, which naturally leads to the assumption that the data characteristic could determine the hyperparameter value.

Chapter 3

H2O AutoML

This chapter describes the H2O AutoML framework. Firstly, a quick overview of the whole framework is provided, followed by an overview of the available models within. Next, the distribution hyperparameter, the core of this thesis, is described – why it is essential for the prediction and how it influences model behavior. And at the end of the chapter, the whole automated machine learning process (as it is in H2O AutoML) is described together with the practical example of usage.

3.1 H2O AutoML

One of the available AutoML frameworks and core of this thesis is *H2O AutoML* [8]. It is an open-source framework. It handles *big data* and can be easily distributed (unlike other AutoML systems). There are APIs in *R, Python, Scala* and *Java* – but the functionality is the same for all. The core of the application is written in *Java*.

Currently (2023), *regression*, *classification*, and the *multi-class classification* supervised tasks are supported. Additional tools, like the *explainability module*, are available for all tasks.

H2O AutoML is built as a distributed system, so the application requires a running backend that provides API (in the aforementioned programming languages). This backend can be distributed over multiple server instances and run locally, for example, through the *Docker containers*. In the practical part of this thesis, one *Docker container* is used.

3.1.1 H2O AutoML models

As of April 2023, *H2O AutoML* uses five selected models from *H2O Open Source* (another *H2O's framework*), together with *Stacked Ensemble models* made from them. Selected models are diverse, each representing a different direction of machine learning algorithms. Since each model used in *H2O AutoML* has a different approach, the resulting *Stacked Ensemble model* usually has good results and does not overfit.

3.1.1.1 Distributed Random Forest

H2O's Distributed Random Forest (DRF) [8] is the implementation of the classical *Random Forest* [21] concerning the possibility of distributing the training over multiple devices, making it faster and able to handle larger datasets.

3.1.1.2 Generalized Linear Model

Generilized Linear Model (GLM) [22] is the generalized version of the classical *linear model*, such as *Linear Regression*.

In the *Linear Regression*, the expected value (denoted as $\mathbb{E}$) of the random variable is predicted while the GLM model predicts the *transformation of the mean*. The transformation function is called the *link function*, and the model uses its inversion. An explained variable must be from the *Exponential family* of distributions. The GLM equation is defined as:

$$\hat{y}_t = \mathbb{E}[y_t \mid x_t, \beta] = g^{-1}(\beta^T x_t) \tag{3.1}$$

GLM can be used with multiple distributions that change the behavior and aim of the model. For instance, GLM with the identity *link function* and *Gaussian distribution* can be used as a classical *linear (Gaussian) regression*.

H2O's GLM [23] has multiple available distributions, supporting the *classification* and *regression tasks*. Supported regression distributions (that are also supported in *H2O AutoML*) are *Gaussian, Poisson, gamma, negative binomial* and *Tweedie*. For *classification tasks*, it is *binomial* and *multinomial*. Each distribution is supposed to be for a different task and has a different training equation.

3.1.1.3 GBM

Gradient Boosting Machine (GBM) [24] is one of the *ensemble learning* methods. It iteratively builds *decision trees*, using residuals to improve. H2O's implementation [8] adds the possibility to parallelize the training similarly to H2O's DRF.

3.1.1.4 XGBoost

Extreme Gradient Boosting (XGBoost) available in the *H2O AutoML* [8] is not H2O's implementation. It is the original XGBoost algorithm [25], wrapped in the H2O's framework. It is the only model in *H2O AutoML* with no custom implementation. The algorithm is quite similar to the GBM, also implemented in the *H2O AutoML* framework, but focuses more on regularization.

3.1.1.5 DeepLearning

The last algorithm included in H2O AutoML represents *artificial neural networks*. H2O's DeepLearning [26] is a multi-layer network trained with stochastic descent. Various activation functions, regularization options, and many other possible hyperparameters exist.

3.1.1.6 Stacked Ensemble

H2O's Stacked Ensemble [8] itself is not exactly another unique model, and it can not be used standalone. The *stacking approach* used there is based on *Super learner algorithm* [27]. It takes the predictions from the base models and creates a new model (also called *meta-model*). The advantage of this approach is increased robustness, reduced possibility of *overfitting*, and the ability to possibly capture patterns and relationships that individual models did not capture.

3.1.2 Distribution hyperparameter

Each of the models above (except for *Stacked ensemble*) has a *hyperparameter* named *distribution* that should determine the distribution of the explained variable and the model usage. In all models (except for *Stacked ensemble*), it sets the *loss function*. It determines the whole training equation for the GLM.

If a user does not set the hyperparameter, it uses *Gaussian* distribution for all the continuous data; for classification tasks, there is a *binomial* or *multinomial* (depending on the number of the target classes).

Not all models support all distributions. If the chosen distribution is not supported on the current model, the distribution is set as *Gaussian*. Available *regression distributions* for each model can be seen in table 3.1. All the distributions were described in section 1.2.

Table 3.1 Available regression distributions in H2O AutoML [8]

	DeepLearning	DRF	GBM	GLM	XGBoost
Gaussian	X	X	X	X	X
Poisson	X		X	X	X
Gamma	X		X	X	X
Negative binomial				X	
Laplace	X		X		
Huber	X		X		
Tweedie	X		X	X	X

3.1.3 Distribution & Loss function

For all the models, the *distribution* parameter influences the *loss function*. Below is a brief overview of the different loss functions for different distributions – they are similar for all the models, but the differences are mainly in regularization. The regularization helps with the predictive performance of the models, and thus it is very practical to keep it turned on; however, it can make a task more difficult to show an improvement.

Those *loss functions* are common for all the *H2O AutoML* base models (*Deep learning*, GBM, GLM, XGBoost) except for DRF [1].

Note that for simplicity, the loss functions below are cropped of the variable w representing *weight* that a user can set if one wants to assign different weights for the observations and regularization.

- Gaussian loss function equals to MSE already mentioned in 1.1.2.
- Poisson loss function

$$L(y,\hat{y}) = 2 \cdot \left(y \cdot \log \hat{y} - e^{\log \hat{y}}\right) \tag{3.2}$$

- Gamma loss function

$$L(y,\hat{y}) = 2 \cdot \left(y \cdot e^{\log \hat{y}} + \log \hat{y}\right) \tag{3.3}$$

- Negative binomial loss function. Parameter θ is automatically learned during the training.

$$L(y,\hat{y}) = e^{-\hat{y}} \cdot \frac{\Gamma(y+\theta^{-1})}{\Gamma(\theta^{-1}) \cdot \Gamma(y+1)} \cdot \left(\frac{\theta^{-1}}{\theta^{-1}+\hat{y}}\right)^{\theta^{-1}} \cdot \left(\frac{\hat{y}}{\theta^{-1}+\hat{y}}\right)^{y} \tag{3.4}$$

- Laplace loss fuction equals to MAE already mentioned in 1.1.1.
- Tweedie loss function, p is the *Tweedie power* parameter.

$$L(y,\hat{y}) = 2 \cdot y^{(2-p)/((1-p)\cdot(2-p))} - y \cdot e^{\hat{y}\cdot(1-p)}/(1-p) + e^{\hat{y}\cdot(2-p)}/(2-p) + e^{\hat{y}\cdot(2-p)}/(2-p) \tag{3.5}$$

All the info was taken from [23] and from online documentation [28].

[1] *Random forests* are trained differently from the other models that treat the loss function another way. For this reason, *regression tasks* for DRF supports only one loss function marked as *Gaussian* in the context of *distribution* hyperparameter.

3.1.3.1 Huber

It can be misleading that the hyperparameter is called *distribution* when *Huber distribution* is not defined. In *H2O AutoML*, if the distribution is set as *Huber*, the *Huber loss function* [29] is applied. It combines MSE and MAE. A hyperparameter δ is set. An error is linear if the distance between the prediction and the true value is greater than δ. If it is lower, the error is quadratic. This reduces the influence of outliers on the model.

$$L_\delta(y, f(x)) = \begin{cases} \frac{1}{2}(y - f(x))^2 & \text{if } |y - f(x)| \leq \delta \\ \delta|y - f(x)| - \frac{1}{2}\delta^2 & \text{otherwise} \end{cases} \tag{3.6}$$

3.1.4 Distribution & GLM

As was mentioned, a special case regarding distribution hyperparameter is the GLM model. The artificial data generation[2] is based on the GLM equations 3.1. Each distribution has a different *link function*, those functions may vary in different implementations. H2O's implementation is shown in table 3.2 – but only for the distributions relevant to the heuristic creation and for the distributions available in H2O's GLM (see table 3.1).[3]

Table 3.2 Link functions

Distribution	Link function	Inverse of link
Gaussian	identity	identity
Poisson	log	exp
Gamma	log	exp
Negative binomial	log	exp
Tweedie	log	exp

3.2 H2O AutoML process

Assume that *H2O AutoML* is already running, and the dataset is uploaded to the backend. The whole process begins with calling the method `train`, followed by the following steps:

1. **Data preprocessing**. *Machine learning* algorithms have specific requirements for the input data and *H2O AutoML* is no exception. *Missing values imputation*, *one-hot encoding*, *target encoding* and *normalization* (when required) is being done in the first phase of the AutoML process. Those are current steps, but others are in the framework's *roadmap*.

2. **Base models**. To ensure a solid starting point, the training of the first few models is predefined. That means the training has given order, and the models have set hyperparameters. The hyperparameter values are chosen regarding the various benchmark results and strong domain knowledge of H2O's data scientists. The goal is to obtain a solid baseline through the models that are generally behaving well, with most kinds of the data (like XGBoost). After each of those groups, *Stacked Ensembles* are also trained to obtain possibly better results.

3. **Random search**. After the initial baseline is set, the training is more randomized. There is a *random search* (that is, by default *cross-validated*) within the set of chosen hyperparameters and models.

[2] Described later in the chapter 4

[3] For *Tweedie* distribution, *link function* differs based on *tweedie variance power* parameter. In *H2O AutoML*, this parameter is confined to $1 < p < 2$, so the link function is the same.

4. **Stacked ensembles**. *Random search* provides a diverse set of models, and from that, *Stacked Ensemble models* are trained. Thanks to the diversity of the models, the results are usually better than any of the before-trained models (but not always). They are also trained after the *base models*.

5. **Leaderboard**. It is up to the user how extensive *random search* (parts 3 & 4) will be. Users can confine the framework either by the maximum number of trained models or the training time. Also, *early-stopping* mechanisms are available. A *leaderboard* is provided when the training ends, containing various metrics and training info. An example may be seen in section 3.3.

All the info for this section was taken from [8].

3.3 H2O AutoML example

An example of the *H2O AutoML* framework is shown in the following lines. The demonstration is done on the *Medical Cost Personal dataset* [30], where the *individual medical costs* are predicted based on basic personal data (6 features). This whole example is available in the *jupyter notebook* in the attachment.

Since the *H2O AutoML* framework is suitable for *big data*, it is expected to be on some remote server that is computationally superior to standard computers. But it can still be run locally via *Docker container* that will provide API for the framework.

Data is uploaded to the cluster; the response column is set. Then, *H2O AutoML* is initialized. Note that distribution is set as *auto*, meaning that the default value – *Gaussian distribution* will be used. Next, 13 base models without further hyperparameter optimization are trained. And that is all. There are various ways how to set how many models will be trained. One can set the total models count or confine the training with a time limit. The time limit approach was used in the AutoML benchmark described in 2.3. The code described in this paragraph is available in the code 3.1.

```
train = h2o.upload_file("insurance.csv")

X = train.columns
y = "charges"
X.remove(y)

# Run AutoML for 13 base models
aml = H2OAutoML(max_models=13, seed=13, distribution="auto")
aml.train(x=X, y=y, training_frame=train)
```

Code listing 3.1 Example of the H2O AutoML

Results are obtained when the training ends. The value of the RMSLE[4] is approximately 0.407, which is a decent baseline. Regarding the leaderboard (3.1) of all trained models, the best models were *Stacked ensembles* of others, closely followed by the XGBoost.

With the trained model, one can examine the possibilities of H2O's explainability module. Simply by calling `automl.explain(valid_set)`, the user will get[5] *leaderboard, residual analysis, learning curve plot, variable importance, and its heatmap, correlation of models, SHAP values and plots for each feature, partial dependence plots for each feature and model, ICE plots* and other possible outcomes, based on parameters.

All this can be obtained without data preprocessing, model selection, etc... This approach gave fast first results providing insights into the data and inner structure.

[4] On cross-validation data

[5] This example is for the regression tasks. In the case of classification, some items will be different.

Figure 3.1 Leaderboard of the best models, ordered by RMSLE). Ordering can be changed. Column *model id* determines the used model; the rest are metrics, described in 1.1.

	model_id	rmse	mse	mae	rmsle	mean_residual_deviance
0	StackedEnsemble_BestOfFamily_1_AutoML_22_20230...	4493.738406	2.019368e+07	2428.147171	0.407778	2.019368e+07
1	StackedEnsemble_AllModels_1_AutoML_22_20230508...	4499.423606	2.024481e+07	2435.660403	0.410949	2.024481e+07
2	XGBoost_grid_1_AutoML_22_20230508_80303_model_1	4514.519718	2.038089e+07	2398.695239	0.411722	2.038089e+07
3	GBM_4_AutoML_22_20230508_80303	4709.750661	2.218175e+07	2702.171413	0.470540	2.218175e+07
4	GBM_2_AutoML_22_20230508_80303	4750.766129	2.256978e+07	2802.340424	0.479540	2.256978e+07
5	DRF_1_AutoML_22_20230508_80303	4798.550557	2.302609e+07	2758.264493	0.438675	2.302609e+07

Chapter 4

Data generation

This chapter describes the generation of artificial data, how it is generated, which properties are used to create noise between the predictors and predicted variables, and other additions to make the artificial data more diverse.

4.1 Introduction

This thesis aims to create a heuristic to tell which available distributions one should use to train the *H2O AutoML* framework. This task can also be described as finding the best hyperparameter value based on the dataset's properties. The initial step was creating artificial datasets with predictors covering all the required distributions – to see whether it was possible to distinguish the most suitable distribution based on artificial data properties.

4.2 Generating artificial data

Creating artificial data is problematic regarding covering as much variability as possible. Data generation is done with the same approach as is used in the GLM (equation 3.1) – the equation through that data is generated has a form:

$$y_i = \theta_d(g^{-1}(\beta^T x_i)) \tag{4.1}$$

- Where y_i is the i-th term in the explained variable vector,
- θ_d is the function that returns a *random variety* from the specified distribution d,
- g^{-1} is the inversion of the *link function* (3.2),
- β is the vector of *coefficients*,
- x_i is the i-th row in the training set X.

The correctness of the generated data can be tested through the GLM. Since the equation is the same as GLM has, its coefficients can also be used for comparison/evaluation.

The following sections show how the formula is created.

4.2.1 Train set

The explained variable y is calculated using the train set X. For the creation, the simplest approach is used – all the features are numeric and *i.i.d.* – but still, multiple factors must be considered.

Dataset size It is expectable that larger datasets may perform differently than smaller datasets. Regarding the computational power of the benchmark device, there are defined two sets of numbers – rows and columns. The resulting shapes are derived from the *Cartesian product* between those two sets.

- Row counts: 50000, 500
- Feature counts: 1250, 500, 100, 2

Intercept Different intercepts for the explained variable increase the variability of the generated data. Two values – 0.15 and 8 – were chosen. To include datasets near zero as well as further values.

Feature values As was stated, only numeric values are acceptable. Features are *i.i.d.* and each feature X_i has the *uniform distribution*, $X_i \sim Uniform(0, 1)$.

Coefficient values In the equation, coefficients are represented with the vector β. To make the data more diverse, there are two types of them:

- Small coefficients, each row $c_i \sim Uniform(-1, 1)$
- Big coefficients, each row $c_i \sim Uniform(-50, 50)$

4.2.2 Explained variable

With the defined training set X, an explained variable y is calculated. Relation between the training set and the explained variable can be represented through the formula 4.1.

Given this relation, explained variable can be constructed, and exact parameters in β can be calculated through the GLM. But that is not enough; real-world data is messy, and there is some noise.

4.2.3 Noise

For data generation, there are two types of features:

Informative feature This feature is directly included in the resulting value of the explained variable. In the GLM equation, it is represented as x_i.

Non-informative feature This feature was not used during the data generation process; thus, it contains no useful information. The problem is that the model does not know it and needs to figure it out.

Noise is introduced via *non-informative* features; those features are randomly chosen from the rest. The proportion of *informative features* is set to the three options:

- 100% of the features are informative
- 75% of the features are informative
- 25% of the features are informative

The special case is for the datasets with only two features; in this case, the choice is reduced to 100% and 50%.

Regarding the GLM formula 3.1, an optimal model would have *non-informative features* coefficients β_i set to zero.

4.2.4 Modality

And last addition is *modality*. Data generated through the GLM formula now have one *mode*. Unfortunately, that is not a rule for the real-world data. For this reason, another mode is introduced. That is achieved with the categorical variable X_c with the *Bernoulli distribution*, $X_c \sim Be(9/20)$. In the construction of y, this variable is multiplied by the higher constant (25), so the explained variable y is divided into *two clusters*, each having its *mode*.

This is not applied to all the datasets.

4.2.5 Link functions

To follow the GLM formula, *inverse link function* must be applied. A sum of coefficients with the features labeled as *informative* is put into the *link function*. List of *link functions* is available in table 3.2.

4.2.6 Distributions

All the regression distributions available in *H2O AutoML* were used, accompanied by the *Cauchy distribution* to create artificial data with more outliers. Distributions are described in section 1.2.

4.2.7 Random variates

The last step is to generate random variate from the appropriate distribution. The Python package *Scipy* [31] is used for this. For each distribution, a function `rvs` takes the given input (and distribution parameters), treats it as a *mean* of the distribution, and returns a random number from it.

4.2.7.1 Random variates parameters

As was mentioned, some distributions have additional parameters. The overview may be seen in table 1.1. Parameters that are not related to the input data are required at *Negative binomial* (1.2.4) and *Tweedie* (1.2.6) distributions. Those parameters are taken from the predefined, *uniformly distributed* grid in both cases.

For *Negative binomial*, it is parameter $p \in [0, 1]$ representing the probability of success (see 1.2.4). For *Tweedie*, it is *Tweedie variance power* $p \in (1, 2)$ and *scale* $\phi \in \mathbb{R}^+$ (see 1.2.6).

4.2.8 Example

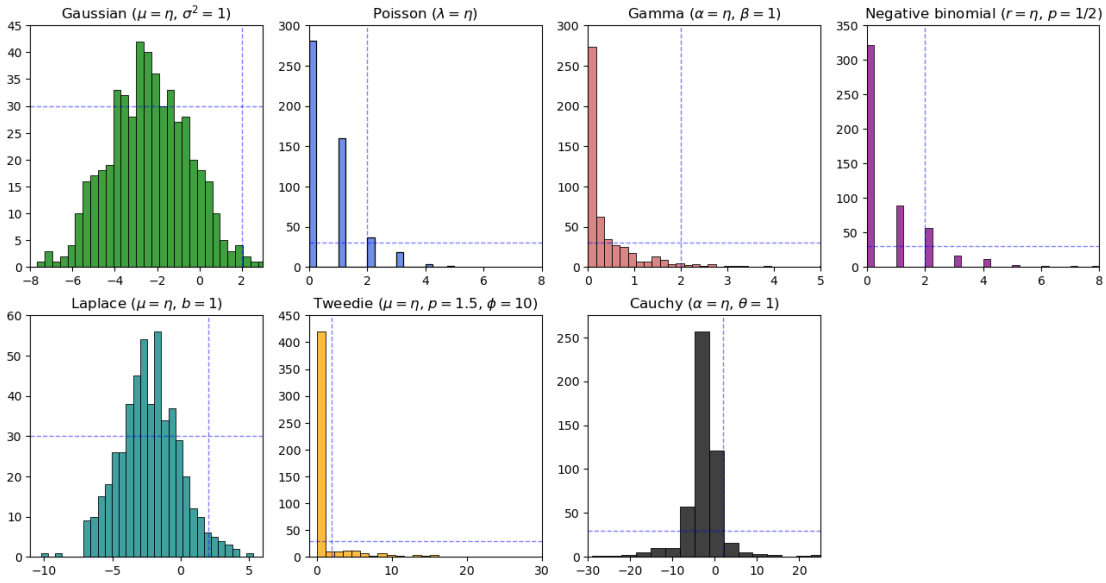
Since the theoretical description may be confusing, there is a *Jupyter notebook* in the attachment with the complete code. Also, in figure 4.1 is an example of the histograms from one randomly chosen generated explained variable.

4.3 Conclusion

Eight unique datasets came from the data generating described above. Each dataset has multiple explained variables – each generated from the same training data but with the different parameters, informative indices, etc. . .

There are 1980 unique target variables, 792 for the *Tweedie distribution*, 396 for the *Negative binomial distribution*, and 132 for the rest. The higher amount for *Tweedie* and *Negative binomial* is caused by other required parameters, explained in 4.2.7.1.

Figure 4.1 The figure shows an example of the explained variable histograms of one randomly chosen dataset. Seven generated distributions are generated from the same base (same coefficients, informative features, ...). The dotted line show where $x = 2$ and $y = 30$ to highlight the different proportions of each distribution.
Parameter η represents the generating parameter obtained during the generation process.



Chapter 5

Heuristic creation

This chapter describes the creation of the heuristic. Firstly, requirements for the heuristic are described, followed by the description of the chosen meta-features from which the heuristic is built. The creation can be described by three main attempts; the last one is the only success. At the end of the chapter is described how the IF... ELSE rules for the heuristic are created.

5.1 Introduction

The purpose of the heuristic was described in previous chapters. This chapter describes its creation.

The general requirements for the heuristic are:

Explainability One of the core requirements is the ability to clearly see (and explain) the decisions that are being made by the heuristic. Ideally, it should consist of IF... ELSE statements.

Speed Another requirement is the speed of the decision - it should be done almost instantly, without long computations. The computation time would increase in case of a *big data* and would slow down the whole framework by a significant amount of time.

Reliability The usage of the heuristic should improve the performance of *H2O AutoML*, so it will generally be more beneficial to use it instead of the current behavior (*Gaussian* for regression).

After the consultation with the thesis supervisor, it was decided that to fulfill all three requirements above, the heuristic itself will be created through the *decision tree* that will be trained using *meta-learning* (described in 2.5). This approach meets the *explainability* requirement because it is easy to derive the rules from the given tree in the form of IF... ELSE statements. This approach also partially fulfills the requirement of *speed* because getting the decision from the conditions is almost instant.

The problematic part about speed is getting the variables for the decision making, which is described in the section 5.2.

Three different heuristics were created with similar approaches – *decision tree* from *meta-learning*. This approach was based on performance, not considering distributions intentions and purpose.

And a reminder from section 1.2, this hyperparameter sets the *loss function* for all the models.

H1 – Artificial data The first heuristic was built purely on artificial data. The initial motivation was to find out if there was an improvement. Using artificial data has the advantage of

knowing the coefficients used for generating. Those coefficients can be obtained through the GLM model since the generating was done with its equation, so I could see how much they differed.

The results were poor, it did not generalize well on the real-world data, but it showed that creating *meta-features* from the training set is unnecessary and that creating it from the target variable is enough. More described in section 5.4.

H2 – Solo models The second heuristic was created on open datasets from *OpenML platform* using standalone models available in *H2O AutoML*. This was done with the expectation of higher generalization – but did not satisfy it. Also, the computational requirements were far smaller than in the case of using *AutoML*. For one dataset and one distribution, only four models were trained (GLM, GBM, XGBoost, and *Deep learning*)[1].

This approach had much better results than the first one, but on the standalone models – not on *H2O AutoML*. More described in section 5.5.

H3 – AutoML model The final heuristic with the good overall results, created through the *H2O AutoML* itself. More described in section 5.6.

Before describing the heuristic itself, used *meta-features* will be introduced.

5.2 Meta-features

All three heuristics were created through the *meta-learning*, based on the *meta-features* from the input dataset. The selection of those features was mainly made in the first heuristic – based on the *decision's tree* feature importance (via multiple approaches of importance calculation). But the features themselves will be described here.

The selection of the initial features was mainly based on the chapter *Dataset Characteristics (Meta-features)* from the book *Meta-learning* [32], and from the article, [33], which shows how to treat bivariate statistics. Some other *meta-features* were introduced and tested by me.

5.2.1 Chosen features

Most of the features selected for the final usage were chosen based on the *feature importance* (done with the *random feature elimination* algorithm and with the *feature importance* from the *decision tree*) and heuristic performance. The next considered criterion was the correlation between the features. Keeping highly correlated features (like mean and median) would increase the complexity of the rules while adding a negligible amount of new information.

Lastly, some features were added/removed based on domain knowledge. According to figure 5.1 that shows the importance ranks of the selected features (the lower the rank, the better), the least useful chosen features are *Only integers, Outliers ratio* and *Minimum*. While the rank is higher, they are still crucial for the correct prediction of some distributions (especially *Poisson* and *Negative binomial*).

The statistics formulas below correspond with the calculation within the framework, using the *Scipy package* [31] and are calculated on the explained variable y[2].

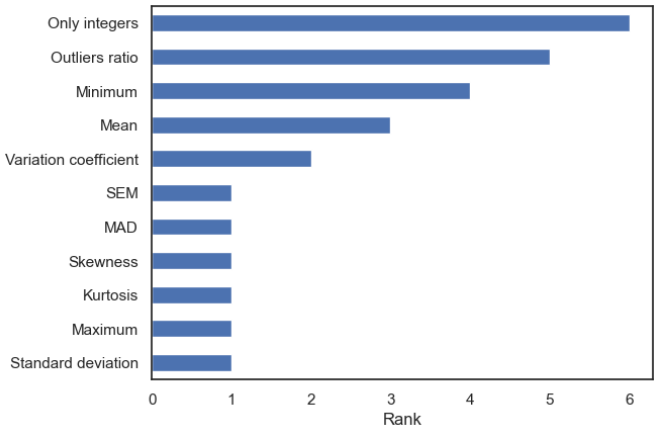
Outliers ratio Some *loss functions* (like *Laplace* or *Huber*) are more fitting for the data with many outliers. The ratio tells the percentage of terms out of the range using Interquartile range (IQR). The term x is considered an outlier if it is outside of the interval:

$$[Q_1 - 5 \cdot \text{IQR},\ Q_3 + 5 \cdot \text{IQR}] \tag{5.1}$$

[1] The third heuristic using *AutoML* had to train 13 models for each dataset and each distribution.

[2] As a matter of fact, in the proposed heuristic implementation, *Scipy* is not used, the calculations are done with the *H2O framework* itself.

Figure 5.1 Feature importance based on the *random feature elimination* algorithm from the *scikit-learn* [13]. Features are ranked from the most important (rank 1) to the least (rank 6) and built on the *final heuristic* using *cross-validation*.



Where Q_1 is the first quantile, Q_3 is the third quantile and IQR $= Q_3 - Q_1$. Note that this outlier detection method usually has a constant of 1.5 instead of 5. The number was risen to mark only the furthest outliers and was estimated through the multiple benchmarks with the pre-defined grid of possible values.

Coefficient of variation Measure of the relative variability. The lower value indicates lower variability.

$$CV = \frac{\sigma}{\bar{x}} \tag{5.2}$$

Where x is a *random variable*.

Standard deviation Well-known statistics showing the dispersion of the variable.

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n}} \tag{5.3}$$

Where x is a *random variable* and n is the number of observations.

Minimum Useful for determining if distribution contains negative numbers or zeros.

Maximum Proved to be useful. It provides information about outliers.

Mean Another well-known statistics. It is also used for the computation of almost all other *meta-features*.

Skewness Third moment, it is a measure of a distribution's asymmetry. *Scipy's* calucaltion uses *Fisher-Pearson* formula.

$$\gamma_1 = \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^3}{\left(\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2\right)^{3/2}} \tag{5.4}$$

Where x is a *random variable* and n is the number of observations.

Kurtosis Fourth moment. It describes the shape of a probability distribution. In *Scipy*, it is calculated through the *Fisher's formula*.

$$\gamma_2 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^4/n}{\left(\sum_{i=1}^{n}(x_i - \bar{x})^2/n\right)^2} - 3 \tag{5.5}$$

Where x is a *random variable* and n is the number of observations.

Median absolute deviation (MAD) Similar to the *standard deviation*, it is a measure of dispersion but less affected by outliers.

$$\text{MAD} = \text{median}\left(|x_1 - \tilde{x}|, |x_2 - \tilde{x}|, \ldots, |x_n - \tilde{x}|\right) \tag{5.6}$$

Where x is a *random variable* and $\tilde{x}$ is its median.

Standard error of mean (SEM) Measurement of the likelihood that the sampled value would differ from the population mean. Smaller values indicate that the sample mean is more likely to be close to the population mean.

$$\text{SEM}_x = \frac{\sigma}{\sqrt{n}} \tag{5.7}$$

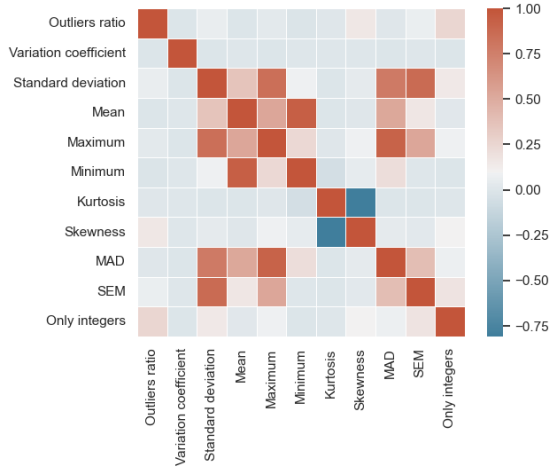
Where x is a *random variable* and n is the number of observations in x.

The selection of this feature was based more on randomness than on the fundament – the trees containing this feature generally performed better. It is similar to the *variation coefficient* that is also included. But while *variation coefficient* represents relative variability, *SEM* measures the precision of the sample mean estimate – so fundamentally, the features are different, bringing different information.

Only integers indicator The only selected feature that is not a descriptive statistic but a categorical variable. It tells if the random variable x consists only of integers or not.

Summarized, *eleven* different features were chosen. The selection was done with the first heuristic and validated later in the second and third. Figure 5.2 shows the correlation between the features. Since *standard deviation* and *mean* are used in other feature calculations, it is no surprise that they are more correlated with others.

Figure 5.2 Pearson's correlation between *meta-features*. It was done on the dataset from the *final heuristic*.



5.2.2 Other tested features

Three types of features were also tested and considered but were not chosen for the heuristic creation.

Other target variable statistics All the selected features (except for *Only integers*) are descriptive statistics of the target variable. Others were tested, like *quantiles, trimmed mean, IQR, ...*, but proved irrelevant.

Dataset characteristics I experimented with features like *data density* (*number of rows* divided by *number of columns*) or just solo dataset shapes. While in some cases it can be decisive, for the whole *H2O AutoML* had those features a little effect and were rather contraproductive.

Bivariate statistics Statistics (*Pearson's correlation, kurtosis, skewness*) between the target variable and training features and within the features in the training set. It was done with the approach proposed in the [33]. If there was a new feature for every pair of features between which would be statistics computed, the dimensionality would drastically grow. Therefore those statistics were computed, and then *binned* into the bins, [33] referred to those bins as *histograms*. The interval of bins is defined after the computation, and all the bins have a similar number of terms within them. In each bin is the number of features within this range, and those are the resulting *meta-features*.

The ideal number of bins was benchmarked and tested. It was suggested by [33] to use *ten bins*, but better results were obtained with *five bins*.

Even though there was some improvement in the heuristic (after discarding the bins that were not considered as important by the feature selection), the calculation took too long and contradicted the requirement for speed. All those statistics would have to be calculated for each input dataset, and that is too much time for a little improvement.

5.3 Dataset creation and heuristic evaluation

To apply a *meta-learning*, I first needed a dataset that could be learned from. All three heuristics are very similar in the form of the creation. What makes them distinct is the *meta-dataset* from which they were created. Each approach is described at the corresponding heuristic.

Meta-dataset consists of the *meta-features* (5.2.1) representing training set X, and the explained variable y is represented as a distribution with the *best results – multinomial classification*.

Defining the criteria at which is the given distribution better than the *Gaussian* is a complex problem, so the most straightforward approach was taken. There would be actually *four meta-datasets* that differ at the values of explained variables – its value is determined by the four different criteria: MAE, RMSE, RMSLE and ERD. For the MAE criterion, the distribution providing the lowest MAE score is chosen – and analogically for the rest.

The second approach (and the used one) is to create an *incosistent dataset*[3]. This means, e.g., that if the dataset outperformed the default distribution with *Poisson* and *Tweedie*, the *meta-dataset* will have two same rows in training set X, each with a different value of the explained variable (one *Poisson* and one *Tweedie*). Despite the inconsistency, this approach enlarges the training dataset and outperforms the consistent one. *Decision trees* can easily treat these inconsistencies by introducing the probabilities to the computations.[4]

Gaussian distribution was set as a target variable if the results on the other distributions were worse.

[3] In other words, if the heuristic predicts a distribution that improves over Gaussian distribution, it's considered as a correct prediction, even if it is not the best prediction.

[4] The inconsistent dataset was only the training set. Validation and test sets were consistent (one row equals one distinct dataset).

The heuristic itself is firstly assessed on the validation data from the *meta-dataset* using modified *confusion matrix* – the value is on the diagonal even though it is not predicted correctly, but still outperforms the default (*Gaussian*) distribution. The second evaluation is done on the test set and more described at each heuristic.

5.4 H1 – The first heuristic

The first attempt at heuristic creation was more kind of an experiment than a genuine attempt. To get some context in this domain and set a starting point. It had terrible results, but those failures defined how the heuristic should be created correctly. Also, feature selection was made mainly on the first heuristic (and validated on the second and third – but with much better results).

Only artificially generated data was used. The idea was to look at how would *H2O AutoML* generalize if it was trained purely on the almost-perfect (in terms of distribution properties) data and tested on the real-world datasets from *OpenML*. I noticed some flaws in this approach right at the beginning. The artificial data was not diverse enough – mainly *skewness* and *kurtosis* was almost the same for every dataset – both were expected to be important for the decisions. Also, they varied significantly on the few datasets from *OpenML*.

5.4.1 H1 – Meta-dataset creation

The artificial dataset (described in chapter 4) was taken, and for each of the 1980 datasets, each individual available model in *H2O AutoML* with each available distribution (3.1) set was benchmarked and results were taken.

Since benchmarks took a lot of time and the results were not good, the benchmarks were not performed for all the datasets. But still, 7878 unique results came from this benchmarking, and another improvement would be negligible at this number.

Analysis of the dataset provided a lot of information. The most important one was the overall performance expressed in counts that may be seen in figure 5.3. It shows that if the "distribution" hyperparameter is set to the according distribution of the explained variable, it outperforms *Gaussian distribution* in a significant amount of cases (for almost all distributions), which proves that creating the heuristic is possible.

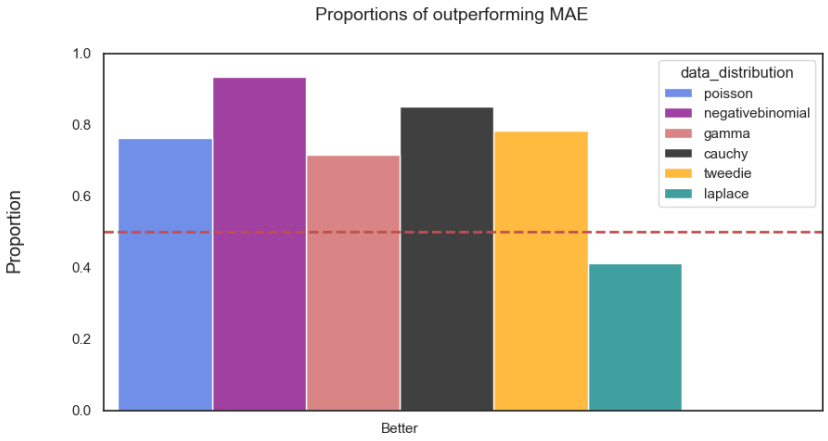
5.4.2 H1 – Conclusion

As was stated, the first heuristic was more the testing one and was not finished. It ended after a few trials with the *decision trees*, whose results had the same value as a random roll with the dice. Although the initial conditions looked optimistic (figure 5.3), it turned out that the quality of *meta-features* is severely low. There was not enough diversity, and the time that would be required to create the new artificial dataset and especially time for benchmarks to create the dataset would be too great. For this reason, I moved to the *OpenML datasets*.

5.5 H2 – The second heuristic

The second attempt for the heuristic creation had much better results on the validation set but very poor results on the test set. Used *meta-feafures* are the same (described in 5.2).

Figure 5.3 First heuristic *meta-dataset* counts. The relative percentage of how many datasets from the given distribution outperformed models with *Gaussian* distribution. A horizontal line is shown at $y = 0.5$ to show the proportion boundary of outperformance.



5.5.1 H2 – Meta-dataset creation

The most significant change was the move from the *artificial data* to the real-world datasets obtained from the *OpenML* platform, which were initially intended as a test set for the heuristic created from the *artificial data*. 832 different datasets were benchmarked (chosen concerning the possible biases described in 2.4).

The benchmarks were done on the standalone models from the *H2O AutoML* with their default parameters, not on *H2O AutoML* itself. This was intended to create more variability and see if the rules derived from the tree were more general or not. As described in 3.2, the models used in *H2O AutoML* are predefined, with chosen hyperparameters and training order.

Meta-dataset itself was tested in consistent and inconsistent form[5]. The inconsistent form had better results. In that form, the dataset had 2727 rows.

5.5.2 H2 – Decision tree

With the built dataset (and split on training, validation, and test parts), *decision tree* was constructed. The most considered hyperparameter was *class weight*, determining the weight for each class. The testing was done on the validation dataset along with *hyperparameter serach* using *Bayesian optimization*. Since the number of records in each class is not equal (but differs a lot), the weight ended up accordingly to the proportions (bigger proportion, lower class weight). Also, class weight for the *Gaussian distribution* rose because it is desirable to obtain predict *Gaussian* rather than the distribution that would end up worse.

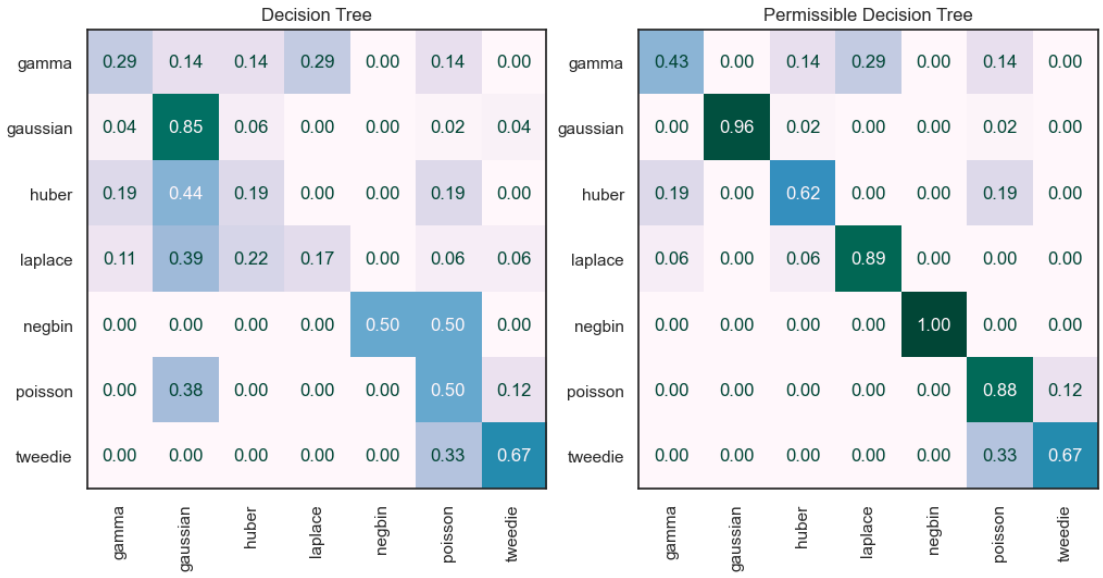
The results of the second heuristic on the validation data can be seen in figure 5.4.

5.5.3 H2 – Benchmarks on H2O AutoML

With this heuristic tweaked and cross-validated, it was time to test it on the *H2O AutoML* itself – on the test dataset that was kept from the training and tuning all the time. The results may be seen in the figure 5.5, and they are also not good.

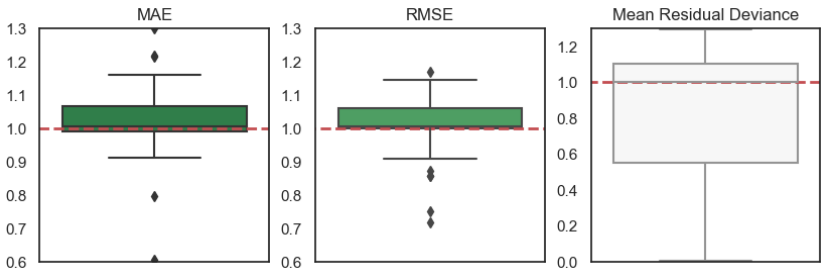
[5] As was described in 5.3

Figure 5.4 Both matrices display each class's relative counts (percentage). The left confusion matrix is the confusion matrix of the base model. The matrix on the right is permissible – marking prediction as true if it outperforms *Gaussian* regardless of whether the value corresponds with the true one.



The only improved metric was ERD. Both MAE and RMSE had worse results. This heuristic could have been used at least for the improvement of the ERD, but that is not that much success since the rest of the criteria are far more frequent and more assessed in *AutoML competitions* 2.3. The ability to generalize from the default models to the *AutoML* models was not transferred.

Figure 5.5 The second heuristic results were performed on the test set. The red dotted line represents the model with the default (*Gaussian*) distribution. Values above this value are worse, and values below are better. The value is obtained as (`Distribution value` / `Gaussian value`)



5.5.4 H2 – Additional experiments

Some additional experiments were performed during the creation. *Dataset balancing techniques* (*under-sampling*, *SMOTE method* and *Tomek links*) were tested. Still, the results on the validation data not had not significantly improved, so it was not used in the final form.

Also, data was augmented with the *artificial data* regarding the balance of each class. In some cases, this outperformed the heuristic on the validation data but not the test data.

For the comparison, *meta-dataset* was also trained on other, more powerful models. Both

Random forest and GBM outperformed the *Decision tree*, but not that significantly. Also, *H2O AutoML* was put on those data, but again the improvement was insignificant.

5.6 H3 – The final heuristic

The third heuristic was successful and outperformed the default distribution in all the tracked criteria.

5.6.1 H3 – Meta-dataset creation

Meta-dataset for the third heuristic was built from the *OpenML* datasets, but apart from the *second heuristic*, the creation was done on the *H2O AutoML* framework itself. The benchmarks ran for two weeks since every dataset was tested with every distribution – many of the datasets failed (because of the low memory of the used benchmark device). *H2O AutoML* was confined to using only thirteen base models. Those are the base models described in 3.2, without the *grid search* and other tunings – only the pure base models with predefined hyperparameters.

As a result, 798 different datasets were benchmarked, and using an inconsistent explained variable approach (described in 5.3), the *meta-dataset* had 1332 rows.

5.6.2 H3 – Decision tree

Four different decision trees (for each of the following criteria – MAE, RMSE, RMSLE and ERD) were built and tuned. Since explained variable differs by each criterion (if MAE ended up better, it does not necessarily mean that other metrics were also better), the proportions of each distribution were different.

Regardless of those proportions, trees with the manual set *class weight* (which was not balanced) ended up better than the balanced ones. Balancing was set and validated with manual tweaking, resulting in *Poisson distribution* to weight 4, *gamma distribution* to weight 3.5, and the rest of the distributions with value 3.

Figure 5.6 Both matrices display each class's relative counts (percentage) and work the same way as the matrices in figure 5.4.

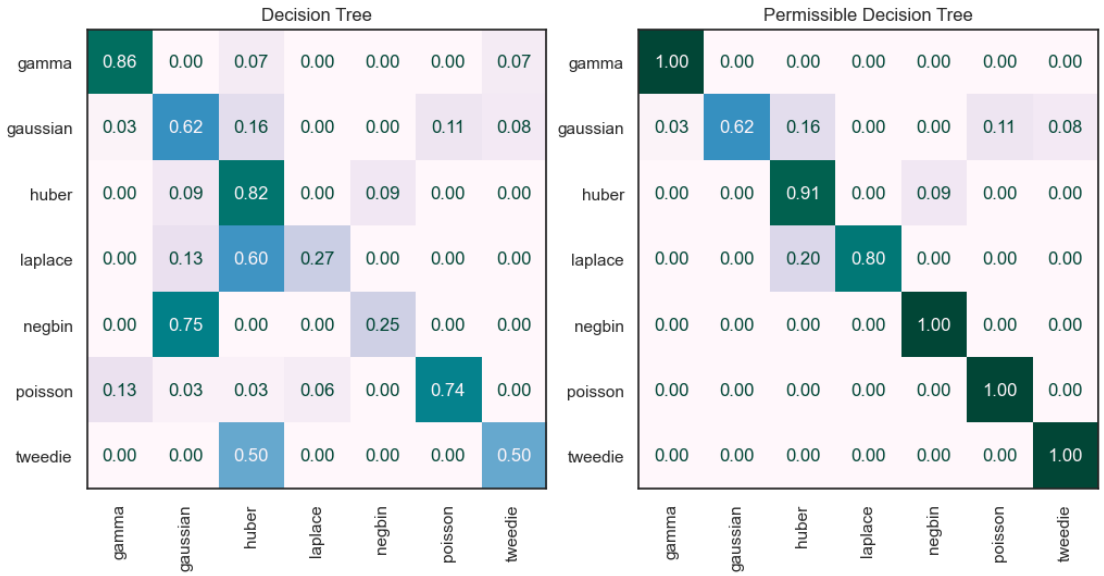


Figure 5.6 shows the confusion matrices on the validation data. The right matrix shows that the heuristic provided the correct prediction of the better distribution for almost five distributions. The *Achilles heel* is the *Gaussian distribution* which is correctly predicted only from the 62%, resulting in the rest 38% being worse.

All the additional techniques mentioned in 5.5.4 were also tested but proved again as not significantly better.

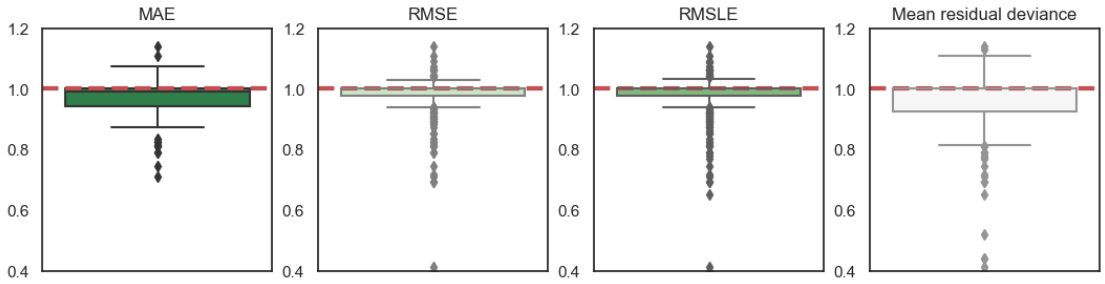
5.6.3 H3 – Results on the H2O AutoML

Since the *Meta-dataset* was generated with *H2O AutoML*, it was unnecessary to rerun benchmarks to see the test results. Instead, a quick comparison, if the chosen distribution outperforms *Gaussian*, was sufficient – since the benchmarks were already done from the creation part.

Every one of the four heuristics (determined by criterion) was tested. The best overall results for all the requirements had a tree built on the MAE criterion. Trees constructed on RMSE and RMSLE performed worse (even at themselves) but still outperformed *Gaussian*. And tree based on ERD ended up as the worst one and is outperformed by *Gaussian distribution*. I attribute this to the fact that the training dataset for the ERD was highly imbalanced and that the criterion targets the quality of fit instead of the prediction quality.

Two evaluation approaches were used to assess the heuristic. The first one is the same visualization technique as was used for the second heuristic – figure 5.5. The results for the MAE criterion (which ended up overall best) may be seen in figure 5.7.

Figure 5.7 The performance of the final heuristic on the *H2O AutoML* built on the MAE criterion. The red dotted line represents the model with the default (*Gaussian*) distribution. Values above this value are worse, and values below are better. The value is obtained as (`Distribution value` / `Gaussian value`)



Evaluation with the boxplots is backtested with the statistical test. The differences between *Gaussian* and the predicted distribution do not follow the *normal (Gaussian) distribution*, so the *Wilcoxon signed-rank test* [34] was used. The *null hypothesis* was set as the differences between the predictions $\hat{y}$ and true values y are symmetric about zero. The alternative hypothesis was set as *less*, meaning that the distribution of the differences is less than a distribution symmetric about zero.

This test was done for all four followed metrics, resulting in four p-values very close to zero. This means that the *null hypothesis* can be rejected and indicates that the alternative hypothesis is likely true – and, in this case, can be confirmed from the boxplots in 5.7.

Regarding those results, I can say that the performance of *H2O AutoML* has improved by 0-5% in the four followed metrics.

5.6.4 Rule extraction

Since the heuristic is created and represented by the *decision tree*, the rule extraction is simple. Rules are extracted from the tree built with the MAE criterion because it performed the best in all four metrics. All the nodes are recursively visited – firstly, left nodes, then right nodes. Within this procedure, rules in the form of Python code are generated with indentation corresponding to the node depth.

```
if y_skewness <= 0.7:
    if y_mad <= 0.12:
        if y_sem <= 0.0:
            if y_std <= 0.0:
                return "huber"
            else:  # if y_std > 0.0
                return "gaussian"
        else:  # if y_sem > 0.0
            # (...)
```

Code listing 5.1 Example of the extracted rules

Extracted rules are long, even with only eleven features, consisting of 685 rows. It may be seen that some rules are futile – for instance, in the code shown in code 5.1, the `y_sem` (Standard error of the mean) can not be negative (by its definition), and the likelihood that it would equals to zero is also almost zero. For greater effectiveness and clarity, rules should be pruned manually. But since this is only a prototype, there is no reason to do that for the thesis.

5.6.5 Conclusion

The results of the third heuristic are positive, meaning the appropriate hyperparameter value can improve performance. My computer limited me; hence not all the datasets could be processed (due to size, etc...), and *H2O AutoML* was strictly confined by the number of models. For those reasons, the creation of the heuristic should be repeated by *H2O*, using more powerful computational devices and more datasets – with the code from the third heuristic. After the creation, it should be manually reviewed to improve effectiveness and remove futile or very specific rules.

Chapter 6

Implementation to H2O AutoML

This chapter describes the prototype implementation of the heuristic to H2O AutoML using the Python interface of the framework. This is followed by the functionality test with the same example as in the previous chapter 3.3, but using the distribution proposed by the heuristic.

6.1 Implementation

The interface is *Pythonic*, but in *H2O AutoML*, code is translated to the language similar to *Lisp*[1]. Then, those expressions are calculated parallelly on the backend, using *MapReduce*.

Automated distribution selection is invoked if the *distribution* hyperparameter is set as *estimate*. The estimation uses the `train` method. If, for some reason, estimation fails, the default *Gaussian* value is used (for regression tasks), so this solution is safe and, in the worst case, changes nothing.

The only affected file is `h2o/automl/_estimator.py`. To the `train` method, after y validation (which is the first thing that is being done) is put a small piece of code referring to the new file `_distribution.py` (code 6.1). There is sent the explained variable in the form of the `H2OFrame` (which is a similar structure to the `Pandas DataFrame`).

```
# (...) y validation

if self.distribution == "estimate":
    self.distribution = distribution_heuristic(self.training_frame[y])

# (...) rest of the train function
```

Code listing 6.1 Integration to the existing function `train`.

This file contains the heuristic in the form of IF... ELSE rules generated in the previous chapter and a function that creates properties of the y (5.2.1). This method returns the estimated distribution or, in case of a failure, the default value (which is *Gaussian*) (code 6.2). Note that all the *meta-features* calculations are being done only using *H2O* package, and in one case, using native *Python* package *math*. No external dependencies are required.

```
from math import sqrt


def distribution_heuristic(y):

```

[1] H2O refers to it as *Rapid expression*.

```
    quantiles = y.quantile(prob=[.25, .50, .75])
    y_q1 = quantiles[0, 1]
    y_median = quantiles[1, 1]
    y_q3 = quantiles[2, 1]

    IQR = y_q3 - y_q1
    outliers = y[(y < (y_q1 - 5 * IQR)) | (y > (y_q3 + 5 * IQR))]
    y_outliers_ratio = len(outliers) / len(y)
    y_std = y.sd()[0]
    y_mean = y.mean()[0]
    y_variation_coefficient = y_std / y_mean
    y_max = y.max()
    y_min = y.min()

    y_kurtosis = y.kurtosis()[0]
    y_skewness = y.skewness()[0]
    y_mad = (abs(y - y_median)).median()[0]
    y_sem = y_std / sqrt(len(y))
    y_only_integers = 1 if (y.dtype == "int") else 0
    try:
        return estimate_distribution(y_outliers_ratio,
    y_variation_coefficient, y_std, y_mean, y_max, y_min, y_kurtosis,
    y_skewness, y_mad, y_sem, y_only_integers)
    except ValueError:
        return "gaussian"
```

Code listing 6.2 Implementation of obtaining meta-features. Function `estimate_distribution` is made of generated rules; part of it is in the code 5.1.

6.2 Testing

An example of *H2O AutoML* was shown in 3.3. I replicate the run, but instead of setting *distribution* parameter as *auto*, I set it as *estimate*. Based on the properties of the explained variable, the heuristic proposed a *Gamma* distribution with that is model trained.

Referring to the leaderboard with the *Gaussian* distribution 3.1, the value of RMSLE was approximately 0.407. The RMSLE of the *H2O AutoML* with the implemented heuristic is 0.399, so there is a small improvement.

Figure 6.1 Leaderboard of the best models with *Gamma* distribution, ordered by RMSLE. Ordering can be changed. Column *model id* determines the used model; the rest are metrics, described in 1.1.

	model_id	rmse	mse	mae	rmsle	mean_residual_deviance
0	XGBoost_3_AutoML_23_20230508_83830	4783.374899	2.288068e+07	2484.764766	0.399417	2.043995e+01
1	DRF_1_AutoML_23_20230508_83830	4798.550557	2.302609e+07	2758.264493	0.438675	2.302609e+07
2	XGBoost_grid_1_AutoML_23_20230508_83830_model_1	4818.818618	2.322101e+07	2694.618675	0.395664	2.040512e+01
3	XGBoost_2_AutoML_23_20230508_83830	4893.508758	2.394643e+07	2615.519096	0.410890	2.044658e+01
4	XGBoost_1_AutoML_23_20230508_83830	4925.729021	2.426281e+07	2711.679625	0.413697	2.044538e+01
5	GBM_4_AutoML_23_20230508_83830	5156.646042	2.659100e+07	3030.291475	0.448196	2.040560e+01

A far bigger improvement is in the ERD. The value for the *Gaussian* model was $2.01 \cdot 10^7$ while for the *Gamma* model, the value is $2.04 \cdot 10^1$, meaning that the model fits the data better by a tremendous difference.[2]

[2] This does not necessarily implicate the improved prediction performance of the model.

Conclusion

This thesis aimed to propose and create a heuristic for the automated hyperparameter selection for the *H2O AutoML framework* that would improve overall performance. This aim was fulfilled, and a heuristic in the form of IF... ELSE rules was created, using *meta-learning* and *meta-features* based on the benchmarks and tests.

This proposed heuristic improves the performance in four metrics – MAE, RMSE, RMSLE and ERD making *H2O AutoML* more accurate in the predictions for a significant proportion of possible use-cases.

For future work and usage, benchmarks should be done again, using a more powerful computational device that would allow larger datasets to increase the variety in the creation, making the heuristic more robust and general.

Also, rules created by the heuristic should be manually reviewed and pruned to achieve greater clarity and effectiveness.

The thesis also showed other possible approaches to creation, showing why they are unsuitable. Anyone who would follow up on this thesis has a solid baseline.

Bibliography

1. *Performance and Prediction — H2O 3.40.0.3 documentation* [online]. [visited on 2023-04-22]. Available from: https://docs.h2o.ai/h2o/latest-stable/h2o-docs/performance-and-prediction.html.
2. Normal Distribution. In: *The Concise Encyclopedia of Statistics* [online]. New York, NY: Springer, 2008, pp. 378–380 [visited on 2023-04-13]. ISBN 978-0-387-32833-1. Available from DOI: 10.1007/978-0-387-32833-1_285.
3. Poisson Distribution. In: *The Concise Encyclopedia of Statistics* [online]. New York, NY: Springer, 2008, pp. 425–427 [visited on 2023-04-13]. ISBN 978-0-387-32833-1. Available from DOI: 10.1007/978-0-387-32833-1_321.
4. Gamma Distribution. In: *The Concise Encyclopedia of Statistics* [online]. New York, NY: Springer, 2008, pp. 215–216 [visited on 2023-04-13]. ISBN 978-0-387-32833-1. Available from DOI: 10.1007/978-0-387-32833-1_157.
5. Negative Binomial Distribution. In: *The Concise Encyclopedia of Statistics* [online]. New York, NY: Springer, 2008, pp. 369–370 [visited on 2023-04-13]. ISBN 978-0-387-32833-1. Available from DOI: 10.1007/978-0-387-32833-1_277.
6. Laplace Distribution. In: *The Concise Encyclopedia of Statistics* [online]. New York, NY: Springer, 2008, pp. 294–295 [visited on 2023-04-13]. ISBN 978-0-387-32833-1. Available from DOI: 10.1007/978-0-387-32833-1_219.
7. DUNN, Peter K; SMYTH, Gordon K. Tweedie family densities: methods of evaluation [online]. 2001, pp. 2–6 [visited on 2023-04-02]. Available from: https://www.researchgate.net/publication/237533744_Tweedie_Family_Densities_Methods_of_Evaluation.
8. LEDELL, Erin; POIRIER, Sebastien. H2O AutoML: Scalable Automatic Machine Learning. *7th ICML Workshop on Automated Machine Learning (AutoML)* [online]. 2020 [visited on 2023-04-13]. Available from: https://www.automl.org/wp-content/uploads/2020/07/AutoML_2020_paper_61.pdf.
9. Cauchy Distribution. In: *The Concise Encyclopedia of Statistics* [online]. New York, NY: Springer, 2008, pp. 60–60 [visited on 2023-04-13]. ISBN 978-0-387-32833-1. Available from DOI: 10.1007/978-0-387-32833-1_46.
10. WIRTH, Rüdiger; HIPP, Jochen. CRISP-DM: Towards a Standard Process Model for Data Mining [online]. [N.d.] [visited on 2023-04-13]. Available from: http://www.cs.unibo.it/~danilo.montesi/CBD/Beatriz/10.1.1.198.5133.pdf.
11. KURGAN, Lukasz A.; MUSILEK, Petr. A survey of Knowledge Discovery and Data Mining process models. *The Knowledge Engineering Review* [online]. 2006, vol. 21, no. 1, pp. 1–24 [visited on 2023-02-22]. ISSN 0269-8889, ISSN 1469-8005. Available from DOI: 10.1017/S0269888906000737.

12. GIJSBERS, Pieter; BUENO, Marcos L. P.; COORS, Stefan; LEDELL, Erin; POIRIER, Sébastien; THOMAS, Janek; BISCHL, Bernd; VANSCHOREN, Joaquin. *AMLB: an AutoML Benchmark* [online]. arXiv, 2022 [visited on 2023-04-13]. No. arXiv:2207.12560. Available from DOI: 10.48550/arXiv.2207.12560.

13. PEDREGOSA, F.; VAROQUAUX, G.; GRAMFORT, A.; MICHEL, V.; THIRION, B.; GRISEL, O.; BLONDEL, M.; PRETTENHOFER, P.; WEISS, R.; DUBOURG, V.; VANDERPLAS, J.; PASSOS, A.; COURNAPEAU, D.; BRUCHER, M.; PERROT, M.; DUCHESNAY, E. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* [online]. 2011, vol. 12, pp. 2825–2830 [visited on 2023-04-13]. Available from: https://scikit-learn.org.

14. FEURER, Matthias; EGGENSPERGER, Katharina; FALKNER, Stefan; LINDAUER, Marius; HUTTER, Frank. Auto-sklearn 2.0: The next generation. *arXiv preprint arXiv:2007.04074* [online]. 2020, vol. 24 [visited on 2023-04-13]. Available from: https://www.researchgate.net/publication/342801746_Auto-Sklearn_20_The_Next_Generation.

15. WANG, Chi; WU, Qingyun; WEIMER, Markus; ZHU, Erkang. FLAML: A fast and lightweight automl library. *Proceedings of Machine Learning and Systems* [online]. 2021, vol. 3, pp. 434–447 [visited on 2023-04-13]. Available from: https://arxiv.org/pdf/1911.04706.pdf.

16. ERICKSON, Nick; MUELLER, Jonas; SHIRKOV, Alexander; ZHANG, Hang; LARROY, Pedro; LI, Mu; SMOLA, Alexander. Autogluon-tabular: Robust and accurate automl for structured data. *arXiv preprint arXiv:2003.06505* [online]. 2020 [visited on 2023-04-13]. Available from: https://arxiv.org/pdf/2003.06505.pdf.

17. VANSCHOREN, Joaquin; RIJN, Jan N. van; BISCHL, Bernd; TORGO, Luis. OpenML: networked science in machine learning. *SIGKDD Explorations* [online]. 2013, vol. 15, no. 2, pp. 49–60 [visited on 2023-03-11]. Available from DOI: 10.1145/2641190.2641198.

18. DEMŠAR, Janez. Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine learning research* [online]. 2006, vol. 7, pp. 1–30 [visited on 2023-04-01]. Available from: https://www.jmlr.org/papers/volume7/demsar06a/demsar06a.pdf.

19. FEURER, Matthias; RIJN, Jan N. van; KADRA, Arlind; GIJSBERS, Pieter; MALLIK, Neeratyoy; RAVI, Sahithya; MUELLER, Andreas; VANSCHOREN, Joaquin; HUTTER, Frank. OpenML-Python: an extensible Python API for OpenML. *arXiv* [online]. 2020, vol. 1911.02490 [visited on 2023-03-11]. Available from: https://arxiv.org/pdf/1911.02490.pdf.

20. BRAZDIL, Pavel; RIJN, Jan N van; SOARES, Carlos; VANSCHOREN, Joaquin. *Metalearning: applications to automated machine learning and data mining*. Springer Nature, 2022. ISBN 978-3-030-67024-5.

21. BREIMAN, Leo. Random Forests. *Machine Learning* [online]. 2001, vol. 45, no. 1, pp. 5–32 [visited on 2023-03-08]. ISSN 1573-0565. Available from DOI: 10.1023/A:1010933404324.

22. NELDER, J. A.; WEDDERBURN, R. W. M. Generalized Linear Models. *Journal of the Royal Statistical Society: Series A (General)* [online]. 1972, vol. 135, no. 3, pp. 370–384 [visited on 2023-03-10]. ISSN 2397-2327. Available from DOI: 10.2307/2344614.

23. NYKODYM, Tomas; KRALJEVIC, Tom; WANG, Amy; WONG, Wendy. Generalized linear modeling with h2o. *Published by H2O. ai Inc* [online]. 2016 [visited on 2023-04-23]. Available from: https://docs.h2o.ai/h2o/latest-stable/h2o-docs/booklets/GLMBooklet.pdf.

24. FRIEDMAN, Jerome H. Greedy Function Approximation: A Gradient Boosting Machine. *The Annals of Statistics* [online]. 2001, vol. 29, no. 5, pp. 1189–1232 [visited on 2023-03-18]. ISSN 0090-5364. Available from: https://www.jstor.org/stable/2699986.

25. CHEN, Tianqi; GUESTRIN, Carlos. XGBoost: A Scalable Tree Boosting System. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* [online]. 2016, pp. 785–794 [visited on 2023-04-23]. Available from DOI: 10.1145/2939672.2939785.

26. CANDEL, Arno; PARMAR, Viraj; LEDELL, Erin; ARORA, Anisha. Deep learning with H2O. *H2O. ai Inc* [online]. 2016, pp. 1–21 [visited on 2023-04-23]. Available from: https://docs.h2o.ai/h2o/latest-stable/h2o-docs/booklets/DeepLearningBooklet.pdf.

27. LAAN, Mark J. van der; POLLEY, Eric C.; HUBBARD, Alan E. Super Learner. *Statistical Applications in Genetics and Molecular Biology* [online]. 2007, vol. 6, no. 1 [visited on 2023-04-01]. ISSN 1544-6115. Available from DOI: 10.2202/1544-6115.1309.

28. *Distribution — H2O 3.40.0.3 documentation* [online]. [visited on 2023-04-22]. Available from: https://docs.h2o.ai/h2o/latest-stable/h2o-docs/data-science/algo-params/distribution.html.

29. HUBER, Peter J. Robust Estimation of a Location Parameter. In: KOTZ, Samuel; JOHNSON, Norman L. (eds.). *Breakthroughs in Statistics: Methodology and Distribution* [online]. New York, NY: Springer, 1992, pp. 492–518 [visited on 2023-04-13]. Springer Series in Statistics. ISBN 978-1-4612-4380-9. Available from DOI: 10.1007/978-1-4612-4380-9_35.

30. *Medical Cost Personal Datasets* [online]. [visited on 2023-05-08]. Available from: https://www.kaggle.com/datasets/mirichoi0218/insurance.

31. VIRTANEN, Pauli; GOMMERS, Ralf; OLIPHANT, Travis E.; HABERLAND, Matt; REDDY, Tyler; COURNAPEAU, David; BUROVSKI, Evgeni; PETERSON, Pearu; WECKESSER, Warren; BRIGHT, Jonathan; VAN DER WALT, Stéfan J.; BRETT, Matthew; WILSON, Joshua; MILLMAN, K. Jarrod; MAYOROV, Nikolay; NELSON, Andrew R. J.; JONES, Eric; KERN, Robert; LARSON, Eric; CAREY, C J; POLAT, İlhan; FENG, Yu; MOORE, Eric W.; VANDERPLAS, Jake; LAXALDE, Denis; PERKTOLD, Josef; CIMRMAN, Robert; HENRIKSEN, Ian; QUINTERO, E. A.; HARRIS, Charles R.; ARCHIBALD, Anne M.; RIBEIRO, Antônio H.; PEDREGOSA, Fabian; VAN MULBREGT, Paul; SCIPY 1.0 CONTRIBUTORS. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods* [online]. 2020, vol. 17, pp. 261–272 [visited on 2023-04-07]. Available from DOI: 10.1038/s41592-019-0686-2.

32. BRAZDIL, Pavel; RIJN, Jan N van; SOARES, Carlos; VANSCHOREN, Joaquin. Dataset Characteristics (Metafeatures). In: *Metalearning: applications to automated machine learning and data mining*. Springer Nature, 2022, pp. 53–75. ISBN 978-3-030-67024-5.

33. KALOUSIS, Alexandros; THEOHARIS, Theoharis. NOEMON: An intelligent assistant for classifier selection [online]. 1999 [visited on 2023-04-08]. Available from: https://www.researchgate.net/publication/2239411_NOEMON_An_Intelligent_Assistant_for_Classifier_Selection.

34. WOOLSON, R. F. Wilcoxon Signed-Rank Test. In: *Wiley Encyclopedia of Clinical Trials* [online]. John Wiley & Sons, Ltd, 2008, pp. 1–3 [visited on 2023-04-22]. ISBN 978-0-471-46242-2. Available from DOI: 10.1002/9780471462422.eoct979.

List of files in attachment

- `README.md` ..... instructions
- `Dockerfile` ..... to create Docker image
- `requirements.txt` ..... required packages
- `AutoML example`
  - `H2O AutoML example.ipynb` ..... Jupyter notebook with example of H2O AutoML
  - `insurance.csv` ..... data source for example
- `Data generation`
  - `data-generation.ipynb` ..... Jupyter notebook containing artificial data generation
  - `gen_params.csv` ..... meta-data from generation
  - `generators.py` ..... generating functions for each distribution
- `Final heuristic`
  - `1 - AML dataset creation.ipynb` ..... Jupyter notebook with dataset generation
  - `2 - Heuristic.ipynb` ..... Jupyter notebook with creation of the proposed heuristic
  - `aml_dataset.csv` ..... dataset created in the first Jupyter notebook
  - `dataset_ids.csv` ..... ids of the regression tasks available in OpenML
  - `failed_datasets.csv` ..... ids of the failed datasets from the first notebook
  - `heuristic.py` ..... proposed heuristic exported from the second notebook
- `Implementation`
  - `_distribution.py` ..... implementation – see instructions
  - `_estimator.py` ..... implementation – see instructions
- `other`
  - `First heuristic` ..... source files to replicate the creation of the first heuristic
  - `Second heuristic` ..... source files to replicate the creation of the second heuristic
- `thesis.zip` ..... Source LaTeX code for thesis