UNIVERSITY OF TARTU
Faculty of Science and Technology
Institute of Computer Science
Data Science (MSc) Curriculum

Dmitri Rozgonjuk

Towards Automated Machine Learning: Hyperparameter Optimization in Online Clustering

Master's Thesis (15 ECTS)

Supervisor: Radwa El Shawi, PhD

Tartu 2023

Towards automated machine learning: hyperparameter optimization in online clustering

Abstract
Machine Learning (ML) has demonstrated significant potential in data-driven applications, particularly in real-time use cases through online ML, which processes data streams and handles concept drift (changes in data distribution) dynamically. Automated ML (AutoML) seeks to streamline ML pipeline tasks like hyperparameter optimization (HPO) and model selection for improved performance. While some efforts have been made to integrate online ML and AutoML, research on automated online clustering remains limited. This thesis focuses on developing a potential HPO solution in online clustering settings. The aim was to propose an ensemble-based approach that leverages more than one internal clustering validation index (CVI) to address the evaluation problem in online clustering. HPO was implemented on top of the `river` framework. To compare the performance of HPO in online clustering, two online clustering algorithms were used on six synthetic datasets with ground truth labels. In HPO, models were separately optimized towards two internal CVIs, the Silhouette score and the Calinski-Harabasz Index, and models were compared by using an external CVI, the Adjusted Rand Index. In the experiments, (a) default online clustering algorithms with default parameters, (b) the best optimized online clustering algorithms, and (c) the ensemble of the best optimized models were compared. The findings revealed that the efficacy of HPO varies depending on the data type. In k-centroid-based datasets, the Silhouette-optimized model and the ensemble model outperformed other clustering solutions, while HPO and ensembling did not yield superior results in S-curve datasets.

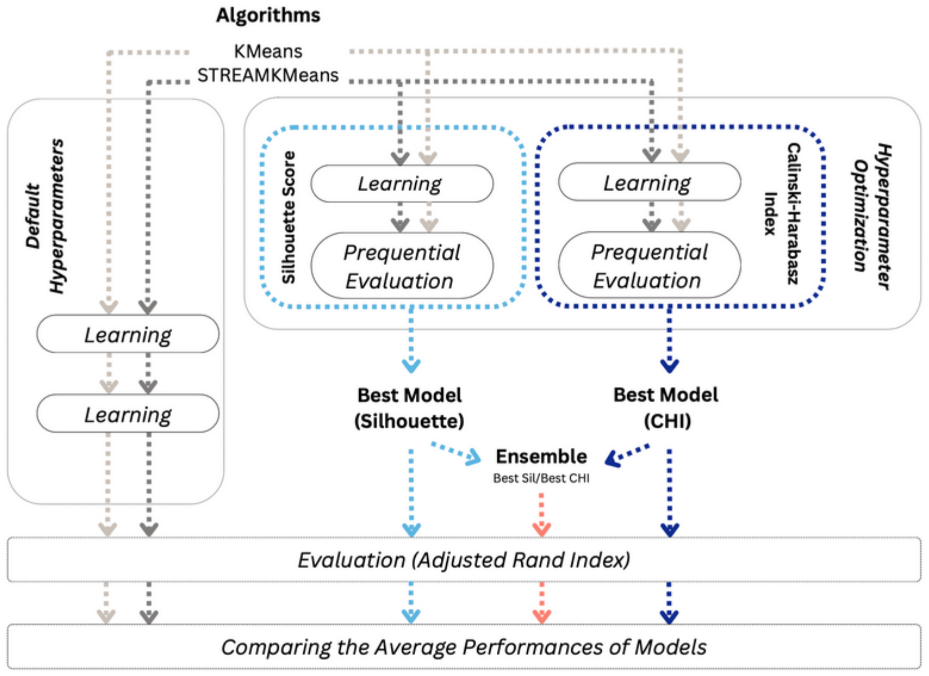
Keywords: autoML; online ML; online clustering; hyperopt; river
CERCS: P170 - Computer science, numerical analysis, systems, control)

Towards automated machine learning: hyperparameter optimization in online clustering

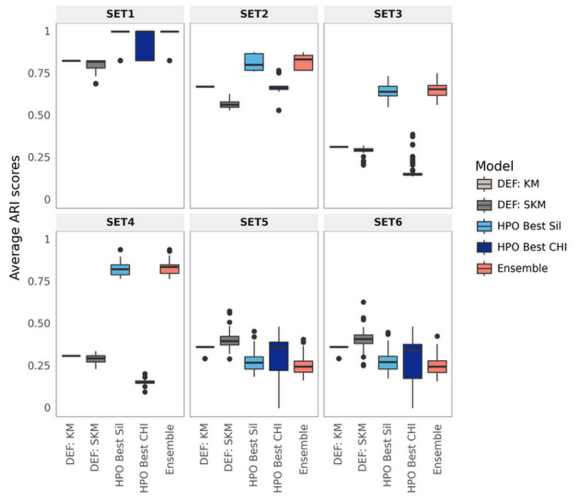
Dmitri Rozgonjuk
Data Science (MSc) 2023
Institute of Computer Science
University of Tartu
Supervisor: Radwa El Shawi, PhD

#UniTartuCS

Workflow



Results



Insights

- **HPO** (especially towards the best Silhouette) **generally outperformed default models** in k-center datasets
- **Ensembling** produced **comparable results** to **best Silhouette-optimized** models
- **Nature of the data matters** - HPO was useful in k-center datasets but not with S-curve datasets
- The results indicate that **HPO may be useful** - but it **likely depends** on the **suitability of algorithms for data**

Automatiseeritud masinõppe suunas: hüperparameetrite optimeerimine *online*-klasterdamises

Lühikokkuvõte

Masinõpe (ingl k *machine learning*; ML) on näidanud suurt potentsiaali andmepõhistes, eriti reaalajas kasutatavates rakendustes, kasutades online-ML-i, mis töötleb andmevooge ning kohandub dünaamiliselt andmejaotuste muutusega. Automatiseeritud ML (AutoML) püüab automatiseerida mitmeid ML töövoos sisalduvaid ülesandeid nagu hüperparameetrite optimeerimist (HPO) ning (parima) mudeli valikut. Kuigi on teadustöid, mis on püüdnud ühendada *online*-ML-i ja AutoML-i, on automatiseeritud *online*-klasterdamise alase töö hulk piiratud. Käesoleva magistritöö fookuses on potentsiaalse HPO lahenduse arendamine *online*-klasterdamises. Eesmärgiks oli arendada mudelite ansamblimisel põhinev lähenemine, mis kasutab rohkem kui ühte sisemist klastrivalideerimisindeksit (KVI), et adresseerida mudeli hindamise probleemi *online*-klasterdamises. HPO rakendamiseks kasutati `river` raamistikku. HPO tulemuste testimiseks on rakendatud kahte *online*-klasterdamise algoritmi kuuel sünteetilisel andmestikul koos klastrikuuluvuse märgenditega. HPO-s optimeeriti mudeleid eraldi KVI-ga (Silhouette-i skoor ning Calinski-Harabaszi Indeks) ning mudeleid võrreldi omavahel välise KVI, Kohandatud Randi Indeksi abil. Eksperimentides võrreldi (a) vaikimisi hüperparameetritega *online*-klasterdamisalgoritme (b) parimate optimeeritud *online*-klasterdamisalgoritmide ning (c) parimatest optimeeritud mudelitest kokku pandud ansambel-mudeli sooritusega. Tulemustest selgus, et HPO sooritus võib sõltuda andmete tüübist. K-tsentroidide põhistes andmestikes näitasid parimat sooritust parimad Silhouette-iga optimeeritud mudelid ning ansambel-mudelid. Samas aga ei olnud HPO-l ning mudelite ansamblil sooritust parandavat efekti S-kõvera põhistes andmestikes.

Võtmesõnad: autoML; online-ML; online-klasterdamine; hyperopt; river

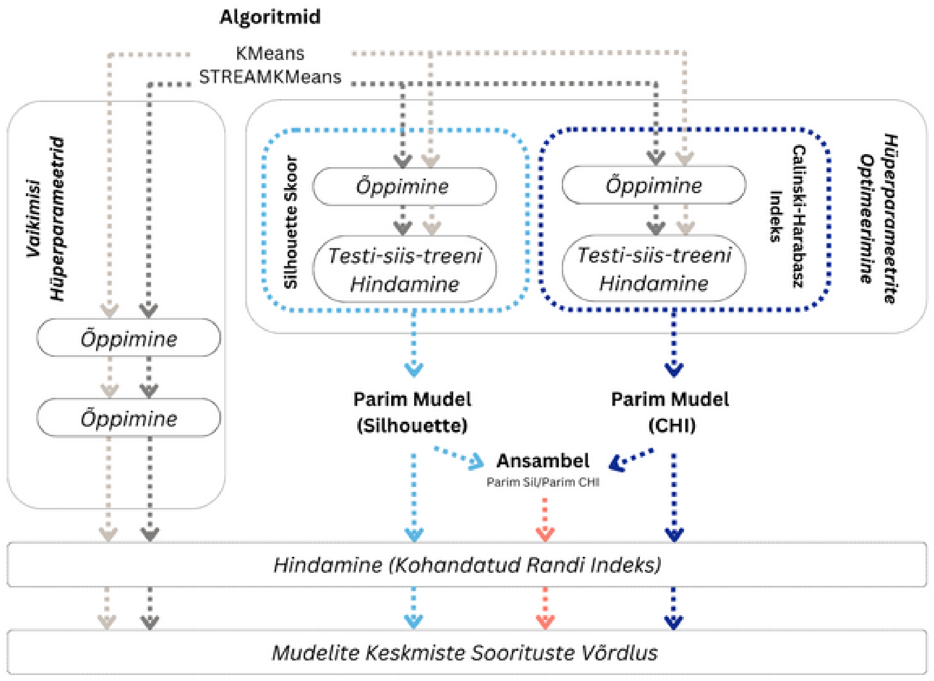
CERCS: P170 - Arvutiteadus, arvutusmeetodid, süsteemid, juhtimine (automaatjuhtimisteooria)

Automatiseeritud masinõppe suunas: hüperparameetrite optimeerimine online-klasterdamises

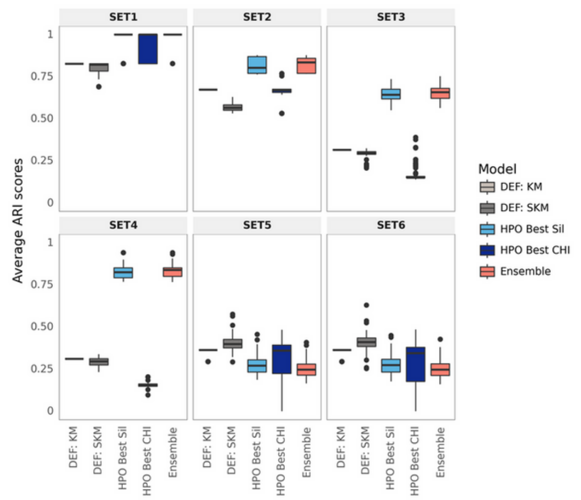
Dmitri Rozgonjuk
Andmeteadus (MSc) 2023
Arvutiteaduste Instituut
Tartu Ülikool
Juhendaja: Radwa El Shawi, PhD

#UniTartuCS

Töövoog


Algoritmid
KMeans
STREAMKMeans
Vaikimisi Hüperparameetrid
Õppimine
Õppimine
Silhouette Skoor
Õppimine
Testi-siis-treeni Hindamine
Õppimine
Testi-siis-treeni Hindamine
Calinski-Harabasz Indeks
Hüperparameetrite Optimeerimine
Parim Mudel (Silhouette)
Parim Mudel (CHI)
Ansambel
Parim Sil/Parim CHI
Hindamine (Kohandatud Randi Indeks)
Mudelite Keskmiste Soorituste Võrdlus

Tulemused



Leiud

- **HPO** (eriti Silhouette-i puhul) **mudelid sooritasid** üldiselt **paremini kui vaikimisi mudelid** k-keskmistega andmestikel
- Mudelite **ansamblite sooritused** olid **võrreldavad parimate Silhouette-optimeeritud mudelitega**
- **Andmete omadused loevad**: HPO oli kasulik k-keskmistega andmestikel, kuid mitte S-kurvidega andmestikel
- Tulemused näitavad, et **HPO võib olla kasulik** - kuid **oleneb ilmselt** sobivate **algoritmide valikust**

Contents

1 Introduction

The amount of data generated by humans and machines is ever-increasing. It has been estimated that by 2025, the amount of data created will exceed 180 zettabytes [64]. The major technological advancements over the past decades, especially those powered by the Internet (e.g., social media, e-commerce platforms, mobile applications, etc.), have been pivotal in this data generation. Additionally, many contemporary applications and technologies, especially those including or relying on sensors (e.g., Internet of Things or IoT devices, autonomous systems such as vehicles, etc.) can produce large volumes of data in real-time. Assuming that there will be more such devices implemented in everyday life in the foreseeable future, the need for solutions for data storage and management, as well as extracting knowledge via analysis, is increasing. Here, the utilization of machine learning (ML) and artificial intelligence (AI) – data-driven frameworks aimed at improving the performance of given tasks [43] – come into play, as these techniques can be used to process the generated data for knowledge extraction, insights, and decision-making.

However, there may be several challenges when using ML. Creating optimal models can be a complex and time-consuming process, as it may involve data preprocessing (e.g., handling missing values, ensuring the right data types for models, scaling the data), feature engineering and extraction (e.g., aggregating features, dimensionality reduction, selecting meaningfully impactful features), as well as choosing the ML algorithm for optimal task performance, tuning the (hyper)parameters in various steps, and deployment [42]. This process is commonly known as constructing an ML pipeline [28], and as can be inferred from the description, may require a high level of expertise from the pipeline engineer.

To that end, over the recent years, the development of automated machine learning (AutoML) systems has been on the rise. The primary task of AutoML systems is to automatically set the hyperparameters of ML pipelines for optimized performance [33]. As discussed in [33], the main use cases for using an AutoML system are to improve the ML pipeline performance, reduce human effort, and make ML pipeline construction more accessible for people with less ML-domain knowledge (the so-called "democratization" of ML), and allowing comparison and benchmarking of ML pipelines (improving reproducibility). Consequently, for example, a data scientist who spent a lot of time on manually tuning an ML pipeline could gain more time for other tasks, while AutoML aims to find an optimal model.

Because data sources are becoming more ubiquitous and faster compared to earlier decades, ML as a field has also been evolving from working with static to handling dynamic data [30]. Yet dynamic data – also known as data streams – introduce additional challenges. In the more traditional ML approach, data is typically processed as a batch, meaning that the entire dataset is analyzed at once. However, because data streams can be generated in real-time at a high rate as a continuous flow, often one instance at a time, ML algorithms would need to be, at least theoretically, updated with every new data instance [13]. This could pose several limitations, such as running into issues like insufficient memory or long model training times. In addition, the distribution of the streaming data could change over time and introduce outlier instances; an ML framework working with these problems should be able to take these possibilities into account. To address these issues, the so-called "online" algorithms have been proposed [35]. In other words, Online ML is designed to work with data streams [48]. An Online ML framework enables continuous model updating with the arrival of new data, typically requires fewer computational resources, and makes the learning process more efficient than when using batch-based ML. Because of this, the framework can be suitable for data applications that require real-time decision-making, as in the Online ML settings, the models are dynamically updated with the most recent data.

Although several AutoML as well as online ML frameworks have been developed over the past years, typically focusing on supervised learning, i.e., classification and regression problems (see *Section 1.1.3*), research on Online AutoML is still scarce, especially for clustering. This may be explained by the complexity of the task, as well as a potentially limited set of (obvious) use cases and application areas. Nevertheless, the ubiquity, volume, and velocity of data generation, and the potential need for knowledge extraction are not ignored, motivating many researchers to invest time, resources, and effort to improve Online AutoML (also in clustering problems).

The aim of this work is to contribute to the research domain of online automated unsupervised ML, specifically, in clustering. While the scope of this thesis does not encompass every aspect of automated online clustering, it offers comprehensive background information on various facets of this approach. The primary objective is to provide context for the thesis and showcase numerous opportunities for expanding upon this work or developing one's own framework within the realm of automated online clustering. In essence, it is hoped that interested readers will find this thesis to be a valuable resource for building their own AutoML systems.

This said, the present thesis is organized as follows: In the *Introduction* section, an overview of AutoML as well as Online ML tasks and characteristics as well as various frameworks is provided. In addition, some potential use cases of Online ML are outlined. The last part of the *Introduction* provides an overview of the scope and the contributions of the thesis, as well as how the solution is planned, and what are the expected results. In the *Methods* section, experiment design and configuration are described, followed by an overview of datasets used for the present study's experiments, the algorithms and their search spaces implemented in this thesis, and a description of how the results of models are compared. In the *Results* section, the main findings are presented, followed by the *Discussion* where the findings are analyzed in the context of the study's goals. The main contributions derived from the findings are discussed alongside limitations and future perspectives. Finally, in the *Conclusions*, the main insights of the present thesis are summarized.

1.1 Automated Machine Learning (AutoML)

Automated machine learning, or AutoML, has been defined as the process in which an ML pipeline – which may include data preprocessing, feature engineering and extraction, model hyperparameter tuning, and selection – is constructed in a data-driven way [28]. In other words, AutoML should minimize the manual work that a person needs to do when constructing such a framework. In turn, this would help to facilitate the democratization of ML, or making a wide variety of ML-related tools, platforms, and data sources more (easily) accessible to more people [15].

1.1.1 Problem Definition

Although the typical tasks of AutoML are described in the next subsection, it is also important to define the main problem that AutoML aims to solve: the *Combined Algorithm Selection and Hyperparameter Optimization*, or the *CASH* problem [22]. This problem can be summarized with the following formula:

$$A^*, \lambda^* = \underset{A_i \in A,\, \lambda_i \in \Lambda}{\operatorname{argmin}} \; L(A_i(\lambda_i), D) \tag{1}$$

In Formula 1, D represents a fixed data set, A is a set of ML algorithms $\{A_1, \ldots, A_m\}$ - in the context of present work, clustering algorithms -, and λ_i represents the hyperparameters of a given algorithm A_i. Altogether, $L(A_i(\lambda_i), D)$ denotes the loss of A_i, given the hyperparameters $\lambda_i \in \Lambda$ on D. Then, as can be seen from the formula, the solution to the CASH problem is to find the joint algorithm and hyperparameter settings that minimize that loss.

The CASH problem can also be adapted for online ML settings (Formula 2).

$$A^*, \lambda^* = \underset{A_i \in A_{OL}}{\operatorname{argmin}} \, L(A_i(\lambda_i, t), D_{t-1}, D_t) \tag{2}$$

Here, A_{OL} denotes the set of online (clustering) algorithms. $A^*_{\lambda,t}$ is continuously trained on the previous data batch D_{t-1} and evaluated on the present data batch D_t. It is worth noting that t here denotes a time window that could either be a batch of data or a continuously arriving number of instances that would trigger training. In the context of clustering, the loss metric is either an internal or external clustering validity index (please see the section 1.2.4 for more details on clustering evaluation).

1.1.2 AutoML tasks

A given AutoML system may aim to optimize a particular task or a set of them. In a review paper [69], eight specific tasks that various developed AutoML frameworks aimed to automate, as well as seven methods to achieve optimally automated task outcomes are outlined. The vast majority of published AutoML systems identified in the review included aiming to solve the CASH problem. In addition, many frameworks also aimed to automate data preparation, feature engineering, and model estimation. Finally, pipeline post-processing and explainability options should also be taken into account. Below is a brief overview of some of the AutoML tasks:

- **Hyperparameter optimization (HPO).** Typically, ML models' hyperparameters need to be set before fitting the model. However, in some cases, the number of hyperparameters, as well as their broad range, can mean spending a lot of time on exploring the optimal hyperparameter configuration. Hence, it may not be surprising that HPO is one of the most explored topics in AutoML research [69]. In the literature, the challenge of finding the best set of hyperparameters for a model is known as the *HPO problem*. There are several techniques for this task [22][1]:
  - *Exhaustive Grid Search* is considered the most basic HPO method, as it specifies a finite set of values for each parameter and the idea is to search through the entire grid of these value combinations. Generally, the advantages of this method are the ease of implementation, taking advantage of parallelism, and the possibility of implementing all parameter types. The limitation of this method is that the search space grows exponentially and that some continuous values need to be discretized. Grid search is an unguided search technique, as there is no reliance on solutions found in previous search runs.
  - *Random Search* is an alternative to grid search where the configurations are sampled randomly from the entire hyperparameter search space. Random search allows for spotting more promising search space areas earlier than grid search. Random search has the same advantages as Grid Search; moreover, random search allows for flexibility in computation budget, i.e., the user of this framework can allocate the computation budget and explore the promising hyperparameter search space areas. The limitation of this technique is poor scalability - when there is a high number of hyperparameters, there are still many samples that need to be covered. Like grid search, random search is an unguided search technique.
  - *Population-based methods* (e.g., genetic and evolutionary algorithms, particle swarm optimization, ant colony optimization, etc.) are optimization algorithms that maintain a set of configurations ("populations") to be improved upon in the next iterations by applying local perturbations ("mutations") and combinations of different optimization model solution parts ("crossover")

[1]The overview of the list of HPO techniques, along with descriptions, is primarily based on [22]. Hence, if other sources are not referenced, it can be assumed that the outlined source was used.

to obtain a "new generation" of better configurations. The advantages are conceptual simplicity, the possibility of all parameter types, and parallelization. However, these methods may introduce difficulty in getting the balance between exploration (discovering promising hyperparameter areas) and exploitation (refining the promising search space) and may not be well-suited for problems like HPO where a high number of evaluations may be needed until appropriate convergence [2].

  - *Bayesian Optimization* (BO) is a global optimization (i.e., finding the optimal solution across the entire search space) framework which is especially useful in optimizing complex and expensive functions. It has two components: (a) a surrogate model and (b) an acquisition function. The former is an approximation of the objective function. The latter refers to a function that is used for guiding the search process; it also aims to balance exploration and exploitation. The main advantages of BO are data-efficient optimization (especially in comparison to grid or random search), balancing exploration and exploitation, implementing prior knowledge from previous searches, and handling noise. The disadvantages are that BO may be computationally expensive (especially if the search space is large) and it does not guarantee finding the global optimal solution.

- **Model Selection and Hyperparameter Tuning.** This is also known as the CASH problem (discussed in section 1.1.1), and it can be viewed as an extension to the HPO problem [22]. In addition to aiming to optimize the parameters of a given model for a given dataset and task, this problem also implies that there are several models which need their hyperparameters to be tuned, and then the best-performing model needs to be selected. In addition to BO and population-based methods mentioned above, the following approaches have been used for solving the CASH problem:

  - *Meta-learning* leverages meta-data on how ML models learn in order to learn new tasks faster [68]. Its general advantages are faster model selection, data efficiency (as it learns from previous experience), and adaptability to new tasks. However, the disadvantages may arise when there is limited information on previous models. Additionally, when the learned tasks do not represent new data well, the learner can produce misleading results.
  - *Portfolio-based methods* approach the algorithm selection problem by first creating a diverse "portfolio" of various algorithm parameter settings, which is then followed by using an algorithm selector to choose the parameter settings that have previously shown good performance [6]. Portfolio-based methods can be generally divided into two subgroups: (a) static and (b)

dynamic portfolios [60]. *Static portfolios* are assembled offline (the models are chosen and fixed before deployment), whereas *dynamic portfolios* are ensemble methods that are adaptive and typically used in online ML. Static portfolios are typically simple to implement and computationally efficient; dynamic portfolios, however, are highly adaptive and suitable for continuous model updates. On the other hand, static portfolios are not as flexible and typically not suitable for adapting to changes, while dynamic models may be complex to implement and could be computationally less efficient.

- **Data pre-processing, feature engineering, and feature selection.** One of the challenges in data analysis is the variety of data types and quality of data [69]. Especially in cases involving real-life data, it is seldom that raw data does not need cleaning, transformations, etc. Here, questions like if (and what kind of) data scaling (method) should be used, how should the missing data be handled, and how to represent categorical variables in the models are highly relevant. Finally, in some cases, not all features in the model are equally informative (i.e., useful). A selection of appropriate features could significantly affect model performance.

- **Other tasks.** AutoML may also aim to implement **post-processing**: for instance, after several candidate models are trained, one may ensemble the best solutions together to improve the total performance [18]. Instead of doing it manually, a data scientist should be able to use automated solutions. Additionally, in some cases, a stakeholder may be interested in acquiring explanations for how the model functions or how it made the predictions [73].

1.1.3 Offline AutoML Frameworks

Below is a (roughly-chronological) overview of various AutoML systems developed until the writing of this thesis (Spring 2023). Each framework is described briefly, and the interested reader has the possibility to get a more in-depth view of a particular framework from the referred publication. It should be noted, however, that this list may not be exhaustive because it is likely that many (new) frameworks are in the development stage, hence, may not be published or at all accessible. While it can be acknowledged that various parties may have or are presently developing proprietary AutoML systems, the present work only covers openly accessible, i.e., open-source, frameworks. In addition, only the frameworks used for classification/regression/clustering tasks are mentioned, excluding frameworks for non-tabular data (i.e., images, audio) as well as deep and reinforcement learning. The interested reader can find some information on these systems in [20]. Several other papers have provided a more in-depth description of the below-described frameworks (for instance, see dedicated chapters in [33] or [20], for a concise overview).

1.1.4 AutoML systems for offline *supervised* ML

In this section, the AutoML frameworks for supervised ML (e.g., classification and regression) are outlined.

- **Auto-Weka** [66]. Written in Java, Auto-WEKA is considered to be the pioneering, first AutoML framework. It implements Bayesian optimization for model selection and hyperparameter tuning.

- **Hyperopt-Sklearn** [36]. Hyperopt-Sklearn implements hyperopt [7] which supports different data pre-processing and regression and classification algorithms, as well as a number of optimization techniques, such as random search and Bayesian optimization techniques.

- **Auto-Sklearn** [23]. In Auto-Sklearn, meta-learning is used for the initialization of combined algorithm selection and hyperparameter tuning. Sequential Model-based Algorithm Configuration (SMAC) is used as a Bayesian optimization technique. Additionally, ensembles are used for improving the performance. Auto-Sklearn has also been further developed to PoSH (POrtfolio Successive Halving) [24], also called as Auto-Sklearn 2.0, where budget allocation as a complementary design choice to model selection was introduced.

- **TPOT** [53]. TPOT (Tree-based Pipeline Optimization Tool) is based on genetic programming and, in essence, it explores various possible pipelines which include data pre-processing, feature engineering, and regression and classification algorithms.

- **RECIPE** [62]. In RECIPE (REsilient ClassifIcation Pipeline Evolution), custom classification pipelines are built from grammar-based genetic programming algorithms (utilizing predefined rules to create structured programs).

- **MLPlan** [44]. In MLPlan, a hierarchical task network algorithm is used, where the search space is depicted as a large tree graph with each node as a goal node of a full pipeline.

- **Auto-MEKAGGP** [61]. Auto-MEKAGGP implements grammar-based genetic programming and is primarily designed for multi-class classification problems. Since the original paper [61], several additions have been developed for the MEKA project [59] for multi-class classification.

- **AutoStacker** [14]. AutoStacker is an ensemble method where different pipelines are combined (as opposed to a single model) to find the best-performing combination. Evolutionary search algorithms are used as the optimization method.

- **`SmartML`** [41]. `SmartML` was the first AutoML package for use in the `R` language and statistical computing environment. Hyperparameter tuning used is based on `SMAC` Bayesian optimization, and meta-learning is also implemented.

- **`Autocompboost`** [16]. `Autocompboost` (Automatic Componentwise Boosting) is a system that can construct an interpretable additive model, which can be fitted using a scalable componentwise boosting algorithm.

- **`GAMA`** [29]. `GAMA` (General Automated Machine learning Assistant) was developed for end-users and AutoML researchers. The tool provides the possibility to generate optimized ML pipelines, which contain data pre-processing, hyperparameter tuning, and classification and regression algorithms.

- **`FLAML`** [70]. `FLAML` does not implement meta-learning or ensembling in order to provide faster computation. It uses Estimated Cost for Improvement (ECI) based sampling of learners, randomized direct search, and ECI-based choice of sample size.

1.1.5 AutoML systems for offline *unsupervised* ML

In this section, the AutoML frameworks for clustering are outlined.

- **`AutoClust`** [58]. `AutoClust` implements meta-learning for algorithm selection, Bayesian optimization for hyperparameter tuning, and the optimization process combines ten internal clustering validity indices (CVIs). In this proprietary framework, two phases can be outlined: (a) the offline learning phase and (b) algorithm selection. In the former, a set of meta-features is extracted for each dataset, different clustering algorithms are applied with various configurations, and the performance (CVIs) of the algorithms is stored. In the algorithm selection phase, the k-nearest neighbor method is implemented to identify the most similar meta-features to the data at hand. The best-performing clustering algorithm is obtained from the meta-knowledge repository.

- **`AutoML4Clust`** [67]. `AutoML4Clust` is an automated clustering system where the hyperparameters of k-center based clustering algorithms are tuned; specifically, only the number of clusters is tuned while using one CVI. Three internal metrics for the evaluation phase are used: Silhouette score, Davies-Bouldin Index (DBI), and Calinski-Harabasz Index (CHI), but they are not ensembled. In the optimization, Random search, Bayesian optimization, Hyperband, and Bayesian Optimization and Hyperband (BOHB) are the options for best model search.

- **`AutoCluster`** [40]. `AutoCluster` leverages meta-feature extraction. This framework implements several clustering techniques (e.g., KMeans and DBSCAN) and allows for iterative clustering solution improvement. In `AutoCluster`, meta-features are extracted, and an ensemble of multiple CVIs (Silhouette score, Davies-Bouldin Index, and Calinski-Harabasz Index) are used.

- **`cSmartML`** [19]. `cSmartML` implements meta-learning for algorithm and evaluation criteria selection in combination with an evolutionary algorithm for hyperparameter tuning and optimization of several CVIs. The framework consists of the following stages: (a) input, (b) algorithm and metric selection, (c) HPO, and (d) output computation and knowledge base update. The algorithm and metric selection consists of two parts: at first, meta-features are extracted, followed by meta-learning recommendation (i.e., which algorithm and evaluation metric are likely to perform well). For optimization, a genetic algorithm (*Non-dominated Sorting Genetic Algorithm II*; NSGA II [17]) is implemented. In order to parallelize the HPO process, hyper-partitioning (subsetting hyperparameter trees with main and conditional hyperparameters) is used. Upon the computation of best solutions, the knowledge base is updated.

- **`TPE-AutoClust`** [21]. `TPE-AutoClust` is an end-to-end automated clustering system that optimizes feature preprocessors and clustering algorithms. It implements meta-learning and an ensemble of CVIs. The process in `TPE-AutoClust` is as follows: first, meta-features are extracted, and the similarity between the new and the previous datasets is computed. In case there are well-performing pipelines in the store, they can be used to warm-start the optimization process where the NSGA-II is used to optimize the model towards multiple internal CVIs (Silhouette score, CHI, DBI). Once best solutions for each CVI are retrieved, a consensus function (majority voting) is used for ensembling the partitions into an optimal data partition.

1.2 Online Machine Learning

The way how ML models are trained and the outputs (i.e., predictions) received could be categorized to *offline* and *online* approaches. The main difference in these processes is how the model learns from data [30]: *Offline ML* (also known as batch learning) uses a fixed dataset to train the model once and typically requires the entire dataset to be available upfront. *Online ML*, also known as streaming (or sometimes: incremental[2]) learning, on the other hand, is characterized by the model learning from a continuous stream of data.

[2]Although one could find literature where "online" and "incremental" learning are treated synonymously, a distinction should be noted. In online learning, the model is updated with each incoming data instance without storing the data, whereas incremental learning involves training the model on *batches* of new data, rather than instance-by-instance [50].

An important factor in online ML is *concept drift* which stands for change in data distribution [71]. Concept drift is a major potential problem in online ML, because it can affect model accuracy. This, in turn, might also increase the number of false positives or false negatives, and this could lead to decreased trust in the model. Additionally, changes in data may require re-training the model to address the accuracy problems, potentially increasing effort for model maintenance.

Generally, the following drift types are differentiated: abrupt, gradual, incremental, and recurrent [63]. *Abrupt drift* means that the data distribution changes suddenly. *Gradual drift* refers to slow and steady change in the underlying data distribution over time. *Incremental concept drift* can be noticed when the data distribution changes over time in small increments. Finally, *recurrent drift* means that the data distribution changes periodically.

In order to take action against concept drift, drift detectors are commonly implemented in online ML. Drift detectors are algorithms that monitor and detect changes in the underlying data distribution during online ML [27]. In supervised ML, where the data has ground truth labels, (one of the) two drift detectors are commonly used: Early Drift Detection Method (EDDM) [3] and ADaptive WINdowing (ADWIN) [9]. The former observes the average distance between classification errors, while the latter monitors the data distribution over a defined window of instances (i.e., the latest batch of data of defined size). EDDM is generally considered to provide better drift detection, especially in gradual drift instances [13].

EDDM is most useful when ground truth is available - this is rarely the case in clustering. For unsupervised drift detection, the Page-Hinkley test can be used. It is a change-point detection technique that can be advantageous in situations where ground truth labels are unavailable or unreliable [49]. Unlike ADWIN or EDDM, which rely on classification errors or data windows, the Page-Hinkley test is based on analyzing the data distribution itself. By detecting shifts in the mean of the underlying data, the test allows clustering algorithms to adapt their models in response to changes in data distribution.

Another key characteristic of online learning is *prequential evaluation*, or interleaved test-then-train evaluation [26]. With each data instance, the model first makes a prediction, evaluates the prediction (e.g., by an error measure), and then updates the model. The advantage of this approach is that the model is always tested on unseen data (similar to test data in offline ML), it is computationally efficient, and is useful in drift detection, i.e., changes in error distributions between the predicted and ground truth values can help with timely adaptation when drift is present (in supervised online ML).

1.2.1 Use Cases of Online ML

As the nature of online ML implies, it is highly suitable for situations where there is a continuous stream of (real-time) data which would make the (re-)training of the original (offline) ML model not feasible. In addition, online ML models can adapt to changes in data distribution which can occur in real-time. To this end, online ML provides several application opportunities. Below are some of the examples - certainly not an exhaustive list of applications, but it should provide insights into the potential of online ML.

- *Anomaly, outlier, and fraud detection* [5, 37]: online clustering can be applied for detecting abnormal data instances that do not fit into identified clusters in contexts like financial transactions (detecting potential fraud), or network activity.
- *Customer segmentation* [11]: customer segments can be very volatile, and online clustering could be used to adapt to these changes in real-time.
- *(Human) Movement patterns* [38]: clustering can help with identifying different movement patterns in humans (can also be used for anomalous movement pattern detection), potentially in real-time applications.
- *Traffic congestion estimation* [45, 56]: online clustering could help with tracking real-time network traffic to, e.g., identify congested areas in traffic.

1.2.2 Online ML frameworks

- **`MOA`** (Massive Online Analysis) [10]. It is a framework designed for online learning from data streams, specifically for classification and clustering.
- **`Vowpal Wabbit`** [39]. `Vowpal Wabbit` has several ML techniques, including online, hashing, allreduce, reductions, learning2search, active, and interactive learning.
- **`River`** [47]. `River` is an online AutoML framework that combines the previously-developed `Scikit-multiflow` [48] and `creme` [31] libraries. It includes methods for supervised (regression, classification) as well as unsupervised (clustering) learning. It is built to take into account online ML specifics, such as concept drift and model evaluation in online learning settings. `River` is also designed to be resource-efficient and user-friendly. Of note, `River` is the first open-source project that includes online clustering [46]. The algorithms and metrics of relevance to the present work are discussed in sections 2.1.2 and 2.1.3.

1.2.3 AutoML systems for online *supervised* ML

By the time of writing the present work, there is a limited number of AutoML systems that have been developed for online ML. In the frameworks (listed below), there are presently no modules for unsupervised ML – though they do typically support regression and classification-based algorithms and approaches.

- **`Cha-Cha`** (`Champion-Challengers`) [72]. The `Champion-Challengers`, or `Cha-Cha`, framework aims to balance computational effort by categorizing choices based on their learning cost and assigning resources only to the most promising algrorithm search spaces. `Cha-Cha` considers one base learning algorithm at a time and tries to find promising hyperparameter settings for the algorithm. The main idea behind `Cha-Cha` is as follows: a data instance is used as input to the "live" model pool (collection of model variants, the pool size is defined by the user). The instances are then predicted and compared to ground truth, starting with the "Champion" model (i.e., the model with the best performance). In case the model performance is significantly worse than the performances of other models in the pool (the "Challengers") based on a statistical test, the better model is "promoted" to be the "Champion", and its search space is further expanded based on the pre-defined resource budget (e.g., run time). At any given time point, the number of models in the "live" pool is constant (e.g., k = 5); this means that as soon as better models are found, the model pool is updated so that the worse models are excluded from the pool.

  The architecture of `Cha-Cha` is relatively complex, and the main limitation of this framework is that although it aims to solve the HPO problem on the go, it is not capable of solving the CASH problem, as `Cha-Cha` does not support optimization of complete ML pipelines (with pre-processing), because it does not explore the large space of learning algorithms.

- **`Online AutoML`** (`OAML`) [13]. The `Online AutoML`, or `OAML`, is a framework that aims to combine AutoML with the adaptability of online learning algorithms. The search space includes online learning algorithms, ensembling methods, and preprocessors. `OAML` simultaneously selects and optimizes hyperparameters by using optimization techniques, such as an asynchronous evolutionary algorithm and asynchronous successive halving. Additionally, the system includes backup ensembles and model stores to counter concept drift.

  The system itself relies on both offline and online learning phases. In the offline phase, the best pipeline is searched based on the allocated resource budget (e.g., runtime) on a batch of streaming data. Once the best pipeline is found, it is fitted on the available data and passed to the online learning phase. In the online learning part, data instances are assumed to arrive one at a time. Here, the prequential evaluation approach is used. By using the performance of the model per each instance, concept drift detection is implemented. If no drift is detected, the model continues instance-by-instance learning; however, in case concept drift is detected, the entire learning process goes back to the offline phase (i.e., searching for the best pipeline).

  `OAML` supports three methods to update the old model: basic, ensemble, and model store approach. The basic method is suitable for abrupt concept drift, and in this case, the old model is completely disregarded, and a new one is built from scratch. In the ensemble approach, when concept drift occurs, the old pipeline is compared to a backup ensemble of pipelines based on predictive values of the sliding window. If the ensemble is better, it replaces the old pipeline. Finally, the model store refers to keeping a user-defined number of individual best pipelines in memory. In essence, it is a history of the best pipelines.

In their work [13], the authors demonstrated that there is no single approach that outperforms others in all cases. For instance, the ensemble approach performed consistently well with various kinds of concept drift, while using a model store seemed to perform the best with recurrent concept drift.

1.2.4 AutoML systems for online *unsupervised* ML

Developing automated online ML systems for unsupervised learning is challenging. In addition to concept drift in online learning, unsupervised learning settings typically lack the ground truth labels that are needed for error measurement. To address this, several works (please see section 1.1.5) rely on (a combination of) clustering validity indices. There are generally two types of validation approaches [32]: *external validation* refers to using previous knowledge about the data (e.g., having ground truth labels), whereas *internal validation* is based only on the intrinsic information retrieved from the data.

Clustering validity indices generally rely on the combination of compactness and separability [32]. *Compactness* refers to the closeness of cluster elements, and it is commonly assessed with variance. *Separability*, on the other hand, aims to depict how distinct two given clusters are; typically, the distance between two different clusters is used as the measure. Based on these concepts, there is a large number of various internal and external CVIs. Below is a list of some commonly used *internal* CVIs [32]:

- *Silhouette Score*: This index evaluates the similarity within clusters and the dissimilarity between clusters. Typically, the score ranges between [-1; 1] with higher scores indicating that the data instance fits the cluster better. However, in the `river` implementation, a Silhouette score of 0 indicates a better fit.

- *Calinski-Harabasz Index* (CHI): This index assesses the ratio of between-cluster dispersion to within-cluster dispersion. Larger values indicate better cluster definitions.

- *Davies-Bouldin Index* (DBI): This metric calculates the average similarity between clusters, aiming to identify sets of clusters that are compact and well-separated. Lower values indicate a better clustering solution.

- *Dunn Index*: This index assesses the proportion of the smallest inter-cluster distance to the largest intra-cluster distance. Larger values correspond to a good clustering solution.

As mentioned, it is rarely the case that there are ground truth labels in real-life datasets. Hence, researchers have either used some commonly known real-life datasets with ground truth labels (usually from multi-classification problems) or synthetic data (e.g., see [21] or [12], for examples). In these cases, *external* CVIs are used. In essence, these indices compare the clustering assignments to true labels to evaluate how well a given clustering algorithm maps the patterns of the data. Below is a list of commonly used *external* CVIs [32]:

- *Adjusted Rand Index* (ARI): By considering all data points, ARI assesses the agreement between two clusterings. The range of this index is between [-1; 1], and 1 indicates perfect agreement.

- *Adjusted Normalized Mutual Information* (ANMI): This index assesses the relationship between two clustering assignments while accounting for chance and normalizing cluster sizes. The range of values is [0; 1]; 1 indicates a perfect solution.

- *Purity*: This metric shows how much a single cluster contains data instances from a single class. The range for Purity is [0; 1], with 1 reflecting perfect purity.

- *Jaccard Index*: This index evaluates the overlap between two clusterings, dividing the number of shared instances by the total number of unique clusters. The values are in the range [0; 1], with higher scores indicating a better clustering solution.

- *Fowlkes-Mallows Index* (FMI): This index computes the geometric mean of precision and recall and uses this to assess the similarity between two clusterings. The values are in the range [0; 1], with higher scores indicating a better clustering solution.

To the best of the knowledge of the author of this thesis at the time of executing this project and writing the thesis, there has been one proposed framework to address automated algorithm selection and configuration in clustering with evolving data streams. This framework is called `ConfStream` and it is the first proof-of-concept for online clustering with AutoML properties [4, 12]. Implemented in `Java` and based on the `MOA` framework [10], `ConfStream` evaluates different configurations of available online clustering algorithms, selects the best configurations, and ensembles them into the optimal clusterer for online learning. Periodically, the ensemble is updated with a better-performing ensemble if one exists. One potential limitation of this framework is that it evaluates models using only the Silhouette score and does not consider a combination of multiple CVIs. Relying on a single metric in online clustering settings may limit the understanding of clustering structure, especially regarding sensitivity to noise, outliers, and cluster characteristics. Therefore, implementing multiple CVIs could provide a more comprehensive clustering solution [21, 67]. Another limitation of `ConfStream` is relying on a single algorithm when a promising algorithm is found from the model pool. However, a single algorithm may not be suitable for all types of data, hence, an ensemble of best algorithms could have a higher potential for a better clustering solution [21].

1.3 Present Work

1.3.1 Scope and Contributions of the Present Thesis

The present work is part of a larger project aimed at developing an automated online clustering framework that addresses the CASH problem in online clustering settings. In this thesis, the main tasks are as follows:

(a) Building a workflow for automatically optimizing the hyperparameters of a given clustering algorithm for a specific internal CVI.

(b) Finding a solution for ensembling the best-optimized online clustering models.

(c) Providing preliminary performance comparison results by contrasting the external CVI scores of:

   (i) Algorithms with default hyperparameter values,

   (ii) Optimized algorithms,

   (iii) Ensemble of the best-optimized algorithms.

1.3.2 Proposed Solution

While an AutoML framework can encompass both offline and online phases, the main focus of this thesis is on the offline part, where models are trained and optimized based on internal CVIs. In the online phase, the models are evaluated using an external CVI. The general workflow of the present work is depicted in Figure 1.

Importantly, for both the `Learning` and `Prequential Evaluation` parts of the process, $N = 1001$ data instances were used both in the learning and prequential evaluation phases (similarly to [13]). The rest of the data (i.e., $N = 10000 - 1001 - 1001 = 7998$ instances; more details on datasets are provided in Table 1) were used for the model evaluation part (computing ARI scores).

As shown in Figure 1, the framework includes two online clustering algorithms. Specifically, there are two separate workflows: one for default models and another for models with optimized hyperparameters. In the case of (a) default models, they are

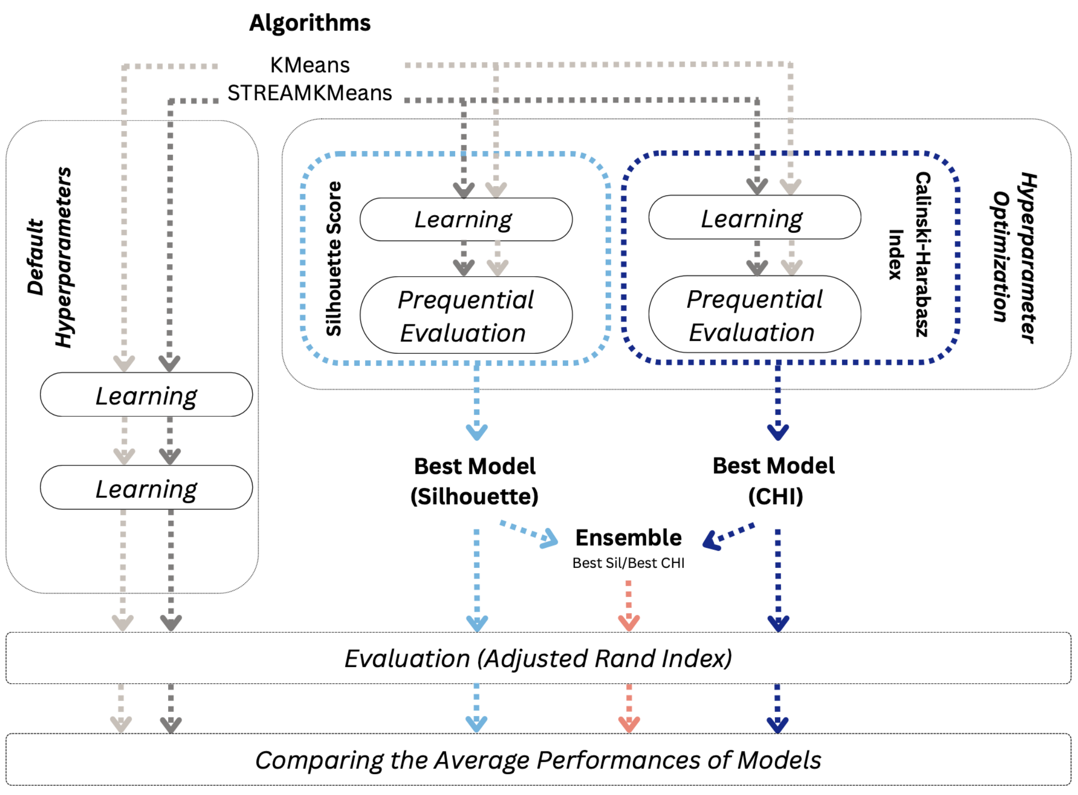


Figure 1. The general workflow overview: model training, optimization and model selection, followed by online evaluation and analytical comparison of results.

applied for online learning using a specific number of samples corresponding to the learning and prequential evaluation parts. In the case of (b) models with optimized hyperparameters, both algorithms are used to optimize the hyperparameters based on an internal CVI (either Silhouette score or CHI). The workflow includes a learning phase, followed by prequential evaluation where clustering labels are predicted, the metric (internal CVI) is updated, and the model is updated accordingly. Once the best scores are computed for each algorithm and metric, the best models for the given internal CVIs are selected based on the last metric value. Additionally, an ensemble clusterer is built based on these best models. Subsequently, all models (defaults, best optimized algorithms, and ensemble) are evaluated against an external CVI (Adjusted Rand Index) with the remaining data, and the results are saved. Finally, these results are compared via statistical testing.

1.3.3 Hypotheses

To test the effectiveness of the proposed solution, hypothesis testing is conducted. The hypotheses of this work are as follows:

- **Hypothesis 1**: *Best optimized models perform better than models with default hyperparameters.*
- **Hypothesis 2**: *The ensemble model performs better than models with default hyperparameters.*
- **Hypothesis 3**: *The ensemble model performs better than the individual best optimized models.*

The proposed framework will be implemented on each dataset used in this thesis for N = 100 iterations. The average external CVI (ARI) scores will be calculated for each model on each iteration; the goal of the model is to produce the highest ARI scores possible. Since the 100 runs will be performed on the same dataset, Friedman's test is appropriate for testing the differences in averages among models that are dependent on the same data. Additionally, Nemenyi's post-hoc tests will be used to determine which models, if any, differ from each other statistically significantly.

2 Methods

Below, the experiment design, data set, and other configurations are presented.

In accordance with the Master's thesis guidelines, I acknowledge utilizing contemporary tools during the writing of the thesis and development of the framework and experiments. Specifically, I made use of ChatGPT [54] for various tasks such as spell and grammar checking, bug fixes, code refactoring and formatting (e.g., docstrings), and providing conceptual and technical explanations.

2.1 Experiment Design

In this section, the performance of the automated online clustering framework is evaluated by using data streams in order to provide an overview of its capabilities. The code and details of these experiments can be found in the associated GitHub repository [3].

2.1.1 Data

In order to simulate an online learning context, synthetic datasets were created by implementing instance-by-instance learning, where each data row is processed one at a time. For this purpose, modified versions of the `make_blobs()` and `make_s_curve()` functions from the `sklearn.datasets` module [57] were utilized. The "blobs" datasets consist of groups of normally distributed points in a multi-dimensional space with k-centers, while the "S-curves" datasets exhibit a non-linear structure where data points form an S-shape in a three-dimensional space. For this thesis, I created the following datasets, aiming to vary the number of clusters and features and include both k-center-based ("blobs") and non-linear ("S-curves") datasets. The dataset descriptions can be found in Table 1.

Table 1. Properties of data sets

Name	Type	N samples	N Features	N Clusters
SET1	Blobs	10000	10	6
SET2	Blobs	10000	3	8
SET3	Blobs	10000	3	19
SET4	Blobs	10000	10	19
SET5	S-curves	10000	3	3
SET6	S-curves	10000	3	8

Notes: For SETS 5 and 6, the `N Clusters` column refers to the number of S-curves.

[3]Project code: https://github.com/qetdr/online-autoclust-hpo

2.1.2 Configuration

Below, the configuration of the framework used in the present study is described.

Online ML framework: `river v0.14.0` was used for online ML in the present work, as described in section 1.2.2. This choice was based on its active development, open-source nature, user-friendliness, and its previous application in supervised online AutoML system development [13].

Optimization configuration: `Hyperopt v0.2.7` [7] was employed for hyperparameter optimization. It is a `Python` library that can perform complex hyperparameter optimization for virtually any `Python` model (given the hyperparameter search space). Several optimization techniques can be implemented within this framework, such as random search, grid search, and Tree Parzen Estimation (TPE). The main advantages of `hyperopt` are compatibility with a range of models (in `Python`) and it is open-source. With regards to the configuration in the present project, TPE was used as the optimization technique. TPE is an efficient technique that converges faster than grid and random search [8]. In each instance of model optimization, the allocated budget was 50 trials per model; upon the depletion of this allocated budget, the best model was retrieved.

Internal CVIs (model optimization): The Silhouette score and Calinski-Harabasz Index (CHI) were implemented for internal CVI evaluation. These metrics are commonly used in clustering research and have demonstrated superior performance compared to other internal CVIs [1]. The Silhouette score and CHI are implemented as (`Silhouette()` and `CalinskiHarabasz()` from the `river` and `river-extra` libraries, respectively). [4]

External CVI (model evaluation): The Adjusted Rand Index (ARI) was used as the external CVI for model evaluation. ARI is a widely used metric in clustering evaluation works [65]. It offers interpretability, chance correction, insensitivity to label permutations and cluster sizes, and suitability for scenarios where clusters may overlap.

The number of experiments: Each dataset was evaluated in $N = 100$ experiments. Each experiment involved running the default models, optimizing two models based on Silhouette and CHI scores, and creating an ensemble model from the best optimized models. Subsequently, all five models were evaluated using ARI. The ARI scores from each iteration were recorded and aggregated to calculate the average ARI score for each model. The experiments were parallelized using the `multiprocessing` library for improved efficiency.

Hardware and operating system. The present MSc thesis experiments were conducted with a 2020 Apple MacBook Pro (macOS Ventura 13.2.1) an Apple M1 chip

[4]It should be noted that, typically, a third CVI is commonly used alongside the Silhouette Score and CHI: the Davies-Bouldin Index. However, in the present implementation (in the `river-extra v. 0.14.0` library), the metric is experiencing unexpected behavior, leading to errors. Hence, this metric was excluded from use in the previous MSc thesis but will be included in the future in the larger project once the bug fixes are applied.

(8-core CPU) and 16 GB RAM, using Python 3.10.9.

2.1.3 Baselines and Search Space

Below, the algorithms used in the present project are described. Additional details about these algorithms can be found in the provided links. [5]

- `KMeans`[6]: This algorithm is an online version of the traditional KMeans clustering algorithm. It iteratively partitions data points into k clusters based on the mean of the data points within each cluster. The implementation in the `river` library adopts a mini-batch approach, updating the cluster centroids incrementally using small batches of data points instead of the entire dataset.

- `STREAMKMeans`[7]: This algorithm serves as an alternative to the STREAMLSEARCH algorithm [52]. STREAMKMeans operates on temporary data chunks comprising new data instances. Once a temporary data chunk reaches its maximum size, KMeans processes the chunk and derives the new cluster centers. When a prediction request is made, the algorithm utilizes the centers retrieved at the time of prediction retrieval.

Table 2 provides comprehensive information about the default parameter values, search spaces, and explanations of the clustering algorithms.

Regarding the baselines, the primary focus of this work is to compare the performance of the optimized solutions against that of the default clustering algorithms. The default values for the online clustering algorithms can be found in the *Default* column of Table 2.

2.1.4 Statistical Analysis for Model Comparisons

Each experiment run provides an average ARI score across all evaluation timepoints, resulting in a total of 100 scores for each data set and model. To compare the performance differences between models, the Friedman test [25] was employed, considering the same underlying data and ensuring result robustness. Additionally, Nemenyi post-hoc tests [51] were conducted to analyze pairwise model differences. These tests were performed separately for each dataset, with a significance level of 0.05.

[5] Although there is potential for implementing more algorithms and expanding the search spaces, the scope of the present work is limited in terms of the number of clustering algorithms and their associated search spaces. This limitation arises from the fact that during the development of the framework for this study, certain algorithms in the `river` library behaved in an unexpected manner and exhibited errors that were hard to trace. The OAML paper [13] also acknowledges that while the `river` library has great potential for developing online ML systems, it presently contains bugs that prevent the full utilization of its larger capabilities.

[6] https://riverml.xyz/dev/api/cluster/KMeans/

[7] https://riverml.xyz/dev/api/cluster/STREAMKMeans/

Table 2. Hyperparameter search spaces for given algorithms in the present work.

Algorithm	Hyperparameter	Search Space	Default	Explanation
KMeans	n_clusters	[3; 20]	5	Maximum number of clusters to assign.
	halflife	[0.01; 1]	0.5	Amount by which to move the cluster centers, a reasonable value if between 0 and 1.
	mu	[0.01; 2.0]	0	Mean of the normal distribution used to instantiate cluster positions.
	sigma	[0.01; 2.0]	1	Standard deviation of the normal distribution used to instantiate cluster positions.
	p	[1, 2]	2	Power parameter for the Minkowski metric. When p=1, this corresponds to the Manhattan distance, while p=2 corresponds to the Euclidean distance.
	seed	-	None	Random seed used for generating initial centroid positions.
STREAMKMeans	chunk_size	[5; 49]	10	Maximum size allowed for the temporary data chunk.
	n_clusters	[3; 20]	2	Number of clusters generated by the algorithm.
	halflife	[0.01; 1]	0.5	Amount by which to move the cluster centers, a reasonable value if between 0 and 1.
	mu	[0.01; 2.0]	0	Mean of the normal distribution used to instantiate cluster positions.
	sigma	[0.01; 2.0]	1	Standard deviation of the normal distribution used to instantiate cluster positions.
	p	[1, 2]	2	Power parameter for the Minkowski metric. When p=1, this corresponds to the Manhattan distance, while p=2 corresponds to the Euclidean distance.
	seed	-	None	Random seed used for generating initial centroid positions.

3 Results

The results for the data sets are presented below. Friedman's tests were conducted to investigate whether there were significant differences in model performance (operationalized as the average ARI score) among the models for each data set. Furthermore, Nemenyi's post hoc tests were performed to examine specific pairwise model differences. The total computation time for all experiments, using the configuration described in the *Methods* section, was approximately 8 hours and 18 minutes.

The distribution of model performances across all data sets is illustrated in boxplots in Figure 2.

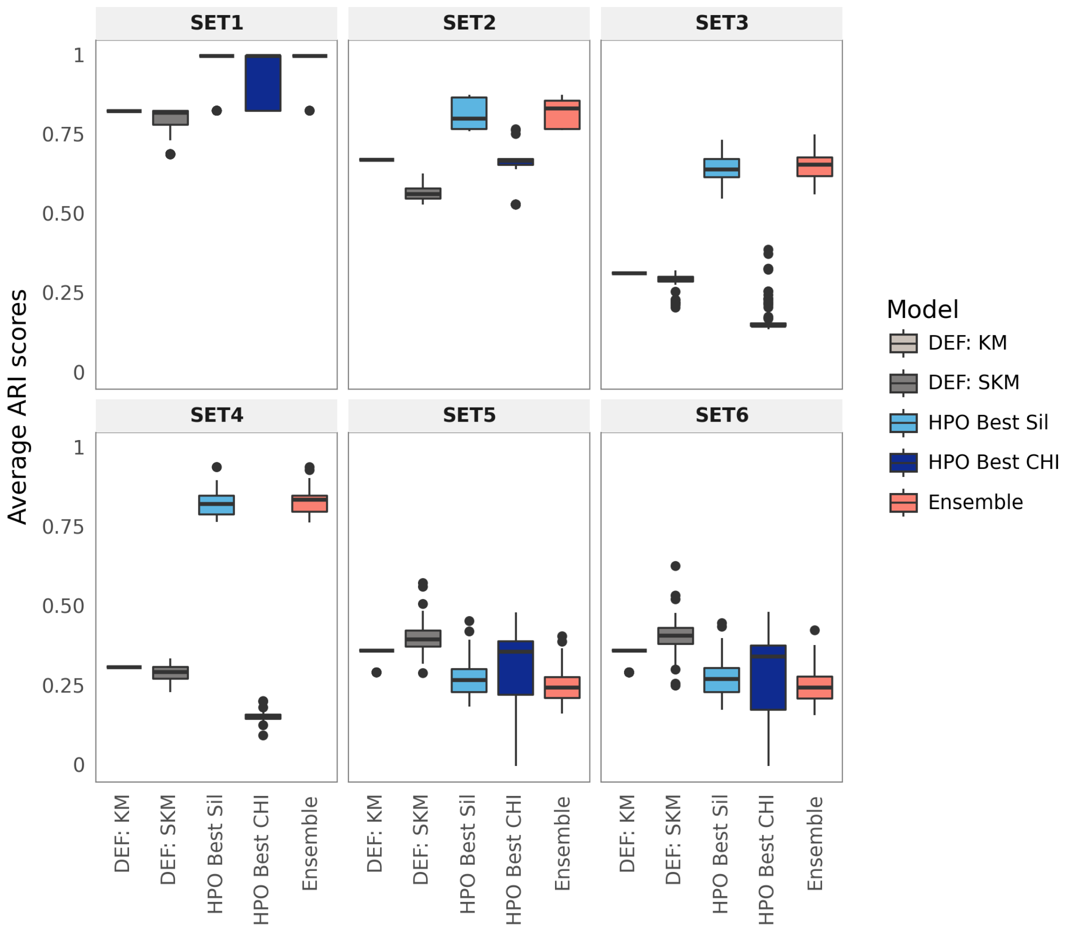


Figure 2. The model evaluation results (average ARI scores) for each clustering solution across each data set (SET1-SET6). *Notes. DEF* = model with default hyperparameters; *HPO* = best optimized model; *Sil* = Silhouette Score, *CHI* = Calinski-Harabasz Index.

Several observations can be made from Figure 2. Firstly, there is no single model that consistently outperforms all other models across all data sets. Secondly, the results suggest the relevance of the data set itself. In Sets 1 to 4, the model optimized towards the Silhouette score and the ensemble model tend to perform better than other models, while this is not the case for Sets 5 and 6. The former sets consist of k-center-based "blobs" data, while the latter sets are three-dimensional S-curves with varying numbers of these curves. Thirdly, the model optimized towards the Silhouette score and the ensemble model appear to perform similarly overall. Fourthly, the model optimized towards the best CHI score tends to perform relatively poorly and is sometimes outperformed by the default models.

In addition to visual inspection, Friedman tests with Nemenyi post-hoc tests were conducted to determine the statistical significance of the observed differences between models for each data set. The results of the Friedman tests are presented in Table 3, indicating statistically significant model performance differences in all case

Table 3. Results of the Friedman test in each data set (N = 100).

Dataset	**Friedman χ^2**	**df**	**p**
SET1	329.121	4	$< .001$
SET2	354.000	4	$< .001$
SET3	367.976	4	$< .001$
SET4	368.538	4	$< .001$
SET5	229.457	4	$< .001$
SET6	214.54	4	$< .001$

To identify pairwise differences between models for each data set, Nemenyi post-hoc tests were performed, and the results are presented in Table 4. Notably, as observed in Figure 2, the best model optimized towards the Silhouette score and the ensemble model outperform other clustering solutions in Sets 1 to 4. Although the ensemble models have slightly higher average ARI scores than the Silhouette-optimized models, the post-hoc test results indicate that the differences in model performances are not statistically significant. In other words, for "blobs"-type data sets, comparable results can be achieved by optimizing a model towards the best Silhouette score and when ensembling it with the CHI-optimization based model.

Interestingly, in the "S-curves" data sets (Sets 5 to 6), which have notably lower average ARI scores compared to "blobs" data sets, ensembling the best hyperparameter-optimized models yielded the poorest performance. In fact, the `STREAMKMeans` algorithm with default values outperformed all other models in the "S-curves" data sets.

Table 4. Averages (M) and standard deviations (SD) for ARI scores for each model in each data set across 100 experiments, and p-values for Nemenyi post-hoc test results.

Dataset	**Model**	**M (ARI)**	**SD (ARI)**	**Post-hoc p-values**			
				1	**2**	**3**	**4**
SET1	**1**. DEF: KM	.828	.000	1			
	2. DEF: SKM	.801	.040	**.015**	1		
	3. HPO: BEST SIL	.986	.047	**.001**	**.001**	1	
	4. HPO: BEST CHI	.938	.083	**.001**	**.001**	**.047**	1
	5. ENSEMBLE	.995	.029	**.001**	**.001**	.900	**.010**
SET2	**1**. DEF: KM	.673	.000	1			
	2. DEF: SKM	.568	.022	**.001**	1		
	3. HPO: BEST SIL	.816	.048	**.001**	**.001**	1	
	4. HPO: BEST CHI	.659	.042	.900	**.001**	**.001**	1
	5. ENSEMBLE	.825	.039	**.001**	**.001**	.796	**.001**
SET3	**1**. DEF: KM	.316	.00	1			
	2. DEF: SKM	.293	.024	**.003**	1		
	3. HPO: BEST SIL	.644	.041	**.001**	**.001**	1	
	4. HPO: BEST CHI	.171	.050	**.001**	**.001**	**.001**	1
	5. ENSEMBLE	.655	.044	**.001**	**.001**	.263	**.001**
SET4	**1**. DEF: KM	.311	.000	1			
	2. DEF: SKM	.294	.023	.239	1		
	3. HPO: BEST SIL	.821	.038	**.001**	**.001**	1	
	4. HPO: BEST CHI	.155	.013	**.001**	**.001**	**.001**	1
	5. ENSEMBLE	.834	.038	**.001**	**.001**	.381	**.001**
SET5	**1**. DEF: KM	.361	.015	1			
	2. DEF: SKM	.404	.044	**.001**	1		
	3. HPO: BEST SIL	.276	.053	**.001**	**.001**	1	
	4. HPO: BEST CHI	.295	.145	.542	**.001**	**.003**	1
	5. ENSEMBLE	.253	.050	**.001**	**.001**	**.001**	**.001**
SET6	**1**. DEF: KM	.359	.019	1			
	2. DEF: SKM	.409	.052	**.001**	1		
	3. HPO: BEST SIL	.279	.057	**.001**	**.001**	1	
	4. HPO: BEST CHI	.289	.136	**.015**	**.001**	.477	1
	5. ENSEMBLE	.253	.052	**.001**	**.001**	**.001**	**.001**

Notes: ARI = Adjusted Rand Index; DEF: KM = default KMeans model; DEF: SKM = default STREAMKMeans model; HPO: BEST SIL = best model optimized for Silhouette score; HPO: BEST CHI = best model optimized for Calinski-Harabasz Index; ENSEMBLE = ensemble model of the best optimized models. Statistically significant p-values ($p < .05$) are highlighted in **bold** font. The average ARI scores for best models for each data set are underlined.

4 Discussion

The main aim of the present thesis was to develop the hyperparameter optimization part for AutoML research in online clustering settings. In this regard, the focus of the thesis was primarily on the AutoML (offline) phase, specifically aiming to develop a framework for automated hyperparameter optimization (HPO) and model selection. To achieve this, the ARI scores of default models were compared against the ARI scores of best optimized models (based on the Silhouette score and CHI) and the ARI scores of the ensemble of the named best optimized models. The key findings, implementation details, and limitations and future perspectives are discussed below.

4.1 Main Findings

In the first part of the thesis, three hypotheses were posed. Overall, it can be said that none of the hypotheses received *full* support from the data. However, as discussed below, certain circumstances may be necessary for the hypotheses to be accepted.

According to Hypothesis 1, it was expected that the models with automatically optimized hyperparameters would outperform the default clustering algorithms. This was indeed the case in three out of six data sets (SETS 1, 3, and 4). However, in SET 2, only the Silhouette-optimized model outperformed all the default algorithms, while there was no significant difference in performance between the model optimized towards optimal CHI and the default KMeans. Surprisingly, the optimized models performed *worse* than the default online clustering algorithms in two data sets (SETS 5 and 6).

For Hypothesis 2, it was expected that the ensemble model would outperform the default algorithms. The findings were similar to Hypothesis 1, with the ensemble outperforming the default configurations in four out of six data sets (SETS 1 to 4). However, it should be noted that in SETS 5 and 6, the ensemble model was surpassed by the default configurations.

Finally, according to Hypothesis 3, it was expected that the ensemble clusterer formed from the best optimized models would outperform both of these individual models. Here, the ensemble only outperformed the models optimized towards optimal CHI in SETS 1 to 4. However, the ensemble models were not statistically significantly better-performing than the Silhouette-optimized models. This suggests that the ensemble may have been more influenced by the best Silhouette model, incorporating primarily its clustering solution. In the future, experimenting with model weights, i.e., determining the contribution of each model to the ensemble formation, could be explored. Interestingly, the ARI scores for ensembles were the poorest in the "S-curves" data sets.

Based on the findings, it can be inferred that HPO can be leveraged but within certain limitations that include the suitability of specific algorithms for particular data types. In general, the performance was better for data sets with distinct centers, such as the "blobs" data sets. In contrast, for non-linear manifold data structures like the "S-curves," the performance of all models, including HPO, was poorer compared to the default models. One potential reason for these findings may be that the "blobs" data sets are well-separated, roughly spherical, and have similar densities, which align well with algorithms like KMeans and STREAMKMeans, as these algorithms are designed to handle clearly-separable linear data with relatively clear boundaries and shape [34]. Hence, a potential extension of these experiments could involve including other algorithms, such as DBSCAN, to handle non-linear data more effectively [55].

4.2 Contribution

This thesis has both practical and theoretical merit. From the theoretical perspective, the main contribution is providing insights into the possible benefits of automated HPO in online clustering (by implementing the `hyperopt` framework). In this study, default online clustering algorithms were compared against their optimized counterparts, and an ensemble of the latter. The results demonstrated that while optimization may help to improve the performance of the default algorithms, this may be so in particular data and algorithm types.

In terms of practical contributions, the thesis provides a code repository that can be valuable for advancing HPO in the context of online clustering. The repository allows for potential users to explore and modify configurations, such as incorporating additional algorithms or expanding the search spaces. By automating the process, the framework developed in this thesis can save time and effort for users.

Furthermore, the thesis offers a broader overview of AutoML and online machine learning. It serves as a starting point for interested readers to delve into additional learning resources and explore the capabilities of AutoML and online ML beyond the scope of this study.

4.3 Limitations and Future Perspective

The framework proposed in this thesis has a limited scope, and there are several opportunities for further improvement and development. Here are some potential ways to enhance the automated online clustering framework:

Expanding the search space: The current framework can be extended by incorporating a wider range of algorithms, hyperparameter configurations, and search duration (e.g., number of trials in `hyperopt`). Algorithms specifically designed for evolving data streams should be explored and made compatible with the framework. Additionally, the framework should consider efficient optimization techniques and explore ensembling methods suitable for algorithms with temporal solutions, such as those utilizing micro-clusters. Furthermore, distribution-based clustering algorithms can be investigated for inclusion.

Expanding capabilities in the online clustering pipeline: The framework should be enhanced to include data pre-processing steps. Data scaling, handling missing data, and transformations should be considered as integral parts of an automated online clustering workflow.

Alternative optimization and model selection techniques: While the present work utilized the `hyperopt` library for optimization, alternative methods like genetic programming-based approaches (e.g., `GAMA`) can be explored. The inclusion of a class like `ClustGAMA` (as a reference to the presently-implemented `ClustHyperopt`) in the optimization codebase could be considered to provide additional optimization capabilities.

Handling concept drift: Although handling concept drift is an important part in online ML, dealing with concept drift was not within the scope of the present thesis. Solutions for adapting to frequent and recurrent drifts can be explored, as well as considering the necessity and utility of explicit drift detection mechanisms. Recent work on drift adaptation in online clustering can provide valuable insights in this area [74].

Improving scalability: Although the ultimate goal of this framework would be usability in real-time (or with minimal lag) context, the present solution is not quite there yet. The model training part may take a considerable amount of time and is not, hence, suitable for real-time applications. The present framework could be improved in later iterations by implementing the use of continuous sliding window batch training outlined in Formula 2. In addition, presently, the ensembling part of the workflow is time-consuming. Future work could also improve the present framework by including a more efficient, faster ensembling method.

Different data types: the present work only utilized synthetic datasets generated with the `sklearn` module for basic functionality comparison. Although the larger project aims to include more complex datasets, including real-world data, additional work may be necessary to provide a compatible framework. It may be fruitful to include additional data sets that differ in distribution. Additionally, when it comes to drift detection, it may be also a good idea to test the framework by including drift in the data.

It is important to note that addressing all of these aspects is beyond the scope of an MSc thesis. The present work might serve as a preliminary foundation and takes steps towards developing an automated online clustering framework, but future iterations and research are required to fully realize the potential of an automated online clustering framework.

5 Conclusions

The aim of the present work was to contribute to the field of online automated machine learning by developing an automated hyperparameter optimization and model selection solution in online clustering settings. The experiments focused on two algorithms, namely KMeans and STREAMKMeans. The performance of the default algorithms was compared against the best-optimized models, where one model was selected based on the optimal Silhouette score and the other based on the optimal CHI. Additionally, an ensemble clusterer was created from these best models. A total of five clustering solutions were evaluated on six different data sets, four of which were variations of spheric "blobs" with clear cluster boundaries, and two data sets were three-dimensional "S-curves" (with a varying number of "S-curves").

The results of the study showed that the Silhouette-optimized model and the ensemble clusterer generally outperformed the other models in the "blobs" data sets. Importantly, there was no statistically significant difference in performance between these two clustering solutions. On the other hand, the CHI-optimized models did not demonstrate particularly strong performance. Interestingly, the default STREAMKMeans algorithm outperformed all other models in the "S-curves" data sets, while the ensemble clusterers had the lowest performance among all the models in those data sets.

These findings highlight the potential for automated hyperparameter optimization in online clustering, although the effectiveness may depend on the choice of algorithm, internal clustering validity index (loss metric), and data set characteristics.

6 Acknowledgments

I would like to extend my gratitude to my thesis advisor, Dr. Radwa El Shawi, for her unwavering support and guidance throughout this journey. Her invaluable insights, expertise, and encouragement have been instrumental in shaping my domain knowledge. Thank you for your patience and dedication in nurturing my academic development.

Ich möchte meine tiefe Dankbarkeit zum Ausdruck bringen, dass ich auf dieser Reise nicht allein war und stets die Unterstützung meiner Partnerin Nana hatte. Es ist schwierig, das volle Ausmaß meiner Dankbarkeit für die Hilfe, die du mir entgegengebracht hast, in Worte zu fassen – von der ersten Frage "Soll ich mich im Master-Programm einschreiben?" bis zum Abschluss des Programms, in dem ich während der Arbeit an meiner Thesis eine Achterbahnfahrt aus Erfolgen und Herausforderungen erlebt habe. Ich bin überzeugt, dass diese Reise ohne dich – wenn nicht unmöglich – weitaus schwieriger und weniger erfüllend gewesen wäre.

In diesem Zusammenhang möchte ich mich auch herzlich bei der Familie Löchner (Eva, Rüdiger, Nico) bedanken, für die entspannten Momente, die wir gemeinsam verbracht haben. Zum Schluss möchte ich auch den zahlreichen deutschen Freunden meinen Dank aussprechen, die mich während meines Studiums tatkräftig unterstützt haben.

Soovin tänada oma peret ning paljusid sõpru. Eriti suured tänud teile, Max ning Jana (ja Cezikowski!), et olete olnud sellel teekonnal alati toeks. Olen siiralt rõõmus, et selle kraadi omandamise algusest on paljud sõbrad olnud asendamatuks toeks - olgu siis huvitavate ja inspireerivate vestluste, tehniliste nõuannete või öömaja pakkumisega Tartus. Suured aitähid ka mitmetele kursusekaaslastele (eriti tahaksin välja tuua Heidi, Triinu ja Marilini), kellega koos oleme paljusid asju arutades kindlasti ühiselt targemaks saanud. Lõpetuseks soovin tänu avaldada ka paljudele andmeteaduse magistriõppekava õppe- ja taustajõududele, kel on olnud oluline roll nii domeenialases arengus kui ka kes on abiks olnud õppekorralduslikes küsimustes.

References

[1] Olatz Arbelaitz et al. “An extensive comparative study of cluster validity indices”. In: *Pattern Recognition* 46.1 (2013), pp. 243–256. ISSN: 0031-3203. DOI: https://doi.org/10.1016/j.patcog.2012.07.021.

[2] Asegunloluwa Eunice Babalola, Bolanle Adefowoke Ojokoh, and Julius Beneoluchi Odili. “A Review of Population-Based Optimization Algorithms”. In: *2020 International Conference in Mathematics, Computer Engineering and Computer Science (ICMCECS)*. 2020, pp. 1–7. DOI: 10.1109/ICMCECS47690.2020.240856.

[3] Manuel Baena-Garcıa et al. “Early drift detection method”. In: *Fourth international workshop on knowledge discovery from data streams*. Vol. 6. Citeseer. 2006, pp. 77–86.

[4] Maroua Bahri et al. “AutoML for Stream k-Nearest Neighbors Classification”. In: *2020 IEEE International Conference on Big Data (Big Data)*. 2020, pp. 597–602. DOI: 10.1109/BigData50022.2020.9378396.

[5] Maroua Bahri et al. “AutoML: state of the art with a focus on anomaly detection, challenges, and research directions”. In: *International Journal of Data Science and Analytics* 14.2 (Aug. 2022), pp. 113–126. DOI: 10.1007/s41060-022-00309-0.

[6] Maria-Florina Balcan, Tuomas Sandholm, and Ellen Vitercik. “Generalization in Portfolio-Based Algorithm Selection”. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 35.14 (May 2021), pp. 12225–12232. DOI: 10.1609/aaai.v35i14.17451.

[7] James Bergstra, Dan Yamins, David D Cox, et al. “Hyperopt: A python library for optimizing the hyperparameters of machine learning algorithms”. In: *Proceedings of the 12th Python in science conference*. Vol. 13. Citeseer, 2013, p. 20.

[8] James Bergstra, Daniel Yamins, and David Cox. “Making a Science of Model Search: Hyperparameter Optimization in Hundreds of Dimensions for Vision Architectures”. In: *Proceedings of the 30th International Conference on Machine Learning*. Ed. by Sanjoy Dasgupta and David McAllester. Vol. 28. Proceedings of Machine Learning Research 1. Atlanta, Georgia, USA: PMLR, 17–19 Jun 2013, pp. 115–123.

[9] Albert Bifet and Ricard Gavaldà. “Learning from Time-Changing Data with Adaptive Windowing”. In: *Proceedings of the 2007 SIAM International Conference on Data Mining (SDM)*, pp. 443–448. DOI: 10.1137/1.9781611972771.42.

[10] Albert Bifet et al. “Moa: Massive online analysis, a framework for stream classification and clustering”. In: *Proceedings of the first workshop on applications of pattern analysis*. PMLR. 2010, pp. 44–50.

[11] Matthias Carnein and Heike Trautmann. “Customer Segmentation Based on Transactional Data Using Stream Clustering”. In: *Advances in Knowledge Discovery and Data Mining*. Ed. by Qiang Yang et al. Cham: Springer International Publishing, 2019, pp. 280–292.

[12] Matthias Carnein et al. “confstream: Automated algorithm selection and configuration of stream clustering algorithms”. In: *Learning and Intelligent Optimization: 14th International Conference, LION 14, Athens, Greece, May 24–28, 2020, Revised Selected Papers 14*. Springer. 2020, pp. 80–95.

[13] Bilge Celik, Prabhant Singh, and Joaquin Vanschoren. “Online AutoML: an adaptive AutoML framework for online learning”. In: *Machine Learning* (Dec. 2022). DOI: 10.1007/s10994-022-06262-0.

[14] Boyuan Chen et al. “Autostacker: A Compositional Evolutionary Learning System”. In: (2018). DOI: 10.48550/ARXIV.1803.00684.

[15] Paul D. Clough and Jahna Otterbacher. “Democratizing AI: from theory to practice”. In: *Handbook of Research on Artificial Intelligence, Innovation and Entrepreneurship*. Ed. by Elias Carayannis and Evangelos Grigoroudis. Edward Elgar Publishing, Feb. 2023, pp. 402–418. DOI: 10.4337/9781839106750.00039.

[16] Stefan Coors et al. “Automatic Componentwise Boosting: An Interpretable AutoML System”. In: (2021). DOI: 10.48550/ARXIV.2109.05583.

[17] K. Deb et al. “A fast and elitist multiobjective genetic algorithm: NSGA-II”. In: *IEEE Transactions on Evolutionary Computation* 6.2 (2002), pp. 182–197. DOI: 10.1109/4235.996017.

[18] Xibin Dong et al. “A survey on ensemble learning”. In: *Frontiers of Computer Science* 14 (2020), pp. 241–258.

[19] Radwa ElShawi, Hudson Lekunze, and Sherif Sakr. “cSmartML: A Meta Learning-Based Framework for Automated Selection and Hyperparameter Tuning for Clustering”. In: *2021 IEEE International Conference on Big Data (Big Data)*. Orlando, FL, USA: IEEE, Dec. 2021, pp. 1119–1126. DOI: 10.1109/BigData52589.2021.9671542.

[20] Radwa ElShawi, Mohamed Maher, and Sherif Sakr. “Automated Machine Learning: State-of-The-Art and Open Challenges”. In: (2019). DOI: 10.48550/ARXIV.1906.02287.

[21] Radwa ElShawi and Sherif Sakr. “TPE-AutoClust: A Tree-based Pipline Ensemble Framework for Automated Clustering”. In: *2022 IEEE International Conference on Data Mining Workshops (ICDMW)*. Orlando, FL, USA: IEEE, Nov. 2022, pp. 1144–1153. DOI: 10.1109/ICDMW58026.2022.00149.

[22] Matthias Feurer and Frank Hutter. “Hyperparameter optimization”. In: *Automated machine learning: Methods, systems, challenges* (2019), pp. 3–33.

[23] Matthias Feurer, Jost Springenberg, and Frank Hutter. “Initializing Bayesian Hyperparameter Optimization via Meta-Learning”. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 29.1 (Feb. 2015). DOI: 10.1609/aaai.v29i1.9354.

[24] Matthias Feurer et al. “Auto-sklearn 2.0: Hands-free automl via meta-learning”. In: *Journal of Machine Learning Research* 23.261 (2020), pp. 1–61.

[25] Milton Friedman. “The Use of Ranks to Avoid the Assumption of Normality Implicit in the Analysis of Variance”. In: *Journal of the American Statistical Association* 32.200 (1937), pp. 675–701. DOI: 10.1080/01621459.1937.10503522.

[26] João Gama et al. “A survey on concept drift adaptation”. In: *ACM Computing Surveys (CSUR)* 46 (2014), pp. 1–37.

[27] João Gama et al. “Learning with Drift Detection”. In: *Advances in Artificial Intelligence – SBIA 2004*. Ed. by David Hutchison et al. Vol. 3171. Series Title: Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 286–295. DOI: 10.1007/978-3-540-28645-5_29.

[28] Pieter Gijsbers. “Systems for AutoML Research”. English. ISBN: 978-90-386-5510-9 Series: SIKS Dissertation Series. PhD Thesis. Mathematics and Computer Science, May 2022.

[29] Pieter Gijsbers and Joaquin Vanschoren. “GAMA: A General Automated Machine Learning Assistant”. In: *Machine Learning and Knowledge Discovery in Databases. Applied Data Science and Demo Track*. Ed. by Yuxiao Dong et al. Vol. 12461. Cham: Springer International Publishing, 2021, pp. 560–564. DOI: 10.1007/978-3-030-67670-4_39.

[30] Heitor Murilo Gomes et al. “Machine learning for streaming data: state of the art, challenges, and opportunities”. In: *ACM SIGKDD Explorations Newsletter* 21.2 (Nov. 2019), pp. 6–22. DOI: 10.1145/3373464.3373470.

[31] Max Halford et al. *creme, a Python library for online machine learning*. June 2020. URL: https://github.com/MaxHalford/creme.

[32] Maria Halkidi, Yannis Batistakis, and Michalis Vazirgiannis. “Cluster validity methods: part I”. In: *ACM Sigmod Record* 31.2 (2002), pp. 40–45.

[33] Frank Hutter, Lars Kotthoff, and Joaquin Vanschoren. *Automated machine learning: methods, systems, challenges*. Springer Nature, 2019.

[34] A. K. Jain, M. N. Murty, and P. J. Flynn. “Data Clustering: A Review”. In: *ACM Comput. Surv.* 31.3 (Sept. 1999), pp. 264–323. DOI: 10.1145/331499.331504.

[35] Richard M Karp. “On-line algorithms versus off-line algorithms: How much”. In: *Algorithms, Software, Architecture: Information Processing 92: Proceedings of the IFIP 12th World Computer Congress*. Vol. 1. 1992, p. 416.

[36] Brent Komer, James Bergstra, and Chris Eliasmith. “Hyperopt-sklearn: automatic hyperparameter configuration for scikit-learn”. In: *ICML workshop on AutoML*. Vol. 9. Citeseer Austin, TX, 2014, p. 50.

[37] Shanmugam Shan Kulandaivel and Cody Irwin. *Use real-time anomaly detection reference patterns to combat fraud*. 2020. URL: https://cloud.google.com/blog/products/data-analytics/using-automated-ml-streaming-architecture-to-find-anomalies (visited on 03/12/2023).

[38] Doi Thi Lan and Seokhoon Yoon. “Trajectory Clustering-Based Anomaly Detection in Indoor Human Movement”. In: *Sensors* 23.6 (Mar. 2023), p. 3318. DOI: 10.3390/s23063318.

[39] John Langford, Lihong Li, and Alex Strehl. *Vowpal wabbit online learning project*. 2007.

[40] Yue Liu, Shuang Li, and Wenjie Tian. “AutoCluster: Meta-learning Based Ensemble Method for Automated Unsupervised Clustering”. en. In: *Advances in Knowledge Discovery and Data Mining*. Ed. by Kamal Karlapalem et al. Vol. 12714. Series Title: Lecture Notes in Computer Science. Cham: Springer International Publishing, 2021, pp. 246–258. DOI: 10.1007/978-3-030-75768-7_20.

[41] Mohamed Maher and Sherif Sakr. *SmartML: A Meta Learning-Based Framework for Automated Selection and Hyperparameter Tuning for Machine Learning Algorithms*. 2019. DOI: 10.5441/002/EDBT.2019.54.

[42] Fernando Martinez-Plumed et al. “CRISP-DM Twenty Years Later: From Data Mining Processes to Data Science Trajectories”. In: *IEEE Transactions on Knowledge and Data Engineering* 33.8 (Aug. 2021), pp. 3048–3061. DOI: 10.1109/TKDE.2019.2962680.

[43] Tom M. Mitchell. *Machine Learning*. McGraw-Hill series in computer science. New York: McGraw-Hill, 1997. ISBN: 978-0-07-042807-2.

[44] Felix Mohr, Marcel Wever, and Eyke Hüllermeier. “ML-Plan: Automated machine learning via hierarchical planning”. In: *Machine Learning* 107.8-10 (Sept. 2018), pp. 1495–1515. DOI: 10.1007/s10994-018-5735-z.

[45] Md. Ashifuddin Mondal and Zeenat Rehena. “Identifying Traffic Congestion Pattern using K-means Clustering Technique”. In: *2019 4th International Conference on Internet of Things: Smart Innovation and Usages (IoT-SIU)*. 2019, pp. 1–5. DOI: 10.1109/IoT-SIU.2019.8777729.

[46] Jacob Montiel et al. "Online Clustering: Algorithms, Evaluation, Metrics, Applications and Benchmarking". In: *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. Washington DC USA: ACM, Aug. 2022, pp. 4808–4809. DOI: 10.1145/3534678.3542600.

[47] Jacob Montiel et al. "River: Machine Learning for Streaming Data in Python". In: *Journal of Machine Learning Research* 22.1 (July 2022).

[48] Jacob Montiel et al. "Scikit-Multiflow: A Multi-output Streaming Framework". In: *Journal of Machine Learning Research* 19.72 (2018), pp. 1–5.

[49] H. Mouss et al. "Test of Page-Hinckley, an approach for fault detection in an agro-alimentary production system". In: *2004 5th Asian Control Conference (IEEE Cat. No.04EX904)*. Vol. 2. 2004, 815–818 Vol.2.

[50] Dinithi Nallaperuma et al. "Online Incremental Machine Learning Platform for Big Data-Driven Smart Traffic Management". In: *IEEE Transactions on Intelligent Transportation Systems* 20.12 (2019), pp. 4679–4690. DOI: 10.1109/TITS.2019.2924883.

[51] Peter Nemenyi. "Distribution-free multiple comparisons". PhD thesis. Princeton University, 1963.

[52] L. O'Callaghan et al. "Streaming-data algorithms for high-quality clustering". In: *Proceedings 18th International Conference on Data Engineering*. 2002, pp. 685–694. DOI: 10.1109/ICDE.2002.994785.

[53] Randal S. Olson and Jason H. Moore. "TPOT: A Tree-based Pipeline Optimization Tool for Automating Machine Learning". In: *Proceedings of the Workshop on Automatic Machine Learning*. Ed. by Frank Hutter, Lars Kotthoff, and Joaquin Vanschoren. Vol. 64. Proceedings of Machine Learning Research. New York, New York, USA: PMLR, June 2016, pp. 66–74.

[54] OpenAI. *ChatGPT)*. 2023. URL: https://chat.openai.com (visited on 05/06/2023).

[55] Tinghui Ouyang and Xun Shen. "Online structural clustering based on DBSCAN extension with granular descriptors". In: *Information Sciences* 607 (2022), pp. 688–704. DOI: https://doi.org/10.1016/j.ins.2022.06.027.

[56] Vishwajeet Pattanaik et al. "Smart real-time traffic congestion estimation and clustering technique for urban vehicular roads". In: *2016 IEEE Region 10 Conference (TENCON)*. 2016, pp. 3420–3423. DOI: 10.1109/TENCON.2016.7848689.

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

[58] Yannis Poulakis, Christos Doulkeridis, and Dimosthenis Kyriazis. “AutoClust: A Framework for Automated Clustering Based on Cluster Validity Indices”. In: *2020 IEEE International Conference on Data Mining (ICDM)*. Sorrento, Italy: IEEE, Nov. 2020, pp. 1220–1225. DOI: 10.1109/ICDM50108.2020.00153.

[59] Jesse Read, Peter Reutemann, and Joerg Wicker. *MEKA*. 2023. URL: http://waikato.github.io/meka/ (visited on 03/03/2023).

[60] Mattia Rizzini et al. “Static and Dynamic Portfolio Methods for Optimal Planning: An Empirical Analysis”. In: *International Journal on Artificial Intelligence Tools* 26.01 (2017), p. 1760006. DOI: 10.1142/S0218213017600065.

[61] Alex G. C. de Sá, Alex A. Freitas, and Gisele L. Pappa. “Automated Selection and Configuration of Multi-Label Classification Algorithms with Grammar-Based Genetic Programming”. In: *Parallel Problem Solving from Nature – PPSN XV*. Ed. by Anne Auger et al. Vol. 11102. Cham: Springer International Publishing, 2018, pp. 308–320. DOI: 10.1007/978-3-319-99259-4_25.

[62] Alex G. C. de Sá et al. “RECIPE: A Grammar-Based Framework for Automatically Evolving Classification Pipelines”. In: *Genetic Programming*. Ed. by James McDermott et al. Vol. 10196. Cham: Springer International Publishing, 2017, pp. 246–261. DOI: 10.1007/978-3-319-55696-3_16.

[63] Vinicius M. A. Souza et al. “Challenges in benchmarking stream learning algorithms with real-world data”. en. In: *Data Mining and Knowledge Discovery* 34.6 (Nov. 2020), pp. 1805–1858. DOI: 10.1007/s10618-020-00698-5.

[64] Statista. *Volume of data/information created, captured, copied, and consumed worldwide from 2010 to 2020, with forecasts from 2021 to 2025*. 2022. URL: https://www.statista.com/statistics/871513/worldwide-data-created/ (visited on 02/28/2023).

[65] Douglas Steinley, Michael J. Brusco, and Lawrence Hubert. “The variance of the adjusted Rand index.” en. In: *Psychological Methods* 21.2 (2016), pp. 261–272. DOI: 10.1037/met0000049.

[66] Chris Thornton et al. “Auto-WEKA: combined selection and hyperparameter optimization of classification algorithms”. In: *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*. Chicago Illinois USA: ACM, Aug. 2013, pp. 847–855. DOI: 10.1145/2487575.2487629.

[67] Dennis Tschechlov, Manuel Fritz, and Holger Schwarz. *AutoML4Clust: Efficient AutoML for Clustering Analyses*. en. 2021. DOI: 10.5441/002/EDBT.2021.32.

[68] Joaquin Vanschoren. “Meta-learning”. In: *Automated machine learning: methods, systems, challenges* (2019), pp. 35–61.

[69] Hernan Ceferino Vazquez. “A General Recipe for Automated Machine Learning in Practice”. In: *Advances in Artificial Intelligence–IBERAMIA 2022: 17th Ibero-American Conference on AI, Cartagena de Indias, Colombia, November 23–25, 2022, Proceedings*. Springer, 2023, pp. 243–254.

[70] Chi Wang et al. “FLAML: A Fast and Lightweight AutoML Library”. In: *Proceedings of Machine Learning and Systems*. Ed. by A. Smola, A. Dimakis, and I. Stoica. Vol. 3. 2021, pp. 434–447. URL: `https://proceedings.mlsys.org/paper/2021/file/92cc227532d17e56e07902b254dfad10-Paper.pdf`.

[71] Gerhard Widmer and Miroslav Kubat. “Learning in the presence of concept drift and hidden contexts”. In: *Machine Learning* 23.1 (Apr. 1996), pp. 69–101. DOI: `10.1007/BF00116900`.

[72] Qingyun Wu et al. “ChaCha for Online AutoML”. In: *Proceedings of the 38th International Conference on Machine Learning*. Ed. by Marina Meila and Tong Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, July 2021, pp. 11263–11273.

[73] Feiyu Xu et al. “Explainable AI: A brief survey on history, research areas, approaches and challenges”. In: *Natural Language Processing and Chinese Computing: 8th CCF International Conference, NLPCC 2019, Dunhuang, China, October 9–14, 2019, Proceedings, Part II 8*. Springer, 2019, pp. 563–574.

[74] Alaettin Zubaroğlu and Volkan Atalay. “Online embedding and clustering of evolving data streams”. In: *Statistical Analysis and Data Mining: The ASA Data Science Journal* 16.1 (2023), pp. 29–44. DOI: `https://doi.org/10.1002/sam.11590`.

Appendix

I. Glossary

The software code written for this project as well as respective documentation can be found in the following repository:

https://github.com/qetdr/online-autoclust-hpo

II. Licence