Regression-Stratified Sampling for Optimized Algorithm Selection in Time-Constrained Tabular AutoML

Mehdi Bahrami [1] **So Hasegawa** [1] **Lei Liu** [1] **Wei-Peng Chen** [1]

Abstract

The selection of a machine-learning (ML) algorithm is indispensable for tabular AutoML training. Finding an optimized algorithm from a search space can be expensive for large tabular datasets, especially under time constraints. In this study, we introduce a novel *Regression-Stratified Sampling* approach that optimizes algorithm selection by minimizing distribution distance between a subset of data and the target variable(s) in the full-scale dataset via Probability Density Function (PDF). Additionally, we introduce a *PDF Energy* metric, based on relative entropy, to identify an optimized ML algorithm from the search space. Our comprehensive evaluation results demonstrate that the proposed approach successfully selects optimized algorithms from a search space of atomic and ensemble models, outperforming simple random sampling methods. We also conduct a thorough evaluation against Kullback-Leibler (KL) divergence, where the *PDF Energy* metric proves superior in algorithm selection. Furthermore, we validate our approach for ML algorithm selection in an end-to-end scenario across 31 public datasets using 6 tabular AutoML tools. The empirical results indicate that our proposed method efficiently utilizes *Regression-Stratified Sampling* and reliably identifies an optimized machine learning algorithm for tabular data through the *PDF Energy* metric under time constraints.

1. Introduction

An Automated Machine Learning (AutoML) platform aims to automate the entire process of feature engineering, data engineering (Milo & Somech, 2020), hyperparameter optimization, model training, prediction, and deployment, where it requires minimal human supervision at all stages (He et al., 2021). One of the popular applications of AutoML platforms is tabular (structured) datasets for classification and regression tasks (Chui et al., 2018). Tabular AutoML can be used in diverse domains, such as material science (Conrad et al., 2022), healthcare domains (Mustafa & Rahimi Azghadi, 2021) and many other domains (Santu et al., 2020). Due to the popularity and widely used of structured datasets across domains (Feurer et al., 2015), the use of AutoML on tabular data is growing rapidly. Additionally, tabular datasets are the most common market for artificial intelligence (Nti et al., 2020). Traditional machine learning algorithms tend to perform well on tabular datasets (Shwartz-Ziv & Armon, 2022) in compared to deep learning algorithms (Grinsztajn et al., 2022). The study by Grinsztajn et al. shows that i) deep learning models are sensitive to unbalanced features, while tree-based models (Liang et al., 2019) tend to be more robust; and ii) tree-based models tend to easily learn irregular functions (Gorishniy et al., 2021; Shwartz-Ziv & Armon, 2022), unlike deep learning models(Zhu et al., 2023). Recently, (Jin et al., 2023) introduced AutoKeras, which leverages prior knowledge of the search space. However, these studies leave readers with the question, "*How do we select an optimized algorithm from the search space, including tree-based ML algorithms, for a time-constrained setup?*" Algorithm selection (Kotthoff et al., 2019; Yang et al., 2019) is indispensable for training a model as it determines an optimized machine learning algorithm for a given input dataset.

Sampling. One of the popular approaches for exploring different machine-learning algorithms *under time constraints* is a simple random sampling approach (Rao, 1964), which involves selecting a subset of the dataset to find an optimized machine-learning algorithm from a search space. However, this approach can result in poor algorithm selection (Yakovlev et al., 2020) due to the limited nature of the observation and it can be unreliable when the sampling ratio is changed. Stratified sampling (Fisher, 1936) divides the population of a target value into strata. Then, it uses random sampling to draw a set of samples from each of the strata. *However, stratified sampling is widely used for target values with a single target in classification tasks.* (Trost, 1986;

[*]Equal contribution [1]Fujitsu Research of America, Sunnyvale, California. Correspondence to: Mehdi Bahrami <mbhrami@fujitsu.com>.

Accepted by the Structured Probabilistic Inference & Generative Modeling workshop of ICML 2024, Vienna, Austria.

Meng, 2013; Liberty et al., 2016). To the best of our knowledge, for the first time our study introduces an investigation into the effectiveness of optimized algorithm selection and its application in the context of tabular AutoMLs for regression tasks by utilizing stratified sampling, as opposed to the widely used stratified sampling for classification tasks.

Contributions. In this study, we introduce a stratified sampling approach for regression tasks that aims to select an optimized algorithm from a search space of both atomic and ensemble models. This study reduces the required computation time for structured tabular AutoML algorithm selection by using a subset of data. The following are our key contributions in this study.

A. Develop a regression stratified sampling approach that can be divided into two steps: **i)** drawing samples from a dataset in a way that ensures a maximal distribution similarity of Probability Density Function (PDF) (Pearson, 1894) between the target variable(s) in the sample data, and the full-scale data; **ii)** defining a novel evaluation metric, PDF Energy which intuitively combines standard metric evaluation and relative entropy methods. PDF Energy computes performance of each algorithm on sampled data with respect to PDF distribution errors.

B. We conduct two comprehensive performance evaluations on two benchmarks of 31 and 14 different real-world public datasets. **i)** we compare the performance of the proposed approach for algorithm selection in different ratios and compare the performance results against widely used a simple random sampling (SRS) approach; we also compare our proposed PDF Energy for an algorithm selection against Kullback-Leibler Divergence(Csiszár, 1975); **ii)** We conduct another thorough empirical evaluation of our proposed approach within an end-to-end scenario in the tabular AutoML context. We compare the performance of our proposed approach against six tabular AutoML tools for predicting outcomes across two benchmarks for a regression task. The evaluation includes a baseline, and 6 popular tabular AutoML tools including MLJAR (Płońska & Płoński, 2021), FLAML (Wang et al., 2021), H2O (LeDell & Poirier, 2020), TPOT (Olson & Moore, 2016), Auto-ScikitLearn (Feurer et al., 2015), and AutoGluon (Erickson et al., 2020). Our evaluation results demonstrate that our proposed approach was able to identify an optimized algorithm within a time-constrained budget, compared to other baseline methods and popular tabular AutoML tools.

2. Related Works

Simple random sampling (SRS) (Fisher, 1936) is the primary sampling approach that randomly draw a sample from the population with an equal probability for each selection. However, selecting an an algorithm based on a subset of data may not represent the correct representation of a full-scale dataset. For example, some linear and logistic regression algorithms are sensitive to outliers (Feng et al., 2014), therefore if the subset does not cover the outliers (edge cases), it causes an unreliable and poor algorithm selection that required additional outlier detection(OD) methods, such as RoLR (Feng et al., 2014). Meng presented a scalable simple random sampling algorithm (ScaSRS), which uses probabilistic thresholds to accept, reject, or wait-list. However, the study focuses on classification problems where it aims to select samples on-the-fly without consideration of algorithm selection. (Li et al., 2023) studies effect of pre-processing of tabular data on algorithm selection and (Hollmann et al., 2024) is a similar study on feature engineering but the authors utilizes LLMs. (Wu et al., 2024) also introduce the effectiveness of an autoencoders for tabular data. (Pérez-Cruz, 2008) introduced a method for estimating the KL divergence between continuous densities and proved it converges almost surely. We utilize this approach as a baseline in our evaluation of continuous variable targets (regression tasks). Additionally, our extension to relative entropy-based algorithm selection permits the algorithm of a trained model to be chosen, if it exhibits superior relative entropy, employing a *PDF Energy* in place of KL. Later in experiment section, we demonstrate that our results indicate that our proposed approach provides superior algorithm selection in compared to relative entropy computation using KL divergence. Yang et al. introduced a filtering approach to select a model; however, the adaptation metric is sensitive and a generic approach is required. Sampling approaches have been widely used previously, but there have been limited studies on machine-learning algorithm selection. For instance, Zogaj et al. conducted an empirical study on down-sampling for tabular AutoMLs. In a recent study, Vincent & Jidesh introduces an improved hyperparameter optimization framework for AutoML systems using evolutionary algorithms; however, it is mainly applied to high-dimensional data (i.e., images). In comparison to our study, we introduce a novel sampling approach and demonstrate its effectiveness both theoretically and empirically. In another study, Nayar studies how to optimize resource allocation for algorithm selection by down-sampling data. However, the author found that additional studies needed in order to have a reliable algorithm selection in compared to simple random sampling. In Section 4.2, we demonstrate that our proposed approach is both theoretically and empirically reliable compared to SRS when applied to a large number of datasets. The results can have a broader impact due to the widespread use of SRS. Our proposed approach not only offers a superior and efficient sampling method with respect to the PDF of the target variable(s) but also provides a performance gain prediction on regression task by selecting an optimized algorithm based on sub-sampled data. Additionally, there have been few attempts by machine-learning researchers and

practitioners (Boehmke & Greenwell, 2019) to manually handle regression target variables using data engineering and feature engineering as two use cases to enhance model performance. For example, mapping each cluster of data to a class might improve model predictions. However, to the best of our knowledge, neither researchers from academia nor the industry sector have focused on automating this process (instead of defining a custom class), nor have they studied the effects of hyperparameter selections, such as the number of strata. Our comprehensive evaluation in this study can bridge the gap between the practical use case of regression stratified sampling and manual feature/data engineering.

3. Preliminary

Let $\mathcal{D}$ be a given input tabular dataset to a tabular AutoML. We define the distribution of the target variables ($\mathcal{Y}$) for a regression task as $\mathbb{D}_{\mathcal{Y}}$ where $\mathbb{D}_{\mathcal{Y}} := \mathbb{R}^{s\times k}$ and s denotes the number of stratum of $\mathcal{Y}^{(k)}$. We consider a single target output when $k = 1$, and a multi-target outputs when $k > 1$. Let M be a Bayesian model in a supervised setting for the given input $\mathcal{X}$ to predict $\mathcal{Y}$ with a parameter of θ with $\mathbb{D}_{\mathcal{Y}}$ distribution as follows.

$$\mathcal{P}(y, \theta|x) = \mathcal{P}(y|x, \theta)\mathcal{P}(\theta) \tag{1}$$

Since we have a Bayesian model, parameters follow a distribution of $\mathcal{P}(\theta)$ (Kirsch & Gal, 2022) where θ is latent parameters as proven by Harrison et al.. The prediction denotes as $\mathcal{P}(y|x) = \mathbb{E}[\mathcal{P}(y|x, \theta)]$. Let $\mathcal{A}(.)$ denote a function that selects a machine-learning algorithm by maximizing the prediction via $\mathbb{E}[\mathcal{P}(y|x, \theta)]$ and returns an optimized algorithm from n number of algorithm choices (search space). Our objective is to identify an optimized algorithm, an unknown function, based on the distribution of the posterior predictive density. Let $\mathcal{PDF}(f(\cdot))$ represent the probability density function of the unknown function $f(\cdot)$. For our purposes, we consider $\mathcal{PDF}(\cdot)$ as equivalent to $\mathbb{D}$, and $\mathbb{D}_{\hat{\mathcal{Y}}}$ is the probability density function of the predicted values for $f(\mathcal{X}^{\rho})$, where $\mathcal{X}^{\rho} \in \mathcal{X}$ and ρ denotes a subset (sampled) of observations. Our hypothesis in this study to be tested is $\mathcal{PDF}(f(\mathcal{X}^{\rho})) \approx PDF(f(\mathcal{X}))$ if $PDF(\mathcal{Y}^{\rho}) \approx \mathcal{PDF}(\mathcal{Y})$. In another word, if we select a set of sample data ($\mathcal{X}^{\rho}$) where its target variable(s) ($\mathcal{Y}^{\rho}$) has the same distribution to the full-scale target variables ($\mathcal{Y}$), the distribution of prediction differences between full-scale data and sampled data are close to zero. Finally, the distribution of predictions for a subset of data of $f(\mathcal{X}^{\rho})$ follows the same distribution of $f(\mathcal{X}^{\rho})$, $\mathbb{D}_{\mathcal{Y}}$ with a parameter distribution of $f(\mathcal{P}(\theta))$ as indicated in Eq. 1.

4. Approach

In this section, we introduce our sampling approach, which defines the $PDF(\cdot)$ function to select a subset of continuous target variable(s). The function draws samples from $\mathcal{D}$, which is the posterior distribution of $\mathcal{Y}$, and defines $\mathcal{X}^{\rho}$ and $\mathcal{Y}^{\rho}$ for sub-sampled feature and target variables, respectively.

Problem Definition. Our objective is to select an algorithm that achieves superior performance on the $\mathcal{D}$ dataset. In practice, a finite set of models $[\mathcal{M}^1, \mathcal{M}^2, ..., \mathcal{M}^n]$ can be trained from $\mathcal{D}$ where each $\mathcal{M}^i$ is produced by fitting a machine-learning algorithm to $\mathcal{D}$ (including different hyperparameters). $\mathcal{M}^i$ can be either an atomic model (i.e., a linear regression) or an ensemble model (i.e., a stack of atomic models or any other ensemble structure) for tabular data. Some studies, such as Zhang et al. and (Erickson et al., 2020) have found that decoupling the algorithm from hyperparameter optimization is beneficial, especially for tabular data. There are two problems that need to be addressed. First, how to construct $\mathcal{D}^{\rho}$ with minimal distribution distance to $\mathcal{D}$ so that we can test our hypothesis. Second, which algorithm is the optimized choice? For the first problem, we introduce Regression Stratified Sampling (RSS) in the next section. This method divides the continues target variables into a set of strata, and a subset of each stratum is selected to ensure that it constructs minimal distribution distance between $PDF(\mathcal{Y}^{\rho})$ and $\mathcal{PDF}(\mathcal{Y})$. For the second problem, we define a loss function $\mathcal{L}(\cdot)$, where a model based on the selected optimized algorithm minimizes the distribution distance between $\mathcal{D}$ and $\mathcal{D}^{\rho}$. Let $\mathcal{M}^O$ be an optimized model that minimizes a loss function, and let a model be selected via $\mathcal{A}(\cdot)$, where $\mathcal{A} = \frac{1}{m}\sum_{i=1}^{m}\mathcal{L}_i(\cdot)$ and m represents the number of sample data in the full-scale dataset. $\mathcal{M}^O$ selects an algorithm from a search space of n algorithm choices including different hyperparameters based on the loss function as follows.

$$\mathcal{M}^o = \underset{i=[1,..,n]}{\operatorname{argmin}}(\mathcal{A}(\mathcal{L}_i(\mathcal{D}))) \tag{2}$$

The loss function $\mathcal{L}_i(\mathcal{D})$ defines the distribution error to the full-scale dataset for the ith model ($\mathcal{M}^i$). Similarly, a model can be selected on sampled data ($\mathcal{D}^{\rho}$) as follows.

$$\mathcal{M}^{\rho} = \underset{i=[1,..,n]}{\operatorname{argmin}}(\mathcal{A}(\mathcal{L}_i(\mathcal{D}^{\rho}))) \tag{3}$$

In an ideal case, where $\mathcal{M}^o = \mathcal{M}^{\rho}$, this indicates that the algorithm selection on $\mathcal{D}^{\rho}$ is equivalent to an optimized algorithm selection on $\mathcal{D}$ *based on given search space*. The ultimate loss function can be defined as follows.

$$\mathcal{L} = \mathcal{PDF}(f(\mathcal{X}^{\rho})) - PDF(f(\mathcal{X})) \tag{4}$$

The loss function computes the distribution divergence between the probability density function (PDF) error distribution of the target variable(s) in the full-scale data and

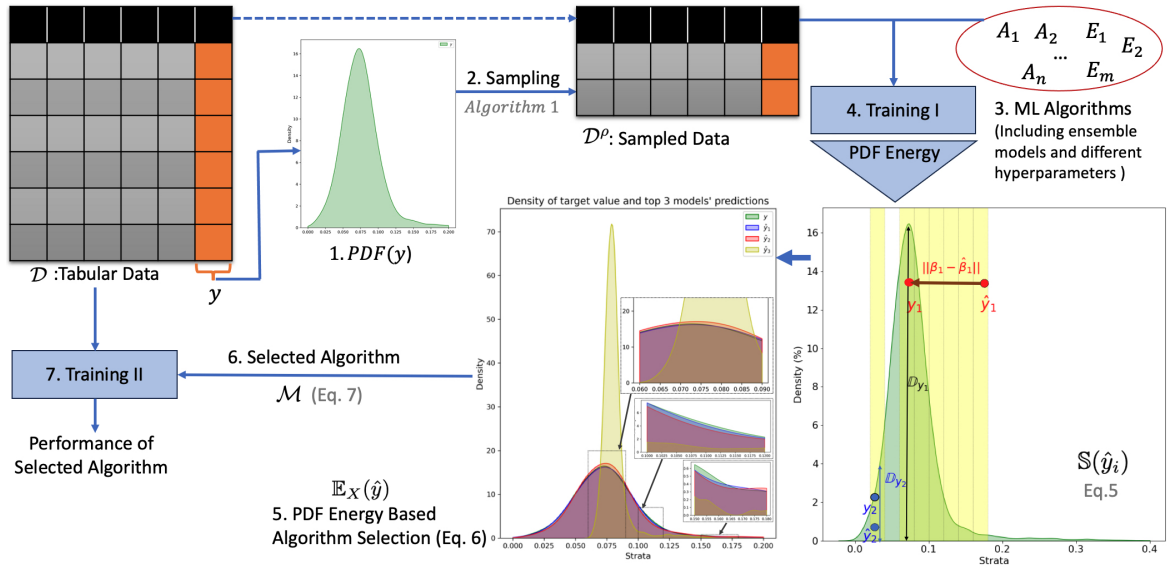


Figure 1. Overview of the proposed approach for algorithm selection in tabular AutoML

the distribution of predicted values for each trained model. Various methods exist for computing the differences between two distributions, and Kullback-Leibler (KL) divergence (Csiszár, 1975) is a widely employed approach. Note that pointwise KL-divergence for Eq. 4 can be defined as $\mathcal{L}^{KL}(p,q) = \int p(x^\rho) \log \frac{p(x^\rho)}{q(x)}\, dx$ where $p_i(x^\rho)$ is the PDF of the predicted distribution for model i and $q(x)$ is the PDF of the true distribution. In the next section, we introduce the concept of *PDF Energy*, which can be employed to identify the minimal distribution error between the target variable in full-scale data and the distribution of predictions from all trained models based on algorithm search space candidates. In Section 4.2, we empirically evaluate our proposed *PDF Energy* against KL where both methods aim to identify the closest distributions based on prediction on sampled data.

4.1. Regression Stratified Sampling

Figure 1 shows the overall process of our proposed approach. In the Step 1, we produce PDF of each target variable and use Algorithm 1 in Step 2 to draw samples from $\mathcal{D}$. In this algorithm, Line 2-4 split each target variable into a set of number of strata ($||\beta||$) where β is a hyperparameter in our algorithm. Next, Line 6-9 represent processing of multi-target outputs by computing PDF on each target variable and finding intersection of data point per strata across all target variables. For instance, if $[d^1, d^5] \in \beta_2^1$ (where d^1 refers to 1^{st} data point which is in range of β_2^1, and it refers to 1^{st} stratum of the 2^{nd} target variable) and $[d^4, d^5] \in \beta_1^1$, therefore $\overrightarrow{UID}_{\mathcal{B}} = [d^5]$ (a candidate for the 1^{st} stratum across two target variables). In the case of a single target, $\overrightarrow{UID}_{\mathcal{B}}$ applies only to one target variable. Finally, Lines 11-13 utilize uniform random sampling to achieve a proportional allocation of elements (ρ) in each intersection stratum ($\overrightarrow{UID}_{\mathcal{B}}$), where ρ and β are two hyperparameters. In Step 3 of Figure 1, a set of algorithms, including different hyperparameters, can be listed as the search space. An algorithm may produce an atomic model (A_i) or an ensemble model (E_j). Each algorithm from the search space can be applied to $\mathcal{D}^\rho$ in Step 4. Note that since $||\mathcal{D}^\rho|| \ll ||\mathcal{D}||$, this step can be performed quickly in compared to Step 7. Furthermore, ρ can be dynamically selected based on $||\mathcal{D}||$ (i.e., $\rho = 0.01$ or $\rho = 0.3$ if $\mathcal{D}$ is a large dataset or a small dataset, respectively).

Why Regression Stratified Sampling?

In practice, as shown in Eq. 4, a trained model in Step 4 is expected to generate a PDF distribution with minimal divergence to the PDF distribution of the original data, including outliers. We utilize the upper bound value of the total number for drawing samples in each stratum, as described in Line 11 of Algorithm 1, to achieve two main goals: i) generating a distribution with minimal distance to the full-scale target variable(s); ii) covering edge cases because some algorithms are sensitive to outliers (Sugiyama & Borgwardt, 2013; Lucic et al., 2016). Consequently, due to the coverage of edge cases, sensitive algorithms have a chance of being selected. In addition, RSS aims to maximize the chance of exploration by utilizing sampling and efficiently selecting an algorithm.

Limitation. In a multi-target regression task, Lines 6-9 of Algorithm 1 determine the intersection of data points across all strata for all target variables. In a scenario where the number of samples is limited and ρ is also small, the number

of drawn samples for $\overrightarrow{UID}_{\mathcal{B}}$ will be constrained. In such cases, implementing an automated approach to increase the sampling ratio (ρ) can help alleviate this limitation.

Algorithm 1 Regression Stratified Sampling Procedures

procedure REGRESSION_STRATIFIED_SAMPLING($\mathcal{X}, \mathcal{Y}, \beta, \rho$)
▷ Sampling data from a dataset with continuous target variable(s) $\mathcal{Y}$, the list of strata β, and a sampling ratio (ρ)
  for y in $\mathcal{Y}$ **do**: ▷ process each target of given dataset
    $\mathcal{W} = \frac{|max(y)-min(y)|}{||\beta||}$
    $\mathcal{P}_y, ID_y := GET\_DENSE(y, \beta, \mathcal{W})$

  for y in $\mathcal{Y}$ **do**:
    for i in range$[Min(y), Max(y), \mathcal{W}]$ **do**:
      ▷ find common element across targets' strata
      $\overrightarrow{UID}_{\mathcal{B}} := \bigcap \overrightarrow{ID}_y^{\beta_i}$

  ▷ ρ percentage of proportional allocation through drawing uniform random samples from each stratum
  $\overrightarrow{SID}_{\mathcal{B}} \sim$ Uniform$(y_i) \quad | \quad \forall y_i \in \overrightarrow{UID}_{\mathcal{B}}$
  $n = \left\lceil \frac{||\overrightarrow{UID}_{\mathcal{B}}||}{\rho} \right\rceil$ ▷ Covers edge cases within each stratum
  $\mathcal{X}^{\rho} = \mathcal{X}[i] \quad \forall\, i$ in $\overrightarrow{SID}_{\mathcal{B}}$
  $\mathcal{Y}^{\rho} = \mathcal{Y}[i] \quad \forall\, i$ in $\overrightarrow{SID}_{\mathcal{B}}$
return $\mathcal{X}^{\rho}, \mathcal{Y}^{\rho}$
procedure GET_DENSE($y, \beta, \mathcal{W}$)
▷ Obtain the density of target variable, partition the target t into β strata, returns the density and elements within each stratum.
  for i in range$[Min(y), Max(y), \mathcal{W}]$ **do**
    $\mathcal{P}^{\beta} := PDF(y_i) \quad$ s.t. $\quad \beta_i \leq y_i < \beta_{i+1}$
    $\overrightarrow{\mathcal{ID}}^{\beta} := Index(y_i) \quad$ s.t. $\quad \beta_i \leq y_i < \beta_{i+1}]$
return $\mathcal{P}, \overrightarrow{ID}$

4.2. PDF Energy Metric

In Step 5 of Figure 1, our objective is to identify the top-performing model that exhibits the best performance on both $\mathcal{X}$ and $\mathcal{X}^{\rho}$ in accordance with our sampling approach. In other words, the selection of a model is crucial such that it demonstrates superior performance on both the full-scale dataset and a subset of the data, where both possess a minimal probability density function (PDF) distribution distance. For instance, as depicted in the figure, a comparison is made between the PDF distribution of a target variable (y) and the predictions of three trained models ($[\hat{y}_1, \hat{y}_2, \hat{y}_3]$) and it aims to find the lowest distribution distance to the original target. *The underlying hypothesis of this study is that opting for the minimal PDF distribution divergence will guide us towards an optimized algorithm, given that the sampled data replicate the same PDF distribution as the full-scale data*. To select an optimized algorithm for $\mathcal{X}^{\rho}$ and $\mathcal{Y}^{\rho}$, we define a new evaluation metric that extends the concept of relative entropy. This metric selects a model based on the distribution distance between $f(\mathcal{X})$ and the posterior, $f(\mathcal{X}^{\rho})$ as follows.

$$\mathbb{S}(\hat{y}_i) = \begin{cases} \mathbb{D}(y_i) & \beta_i \leq \hat{y}_i < \beta_{i+1} \\ -\mathbb{D}(y_i) * ||\beta_i - \hat{\beta}_i|| & \hat{y}_i < \beta_i \text{ or } \hat{y}_i > \beta_{i+1} \end{cases} \tag{5}$$

where $\mathbb{D}(y_i)$ denotes the density of y_i according to $\mathcal{Y}$ within the range of $[\beta_i, \beta_{i+1}]$, and $||\beta_i - \hat{\beta}_i||$ represents the distribution distance between the actual target strata (β_i) and the predicted target strata ($\hat{\beta}_i$) with respect to the strata distance (strata error). This equation aims to penalize the prediction error of strata based on the probability density function (PDF) of the original data. This equation computes higher energy when $f(\mathcal{X}^{\rho})$ has lower strata allocation error w.r.t. the density of each strata in $\mathcal{Y}$. We explain a simple example of this computation in Appendix B. Next, we compute *PDF Energy* in Step 5 as follows.

$$\mathbb{E}_X(\hat{y}) = \sum_{k=1}^{||\mathbb{D}^{\rho}||} \mathbb{S}(\hat{y}_k) \tag{6}$$

Finally, we identify an optimized model ($\mathcal{M}$) as follows.

$$\mathcal{M} = \underset{\gamma \in \Gamma}{\text{argmax}}(\mathbb{E}_X^{\gamma}(\hat{y})) \tag{7}$$

where Γ corresponds to the total number of algorithms in the search space. n optimized model corresponds to an algorithm and its hyperparameters, which are used for training. In Appendix C, we prove that selecting a model based on RSS results in a better *PDF Energy* score. In the experiment section, we compare our proposed *PDF Energy* score against the Kullback-Leibler (KL) divergence(Csiszár, 1975) for algorithm selection where KL divergence and *PDF Energy* aims to find minimal distance between distributions of PDFs (y versus $\hat{y^i}$ for $i \in [1, .., n]$).

Once an algorithm is chosen using the *PDF Energy* method, the algorithm is then applied to $\mathcal{D}$ in Step 7. Through the utilization of sample data rather than the entire dataset, a Bayesian model-based approach (Eq. 1) effectively incorporates various algorithms with distinct hyperparameters from the search space. This approach facilitates result comparison using the *PDF Energy* metric.

5. Experiments

Benchmark. For the purpose of reproducibility, we utilized a publicly available benchmark from OpenML(Vanschoren et al., 2014) that was introduced by Grinsztajn et al.. We designed two sets of sub-benchmarks: Benchmark #1, which consists of 31 datasets, and Benchmark #2, which includes 14 real-world datasets for regression tasks. While there is overlap between these two benchmarks, we employed

distinct seeds to generate two sets of comprehensive benchmarks for each sub-experiment. The dataset details including the random seeds, and the GitHub repository for result reproducibility, are explained in Appendix D.2.

Search Space. The search space includes 12 widely used machine-learning algorithms for tabular data with hyperparameters. In addition, we define 2 different ensembles models on top of atomic models where it uses 2 different estimators of *Ridge regression* (Hoerl & Kennard, 1970) and Stochastic Gradient Descent Regressor (*SGDRegressor*) (Zhang, 2004) to stack top 3 outperformed atomic models from search space with respect to experiment configuration (i.e., R^2 metric or PDF Energy). The ensemble models are dynamically (on-the-fly) selected based on the top 3 algorithms and built based on experiment configuration. Refer to Appendix D.3.1 for a comparison of our search space against other tabular AutoML tools which indicates that our search space is competitive.

Sampling Ratio(ρ). All experiments were conducted on 25% of hold-out data. We chose a set of reasonable sampling ratios, ranging from 20% to 40%, for our experiments. RSS can be applied to larger datasets with lower sampling ratios (i.e., 1%), as it only requires sufficient examples per stratum.

We repeat each experiment on different configurations (i.e., β, ρ, time budget, and evaluation metric), and report two metrics of R^2/RMSE on test after the algorithm selection. In each experiment set, we partition each dataset D into D^{Train} and D^{Test} using a 75-25 split ratio based on a random seed number. We defines two set of experiments as follows.

5.1. Algorithm Selection

This experiment aims to evaluate: i) the proposed sampling algorithm, and ii) the performance of PDF Energy with different hyperparameters by utilizing Benchmark #2.

i) Sampling. The objective of this experiment is assessing how the use of different sampling algorithms may influence the final outcome in terms of performance on hold-out data. We define three methods for sampling as follows.

- Random Sampling: This method utilizes a simple random sampling (SRS) method to draw sample from full-scale data by employing a widely used implementation of Scikit-Learn (Ojala & Garriga, 2010; Pedregosa et al., 2011) [1].

- PDF Sampling: This method draw samples based on our regression stratified sampling as shown in Algorithm 1, which guarantees in each stratum: **i)** an equal opportunity for selecting samples is provided based on the total number of strata; **ii)** draw sample for possible edge cases (n) as mentioned in Algorithm 1 (Line 11).

[1]Use *model_selection.train_test_split* API call

- Dynamic Stratified PDF Sampling: This method utilizes *PDF Sampling*, but instead of assigning a fixed value for the number of strata, we use $n_h = N_h \frac{KS_h}{\sqrt{C_h}}$ as suggested in (Kish, 2011) to compute the size of strata per dataset dynamically. In this equation, n_h dynamically determines the number of strata per input dataset. The method then employs the highest *PDF Energy* to sort and select the best possible model. Basically, this method utilizes the standard deviation of the target value to determine the number of strata (β). The computed β per each dataset/seed is listed in Appendix (Table 6).

ii) Evaluation Methods. To evaluate each algorithm in Step #4 with respect to the sampling method and ratio (ρ), we assess the prediction results using three evaluation methods in Step #5: the Standard Metric, KL, and PDF Energy. The results can be arranged in ascending order for RMSE and KL, and in descending order for R^2 and *PDF Energy*. Finally, we record the error rate or score based on two standard metrics: Root-mean-square deviation (*RMSE*)(Hyndman & Koehler, 2006)(Armstrong & Collopy, 1992) and R^2 for each selected model. By evaluating all methods based on *RMSE/*R^2 at the end, we ensure a fair comparison across all different sampling and evaluation methods for each input dataset.

- Metric: We use *RMSE* and R^2 as two baseline evaluation methods. These metrics indicate the error rate and coefficient of determination across all predictions, respectively. An algorithm is selected based on the lowest and highest values on sampled data for *RMSE* and R^2 metrics, respectively.

- KL: We use Kullback-Leibler (KL) divergence (Csiszár, 1975) as another baseline where an algorithm is selected based on the lowest relative entropy of $\mathbb{D}_{\mathcal{Y}}$ w.r.t. $\mathbb{D}_{\hat{\mathcal{Y}}}$. There are several approaches for KL constructions; note that in this study, similar to Moreno et al., we report the evaluation based on $\mathbb{D}_{KL}(\mathbb{D}_{\mathcal{Y}}||\mathbb{D}_{\hat{\mathcal{Y}}})$ to recommend an algorithm. This method compares PDF distribution of each candidate trained model against distribution of target(s) in full-scale dataset and it selects the lowest value of $\mathbb{D}_{KL}(.)$ as the best model. Note that once a model is selected based on *KL*, then *RMSE/*R^2 on test data is reported.

- PDF Energy: We employ Eq. 6 to select an algorithm, where the highest energy value indicates the lowest *PDF Error* predictions. It is important to note that the value in the *PDF Energy* does not represent the actual energy value; instead, it reports the *RMSE/*R^2 score of the top-selected algorithm prediction on test data.

5.2. AutoML Evaluation

To evaluate our proposed approach in a real-world, end-to-end scenario, we utilize Benchmark 1 with 31 datasets

primarily for AutoML evaluation, using 10 seeds because AutoML is known to be noisy. We define a **baseline** using a meta-learning-based open-source AutoML tool, SapientML Saha et al., which allows us to conduct feature engineering on the input dataset. Baseline mainly selects a model based on a pre-trained model insist of our method that explore data.In addition, we utilized the most popular and widely used AutoML tools for structured data, including FLAML (Wang et al., 2021), H2O (LeDell & Poirier, 2020), TPOT (Olson & Moore, 2016), AutoScikitLearn (Feurer et al., 2015), and AutoGluon (Erickson et al., 2020), in our AutoML evaluation. Finally, we utilize our *Regression Stratified Sampling* (**RSS**) approach where we add our proposed *PDF sampling* to recommend an algorithm with respect to the highest *PDF Energy*. In the context of AutoML experiments, we use hyperparameters of $\beta = 100$ (the number of stratum) and $\rho = 0.3$ (sampling ratio). There are no limitations on each AutoML tool besides specifying the assigned training data as input and the time budget. We present the average R^2 scores across all random seeds for each dataset and time budgets of (in seconds): $[30, 60, 120, 180, 300, 600, 1200]$. Given that the primary aim of this study is to evaluate AutoML platforms against our PDF sampling approach, our main focus lies on lower time budgets as a stress test. However, it is important to note that in general, these shorter time constraints do not hinder AutoML from producing at least one model.

5.3. Experiment Results

Algorithm Selection Results. Figure 2 shows a comprehensive performance evaluation of Avg. R^2 and its standard deviation on a hold-out data on benchmark #2 with 3 random seeds based on SRS (random) and PDF sampling methods with a sampling ratio between 20% to 40% with 4 different hyperparameters of β. The results indicate that PDF Sampling and utilizing PDF Energy outperformed SRS sampling by considering $\beta = 100$ and $\beta = 1000$. Table 1 shows more detail of this experiment results with two metrics of R^2*/RMSE*. Each value is averaged across 3 different seeds on each sampling ratio of 20%, 30% and 40%. $\beta = 100$ indicates the best configuration across all datasets where our approach (*PDF*) is selected as a champion (the best score) for 10 out of 14 datasets. Furthermore, the average performance of both RMSE/R^2 scores across all datasets show that *PDF* outperforms random sampling across all stratum configurations. The results of lower sampling ratio (0.2) suggest that the models are inconsistent and likely struggling with insufficient data. However, at a sampling ratio of 0.3, the model aims to achieve better generalization compared to SRS. We can also observe a potential overfitting issue with the lower bound hyperparameter of $\beta = 10$. The results indicate that $\beta \geq 100$ leads to a more generalized model selection.

See Appendix E.1 for fined-grain results and further discussions.

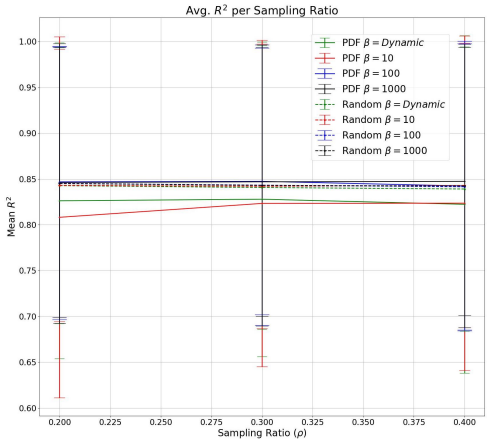


Figure 2. A Comparison of Average R^2 Scores between Random Sampling and PDF Sampling with four different values of hyperparameters for β across Benchmark 2 using three seeds. Note that the y-axis starts at 0.6 for better readability of results.

Relative Entropy. Figure 3 shows the ultimate performance of R^2 on a hold-out data of 14 different datasets where a model is selected based on *relative entropy* comparison based on i) Kullback-Leibler divergence (KL), and ii) *PDF Energy* computation. Each sub-figure shows different values of hyperparameter of β in the range of $[10, 100, 1000$,dynamic$]$ where in dynamic setting, $\bar{\beta} = 6$. The overall average ($\overline{PDF}$) on 4 different hyperparameters, shows that the *PDF Energy* algorithm selection is more effective than using KL (PDF championed in 4 out of 4). Furthermore, KL divergence is asymmetric; however, *PDF Energy* is a symmetric approach and it can be used as a metric to evaluate relative entropy. Detailed results for different sampling ratios (ρ) are explained in Appendix E.3.

Why does PDF Energy outperform standard metrics and KL? PDF Energy combines both metric evaluation and bin density error by penalizing bin errors with respect to the density of the original data (Eq. 5). The evaluation approach computes prediction errors across different algorithms' predictions while also finding the the lowest prediction distribution distance to the expected distribution of the original data. Note that the sampled data have been drawn from the original data, producing a similar distribution as shown in Algorithm 1.

AutoML Evaluation. Table 2 presents the evaluation results based on the R^2 metric across Benchmark #1 (31 datasets), considering 10 random seeds per dataset per time budget. Our objective here is to validate our proposed approach under stringent time budget conditions. These results highlight the following key findings. (**i**) Our approach outperforms across all stress time budget constraints; (**ii**) By utilizing

	β		Final Evaluation on 25% hold-out data							
			10		100		1000		Dynamic Stratified	
	Sampling Method	Eval. Method	$R^2\uparrow$	RMSE↓	$R^2\uparrow$	RMSE↓	$R^2\uparrow$	RMSE↓	$R^2\uparrow$	RMSE↓
Average	Random Sampling	Metric	0.8425 ± 0.012	9.6812 ± 0.359	0.8425 ± 0.012	9.6812 ± 0.3590	0.8425 ± 0.012	9.6812 ± 0.359	0.8425 ± 0.012	9.6812 ± 0.359
	PDF Sampling (our)	PDF Energy (our)	0.8183 ± 0.024	9.5147 ± 0.356	$\mathbf{0.8455 \pm 0.016}$	$\mathbf{9.5432 \pm 0.363}$	$\mathbf{0.8468 \pm 0.01}$	$\mathbf{9.6453 \pm 0.332}$	0.8254 ± 0.036	9.6757 ± 0.87
Total Number of Champions		Metric	3	3	4	4	5	5	5	5
(Top rank across 14 possible datasets)		Equal Results	2	2	0	0	1	1	1	1
		PDF Energy (our)	**9**	**9**	**10**	**10**	**8**	**8**	**8**	**8**

Table 1. Performance Comparison on Benchmark #2; "Eval. Method" refers to the metric evaluation method for algorithm selection; the values are averaged across 3 time budgets of $\rho = [0.2, 0.3, 0.4]$ with 3 seeds per ρ for algorithm selection through *PDF Sampling* (our approach), $\beta = [10, 100, 1000, \text{Dynamic}]$. The best score per setting is marked in **bold**, based on the lowest RMSE and the highest R^2 in each setting. See results per hyperparameter/dataset/time budget in Appendix (Table 13).

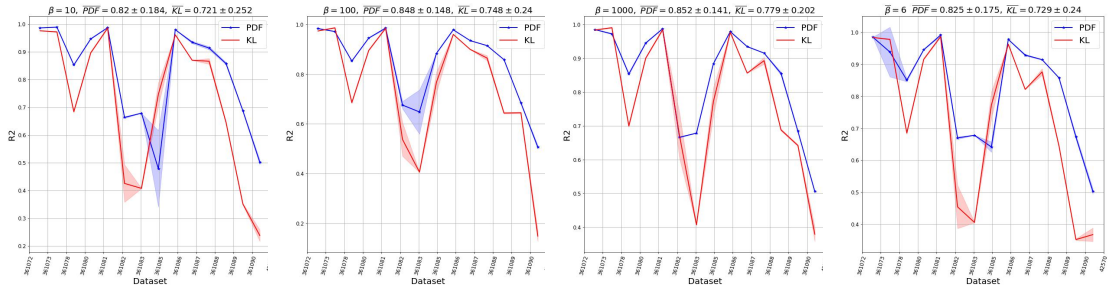


Figure 3. R^2 Performance comparison (higher is better) between KL and PDF (our) for algorithm selection based on relative entropy across Benchmark #2 with different hyperparameter of β; Values are averaged of 9 experiments (3 seeds per each sampling ratios of $[0.2, 0.3, 0.4]$). See Appendix (Figure 7) for detailed results per Sampling Ratio (ρ).

Time (s)	Baseline	MLJAR	FLAML	AutoSKLearn	H2O	TPOT	AutoGluon	RSS (our)
30	0.7601 ± 0.28	0.7662 ± 0.28	0.8069 ± 0.21	0.6607 ± 0.35	0.7885 ± 0.22	0.6597 ± 0.33	0.7047 ± 0.32	$\mathbf{0.8222 \pm 0.21}$
60	0.7663 ± 0.27	0.7764 ± 0.27	0.8152 ± 0.2	0.7179 ± 0.3	0.8004 ± 0.21	0.689 ± 0.32	0.7341 ± 0.3	$\mathbf{0.8239 \pm 0.2}$
120	0.7629 ± 0.28	0.7691 ± 0.28	0.8177 ± 0.2	0.7518 ± 0.27	0.8039 ± 0.22	0.7506 ± 0.27	0.7751 ± 0.27	$\mathbf{0.8242 \pm 0.2}$
180	0.7598 ± 0.28	0.7365 ± 0.28	0.819 ± 0.2	0.7819 ± 0.24	0.8054 ± 0.22	0.7618 ± 0.26	0.777 ± 0.27	$\mathbf{0.8243 \pm 0.2}$
300	0.761 ± 0.28	0.7277 ± 0.28	0.8217 ± 0.2	0.7923 ± 0.23	0.8131 ± 0.21	0.7748 ± 0.25	0.7716 ± 0.28	$\mathbf{0.8262 \pm 0.2}$

Table 2. AutoML evaluation (R^2 score - higher is better) across 31 datasets (Benchmark #1); RSS utilize feature engineering by Baseline, and proposed PDF Sampling/PDF Energy for algorithm selection with $\rho = 0.3$ and $\beta = 100$; Each data point averaged over 310 experiments (10 seeds) with a standard deviation of $\pm$; **Bold** values show the highest score per time budget (row).

the *PDF Sampling* and *PDF Energy* metric, we can identify an optimized algorithm within tight time budget constraints. An increase in the time budget results in a minor performance enhancement, indicating successful algorithm selection even under more restricted time limits (more sustainable); (**iii**) The addition of time budget enables our approach to enhance its performance when the most optimized algorithm selection was missed due to the time constraints (i.e., chance of other algorithm explorations); (**iv**) Across all AutoML tools, our proposed approach achieves the highest overall average R^2 score among all other AutoML tools, and the lowest standard deviation ($std = \pm 0.2$) alongside FLAML. We defer the details and additional AutoML experiments with higher time budgets in the Appendix (Table 12).

6. Conclusion

We examined the effectiveness of the stratified sampling approach on regression tasks for algorithm selection, marking the first instance of such a study. We introduced RSS by utilizing Probability Density Function (PDF) for both sampling and evaluation. Our extensive evaluation results revealed that our proposed approach outperformed Simple Random Sampling (SRS) in algorithm selection across 31 distinct datasets. Our proposed PDF Energy metric identifies an algorithm by combining the concept of relative entropy method and an evaluation metric, where it showed a superior performance compared to the Kullback-Leibler divergence-based selection. This study concludes with a comprehensive empirical evaluation encompassing an end-to-end scenario on 31 datasets using 6 popular tabular AutoML tools. The overall evaluation results indicate that by utilizing *PDF Sampling* algorithm in conjunction with the *PDF Energy* metric, our approach achieves top rankings in identifying an optimized algorithm under time constraints.

References

Armstrong, J. S. and Collopy, F. Error measures for generalizing about forecasting methods: Empirical comparisons. *International journal of forecasting*, 8(1):69–80, 1992.

Boehmke, B. and Greenwell, B. M. *Hands-on machine learning with R*. CRC press, 2019.

Chen, T., He, T., Benesty, M., Khotilovich, V., Tang, Y., Cho, H., Chen, K., et al. Xgboost: extreme gradient boosting. *R package version 0.4-2*, 1(4):1–4, 2015.

Chui, M., Manyika, J., Miremadi, M., Henke, N., Chung, R., Nel, P., and Malhotra, S. Notes from the ai frontier: Insights from hundreds of use cases. *McKinsey Global Institute*, pp. 28, 2018.

Conrad, F., Mälzer, M., Schwarzenberger, M., Wiemer, H., and Ihlenfeldt, S. Benchmarking automl for regression tasks on small tabular data in materials design. *Scientific Reports*, 12(1):1–14, 2022.

Csiszár, I. I-divergence geometry of probability distributions and minimization problems. *The annals of probability*, pp. 146–158, 1975.

Demšar, J. Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine learning research*, 7:1–30, 2006.

Dorogush, A. V., Ershov, V., and Gulin, A. Catboost: gradient boosting with categorical features support. *arXiv preprint arXiv:1810.11363*, 2018.

Erickson, N., Mueller, J., Shirkov, A., Zhang, H., Larroy, P., Li, M., and Smola, A. Autogluon-tabular: Robust and accurate automl for structured data. *arXiv preprint arXiv:2003.06505*, 2020.

Feng, J., Xu, H., Mannor, S., and Yan, S. Robust logistic regression and classification. *Advances in neural information processing systems*, 27, 2014.

Feurer, M., Klein, A., Eggensperger, K., Springenberg, J., Blum, M., and Hutter, F. Efficient and robust automated machine learning. *Advances in neural information processing systems*, 28, 2015.

Feurer, M., Eggensperger, K., Falkner, S., Lindauer, M., and Hutter, F. Auto-sklearn 2.0: The next generation. *arXiv preprint arXiv:2007.04074*, 24, 2020.

Fisher, R. A. Design of experiments. *British Medical Journal*, 1(3923):554, 1936.

Gijsbers, P., Bueno, M. L., Coors, S., LeDell, E., Poirier, S., Thomas, J., Bischl, B., and Vanschoren, J. Amlb: an automl benchmark. *Journal of Machine Learning Research*, 25(101):1–65, 2024.

Gorishniy, Y., Rubachev, I., Khrulkov, V., and Babenko, A. Revisiting deep learning models for tabular data. *Advances in Neural Information Processing Systems*, 34: 18932–18943, 2021.

Grinsztajn, L., Oyallon, E., and Varoquaux, G. Why do tree-based models still outperform deep learning on typical tabular data? In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2022.

Harrison, J., Sharma, A., and Pavone, M. Meta-learning priors for efficient online bayesian regression. In *International Workshop on the Algorithmic Foundations of Robotics*, pp. 318–337. Springer, 2020.

He, X., Zhao, K., and Chu, X. Automl: A survey of the state-of-the-art. *Knowledge-Based Systems*, 212:106622, 2021.

Henry, G. T. *Practical sampling*, volume 21. Sage, 1990.

Hoerl, A. E. and Kennard, R. W. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.

Hollmann, N., Müller, S., and Hutter, F. Large language models for automated data science: Introducing caafe for context-aware automated feature engineering. *Advances in Neural Information Processing Systems*, 36, 2024.

Hyndman, R. J. and Koehler, A. B. Another look at measures of forecast accuracy. *International journal of forecasting*, 22(4):679–688, 2006.

Jin, H., Chollet, F., Song, Q., and Hu, X. Autokeras: An automl library for deep learning. *Journal of Machine Learning Research*, 24(6):1–6, 2023.

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., and Liu, T.-Y. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, 30, 2017.

Kirsch, A. and Gal, Y. Unifying approaches in active learning and active sampling via fisher information and information-theoretic quantities. *Transactions on Machine Learning Research*, 2022.

Kish, L. Survey sampling. In *Survey sampling*, pp. 643–643. 2011.

Kotthoff, L., Thornton, C., Hoos, H. H., Hutter, F., and Leyton-Brown, K. Auto-weka: Automatic model selection and hyperparameter optimization in weka. In *Automated machine learning*, pp. 81–95. Springer, Cham, 2019.

LeDell, E. and Poirier, S. H2o automl: Scalable automatic machine learning. In *Proceedings of the AutoML Workshop at ICML*, volume 2020, 2020.

Li, P., Chen, Z., Chu, X., and Rong, K. Diffprep: Differentiable data preprocessing pipeline search for learning over tabular data. *Proceedings of the ACM on Management of Data*, 1(2):1–26, 2023.

Liang, J., Meyerson, E., Hodjat, B., Fink, D., Mutch, K., and Miikkulainen, R. Evolutionary neural automl for deep learning. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 401–409, 2019.

Liberty, E., Lang, K., and Shmakov, K. Stratified sampling meets machine learning. In *International conference on machine learning*, pp. 2320–2329. PMLR, 2016.

Lucic, M., Bachem, O., and Krause, A. Linear-time outlier detection via sensitivity. *arXiv preprint arXiv:1605.00519*, 2016.

Mardia, K. Families of frequency distributions, 1973.

Meng, X. Scalable simple random sampling and stratified sampling. In *International Conference on Machine Learning*, pp. 531–539. PMLR, 2013.

Milo, T. and Somech, A. Automating exploratory data analysis via machine learning: An overview. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, pp. 2617–2622, 2020.

Moreno, P., Ho, P., and Vasconcelos, N. A kullback-leibler divergence based kernel for svm classification in multimedia applications. *Advances in neural information processing systems*, 16, 2003.

Mustafa, A. and Rahimi Azghadi, M. Automated machine learning for healthcare and clinical notes analysis. *Computers*, 10(2):24, 2021.

Nam, G., Yoon, J., Lee, Y., and Lee, J. Diversity matters when learning from ensembles. *Advances in Neural Information Processing Systems*, 34:8367–8377, 2021.

Nayar, N. Data subsampling for model selection in automl frameworks. 2021.

Nti, I. K., Adekoya, A. F., and Weyori, B. A. A systematic review of fundamental and technical analysis of stock market predictions. *Artificial Intelligence Review*, 53(4): 3007–3057, 2020.

Ojala, M. and Garriga, G. C. Permutation tests for studying classifier performance. *Journal of machine learning research*, 11(6), 2010.

Olson, R. S. and Moore, J. H. Tpot: A tree-based pipeline optimization tool for automating machine learning. In *Workshop on automatic machine learning*, pp. 66–74. PMLR, 2016.

Pearson, K. Contributions to the mathematical theory of evolution. *Philosophical Transactions of the Royal Society of London. A*, 185:71–110, 1894.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Pérez-Cruz, F. Kullback-leibler divergence estimation of continuous distributions. In *2008 IEEE international symposium on information theory*, pp. 1666–1670. IEEE, 2008.

Płońska, A. and Płoński, P. Mljar: State-of-the-art automated machine learning framework for tabular data. version 0.10.3, 2021. URL https://github.com/mljar/mljar-supervised.

Plońska, A. and Ploński, P. Mljar: State-of-the-art automated machine learning framework for tabular data. *Version 0.10*, 3, 2021.

Rao, C. R. Sir ronald aylmer fisher–the architect of multivariate analysis. *Biometrics*, 20(2):286–300, 1964.

Saha, R. K., Ura, A., Mahajan, S., Zhu, C., Li, L., Hu, Y., Yoshida, H., Khurshid, S., and Prasad, M. R. Sapientml: Synthesizing machine learning pipelines by learning from human-writen solutions. In *Proceedings of the 44th International Conference on Software Engineering*, ICSE '22, pp. 1932–1944, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450392211. doi: 10.1145/3510003.3510226. URL https://doi.org/10.1145/3510003.3510226.

Santu, S. K. K., Hassan, M., Smith, M. J., Xu, L., Zhai, C., Veeramachaneni, K., et al. Automl to date and beyond: Challenges and opportunities. *arXiv preprint arXiv:2010.10777*, 2020.

Shwartz-Ziv, R. and Armon, A. Tabular data: Deep learning is not all you need. *Information Fusion*, 81:84–90, 2022.

Sugiyama, M. and Borgwardt, K. Rapid distance-based outlier detection via sampling. *Advances in neural information processing systems*, 26, 2013.

Trost, J. E. Statistically nonrepresentative stratified sampling: A sampling technique for qualitative studies. *Qualitative sociology*, 9(1):54–57, 1986.

Vanschoren, J., Van Rijn, J. N., Bischl, B., and Torgo, L. Openml: networked science in machine learning. *ACM SIGKDD Explorations Newsletter*, 15(2):49–60, 2014.

Vincent, A. M. and Jidesh, P. An improved hyperparameter optimization framework for automl systems using evolutionary algorithms. *Scientific Reports*, 13(1):4737, 2023.

Wang, C., Wu, Q., Weimer, M., and Zhu, E. Flaml: A fast and lightweight automl library. *Proceedings of Machine Learning and Systems*, 3:434–447, 2021.

Wu, J., Chen, S., Zhao, Q., Sergazinov, R., Li, C., Liu, S., Zhao, C., Xie, T., Guo, H., Ji, C., et al. Switchtab: Switched autoencoders are effective tabular learners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 15924–15933, 2024.

Yakovlev, A., Moghadam, H. F., Moharrer, A., Cai, J., Chavoshi, N., Varadarajan, V., Agrawal, S. R., Idicula, S., Karnagel, T., Jinturkar, S., et al. Oracle automl: a fast and predictive automl pipeline. *Proceedings of the VLDB Endowment*, 13(12):3166–3180, 2020.

Yang, C., Akimoto, Y., Kim, D. W., and Udell, M. Oboe: Collaborative filtering for automl model selection. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 1173–1183, 2019.

Zhang, T. Solving large scale linear prediction problems using stochastic gradient descent algorithms. In *Proceedings of the twenty-first international conference on Machine learning*, pp. 116, 2004.

Zhang, Y., Zhou, Z., Yao, Q., and Li, Y. Efficient hyperparameter search for knowledge graph embedding. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2715–2735, 2022.

Zhu, B., Shi, X., Erickson, N., Li, M., Karypis, G., and Shoaran, M. Xtab: Cross-table pretraining for tabular transformers. *arXiv preprint arXiv:2305.06090*, 2023.

Zogaj, F., Cambronero, J. P., Rinard, M. C., and Cito, J. Doing more with less: characterizing dataset downsampling for automl. *Proceedings of the VLDB Endowment*, 14 (11):2059–2072, 2021.

A. Appendix Outline

The Appendix section of this study aims to define all the required details of our experiments, additional experiments, and provide all the necessary information for the reproducibility of our results. The artifacts from our study are available on GitHub at:https://marscod.github.io/Regression_Stratified_Sampling
This appendix is organized as follows.

- Appendix B shows a simple example of *PDF Energy* computation.
- Appendix C proofs that the selected algorithm by RSS computes an optimized *PDF energy* score.
- Appendix D shows the details of experiment setup that includes configuration environment for both algorithm selection and AutoML experiments (Appendix D.1) which includes our benchmarks detail (Appendix D.2), a comparison of algorithm search space between this study and AutoML tools, and the list of utilized APIs in our experiments for reproducibility (Appendix D.3).
- In Appendix E, we provide the detail of evaluation results that include β parameter values for Dynamic PDF Stratified Sampling, where the value vary in each experiment (Appendix E.2), relative entropy comparison where we provide macro analysis of comparison between PDF and KL (Appendix E.3), and Appendix E.4 shows the fine-grained details of our comprehensive AutoML evaluations per dataset/time budget on Benchmark #1 with 31 datasets, the list of AutoML assumptions, and a report on AutoML failure cases.
- Although the objective of our AutoML experiment is validating sampling approach under time stress constraint, Appendix F aims to answer the question on AutoML experiments with higher time budgets.
- Finally, Appendix G explains our disclaimers of this study.

B. Example of PDF Energy Computation

As explained in Eq. 5 and Eq. 6, *PDF Energy* is computed based on both the density of $\mathcal{Y}$ and $\widehat{y^{\rho}}$. In this section, we explain an example of this computation.

In Figure 4, the green curve depicts the density (percentage) of a single target ($\mathcal{Y}$) with two prediction values on the x-axis (height is ignored). The yellow boxes outline the strata. In this example, $\hat{y}_1$ is predicted incorrectly (due to incorrect strata allocation), while $\hat{y}_2$ is predicted correctly because it falls within the y_2 stratum. In this example, a penalty value equal to the density of the original stratum ($\mathbb{D}_1$) times the distribution distance between the two strata ($||\beta_1 - \hat{\beta}_1||$) is computed as a penalty for $\hat{y}_1$ prediction. However, $\hat{y_2}$ is predicted correctly (the same strata), a positive energy equivalent to the density of $\mathbb{D}_{y^2}$ is added.

C. Algorithm Selection

A selected algorithm by RSS computes *PDF energy score* as noted in Eq. 5- 7.

Proof. By considering the PDF of a single continuous target variable of y, the density can be defined as follows (Mardia, 1973).

$$\int_{-\infty}^{\infty} f_X(y)dy \quad = 1 \tag{8}$$

where f_X denotes to an unknown function which is produced by fitting $\mathbb{A}_\Gamma(X)$ and resulting $f_X(y) = \mathbb{A}_\gamma(X)$
$\gamma = \{1, 2, .., \Gamma\}$ where $\mathbb{A}_\gamma$ denotes γth algorithm, and Γ denotes the total number of algorithms in our search space. We can define PDF over target variable of y as follows.

$$\int_{\beta_1}^{\beta_2} f_X(y)dy + \int_{\beta_2}^{\beta_3} f_X(y)dy + .. + \int_{\beta_{L-1}}^{\beta_L} f_X(y)dy \quad = 1 \tag{9}$$

where β_L refers to each strata and L denotes the total number of stratum for the target y values.

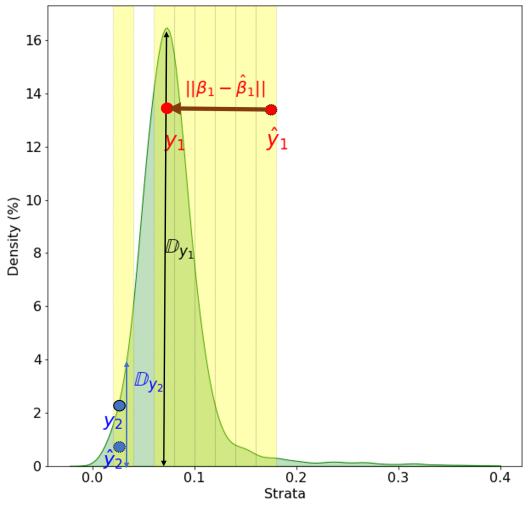


Figure 4. An example of *PDF Energy* computation for a correct ($\hat{y}_2$) and incorrect ($\hat{y}_1$) predictions

Let $\mathcal{S}(.) : \mathbb{R}^d \rightarrow \mathbb{R}^\rho$ be our RSS sampling as explained in Algorithm 1, by using a uniform sampling per strata, $\overrightarrow{SID}_\mathcal{B} \sim \text{Uniform}(y_i) \quad | \quad \forall y_i \in \overrightarrow{UID}_\mathcal{B}$ which draws ρ percentage of samples per strata as a proportional allocation (Henry, 1990) and defines $\frac{n_1}{N_1} = \frac{n_2}{N_2} = ... = \frac{n_L}{N_L}$ where N_L and n_L denotes the total number of samples in $\mathcal{Y}$ and $\mathcal{Y}^\rho$ per strata, respectively.

Let $\mathbb{D}(\mathcal{Y}^t, \beta_i)$ denotes the probability density function of ith strata on tth target variable, where $t = 1, ..., T$ and it is abbreviated as $\mathbb{D}^i_{\mathcal{Y}^t}$ ($T = ||\mathcal{Y}||$). The following equation shows the density of each strata on predicted values.

$$\mathbb{D}^i_{y^\rho} = \sum_{t=1}^{T} \mathbb{D}^i_{\mathcal{Y}^t} + \alpha^t_i \tag{10}$$

where α^t_i denotes the fraction differences after applying $\mathcal{S}(.)$ and $\alpha^t_i \approx 0$; let assume that $\alpha^t_i = 0$, the density of both target values and predicted values per each target and strata remains the same.

$$\sum_{t=1}^{T} \int_{\beta_l - 1}^{\beta_i} f_X(y^\rho_t) dy^\rho = \sum_{t=1}^{T} \int_{\beta_l - 1}^{\beta_i} f_X(y_t) dy_t \quad l = 1, .., L \tag{11}$$

Note that $y^\rho_t = \mathcal{Y}^{t^\rho}$ and $y_t = \mathcal{Y}^t$ are used for the simplicity.

The predicted values from $f_X(y^\rho_t)$ may include an error per strata (ϵ^t_i) and similarly it defines PDF as follows where a prediction error per target variable/strata is added.

$$\sum_{t=1}^{T} \sum_{i=2}^{L} \int_{\beta_{i-1}}^{\beta_i} f_X(\widehat{y^\rho_t}) d\widehat{y^\rho_t} - \epsilon^t_i = ||\mathcal{Y}|| \tag{12}$$

Note that for a single target variable $\sum_{i=2}^{L} \int_{\beta_{i-1}}^{\beta_i} f_X(\widehat{y^\rho_t}) d\widehat{y^\rho_t} - \epsilon^t_i = 1$. In an optimized model selection, $f_X(y^\rho_t)$ predicts values in correct strata and with minimal PDF divergence as follows.

$$
\begin{aligned}
&\sum_{t=1}^{T}(\int_{\beta_1}^{\beta_2} f_X(\widehat{y_t^{\rho}})d\widehat{y_t^{\rho}} - \epsilon_1^t) + (\int_{\beta_2}^{\beta_3} f_X(\widehat{y_t^{\rho}})d\widehat{y_t^{\rho}} - \epsilon_2^t) \\
&+ ... \\
&+ (\int_{\beta_{L-1}}^{\beta_L} f_X(\widehat{y_t^{\rho}})d\widehat{y_t^{\rho}} - \epsilon_m^t) \\
=&\sum_{t=1}^{T}(\int_{\beta_1}^{\beta_2} f_X(y_t)dy_t + \int_{\beta_2}^{\beta_3} f_X(y_t)dy_t \\
&+ ... \\
&+ \int_{\beta_{L-1}}^{\beta_L} f_X(y_t)dy_t) \\
=&||\mathcal{Y}||
\end{aligned}
$$

where we assume that $\epsilon_i^t = 0, \quad i = [1, ..., L]$ and $t = [1, .., T]$ (optimized model selection across all target variables).

As shown in Eq. 6, $\mathbb{E}_X(\hat{y})$ computes the highest value for all correct stratum predictions ($\beta_l^t \leq y_i^t < \beta_{l+1}^t$ where $i = [1, ..., m], \quad T = [1, ..., T]$) and the maximum value of *PDF Energy* is defined as follows.

$$
\begin{aligned}
\max\left(\mathbb{E}_X(\hat{y})\right) &= \sum_{t=1}^{T}\sum_{i=2}^{L} n_{i-1}^t \mathbb{D}_{y_i}^t \\
n_i^t &= ||y_i^t|| \quad s.t. \quad \beta_i^t \leq y_i^t < \beta_{i+1}^t
\end{aligned}
\tag{13}
$$

where n_i^t denotes the total number of samples in range of $[\beta_i^t, \beta_{i+1}^t]$ of $\mathcal{Y}^t$.

Finally, the selected algorithm predicts values where the energy score is optimized and can be defined as follows.

$$
\max\left(\mathbb{E}_X(\hat{y})\right) = \sum_{t=1}^{T}\sum_{i=2}^{L}(n_{i-1}^t \int_{\beta_{i-1}^t}^{\beta_i^t} f_X(\widehat{y_t^{\rho}})d\widehat{y_t^{\rho}}) \tag{14}
$$

□

Note that in certain cases when one of the two condition met, there is a possibility of equal maximum *PDF Energy* between at least two algorithms predictions of $\mathbb{A}_i$ and $\mathbb{A}_j$, where $i \neq j$: i) all values within all strata are predicted correctly; ii) at least two models predicts all values equally. In this scenario, increasing the number of strata defines a fined-grained evaluation by identifying an optimized algorithm among all equal algorithm prediction. However, this is a limitation of *PDF Energy* where it may requires additional computation time on repeating the process with different hyperparameter of β.

D. Experiments Setup Detail

D.1. Configuration Environment.

Our proposed approach process is completed on a machine with Ubuntu, an Intel(R) Xeon(R) Gold 5120 CPU @ 2.20GHz (56 cores) with 128 GB RAM, Quadro P5000 GPU with 16GB graphic RAM and 2 TB disk. All experiments are developed in Python with version '3.9' in Anaconda environment.

As AutoML platforms are known to produce noisy results, we conducted a comprehensive evaluation of our proposed approach by running experiments on Benchmark #1 (31 different real-world datasets). We repeat the experiments across 10 different random seeds, with 5 different time budgets, totaling 1,550 experiments on each AutoML tool.

To produce the AutoML results on Benchmark #1, we used nine Ubuntu virtual machines with identical software and hardware configurations of Intel(R) Xeon(R) Platinum 8176 vCPU @ 2.10GHz and Ubuntu 20.04.3 LTS (GNU/Linux 5.13.0-27-generic x86_64) with 16 GB RAM and 67GB Disk. Processing all datasets per seed on each AutoML/method took approximately six hours of computation time on a standard CPU machine, excluding any debugging issues. We encountered some crashes and restarts with H2O and MLJAR, which slightly prolonged the overall computation time. Overall, each AutoML/method required 57.5 hours of machine computation time in Benchmark #1, and for eight different platforms, it required a total of 460 hours of computation time, which was divided among the nine computing machines and completed in five days.

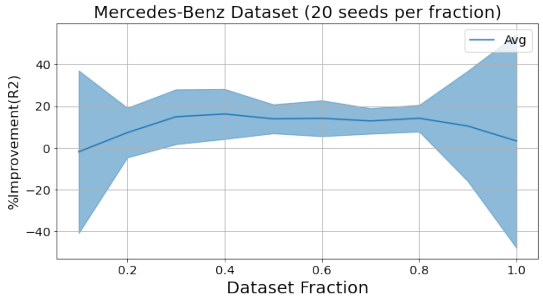


Figure 5. R^2 percentage improvements were compared for 20 different seeds per fraction, ranging from 10% to 100%, with a full-scale R^2 average of 0.487 on the 42570 OpenML dataset.

Algorithm Selection Experiments. To reproduce Table 1, about 29 hours computation is required to complete for *each* configuration (i.e., $\beta = 100, \rho = 20, metric = R^2$) across Benchmark #2 with considering 3 seeds. The following seeds have been used in this experiment: $[181, 185, 189]$ (similar seeds have been used for AutoML experiments of Benchmark #2) to split each dataset into training/test with a ratio of 75/25 by using a simple random sampling. Both RMSE/R^2 scores are reported based on evaluation on unseen test dataset. We use the same feature engineering across all experiments for algorithm selection. Therefore only hyperparameters are changed in each experiment.

AutoML Experiments. Since we restrict each AutoMLs in a set of time budget, each AutoML takes 11 minutes and 30 seconds to complete each experiment per seed in Benchmark #2. It requires around 8 hours computation time for processing all datasets on each AutoML. Overall around 64 hours computation time is required to complete all experiments on all AutoMLs of Benchmark #2.

D.2. Benchmark

we used datasets that were recently collected by Grinsztajn et al. and publicly available from OpenML. In this benchmark, we have both numerical regression [2] and categorical regression [3] and both groups of datasets are publicly available. More detail about the benchmark including downloading scripts can be found in Grinsztajn et al.. First, we use Scikit-Learn *train_test_split* function[4] to split the given input data set of dataset from the benchmark into 75%/25%. In both benchmarks, we use 25% of each dataset as hold-out. We conduct 10 and 3 experiments based on each configuration on each dataset for Benchmark #1 and Benchmark #2, respectively. We use the following random seeds of $[101, 112, 121, 168, 173, 193, 194, 219, 683, 761]$ and $[181, 185, 189]$ to split datasets in Benchmark #1 and #2, respectively. The same random seeds have been used in both algorithm selection and AutoML experiments. Table 3 shows the total number of features and samples per dataset for both benchmarks. As a future work, we plan to extend study by utilizing (Gijsbers et al., 2024).

D.3. Search Space

Table 4 shows the list of algorithms and utilized API libraries for choice of algorithm selection. We use `random_state=0` as a parameter for all algorithms. We also do not discriminate against any algorithm; therefore each algorithm has an equivalent chance of selection based on evaluation which is computed per configuration metric (i.e., *PDF Energy*, R^2 metric or KL divergence). Our objective in this study is: "*how to select a an algorithm effectively?*" and the list of algorithms can be extended or decreased per use case.

[2]OpenML benchmark 297:https://www.openml.org/search?type=benchmark&study_type=task&sort=tasks_included&id=297

[3]OpenML benchmark 300:https://www.openml.org/search?type=benchmark&sort=tasks_included&study_type=task&id=300

[4]https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html

Dataset	361072	361073	361074	361075	361076	361077	361078	361079	361080	361081	361082
#Features	22	27	17	614	12	34	9	17	7	9	7
#Samples	8192	15000	16599	7797	6497	13750	20640	22784	53940	10692	17379
Benchmark No.	1,2	1,2	1	1	1	1	1,2	1	1,2	1,2	1,2

Dataset	361083	361084	361085	361086	361087	361088	361089	361090	361092	361093	361094
#Features	10	16	7	4	14	80	9	6	63	8	5
#Samples	581835	21613	10081	163065	13932	21263	20640	18063	8885	4052	8641
Benchmark No.	1,2	1	1,2	1,2	1,2	1,2	1,2	1,2	1	1	1

Dataset	361095	361096	361097	361098	361099	361101	361102	361103	361104	42570
#Features	10	10	360	12	12	17	18	7	10	377
#Samples	166821	53940	4209	10692	17379	581835	21613	394299	241600	4209
Benchmark No.	1	1	1	1	1	1	1	1	1	2

Table 3. The number of features and samples per dataset in both benchmarks

Algorithm	API	Version
RandomForestRegressor	Scikit-Learn (Pedregosa et al., 2011) [5]	0.24.2
ExtraTreesRegressor	Scikit-Learn	1.1.3
LGBMRegressor	LightGBM(Ke et al., 2017) [6]	3.3.5
XGBRegressor	XGBoost (Chen et al., 2015) [7]	1.7.1
CatBoostRegressor	CatBoost (Dorogush et al., 2018) [8]	1.1.1
GradientBoostingRegressor	Scikit-Learn	1.1.3
AdaBoostRegressor	Scikit-Learn	1.1.3
BaggingRegressor	Scikit-Learn	1.1.3
DecisionTreeRegressor	Scikit-Learn	1.1.3
LinearRegression	Scikit-Learn	1.1.3
StackingRegressor-RidgeCV	Scikit-Learn	1.1.3
StackingRegressor-SGDRegressor	Scikit-Learn	1.1.3

Table 4. Search space algorithms and the list of used API implementations for RSS evaluations

D.3.1. Search Space Detail

RSS aims to select an optimized algorithm from *search space*. We considered 12 different atomic models where each model can be trained from an algorithm, however, any number of choices can be applied to RSS because RSS sub sampled based on the distribution of each trained model. Table 5 shows a comparison between the number of algorithm search space in popular tabular AutoMLs. This summary of algorithm search space across popular tabular AutoML tools, shows that our search space, consisting of 14 base choices, is competitive with traditional ML algorithms like tree-based methods, which perform effectively on tabular data. Our primary objective is centered around selecting an optimized algorithm.

Note that the current experiments are completed on a set of diverse algorithms and it can be extended to any number of ML algorithms. For instance, we consider leveraging *PDF Sampling* and *PDF Energy* metric for Network Architecture Search (NAS) as a future work.

AutoML	**# of Choices**	**Source/Reference**
MLJAR	10	MLJAR Website - Compete mode [9]
AutoGuon	6	Erickson et al.
H2O	5 categories includes 14 algorithms	LeDell & Poirier
Auto-Scikit Learn	15	Feurer et al.
TPOT	6	Olson & Moore
FLAML	6 (w/ hyperparameters)	Wang et al.
Baseline+RSS (Our)	14	this study

Table 5. A comparison of search space for algorithm selection in different tabular AutoMLs

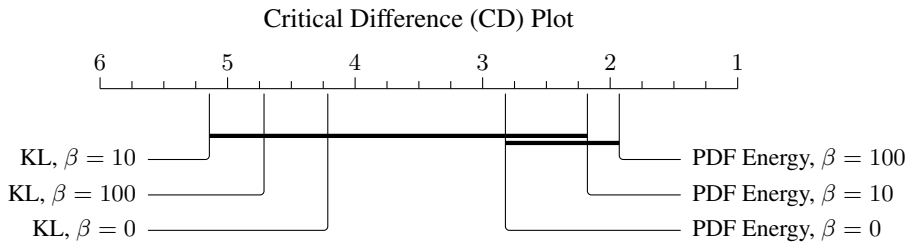


Figure 6. A Critical Difference (CD) diagram of algorithm selection based on KL and *PDF Energy* on R^2 score across three hyperparameters of β (strata) and Benchmark #2; $\beta = 0$ refers to Stratified PDF Sampling with dynamic value where $\bar{\beta} = 6$; See Table 13 for the detail of this experiment.

D.4. Sampling Ratio

Selecting a sampling ratio might be a challenging task. For instance, we use an **oracle search** in a simple experiment by using *PDF Sampling* on a set of ratio to identify the algorithm, and then compare it against SRS. Figure 5 shows percentage improvement that selecting the right ratio may cause significant performance changes where ratio between 20% to 80% has a lower standard deviation (more reliable) in compared to other ratios. A simple heuristic approach can be used to select the ratio according to input dataset size. However as noted earlier, sampling ratio selection is a limitation of our study and it is considered as a future work on selecting the correct ratio for possibility of additional performance gains.

E. Detail of Evaluation Results

E.1. Detail of Algorithm Selection Results

Table 13 shows R^2/RMSE results per dataset with different hyperparameters, which is a fine-grained detail of Figure 2 and Table 1. Note that in some experiments such as *361083 dataset* and $\beta = 10$, the same algorithm has been selected; therefore, the same prediction score can be observed. As shown in this table, the results of both metric evaluations indicate that using *PDF sampling* and using *PDF Energy* with distinct values of hyperparameter of β has a better performance in compared to simple random sampling (SRS) where the best algorithm is selected based on SRS and the performance of standard metrics.

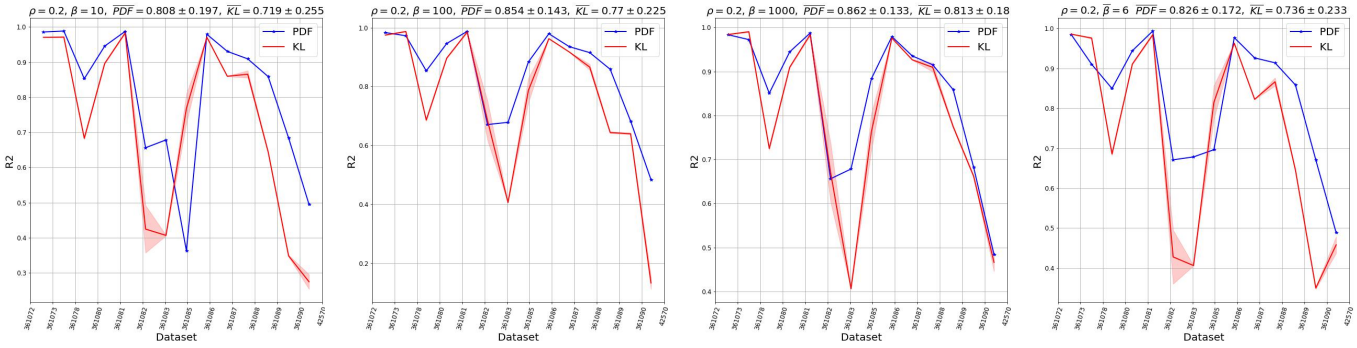
Figure 6 shows a Critical Difference (CD) (Demšar, 2006) diagram of algorithm selection for R^2 score in Benchmark #2 by comparing KL against *PDF Energy* with three hyperparameters of β (strata). In this figure, a lower rank signifies better performance, and the results shown that the utilization of *PDF Energy* led to a lower ranking in terms of CD. Note that in this figure, $\beta = 0$ pertains to Stratified PDF Sampling with a dynamic value, where values are chosen dynamically as discussed in the Relative Entropy experiment (5.3), with $\bar{\beta} = 6$. In this figure, the connected methods and their associated hyperparameters, represented by a bar, do not exhibit significant differences.

E.2. Stratified PDF

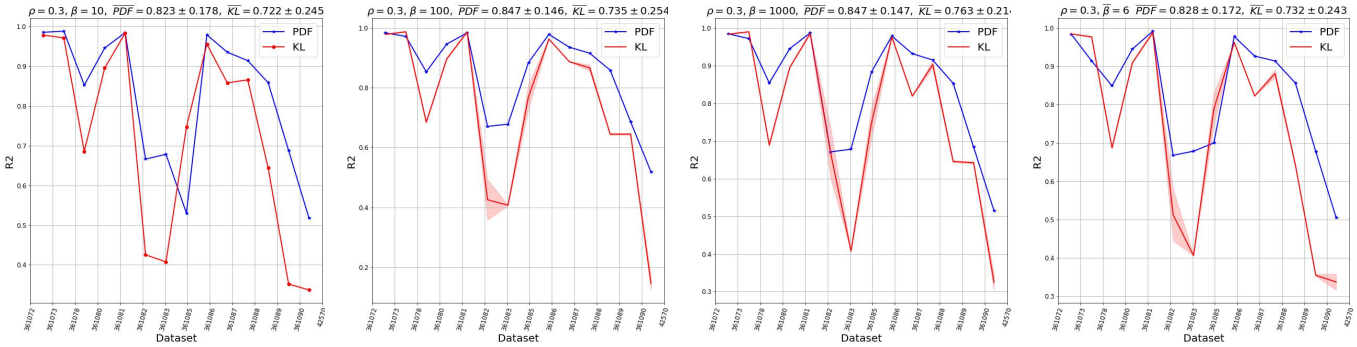
In Stratified *PDF Sampling*, the number of stratum (β) is selected per input dataset. Since we have 3 different seeds β in Benchmark #2, the value can be varied based on target variable(s). Table 6 shows the value of β per dataset/seed which is used in our experiments and reported in Table 1.

E.3. Relative Entropy Comparison

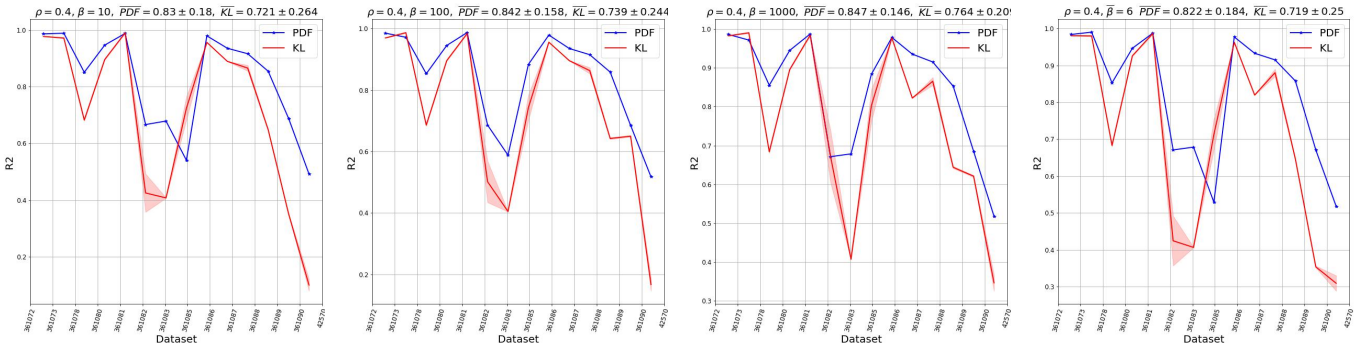
Figure 7 shows the performance evaluation on Benchmark #2 and 3 random seeds) across different sampling ratios (ρ) of $[0.2, 0.3, 0.4]$. Each column represents different values of hyperparameter of β in range of $[10, 100, 1000$,Dynamic$]$ where $Dynamic$ refers to stratified regressor PDF and the value is selected dynamically per dataset/seed as discussed in previous section. The detail results per sampling ratio (ρ), show the effectiveness of algorithm selection across different configurations and sampling ratios through KL and *PDF Energy* where both are relative entropy based approach. The performance is averaged across 3 different seeds. The results demonstrate that PDF is either competitive (equal) or outperforms other methods in all experiments. Overall, the average performance of PDF is remarkable where it achieves the highest R^2 score with the lowest standard deviation in each configuration of β and ρ.



(a) 20% Sampling Ratio ($\rho = 0.2$)



(b) 30% Sampling Ratio ($\rho = 0.3$)



(c) 40% Sampling Ratio ($\rho = 0.4$)

Figure 7. The performance comparison of KL and PDF (our) on hold-out data for algorithm selection across Benchmark #2 with hyperparameters of four different values of β and three sampling ratios of (a) $\rho = 0.2$, (b) $\rho = 0.3$, (c) $\rho = 0.4$. Each data point represents the average of 3 experiments with 3 random seeds.

Dataset	Seed	β	Dataset	Seed	β
361072	181	5	361086	181	6
	185	5		185	6
	189	5		189	6
361073	181	2	361087	181	6
	185	2		185	6
	189	2		189	6
361078	181	6	361088	181	3
	185	6		185	4
	189	6		189	4
361080	181	4	361089	181	5
	185	4		185	5
	189	4		189	5
361081	181	7	36190	181	4
	185	8		185	4
	189	8		189	4
361082	181	4	42570	181	7
	185	4		185	15
	189	4		189	7
361083	181	13	361085	181	20
	185	13		185	18
	189	13		189	18

Table 6. The total number of strata (β) was selected per seed per dataset for stratified PDF sampling across all datasets in Benchmark #2, with an average of $\bar{\beta} = 6$.

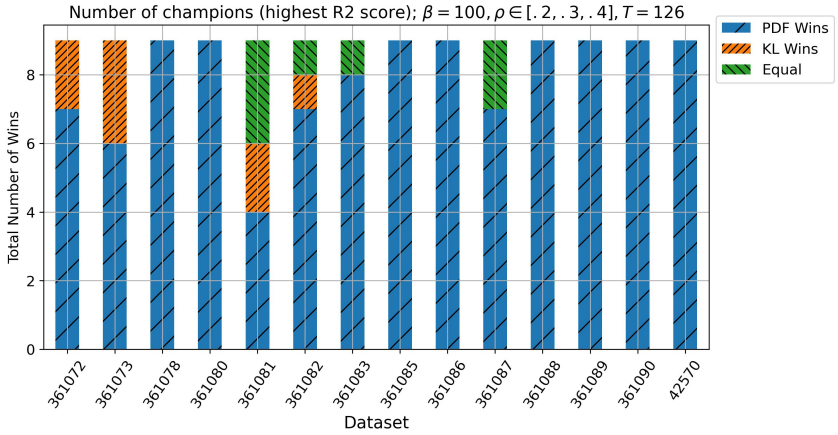
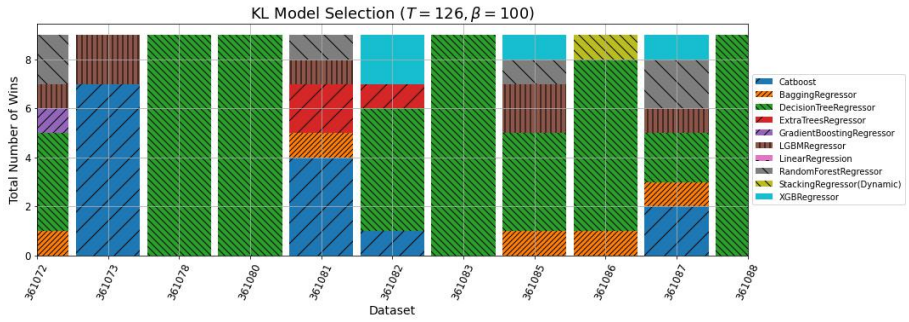
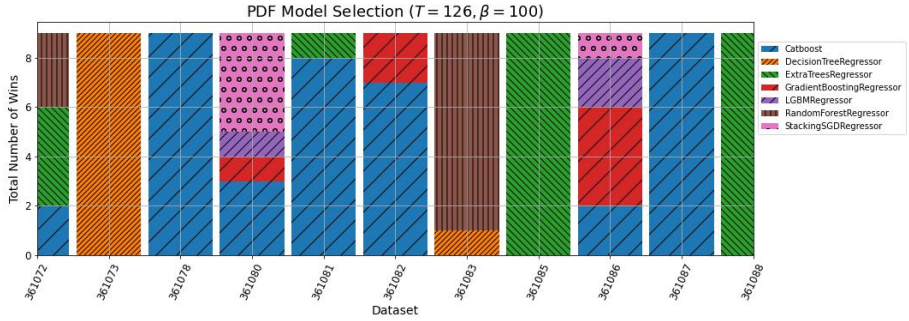


Figure 8. A comparison between KL and PDF (our) for algorithm selection on Benchmark #2 with the total number of champions per dataset based on highest R^2 score.

Figure 8 shows a comparison between KL and PDF in term of the total number of highest R^2 score per dataset. Each dataset includes 9 different experiments (3 seeds per 3 sampling ratios) with the configuration of $\beta = 100$. We also observed the same outcome for each individual ratio where *PDF* outperformed significantly on selected models. With the same configuration (total 126 experiments per method), Figure 9 shows a comparison of the list of recommended algorithms between KL and PDF per dataset across all experiments. Note that stacking represents a model which is trained from top 3 atomic algorithms. In this figure, we do not discriminate different type of stacking (architecture) and all diverse architectures including diverse estimator of ensemble models considered as a single "ensemble" type. These results show that PDF aims to select more diverse algorithms as suggested by Nam et al., which is an important factor when recommending an ensemble model. In this experiments, KL recommends more decision tree regressor algorithm; however, PDF tends more on CatBoost algorithm selection.



(a) Diversity of Algorithm Selection based on KL



(b) Diversity of Algorithm Selection based on PDF Sampling + PDF Energy evaluation

Figure 9. A comparison between (a) KL and (b) PDF (our) for diversity algorithm selection per dataset on Benchmark #2; 9 experiments per dataset that includes 3 different seeds per 3 sampling ratios, considering $\beta = 100$ and total computation time of 126 across all time budgets.

E.4. AutoML Evaluation Detail

As explained in Section 4.2, we use 6 different AutoML platforms on top of baseline to train a model on each training dataset and evaluate each AutoML platform on test dataset by using R^2 metric. We use the latest version of open-source tabular AutoML platforms of FLAML [10] (Wang et al., 2021), MLJAR [11] (Płońska & Płoński, 2021), AutoGluon [12] (Erickson et al., 2020), H2O [13] (LeDell & Poirier, 2020), TPOT [14] (Olson & Moore, 2016) and AutoSK-Learn [15] (Feurer et al., 2020). In each experiment similarly D^{Train} fit into each AutoML and reported the averaged R^2 score across all different seeds on test dataset with respect to given time budget. Note that 210 different experiments have been conducted on each AutoML that include 5 different time budgets restriction with 3 different random seeds on 14 different datasets for Benchmark #2. The configuration and version of each AutoML tool which is used in both experiments, is shown in Table 7.

In Benchmark #2, only TPOT and Auto-SKLearn failed to generate a model in all experiments due to the lack of text processing. By removing failure cases (R^2=0) and considering 185 experiments and 195 experiments for TPOT and AutoSKLearn (out of 210) the overall average is $Avg(\mathcal{A}_{TPOT}) = 0.847$, $Avg(\mathcal{A}_{AutoSKLearn}) = 0.76$. (see failures in Appendix E.4.1). Table 11 shows the performance of all AutoML experiments per dataset on Benchmark #2 where RSS is the top performer by reaching the highest R^2 score among 6 AutoMLs and baseline (8 out of 14 datasets) with an overall average of 0.852 over 210 experiments per platform. These evaluation results on Benchmark #2, similar to Section 5.2 on Benchmark #1, show that **i)** our algorithm selector was able to outperform in all of time budget constrains (4 out of 5 time budgets); **ii)** PDF energy identifies the algorithm with respect to the time budget constrain as early as 30 second time budget because adding additional time budget slightly increases the performance; **iii)** adding time budget enables our proposed

[10] https://microsoft.github.io/FLAML/
[11] https://supervised.mljar.com/
[12] https://auto.gluon.ai/stable/api/autogluon.task.html
[13] https://docs.h2o.ai/h2o/latest-stable/h2o-docs/automl.html
[14] https://github.com/EpistasisLab/tpot
[15] https://automl.github.io/auto-sklearn/master/

#	AutoML	Version	Parameters
1	MLJAR	0.11.2	
2	FLAML	1.0.14	
3	TPOT	0.11.7	
4	H2O AutoML	3.38.0.4	*'max_models': 20*
5	AutoGluon	0.6.2	*presets:'best_quality'*
6	Auto-SKLearn	0.15.0	

Table 7. AutoML Configurations that have been used in both Benchmark #1 and Benchmark #2.

approach to improve its performance in case of missed optimized algorithm (i.e., second choice from sorted algorithm recommendation w.r.t. *PDF Energy*); **iv)** our proposed approach and *H2O* achieved the lowest standard deviation with $std = \pm 0.15$ among all AutoMLs; **v)** our proposed approach achieved the best overall averaged R^2 score of 0.852 across all AutoMLs.

Table 14 shows the detail results on Benchmark #2 by representing R^2 score evaluation per dataset/time budget. As shown in this table, *RSS* selects an optimized algorithm for 7 out 14 datasets within the first time budget of 30 seconds. One further improvement as future work is increasing the number of algorithm exploration to enhance the performance by utilizing *PDF Sampling* and *PDF Energy* metric.

Table 2 shows the overall R^2 score per time budget per AutoML in Benchmark #1. Each data point represents an average of 310 experiments with $\pm$ standard deviation. The champions per time budgets (rows) are highlighted in bold where RSS shows the best performance across all platforms. Note that the baseline does not take into account the time budget (model selection is the same across all time budgets). However, for a fair comparison, we repeat the training and prediction procedure to obtain a more accurate report. Figure 11 shows R^2 performance comparison (higher is better) where each sub-figure represents the performance of all AutoMLs including our RSS approach per selected time budget in seconds. As shown in this figure, RSS is leading in most of the experiments.

In addition to the results as shown in Table 2 and Figure 11, Figure 10 shows AutoML Critical Difference (CD) plot on Benchmark #1. AutoML Tools connected by a bar are not significantly different. The lower rank is better where RSS achieves the best ranking among all AutoML tools by utilizing *PDF Sampling* and *PDF Energy*.

Assumptions. Our evaluation is based on fitting 75% of datasets into different AutoML tools and predicting 25% of the test dataset on all AutoMLs based on all different random seeds. Therefore, we did not consider any update on any specific AutoML platform beside using the same hyperparameters such as *evaluation metric*, *time budget* and etc. The same configuration allows us to have a fair comparison between different AutoML tools according to the given input dataset. Therefore, we assume that any small fixes beside system configuration are out of scope in our evaluation. In addition, AutoML platforms are evaluated based on our split test dataset which has been used to fairly test all AutoML tools.

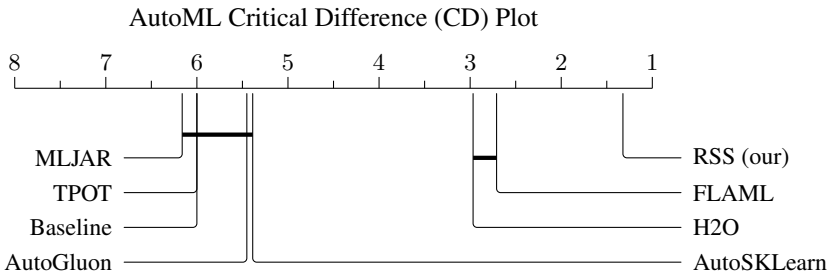


Figure 10. Critical Difference (CD) diagram for test score (R^2) on 31 datasets with 10 random seeds (see Figure 11 for the detail); The lower rank represents the better performance; The AutoML tools which are connected by a bar, are not significantly different.

E.4.1. AutoML Failures.

Note that there were some failure cases in Benchmark #1 where AutoSKLearn and TPOT failed due to the input data type issue. We also observe that H2O failed due to time budget limitations. In each experiment, if there was any error, we marked

it as $R^2 = 0$. After collecting all individual results, we reprocess records with $R^2 = 0$ on another refreshed machine to be ensured that there is no any environment concern. Table 8 are the results of the number of $R^2 = 0$ per AutoML in the first and the second attempt. Note that only baseline, FLAML and our proposed approach (RSS) were able to generate all results without any repetitions.

AutoML	# Failure in first attempt	# Failure in 2nd attempt	Issue
AutoGluon	11	0	-
AutoSKLearn	163	163	data format
H2O	159	48	time budget
MLJAR	5	3	time budget
TPOT	72	65	data format

Table 8. The number of failure cases in the first and the second repetitions of the process

Issues. We observed several issues where only 2 AutoML platforms were not able to produce any model (failure cases) for 40 different experiments of Benchmark #2. Table 9 shows the detail of errors for each AutoML tool.

AutoML	# of Failure	Dataset	Seed	Time Budget	Error/Comment
TPOT					
	10	361083	[181,185,189]	[30, 60]	`Liblinear failed to converge, increase`; due time budget limitation
	15	42570	[181,185,189]	[30, 60, 120,180,300]	could not process text feature
Auto-SKLearn					
	15	42570	[181,185,189]	[30, 60, 120,180,300]	could not process text feature

Table 9. AutoML failures on Benchmark #2 (14 different datasets with 3 different random seeds and 5 different time budgets)

Time (S)	Baseline	MLJAR	FLAML	H2O	TPOT	AutoSKLearn	AutoGluon	RSS (our)
30	0.7795 ±0.2	0.7693 ±0.19	0.8349 ±0.18	0.8391 ±0.15	0.6821 ±0.36	0.4924 ±0.42	0.8384 ±0.16	**0.8510 ±0.15**
60	0.7795 ±0.2	0.8194 ±0.18	0.8348 ±0.2	0.8427 ±0.15	0.7084 ±0.36	0.6899 ±0.35	0.8428 ±0.15	**0.8517 ±0.15**
120	0.7795 ±0.2	0.8469 ±0.15	0.8468 ±0.15	0.8471 ±0.15	0.7757 ±0.28	0.7817 ±0.28	0.8480 ±0.16	**0.8520 ±0.15**
180	0.7795 ±0.2	0.8480 ±0.15	0.8483 ±0.15	0.8476 ±0.15	0.7799 ±0.27	0.7826 ±0.28	0.8508 ±0.15	**0.8520 ±0.15**
300	0.7795 ±0.2	0.8490 ±0.15	0.8511 ±0.15	0.8482 ±0.15	0.7837 ±0.27	0.7833 ±0.28	**0.8530 ±0.15**	0.8522 ±0.15
$Avg(\mathcal{A})$	0.7795±0.21	0.8265±0.16	0.8432±0.16	0.8449±0.15	0.7459±0.29	0.7060±0.28	0.8466±0.16	**0.8518±0.15**

Table 10. AutoML evaluation based on R^2 score (higher is better) on Benchmark #2; averaged over 3 random seeds per dataset for each time budgets in second (averaged of 42 experiments per data point) with a standard deviation of approximately ±.; **bold** values shows the highest score per time budget. See per dataset detail in Table 11 and fined-grain results in Table 14.

Dataset	361072	361073	361078	361080	361081	361082	361083	361085	361086	361087	361088	361089	361090	42570	$Avg(\mathcal{A})$
Baseline	0.984	0.988	0.789	0.890	0.981	0.664	0.301	0.835	**0.979**	0.773	0.896	0.684	0.683	0.467	0.779
MLJAR	0.984	0.991	0.821	0.945	0.976	0.663	0.542	0.802	**0.979**	0.885	0.910	**0.864**	0.690	**0.519**	0.827
FLAML	0.982	0.990	0.848	**0.947**	0.992	0.697	0.646	0.881	**0.979**	0.931	**0.923**	0.848	0.690	0.448	0.843
H2O	0.985	0.988	0.859	**0.947**	**0.994**	0.705	0.648	0.822	0.978	0.930	0.914	0.861	0.690	0.508	0.845
TPOT	0.983	0.985	0.832	0.946	0.986	0.694	0.226	0.817	0.798	0.918	0.747	0.828	0.684	0.000	0.746
AutoSKLearn	0.982	0.990	0.806	0.943	0.789	0.701	0.243	0.835	0.587	0.925	0.713	0.683	0.687	0.000	0.706
AutoGluon	0.978	**0.994**	**0.869**	**0.947**	0.992	**0.708**	0.593	0.873	**0.979**	0.935	0.917	0.860	**0.694**	0.512	0.847
RSS (our)	**0.986**	0.990	0.857	**0.947**	**0.994**	0.689	**0.678**	**0.884**	**0.979**	**0.936**	0.916	0.859	0.691	**0.519**	**0.852**
$Avg(\mathcal{D})$	0.983	0.989	0.835	0.939	0.963	0.690	0.485	0.844	0.907	0.904	0.867	0.811	0.688	0.372	0.806

Table 11. AutoML evaluation per dataset on Benchmark #2; averaged R^2 Score (higher is better) over 3 random seeds for each time budget of $[30, 60, 120, 180, 300]$ (averaged of 15 experiments per data point); highest score per dataset is marked **bold**. See fined-grain results in Table 14.

F. Ablation Study on Higher Time Budget

In the previous experiments, we focused on short time budgets for each experiment, with a maximum of five minutes. Our objective was to test different AutoML tools under stress test conditions (limited time budget) where our proposed sampling approach compared to AutoMLs performances. To further evaluate each AutoML platform, we have set up an additional ablation study with an extra time budget.

Setup. We limited each experiment to 10 and 20 minutes, where the time budgets were 3X+ and 5X+ times higher than in the previous experiments, respectively. We used 10 different datasets (361083, 361085, 361086, 361084, 361080, 361081, 361082, 361072, 361073, 361074) with 10 random seeds (761, 193, 121, 101, 112, 168, 173, 194, 219, 683) for reproducibility. In this ablation study, we allowed RSS to generate an ensemble model with the top 6 and top 10 outperforming models within its time budget (n_stacking=3 in the two previous sets of experiments for Benchmark #1 and #2).

Results. Table 12 shows the results of the time budget ablation study on 10 datasets with 10 random seeds for two time budgets of 10 and 20 minutes. The results of the ablation study indicate that increasing the time budget still leads to improved performance in 10 minutes and competitive with AutoGluon in 20 minutes. Specifically, our study found that increasing the time budget allowed RSS to identify an optimized model more accurately. AutoGluon's performance tends to improve with a higher time budget compared to short time budgets, but it still performs less well than RSS in the 10-minute experiment. Note that AutoGluon is able to expand the search space when there is sufficient time (i.e., ensemble models); however, RSS uses a limited search space in this study. We expect that if we similarly increase the search space (i.e., allowing RSS to ensemble more model), RSS could potentially outperform AutoGluon. Since increasing the search space and higher time budgets are not the main focus of this study, we consider it a potential area for future work. These results support our earlier conclusions, even with the increased time budget. Note that, since our objective is sampling evaluation, drawing a conclusion on a shorter time budget might be more effective in evaluating each AutoML tool against our proposed sampling approach, RSS, for tabular data. Note that since each experiment ran independently, an AutoML may use a different strategy to find an optimized algorithm based on given time budget. As a result, some AutoMLs (i.e., AutoSKLearn) may suffer performance degradation due to the use of different strategies, even with a given additional time budget. Therefore, increasing a time budget does not necessarily mean that an AutoML will always have improved performance.

Time(s)	Baseline	MLJAR	FLAML	AutoSKLearn	H2O	TPOT	AutoGluon	RSS (our)
600	0.8888 ± 0.12	0.8865 ± 0.15	0.8996 ± 0.11	0.5553 ± 0.45	0.8940 ± 0.12	0.8689 ± 0.16	0.8973 ± 0.12	$\mathbf{0.900 \pm 0.11}$
1200	0.8888 ± 0.12	0.8866 ± 0.15	0.8999 ± 0.11	0.5577 ± 0.45	0.8956 ± 0.11	0.8737 ± 0.15	$\mathbf{0.9021 \pm 0.11}$	0.9012 ± 0.11

Table 12. Proposed approach evaluation against AutoMLs in ablation higher time budget study on 10 datasets with 10 different random seeds; Each data point represents an average of 100 experiments with a standard deviation of approximately $\pm$. Champions for each time budget (row) are highlighted in **bold**.

G. Disclaimers

- Although a sampling approach has been introduced in this study, the authors do not anticipate that the sampling algorithm or ML algorithm selection will suffer from bias or discriminatory selection. However, if the full-scale data consist of biased and discriminatory data points, the sampled data and/or algorithm selection may inherently exhibit these issues.

β			10		100		1000		Dynamic Stratified	
Dataset	Sampling Method	Evaluation Method	R^2	RMSE	R^2	RMSE	R^2	RMSE	R^2	RMSE
361072	PDF	PDF Energy	**0.9857 ± 0.002**	**2.1904 ± 0.074**	0.9839 ± 0.003	2.3204 ± 0.136	0.9848 ± 0.003	2.2556 ± 0.133	0.9845 ± 0.002	2.2854 ± 0.1
	Random	Metric	0.9849 ± 0.002	2.255 ± 0.054	**0.9849 ± 0.002**	**2.255 ± 0.054**	**0.9849 ± 0.002**	**2.255 ± 0.054**	**0.9849 ± 0.002**	**2.255 ± 0.054**
361073	PDF	PDF Energy	**0.9884 ± 0.001**	**4.501 ± 0.248**	0.972 ± 0.002	7.0065 ± 0.243	0.9721 ± 0.002	6.9917 ± 0.262	0.9385 ± 0.103	7.7879 ± 7.281
	Random	Metric	**0.9884 ± 0.002**	**4.501 ± 0.427**	**0.9884 ± 0.002**	**4.501 ± 0.427**	**0.9884 ± 0.002**	**4.501 ± 0.427**	**0.9884 ± 0.002**	**4.501 ± 0.427**
361078	PDF	PDF Energy	**0.8519 ± 0.006**	**0.2191 ± 0.005**	**0.8532 ± 0.006**	**0.2181 ± 0.005**	**0.8533 ± 0.006**	**0.2181 ± 0.005**	0.8506 ± 0.008	0.22 ± 0.007
	Random	Metric	0.8515 ± 0.008	0.2194 ± 0.006	0.8515 ± 0.008	0.2194 ± 0.006	0.8515 ± 0.008	0.2194 ± 0.006	**0.8515 ± 0.008**	**0.2194 ± 0.006**
361080	PDF	PDF Energy	**0.946 ± 0.001**	**0.2355 ± 0.003**	**0.9457 ± 0.002**	**0.2363 ± 0.003**	**0.945 ± 0.002**	**0.2377 ± 0.003**	**0.9455 ± 0.001**	**0.2366 ± 0.003**
	Random	Metric	0.9414 ± 0.003	0.2456 ± 0.007	0.9414 ± 0.003	0.2456 ± 0.007	0.9414 ± 0.003	0.2456 ± 0.007	0.9414 ± 0.003	0.2456 ± 0.007
361081	PDF	PDF Energy	0.9865 ± 0.01	0.084 ± 0.041	0.9859 ± 0.012	0.0857 ± 0.042	0.9872 ± 0.01	0.0826 ± 0.038	**0.9912 ± 0.007**	**0.0683 ± 0.032**
	Random	Metric	**0.9874 ± 0.011**	**0.0802 ± 0.042**	**0.9874 ± 0.011**	**0.0802 ± 0.042**	**0.9874 ± 0.011**	**0.0802 ± 0.042**	0.9874 ± 0.011	0.0802 ± 0.042
361082	PDF	PDF Energy	**0.6629 ± 0.021**	**105.1333 ± 3.343**	**0.6759 ± 0.022**	**103.0747 ± 3.567**	**0.6662 ± 0.023**	**104.6016 ± 3.163**	**0.6699 ± 0.022**	**104.0375 ± 3.801**
	Random	Metric	0.6488 ± 0.024	107.2763 ± 3.316	0.6488 ± 0.024	107.2763 ± 3.316	0.6488 ± 0.024	107.2763 ± 3.316	0.6488 ± 0.024	107.2763 ± 3.316
361083	PDF	PDF Energy	**0.6784 ± 0.002**	**0.3382 ± 0.001**	0.6483 ± 0.09	0.3516 ± 0.04	**0.6784 ± 0.002**	**0.3382 ± 0.001**	**0.6784 ± 0.002**	**0.3382 ± 0.001**
	Random	Metric	**0.6784 ± 0.002**	**0.3382 ± 0.001**	**0.6784 ± 0.002**	**0.3382 ± 0.001**	**0.6784 ± 0.002**	**0.3382 ± 0.001**	**0.6784 ± 0.002**	**0.3382 ± 0.001**
361085	PDF	PDF Energy	0.4778 ± 0.229	0.0394 ± 0.012	**0.8837 ± 0.035**	**0.0193 ± 0.005**	**0.8837 ± 0.035**	**0.0193 ± 0.005**	0.6418 ± 0.307	0.0318 ± 0.018
	Random	Metric	**0.8653 ± 0.048**	**0.0207 ± 0.005**	0.8653 ± 0.048	0.0207 ± 0.005	0.8653 ± 0.048	0.0207 ± 0.005	**0.8653 ± 0.048**	**0.0207 ± 0.005**
361086	PDF	PDF Energy	**0.9792 ± 0.0**	**0.0817 ± 0.001**	**0.9789 ± 0.001**	**0.0822 ± 0.002**	**0.9789 ± 0.001**	**0.0823 ± 0.002**	**0.9777 ± 0.001**	**0.0845 ± 0.003**
	Random	Metric	0.977 ± 0.001	0.0858 ± 0.003	0.977 ± 0.001	0.0858 ± 0.003	0.977 ± 0.001	0.0858 ± 0.003	0.977 ± 0.001	0.0858 ± 0.003
361087	PDF	PDF Energy	0.9338 ± 0.005	0.1468 ± 0.005	**0.9354 ± 0.002**	**0.1451 ± 0.002**	0.9344 ± 0.003	0.1463 ± 0.003	0.9289 ± 0.005	0.1522 ± 0.006
	Random	Metric	**0.9347 ± 0.003**	**0.1459 ± 0.003**	0.9347 ± 0.003	0.1459 ± 0.003	**0.9347 ± 0.003**	**0.1459 ± 0.003**	**0.9347 ± 0.003**	**0.1459 ± 0.003**
361088	PDF	PDF Energy	**0.9131 ± 0.011**	**10.1257 ± 0.619**	**0.9152 ± 0.007**	**10.0101 ± 0.408**	**0.9152 ± 0.007**	**10.0101 ± 0.408**	**0.9146 ± 0.008**	**10.0459 ± 0.458**
	Random	Metric	0.9125 ± 0.011	10.163 ± 0.587	0.9125 ± 0.011	10.163 ± 0.587	0.9125 ± 0.011	10.163 ± 0.587	0.9125 ± 0.011	10.163 ± 0.587
361089	PDF	PDF Energy	**0.8573 ± 0.006**	**0.1339 ± 0.003**	**0.8587 ± 0.002**	**0.1332 ± 0.002**	0.855 ± 0.008	0.1349 ± 0.004	**0.8577 ± 0.003**	**0.1337 ± 0.002**
	Random	Metric	0.8568 ± 0.006	0.1341 ± 0.003	0.8568 ± 0.006	0.1341 ± 0.003	**0.8568 ± 0.006**	**0.1341 ± 0.003**	0.8568 ± 0.006	0.1341 ± 0.003
361090	PDF	PDF Energy	**0.6867 ± 0.003**	**0.7587 ± 0.004**	**0.6848 ± 0.003**	**0.7609 ± 0.003**	**0.684 ± 0.003**	**0.762 ± 0.004**	**0.6732 ± 0.009**	**0.7748 ± 0.011**
	Random	Metric	0.6675 ± 0.021	0.7812 ± 0.025	0.6675 ± 0.021	0.7812 ± 0.025	0.6675 ± 0.021	0.7812 ± 0.025	0.6675 ± 0.021	0.7812 ± 0.025
42570	PDF	PDF Energy	**0.5087 ± 0.038**	**9.2177 ± 0.62**	**0.5149 ± 0.036**	**9.1607 ± 0.624**	**0.517 ± 0.033**	**9.1545 ± 0.612**	**0.5036 ± 0.027**	**9.2623 ± 0.452**
	Random	Metric	0.5009 ± 0.03	9.2906 ± 0.545	0.5009 ± 0.03	9.2906 ± 0.545	0.5009 ± 0.03	9.2906 ± 0.545	0.5009 ± 0.03	9.2906 ± 0.545
Average	PDF	PDF Energy	0.8183 ± 0.024	**9.5147 ± 0.356**	**0.8455 ± 0.016**	**9.5432 ± 0.363**	**0.8468 ± 0.01**	**9.6453 ± 0.332**	0.8254 ± 0.036	**9.6757 ± 0.87**
	Random	Metric	**0.8425 ± 0.012**	9.6812 ± 0.359	0.8425 ± 0.012	9.6812 ± 0.359	0.8425 ± 0.012	9.6812 ± 0.359	**0.8425 ± 0.012**	9.6812 ± 0.359
Total Number of Champions		PDF	**9**	**9**	**10**	**10**	**8**	**8**	**8**	**8**
(Total: 14 datasets)		Metric	3	3	4	4	5	5	5	5
		Equal	2	2	0	0	1	1	1	1

Table 13. Model selection performance comparison for 14 different public OpenML datasets (Benchmark #2) on 3 different random seeds with 25% hold-out data and two metrics of RMSE and R^2 evaluation; the values are averaged across 3 seeds where a sampling ratio of $\rho = [0.2, 0.3, 0.4]$ is used per seed. $\beta = [10, 100, 1000, \mathcal{D}]$ for model selection through *PDF Sampling*(our approach). The winner of each experiment per dataset/selected strata is marked in **bold** based on either lower RMSE or higher R^2 on test dataset

Dataset	Time(s)	Baseline	MLJAR	FLAML	H2O	TPOT	AutoSKLearn	AutoGluon	RSS (Our)
361072	30	0.9836 ±0.0	0.9839 ±0.0	0.9829 ±0.0	0.9835 ±0.0	0.9827 ±0.0	0.9797 ±0.01	0.9846 ±0.0	🏆 **0.9861 ±0.0**
	60	0.9836 ±0.0	0.9828 ±0.0	0.9803 ±0.0	0.9856 ±0.0	0.9827 ±0.0	0.9811 ±0.01	0.9454 ±0.07	**0.9861 ±0.0**
	120	0.9836 ±0.0	0.9847 ±0.0	0.9823 ±0.0	0.9853 ±0.0	0.9828 ±0.0	0.9826 ±0.0	**0.9868 ±0.0**	0.9861 ±0.0
	180	0.9836 ±0.0	0.9849 ±0.0	0.9816 ±0.0	0.9851 ±0.0	0.9828 ±0.0	0.9823 ±0.0	**0.9876 ±0.0**	0.9861 ±0.0
	300	0.9836 ±0.0	0.9852 ±0.0	0.9833 ±0.0	0.9851 ±0.0	0.9824 ±0.0	0.9827 ±0.0	**0.9875 ±0.0**	0.9861 ±0.0
361073	30	0.9878 ±0.0	0.9867 ±0.0	0.9897 ±0.0	0.9814 ±0.0	0.9835 ±0.0	0.9828 ±0.0	🏆 **0.9931 ±0.0**	0.9904 ±0.0
	60	0.9878 ±0.0	0.9898 ±0.0	0.9906 ±0.0	0.9878 ±0.0	0.9839 ±0.0	0.9877 ±0.0	**0.9937 ±0.0**	0.9904 ±0.0
	120	0.9878 ±0.0	0.9913 ±0.0	0.9906 ±0.0	0.9897 ±0.0	0.9856 ±0.0	0.9886 ±0.0	**0.9943 ±0.0**	0.9904 ±0.0
	180	0.9878 ±0.0	0.9915 ±0.0	0.9904 ±0.0	0.9898 ±0.0	0.9855 ±0.0	0.9946 ±0.0	**0.995 ±0.0**	0.9904 ±0.0
	300	0.9878 ±0.0	0.9946 ±0.0	0.991 ±0.0	0.9893 ±0.0	0.9859 ±0.01	0.9946 ±0.0	**0.9956 ±0.0**	0.9904 ±0.0
361078	30	0.7885 ±0.08	0.6587 ±0.17	0.8418 ±0.02	0.8541 ±0.01	0.8199 ±0.01	0.6853 ±0.01	🏆 **0.8616 ±0.01**	0.8532 ±0.01
	60	0.7885 ±0.08	0.8588 ±0.01	0.8482 ±0.0	0.8591 ±0.01	0.832 ±0.01	0.7937 ±0.09	**0.8667 ±0.01**	0.8551 ±0.01
	120	0.7885 ±0.08	0.8615 ±0.0	0.8474 ±0.0	0.8608 ±0.0	0.8334 ±0.02	0.8471 ±0.01	**0.8712 ±0.01**	0.8581 ±0.01
	180	0.7885 ±0.08	0.8627 ±0.01	0.8483 ±0.0	0.8589 ±0.0	0.8379 ±0.02	0.8507 ±0.01	**0.8719 ±0.01**	0.8581 ±0.01
	300	0.7885 ±0.08	0.8637 ±0.0	0.8528 ±0.0	0.86 ±0.01	0.8379 ±0.02	0.8511 ±0.01	**0.8726 ±0.01**	0.8581 ±0.01
361080	30	0.8903 ±0.02	0.9369 ±0.0	🏆 **0.9468 ±0.0**	🏆 0.9466 ±0.0	0.9462 ±0.0	0.9364 ±0.01	🏆 0.9454 ±0.0	🏆 0.9456 ±0.0
	60	0.8903 ±0.02	0.9471 ±0.0	0.9471 ±0.0	0.947 ±0.0	0.9462 ±0.0	0.9399 ±0.0	0.9463 ±0.0	**0.9472 ±0.0**
	120	0.8903 ±0.02	0.9473 ±0.0	0.9471 ±0.0	**0.9474 ±0.0**	0.9462 ±0.0	0.9466 ±0.0	0.9473 ±0.0	0.9472 ±0.0
	180	0.8903 ±0.02	0.9474 ±0.0	0.9471 ±0.0	0.9474 ±0.0	0.9464 ±0.0	0.9468 ±0.0	**0.9476 ±0.0**	0.9472 ±0.0
	300	0.8903 ±0.02	0.9473 ±0.0	0.9471 ±0.0	0.9474 ±0.0	0.9465 ±0.0	0.9468 ±0.0	**0.9477 ±0.0**	0.9472 ±0.0
361081	30	0.981 ±0.02	0.949 ±0.0	0.9924 ±0.01	🏆 0.9922 ±0.01	0.9778 ±0.02	-0.0005 ±0.0	0.9937 ±0.01	🏆 **0.9942 ±0.0**
	60	0.981 ±0.02	0.9693 ±0.01	0.9922 ±0.01	0.993 ±0.0	0.9778 ±0.02	0.988 ±0.01	**0.998 ±0.0**	0.9942 ±0.0
	120	0.981 ±0.02	0.9855 ±0.01	0.9918 ±0.01	0.9947 ±0.0	0.989 ±0.01	0.9869 ±0.01	**0.9952 ±0.0**	0.9942 ±0.0
	180	0.981 ±0.02	0.9859 ±0.01	0.9927 ±0.01	0.994 ±0.0	0.989 ±0.01	0.9864 ±0.01	0.9884 ±0.02	**0.9942 ±0.0**
	300	0.981 ±0.02	0.99 ±0.01	0.9934 ±0.01	**0.9944 ±0.0**	0.9943 ±0.0	0.9866 ±0.01	0.9835 ±0.02	0.9942 ±0.0
361082	30	0.6637 ±0.01	0.5516 ±0.01	0.6915 ±0.02	0.7017 ±0.01	0.6918 ±0.01	0.6939 ±0.01	🏆 **0.7075 ±0.01**	0.6861 ±0.0
	60	0.6637 ±0.01	0.6832 ±0.01	0.6968 ±0.0	**0.7058 ±0.01**	0.692 ±0.01	0.7029 ±0.01	0.7052 ±0.01	0.6899 ±0.01
	120	0.6637 ±0.01	0.692 ±0.01	0.699 ±0.01	0.7065 ±0.01	0.6948 ±0.01	0.7028 ±0.01	**0.7088 ±0.01**	0.6899 ±0.01
	180	0.6637 ±0.01	0.6907 ±0.01	0.699 ±0.01	0.7066 ±0.01	0.6948 ±0.01	0.7026 ±0.01	**0.7082 ±0.01**	0.6899 ±0.01
	300	0.6637 ±0.01	0.6995 ±0.01	0.699 ±0.01	0.7065 ±0.01	0.6948 ±0.01	0.7035 ±0.01	**0.7086 ±0.01**	0.6899 ±0.01
361083	30	0.3006 ±0.0	0.4043 ±0.0	0.6157 ±0.03	0.6361 ±0.01	0.0 ±0.0	-0.0 ±0.0	0.5356 ±0.0	🏆 **0.6784 ±0.0**
	60	0.3006 ±0.0	0.4043 ±0.0	0.6452 ±0.01	0.6483 ±0.0	0.0 ±0.0	0.3034 ±0.0	0.5821 ±0.0	**0.6784 ±0.0**
	120	0.3006 ±0.0	0.6306 ±0.0	0.6304 ±0.03	0.6496 ±0.0	0.3446 ±0.08	0.3034 ±0.0	0.5811 ±0.01	**0.6784 ±0.0**
	180	0.3006 ±0.0	0.6335 ±0.0	0.6551 ±0.02	0.6549 ±0.0	0.3918 ±0.08	0.3034 ±0.0	0.6168 ±0.01	**0.6784 ±0.0**
	300	0.3006 ±0.0	0.6371 ±0.0	**0.6836 ±0.01**	0.6513 ±0.0	0.3918 ±0.08	0.3034 ±0.0	0.6517 ±0.01	0.6784 ±0.0
361085	30	0.8351 ±0.05	0.7053 ±0.02	0.8739 ±0.04	0.7954 ±0.05	0.8122 ±0.04	0.8262 ±0.05	0.8696 ±0.04	🏆 **0.8837 ±0.04**
	60	0.8351 ±0.05	0.7353 ±0.03	**0.8864 ±0.04**	0.7986 ±0.05	0.8122 ±0.04	0.8363 ±0.05	0.8718 ±0.04	0.8839 ±0.04
	120	0.8351 ±0.05	0.8521 ±0.06	**0.8871 ±0.04**	0.8338 ±0.05	0.8209 ±0.04	0.8355 ±0.05	0.8743 ±0.05	0.8839 ±0.04
	180	0.8351 ±0.05	0.8624 ±0.06	0.8809 ±0.04	0.836 ±0.05	0.8209 ±0.04	0.8355 ±0.05	0.8742 ±0.05	**0.8839 ±0.04**
	300	0.8351 ±0.05	0.8542 ±0.06	0.8779 ±0.04	0.8449 ±0.04	0.8209 ±0.04	0.8416 ±0.05	0.8739 ±0.05	**0.8839 ±0.04**
361086	30	🏆 0.9786 ±0.0	🏆 0.9764 ±0.0	🏆 **0.9793 ±0.0**	0.9775 ±0.0	0.4023 ±0.51	-0.0 ±0.0	🏆 0.9791 ±0.0	🏆 0.9789 ±0.0
	60	0.9786 ±0.0	0.9791 ±0.0	**0.9793 ±0.0**	0.9784 ±0.0	0.652 ±0.56	-0.0 ±0.0	**0.9793 ±0.0**	0.9789 ±0.0
	120	0.9786 ±0.0	0.9793 ±0.0	**0.9794 ±0.0**	0.9787 ±0.0	0.9783 ±0.0	0.9789 ±0.0	**0.9794 ±0.0**	**0.9794 ±0.0**
	180	0.9786 ±0.0	**0.9795 ±0.0**	0.9794 ±0.0	0.9788 ±0.0	0.9783 ±0.0	0.9791 ±0.0	**0.9795 ±0.0**	0.9794 ±0.0
	300	0.9786 ±0.0	**0.9795 ±0.0**	0.9794 ±0.0	0.9791 ±0.0	0.9783 ±0.0	0.9793 ±0.0	0.9794 ±0.0	0.9794 ±0.0
361087	30	0.7728 ±0.0	0.6925 ±0.01	0.9309 ±0.0	0.9257 ±0.0	0.9166 ±0.0	0.9226 ±0.0	0.9321 ±0.0	🏆 **0.9354 ±0.0**
	60	0.7728 ±0.0	0.9293 ±0.0	0.9305 ±0.0	0.9263 ±0.0	0.9166 ±0.0	0.9242 ±0.0	0.9327 ±0.0	**0.9354 ±0.0**
	120	0.7728 ±0.0	0.9334 ±0.0	0.9315 ±0.0	0.9331 ±0.0	0.9166 ±0.0	0.9242 ±0.0	**0.9358 ±0.0**	0.9356 ±0.0
	180	0.7728 ±0.0	0.9338 ±0.0	0.9314 ±0.0	0.9332 ±0.0	0.9209 ±0.01	0.9259 ±0.0	**0.9375 ±0.0**	0.9356 ±0.0
	300	0.7728 ±0.0	0.934 ±0.0	0.9318 ±0.0	0.9331 ±0.0	0.9209 ±0.01	0.9267 ±0.0	**0.938 ±0.0**	0.9356 ±0.0
361088	30	0.8964 ±0.01	0.8576 ±0.11	🏆 **0.9194 ±0.01**	0.9107 ±0.0	0.5104 ±0.47	0.1847 ±0.0	0.896 ±0.01	0.9152 ±0.01
	60	0.8964 ±0.01	0.9218 ±0.01	**0.9231 ±0.01**	0.9131 ±0.0	0.6121 ±0.53	0.6661 ±0.42	0.9154 ±0.01	0.9152 ±0.01
	120	0.8964 ±0.01	0.9234 ±0.01	0.924 ±0.01	0.9142 ±0.01	0.8569 ±0.11	0.9053 ±0.01	**0.9242 ±0.01**	0.9152 ±0.01
	180	0.8964 ±0.01	0.9239 ±0.01	0.9235 ±0.01	0.9148 ±0.01	0.8569 ±0.11	0.9054 ±0.01	**0.9257 ±0.01**	0.9152 ±0.01
	300	0.8964 ±0.01	0.9237 ±0.01	0.9238 ±0.01	0.9171 ±0.0	0.9007 ±0.03	0.9054 ±0.01	**0.9257 ±0.01**	0.9168 ±0.01
361089	30	0.6844 ±0.0	🏆 **0.8594 ±0.0**	0.8493 ±0.01	0.8574 ±0.0	0.8269 ±0.01	-0.0007 ±0.0	0.8386 ±0.0	0.8587 ±0.0
	60	0.6844 ±0.0	**0.8638 ±0.0**	0.8477 ±0.0	0.8577 ±0.0	0.8269 ±0.01	0.8488 ±0.0	0.8571 ±0.0	0.8594 ±0.0
	120	0.6844 ±0.0	**0.8662 ±0.0**	0.8457 ±0.0	0.863 ±0.01	0.8269 ±0.01	0.8551 ±0.0	0.8646 ±0.0	0.8597 ±0.0
	180	0.6844 ±0.0	0.8662 ±0.0	0.8469 ±0.0	0.8636 ±0.0	0.8291 ±0.01	0.8551 ±0.0	**0.8694 ±0.0**	0.8597 ±0.0
	300	0.6844 ±0.0	0.8662 ±0.0	0.8528 ±0.01	0.8645 ±0.0	0.8282 ±0.01	0.8557 ±0.0	**0.8722 ±0.0**	0.8597 ±0.0
361090	30	0.6826 ±0.0	0.6904 ±0.0	0.6876 ±0.0	0.6859 ±0.0	0.6787 ±0.0	0.683 ±0.01	🏆 **0.6911 ±0.0**	0.6897 ±0.0
	60	0.6826 ±0.0	0.6893 ±0.0	0.6899 ±0.0	0.6911 ±0.0	0.683 ±0.01	0.687 ±0.0	**0.6931 ±0.0**	0.6911 ±0.0
	120	0.6826 ±0.0	0.69 ±0.0	0.6896 ±0.0	0.6919 ±0.0	0.6835 ±0.01	0.6872 ±0.0	**0.6947 ±0.0**	0.6911 ±0.0
	180	0.6826 ±0.0	0.6901 ±0.0	0.6904 ±0.0	0.6915 ±0.0	0.6842 ±0.01	0.688 ±0.0	**0.6953 ±0.0**	0.6911 ±0.0
	300	0.6826 ±0.0	0.6912 ±0.0	0.6919 ±0.0	0.6904 ±0.0	0.6886 ±0.0	0.6882 ±0.0	**0.6955 ±0.0**	0.6911 ±0.0
42570	30	0.4675 ±0.03	🏆 0.5175 ±0.02	0.3872 ±0.27	0.4998 ±0.03	0.0 ±0.0	0.0 ±0.0	0.5089 ±0.03	🏆 **0.5187 ±0.04**
	60	0.4675 ±0.03	0.5181 ±0.02	0.3291 ±0.36	0.5054 ±0.04	0.0 ±0.0	0.0 ±0.0	0.5117 ±0.03	**0.5188 ±0.04**
	120	0.4675 ±0.03	**0.5196 ±0.02**	0.5088 ±0.04	0.5101 ±0.02	0.0 ±0.0	0.0 ±0.0	0.5147 ±0.03	0.5193 ±0.04
	180	0.4675 ±0.03	**0.52 ±0.02**	0.5089 ±0.04	0.5118 ±0.02	0.0 ±0.0	0.0 ±0.0	0.5139 ±0.03	0.5193 ±0.04
	300	0.4675 ±0.03	**0.5201 ±0.02**	0.5082 ±0.04	0.5121 ±0.03	0.0 ±0.0	0.0 ±0.0	0.5104 ±0.03	0.5193 ±0.04
$Avg(\mathcal{A})$		0.7795	0.8265	0.8432	0.8449	0.7459	0.706	0.8466	**0.8518**

Table 14. A comparison between our proposed approach against baseline and 6 different AutoMLs on Benchmark #2; each value is an average of 3 different experiments (3 random seeds); champions per dataset/time budget (row) marked in **bold**; 🏆 indicates Method/AutoM champion per dataset (rows) across all AutoMLs based on the average of 3 seeds and 5 time budgets (15 experiments).

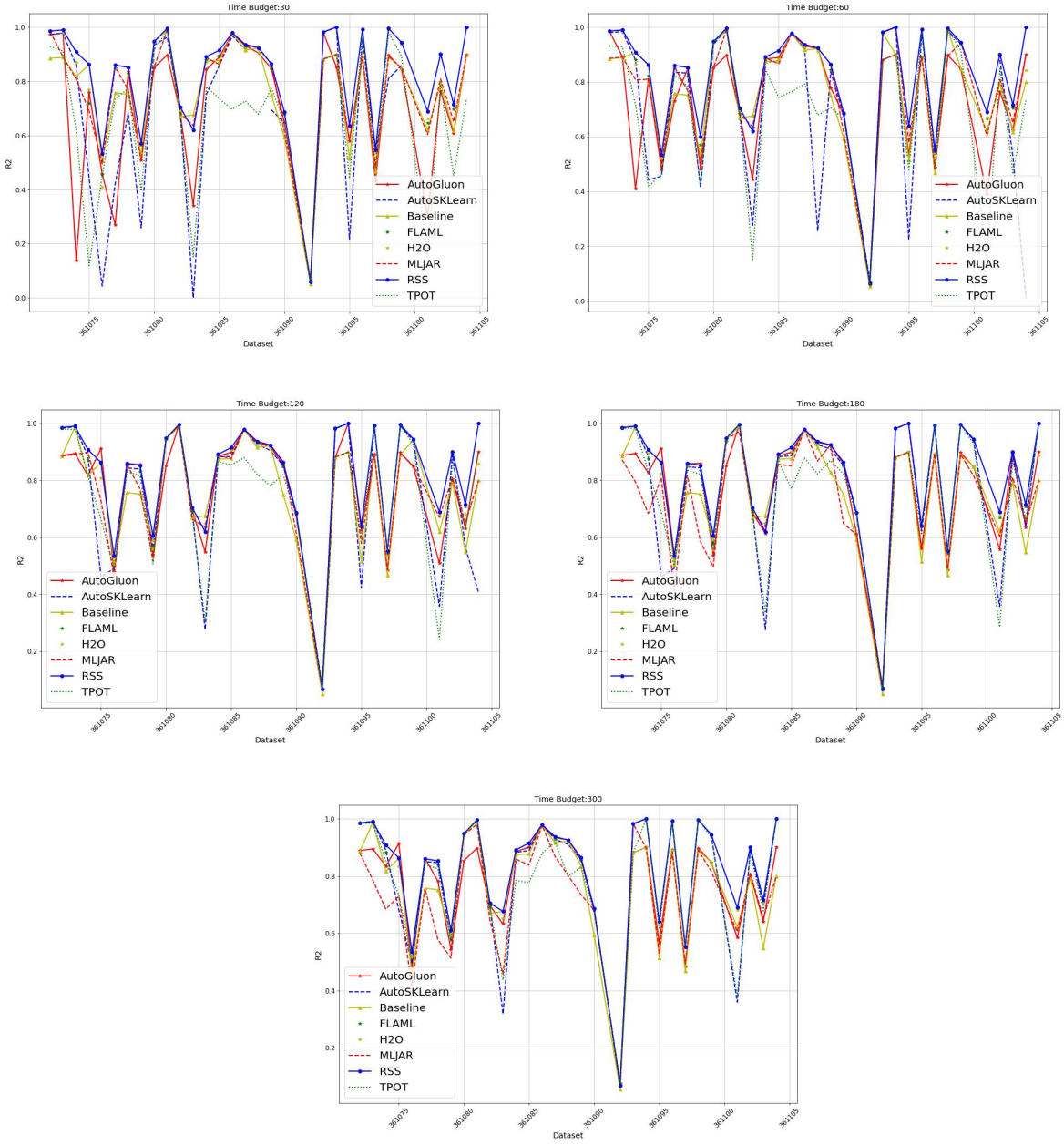


Figure 11. R^2 performance comparison (higher values indicate better performance) of RSS (our) against a baseline and 6 different AutoML tools per time budget; each sub-figure represents the performance of tool/method for a given time budget (second); Note that the x-axis represents the sorted names of 31 datasets from Benchmark #1 (24 labels hidden for readability).