Demonstrating SubStrat: A Subset-Based Strategy for Faster AutoML on Large Datasets

Teddy Lazebnik
University College London
London, UK
t.lazebnik@ucl.ac.uk

Amit Somech
Bar-Ilan University
Ramat Gan, Israel
somecha@cs.biu.ac.il

ABSTRACT

Automated machine learning (AutoML) frameworks are gaining popularity among data scientists as they dramatically reduce the manual work devoted to the construction of ML pipelines while obtaining similar and sometimes even better results than manually-built models. Such frameworks intelligently search among millions of possible ML pipeline configurations to finally retrieve an optimal pipeline in terms of predictive accuracy. However, when the training dataset is large, the construction and evaluation of a single ML pipeline take longer, which makes the overall AutoML running times increasingly high.

To this end, in this work we demonstrate SubStrat, an AutoML optimization strategy that tackles the dataset size rather than the configurations search space. SubStrat wraps existing AutoML tools, and instead of executing them directly on the large dataset, it uses a genetic-based algorithm to find a small yet representative *data subset* that preserves characteristics of the original one. SubStrat then employs the AutoML tool on the generated subset, resulting in an intermediate ML pipeline, which is later refined by executing a restricted, much shorter, AutoML process on the large dataset. We demonstrate SubStrat on both AutoSklearn, TPOT, and H2O, three popular AutoML frameworks, using several real-life Kaggle datasets.

CCS CONCEPTS

• **Information systems** → *Data analytics*; *Data mining*; • **Computing methodologies** → *Machine learning*.

KEYWORDS

Automated Machine Learning (AutoML), Data Reduction

ACM Reference Format:
Teddy Lazebnik and Amit Somech. 2022. Demonstrating SubStrat: A Subset-Based Strategy for Faster AutoML on Large Datasets. In *Proceedings of the 31st ACM International Conference on Information and Knowledge Management (CIKM '22), October 17–21, 2022, Atlanta, GA, USA.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3511808.3557160

1 INTRODUCTION

Automated machine learning (AutoML) frameworks aim to facilitate the time-consuming task of developing a machine learning (ML) model [9], assisting even inexperienced users in building accurate and robust models for a given dataset. For each dataset, AutoML frameworks compare a multitude of ML pipeline configurations – typically composed of pre-processing, feature engineering, model selection and hyper-parameters optimization [4] – and select the configuration that yields the most accurate model [9].

Since the configuration space of all possible ML pipelines is tremendously large, AutoML frameworks employ a variety of optimizations and search heuristics, such as Bayesian optimization [7], meta-learning [3, 5], and genetic algorithms [13], in order to reduce the number of compared pipelines.

Nonetheless, when the training data is large, each pipeline execution takes a longer time to run. This can add up to hours of search time, even when using state-of-the-art AutoML frameworks [4]. To tackle this challenge, we propose SubStrat, a new strategy for reducing AutoML computation costs, tackling the *data size* rather than the *configuration search space*. Hence, instead of employing an AutoML tool directly on the large dataset, SubStrat first generates a special *data subset* (DST) which preserves some characteristics of the original one. It then executes the AutoML tool on the small subset (which is significantly faster than on the large dataset) and last, refines the resulted ML pipeline configuration by executing a limited, shorter AutoML process back on the large dataset. This allows SubStrat to significantly reduce AutoML times with a minimal decrease in the final model accuracy. To our knowledge, SubStrat is the first AutoML optimization strategy that utilizes data subsets in order to speedup the AutoML process.

Although it is quite obvious that employing AutoML on a fraction of the data takes less time, not all subsets result in good-enough AutoML performance. For instance, as shown in Figure 3, using a randomly selected subset in our approach is indeed much faster, yet reduces the model accuracy by 30%. Therefore, finding an adequate DST, with a minimal impact on the final accuracy in a timely fashion is a challenge. To overcome this, SubStrat uses Gen-DST, a novel, genetic-based algorithm that finds a DST which preserves the original *dataset entropy*. Given a dataset of N rows and M columns, Gen-DST finds a DST of size $n \times m$ (s.t. $n << N$ and $m << M$) by producing generations of candidate DSTs, and performing an evolution-like selection of the ones that obtain the closest entropy to the original data.

The main advantage of our system is the compatibility with state-of-the-art existing AutoML tools, allowing data scientists to continue using their favorite frameworks while significantly reducing computation times.

Related work. Auto-ML is a growing research field, aiming to automate the process of developing ML models for a given task and dataset [2, 4, 5, 7, 15, 17, 19]. Existing AutoML can be roughly divided into two main categories: search-space optimizations and meta-learning solutions (as well as the combination thereof). Search space optimizations employ strategies such as Bayesian optimization [2, 7], directed search [17, 19] and genetic programming [14], to reduce the configuration search space, while meta-learning solutions [3, 5, 20] utilize a corpus of datasets to learn some of the configuration elements.

Differently from these works, the goal of SubStrat is to reduce the size of the input dataset, rather than the search space of pipeline configurations. The goal of SubStrat is *not* to replace existing AutoML tools, but to *improve the running time* of such search-based tools by executing the majority of computation on a significantly smaller data subset, discovered by Gen-DST, our genetic-based algorithm.

Last, note that SubStrat is currently designed to assist in traditional AutoML tools (aimed at tabular datasets), whereas AutoML frameworks (e.g.,[8] exists also for Neural-Network Architecture search (NAS). We aim to tackle NAS optimization in future work.

Demo. We show how SubStrat is easily used with three popular AutoML frameworks, AutoSklearn [3], TPOT [14], and H2O [12] on a variety of publicly available datasets. After selecting a dataset, we demonstrate how SubStrat integrates with the AutoML tools and is able to significantly reduce their running times with a minimal impact on accuracy. To further gauge the performance of SubStrat, we provide a comparison utility that presents a side-by-side evaluation of running times and performance of both SubStrat, the full AutoML process, and a baseline approach that uses a Monte-Carlo based DST (See Figure 3).

Last, note that a full report on SubStrat, including full algorithmic details and experimental results, is provided in [11]. We further publish our source code in [16].

2 APPROACH

In a typical AutoML scenario, a data scientist desires to build an ML model for predicting the value of some target feature y in dataset D. Rather than manually constructing the model, the data scientist employs an AutoML tool A that intelligently scans multitudes of ML pipelines (composed of feature engineering, model selection, and hyper-parameters optimizations), and results in a particular configuration that achieves the highest predictive performance. We denote the application of an AutoML tool A over dataset D to predict the target y by $A(D, y) \rightarrow M^\star$, where $M^\star$ is the best configuration that A could find. As mentioned above, the larger the dataset, the higher the computational cost of the AutoML, since each candidate pipeline takes longer to execute. Let $Time(M^\star)$ be the time it takes for the AutoML tool A to generate $M^\star$, with final model accuracy, denoted by $Acc(M^\star)$.

The goal of SubStrat, our subset-based strategy, is to utilize a data *subset* in order to reduce AutoML computation times – while retaining the output model performance. Namely, to generate a model configuration M_{sub} s.t. $Time(M_{sub}) << Time(M^\star)$ but $Acc(M_{sub}) \approx Acc(M^\star)$.

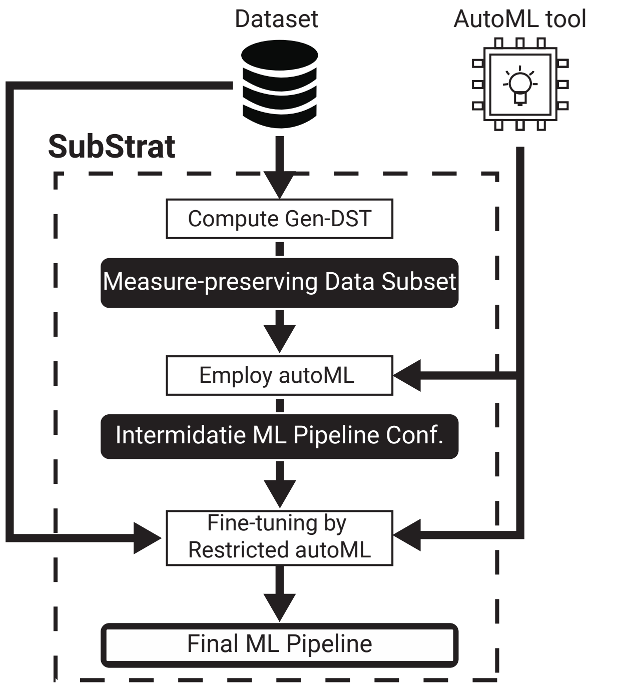
Dataset
AutoML tool
SubStrat
Compute Gen-DST
Measure-preserving Data Subset
Employ autoML
Intermidatie ML Pipeline Conf.
Fine-tuning by Restricted autoML
Final ML Pipeline

Figure 1: SubStrat Workflow

Unfortunately, as is also shown in [11], a simple random sample is highly ineffective: When running the AutoML tool on a random subset of the data, the process is indeed much faster (time is reduced by more than 96%), yet the final model's accuracy significantly drops. On average, the accuracy of the random-sample AutoML model is 27% lower compared to the accuracy of the $M^\star$, the output model of the AutoML tool when running on the full dataset. Naturally, such low accuracy makes the AutoML process completely futile.

We next describe how SubStrat overcomes this problem, able to reduce running time by an average of 78% with an accuracy reduction of less than 2% [11].

SubStrat workflow. Figure 1 describes the architecture and workflow of SubStrat.

The system takes as input a dataset, and an existing AutoML tool (e.g., TPOT [14], AutoSKlearn [3]). It then works in three steps: (1) SubStrat first intelligently searches for a small DST that preserves the entropy of the full dataset (See Section 2.1). This is done using Gen-DST, our novel genetic-based algorithm (See Section 2.2. (2) Once the DST is ready, SubStrat employs the AutoML tool over the DST, instead of the full dataset. This naturally, takes a fraction of the time (See Figure 2). This phase results in an (intermediate) ML pipeline configuration. (3) Last, SubStrat refines the intermediate ML pipeline, by executing the AutoML tool on the full dataset, yet in a much shorter, restricted manner. This results in a final ML pipeline configuration.

We next describe the components of SubStrat in more detail.

2.1 Entropy-Preserving Data Subsets

Let D be a dataset of N rows and M columns. Denote its row and column indices by $R = 1, 2, \ldots, N$ and $C = 1, 2, \ldots, M$, respectively. Intuitively, a DST of a full dataset D is simply a subset of the rows of

D, projected over a subset of the columns. Formally, given a dataset D with row-indices R and column-indices C, a data subset (DST) of size $n \times m$ is defined as follows. Let $[R]^n$ be the set of all n-subsets of R, i.e., $[R]^n = \{R' | R' \subseteq R \wedge |R| = n\}$, and $[C]^m$ be the set of all m-subsets of C. Then, given $r \in [R]^n$ and $c \in [C]^m$, the data subset is defined by the rows in D indicated in r, projected over the columns indicated in c.

Our goal is therefore to find a *representative* DST, that preserves characteristics of the original dataset. Inspired by works in feature selection [21] and decision tree learning [18], we use an entropy-based measure to amount the preservation of such characteristics. Our *dataset entropy* function, as defined below, assesses the "amount of information" conveyed in the entire dataset.

Definition 2.1 (Dataset Entropy). Given dataset D of size $N \times M$, Let D_{ij} be the value in row i and column j.

$$H(D) = \frac{\sum_{j=1}^{M} \left(\sum_{i=1}^{N} P_j(D_{ij}) \cdot Log_2 P(D_{ij}) \right)}{M} \tag{1}$$

Where $P_j(D_{ij})$ is a probability function corresponding to the frequency of the value in D_{ij} w.r.t. Column j. Namely, if $D_{ij} = v$ then:

$$P_j(v) = \frac{\sum_{k=1}^{N} I[D_{kj} = v]}{N} \tag{2}$$

Ideally, for a dataset D, we would like to obtain the best entropy-preserving DST d of size $n \times m$. Naively, one can employ a brute-force search that traverses through all possible DSTs of size $n \times m$. Clearly, this becomes infeasible for large datasets or when a larger DST is needed. Therefore, we define an optimization problem, which is to minimize the entropy difference between the DST d and the original dataset D, as follows:

$$\mathcal{L}(r, c) = |H(D[r, c]) - H(D)|\,. \tag{3}$$

In what comes next we describe Gen-DST, a genetic-based algorithm that minimizes $\mathcal{L}(r, c)$, while also obtaining short convergence time. The latter is highly important since if the DST generation process takes too much time, the efficacy of our overall solution in reducing AutoML times is significantly diminished.

2.2 Gen-DST: A Genetic-Based DST Algorithm

Our framework employs a Genetic Algorithm (GA) [6], a well-known and commonly-used meta-heuristic search method, based on the biological theory of evolution [6]. We next briefly describe Gen-DST our genetic-based algorithm for finding DSTs (Refer to [11] for full details and analysis).

Briefly, the genetic representation of a candidate-DST, denoted G, comprises of $n + m$ chromosomes: n *row*-chromosomes, that correspond to n row indices of dataset D, and m *column*-chromosomes, that correspond to m column-indices. Namely, $G := (r, c)$, $r \in [R]^n, c \in [C]^m$, where for a dataset D, recall that R and C denote its rows and columns indices. The *fitness function*, i.e., the goal of the evolutionary process, is defined as the negative loss of the DST-candidate: $F(G) := -\mathcal{L}(r, c)$ (using the entropy difference, see Equation 3).

Gen-DST finds the optimal DST, by first initializing (at random) a population P of candidate DSTs, then iteratively employing designated genetic operations: *mutation*, *cross-over*, and *selection*: The mutation operator introduces random noise into the DST candidates; The cross-over operator generates new DST candidates by combining chromosomes of older candidates; Last, the selection operator filters out the unfitted candidates (using Equation 3). For each generation i (out of ψ generations), the fittest DST candidate is kept if it surpasses the best DST from the previous generations. Finally, Gen-DST returns the best DST out of all ψ generations.

Last, note that the running times of Gen-DST depend on numerous factors such as the number of generations ψ, population size $|P|$, the full data as well as the DST size. We discuss below the running time analysis of SubStrat compared to a full AutoML process.

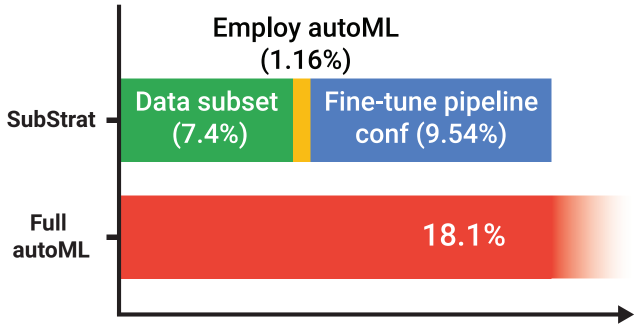

Figure 2: SubStrat Computation Breakdown. Finding the DST using Gen-DST takes 7.4% out of the full AutoML process (on the large dataset). Then, employing the AutoML tool takes another 1.16% and the fine-tuning stage takes another 9.54%. The entire SubStrat process takes, on average, only 18.1% of the time used by executing a full AutoML process.

Fine-Tuning the Configuration. After SubStrat generates a DST d using Gen-DST, it runs the AutoML tool on d and obtains an intermediate ML configuration, denoted M'. This configuration needs to be adapted back to fit dataset D. To do so, we employ A back on D, but restrict its search space, forcing it to only consider configurations that use the same ML model as specified in M'. Such a simple yet efficient restriction allows SubStrat to retain short running times, while considerably improving the intermediate configuration M', reaching almost the same accuracy as the best configuration $M^{\star}$.

Running Times Analysis. Figure 2 shows the mean computation time of each of the three components in SubStrat, relatively to the Full-AutoML execution (100%). The proportions were obtained empirically over 10 different datasets, as described in [11]. The first component of finding an entropy-preserving DST using Gen-DST is rather costly, but still takes only 7.4% of the full AutoML process. The second component, in which we employ the AutoML tool on the obtained DST, takes only 1.16% of the full AutoML computation. This natural speedup is since the DST, with a default size of $\sqrt{N}$ rows and $0.25M$ columns, is on average 502X smaller than the full dataset. Last, as expected, the fine-tuning phase is more costly (9.54%), since each tested pipeline configuration is now executed on the full dataset. Nevertheless, *see that SubStrat execution takes only 18.1% on average than the full AutoML case.*

```
1 import autosklearn.classification as autoML # load the autoML library
2 from SubStrat import SubStrat # import the SubStrat module
3 full_cls = autoML.AutoSklearnClassifier() # initialize an AutoML object
4 X_train, X_test, y_train, y_test = train_test_split(flights_df.iloc[:, :-1], flights_df.iloc[:, -1], test_size=0.2)
5 SubStrat_cls = SubStrat(dataset_x=X_train, dataset_y=y_train, auto_ml=full_cls, size="auto", algo="Gen-DST")
6 SubStrat_cls.add_compare("Full", full_cls.fit(X_train, y_train))
7 SubStrat_cls.add_compare("SubStrat\nMC", SubStrat(dataset_x=X_train, dataset_y=y_train, auto_ml=full_cls, size="auto", algo="MC"))
8 SubStrat_cls.plot_compare_graph()
```

```
Mean entropy: SubStrat-MC = 2.174, Full = 3.847, SubStrat-Gen-DST = 3.679
```

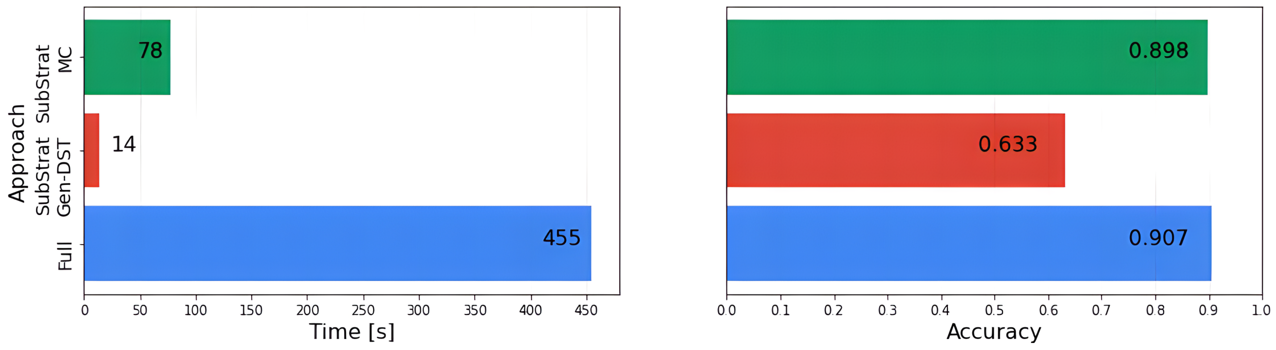

Figure 3: Example usage of the SubStrat library on the flights dataset [1] and AutoSKlearn

3 UI AND DEMONSTRATION OVERVIEW

We implemented SubStrat as a Python library that wraps existing AutoML tools, so users can continue using their favorite AutoML library with the same pipelines they are used to. The hyper parameters of Gen-DST as well as the DST size, are inferred using a meta-learning approach (Similarly to [2]), optimized on a system setting that results in up to a 5% decrease of the model accuracy.

In our demonstration, we invite participants to use SubStrat in a Jupyter notebook [10] environment, as depicted in Figure 3. Participants will first select a dataset from a collection of 10 public Kaggle datasets and predictive tasks. Then, the participants will be invited to choose an AutoML tool (TPOT, AutoSKlearn, H2O), and observe how SubStrat significantly reduces its running times. Figure 3 illustrates an example such demonstration scenario, using the *flight delays dataset* [1] and the AutoSKlearn [3] tool. The ML task associated with the dataset is to *predict whether a flight is delayed or not.*

To do so, as illustrated in Figure 3, users first need to import the AutoML tool (Line 1) and the SubStrat library (Line 2). Next, the users need to initialize the AutoML object (Line 3) and split the data to train and test (Line 4). Then, SubStrat is employed (See Line 5), taking as input the training dataset, the target column (e.g., the rightmost, *"is_delayed"* binary feature in the flight-delays example scenario), and the AutoML object. Users can optionally specify the DST size (default is "auto", which will derive the size w.r.t. the input dataset as described above) and the algorithm (default is Gen-DST).

SubStrat then outputs an optimized ML model which can be used to perform predictions over the test dataset.In our example scenario (Figure 3) SubStrat returns an AutoSklearn Estimator (cls) object, yet in our additional demonstration scenarios, we show interested users an alternative SubStrat process also for TPOT and H2O.

Next, to gauge the performance of SubStrat in terms of running times and accuracy, we will use the SubStrat *comparison utility*, which allows users to inspect the performance of SubStrat against custom baselines. For instance, as depicted in Lines 6-8 in Figure 3, we compare SubStrat to the naive, full AutoML process (which executes AutoSklearn over the full dataset), as well as to a Monte Carlo baseline, which selects the best entropy-preserving DSTs among 100 randomly generated DSTs, instead of using Gen-DST.

The comparison tool then outputs two plots that depict the performance of the selected baselines. The left-hand plot in Figure 3 compares the running time performance. We can see that the full AutoML process took 455 seconds, compared to 78 seconds by SubStrat, and 14 seconds by the Monte Carlo algorithm. Then, in the right-hand plot shows an accuracy comparison. While the full AutoML resulted in a model with 0.907 accuracy for the flights delay prediction task, the model generated by SubStrat obtained 0.898 – less than 1 point difference. The Monte Carlo approach, however, yields substantially inferior accuracy, 0.633, which demonstrates a decrease of more than 27 points compared to the full AutoML.

Last, we invite the participants to look under the hood of SubStrat, and shed light on its computation process. We will present, for example, which columns and rows from the original dataset were used in the generated DST; what is the dataset entropy difference between the resulted DST and the full dataset; and present the difference in the output model configurations.

Additional Scenarios. As mentioned above, other than the Flights dataset scenario, we will allow users to select a different dataset and AutoML algorithm, and inspect the optimization process performed by SubStrat. Participants can choose to employ either AutoSklearn or TPOT over 10 additional datasets from various domains and sizes (up to 2M rows and 1000 columns) such as the Kaggle Car Insurance Dataset, Mushroom Classification, Bike Demand, and more.

REFERENCES

[1] Kaggle Flight Delays Dataset. 2022. https://www.kaggle.com/usdot/flight-delays.

[2] Matthias Feurer, Katharina Eggensperger, Stefan Falkner, Marius Lindauer, and Frank Hutter. 2020. Auto-sklearn 2.0: Hands-free automl via meta-learning. *arXiv preprint arXiv:2007.04074* (2020).

[3] M. Feurer, A. Klevin, K. Eggensperger, J. T. Springenberg, M. Blum, and F. Hutter. 2019. *Auto-sklearn: Efficient and Robust Automated Machine Learning.*

[4] Xin He, Kaiyong Zhao, and Xiaowen Chu. 2021. AutoML: A Survey of the State-of-the-Art. *Knowledge-Based Systems* 212 (2021), 106622.

[5] Yuval Heffetz, Roman Vainshtein, Gilad Katz, and Lior Rokach. 2020. Deepline: Automl tool for pipelines generation using deep reinforcement learning and hierarchical actions filtering. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining.* 2103–2113.

[6] J. H. Holland. 1992. Genetic Algorithms. *Scientific American* 267, 1 (1992), 66–73.

[7] Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. 2011. Sequential model-based optimization for general algorithm configuration. In *International conference on learning and intelligent optimization.* Springer, 507–523.

[8] Haifeng Jin, Qingquan Song, and Xia Hu. 2019. Auto-keras: An efficient neural architecture search system. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining.* 1946–1956.

[9] Shubhra Kanti Karmaker, Md Mahadi Hassan, Micah J Smith, Lei Xu, Chengxiang Zhai, and Kalyan Veeramachaneni. 2021. AutoML to Date and Beyond: Challenges and Opportunities. *ACM Computing Surveys (CSUR)* 54, 8 (2021), 1–36.

[10] Thomas Kluyver, Benjamin Ragan-Kelley, Fernando Pérez, Brian Granger, Matthias Bussonnier, Jonathan Frederic, Kyle Kelley, Jessica Hamrick, Jason Grout, Sylvain Corlay, Paul Ivanov, Damián Avila, Safia Abdalla, and Carol Willing. 2016. Jupyter Notebooks – a publishing format for reproducible computational workflows. In *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, F. Loizides and B. Schmidt (Eds.). IOS Press, 87 – 90.

[11] Teddy Lazebnik, Amit Somech, and Abraham Itzhak Weinberg. 2022. SubStrat: A Subset-Based Strategy for Faster AutoML. *arXiv preprint arXiv:2206.03070* (2022).

[12] Erin LeDell and Sebastien Poirier. 2020. H2o automl: Scalable automatic machine learning. In *Proceedings of the AutoML Workshop at ICML*, Vol. 2020.

[13] Randal S Olson and Jason H Moore. 2016. TPOT: A tree-based pipeline optimization tool for automating machine learning. In *Workshop on automatic machine learning.* PMLR, 66–74.

[14] R. S. Olson and J. H. Moore. 2016. TPOT: A Tree-based Pipeline Optimization Tool for Automating Machine Learning. In *JMLR: Workshop and Conference Proceedings*, Vol. 64. 66–74.

[15] Esteban Real, Chen Liang, David So, and Quoc Le. 2020. Automl-zero: Evolving machine learning algorithms from scratch. In *International Conference on Machine Learning.* PMLR, 8007–8019.

[16] SubStrat Github Repository. 2022. https://github.com/teddy4445/SubStrat.

[17] Chi Wang, Qingyun Wu, Markus Weimer, and Erkang Zhu. 2021. FLAML: a fast and lightweight AutoML Library. *Proceedings of Machine Learning and Systems* 3 (2021), 434–447.

[18] Duch Wlodzislaw, Tadeusz Wieczorek, Jacek Biesiada, and Marcin Blachnik. 2004. Comparison of feature ranking methods based on information entropy, Vol. 2. 1415 – 1419 vol.2. https://doi.org/10.1109/IJCNN.2004.1380157

[19] Qingyun Wu, Chi Wang, and Silu Huang. 2021. Frugal optimization for cost-related hyperparameters. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 10347–10354.

[20] Anatoly Yakovlev, Hesam Fathi Moghadam, Ali Moharrer, Jingxiao Cai, Nikan Chavoshi, Venkatanathan Varadarajan, Sandeep R Agrawal, Sam Idicula, Tomas Karnagel, Sanjay Jinturkar, et al. 2020. Oracle automl: a fast and predictive automl pipeline. *PVLDB* 13, 12 (2020), 3166–3180.

[21] Xiao Zhang, Changlin Mei, Degang Chen, and Jinhai Li. 2016. Feature selection in mixed data: A method using a novel fuzzy rough set-based information entropy. *Pattern Recognition* 56 (2016), 1–15.