Man versus Machine: AutoML and Human Experts’ Role in Phishing Detection

Rizka Purwanto\textsuperscript{1}, Arindam Pal\textsuperscript{1,2,3}, Alan Blair\textsuperscript{1}, and Sanjay Jha\textsuperscript{1}

\textsuperscript{1}University of New South Wales, Australia
\textsuperscript{2}Data61, CSIRO, Sydney, Australia
\textsuperscript{3}Cyber Security Cooperative Research Centre, Australia

This work has been submitted to the IEEE for possible publication. Copyright may be transferred without notice, after which this version may no longer be accessible.

Abstract

Machine learning (ML) has developed rapidly in the past few years and has successfully been utilized for a broad range of tasks, including phishing detection. However, building an effective ML-based detection system is not a trivial task, and requires data scientists with knowledge of the relevant domain. Automated Machine Learning (AutoML) frameworks have received a lot of attention in recent years, enabling non-ML experts in building a machine learning model. This brings to an intriguing question of whether AutoML can outperform the results achieved by human data scientists. Our paper compares the performances of six well-known, state-of-the-art AutoML frameworks on ten different phishing datasets to see whether AutoML-based models can outperform manually crafted machine learning models. Our results indicate that AutoML-based models are able to outperform manually developed machine learning models in complex classification tasks, specifically in datasets where the features are not quite discriminative, and datasets with overlapping classes or relatively high degrees of non-linearity. Challenges also remain in building a real-world phishing detection system using AutoML frameworks due to the current support only on supervised classification problems, leading to the need for labeled data, and the inability to update the AutoML-based models incrementally. This indicates that experts with knowledge in the domain of phishing and cybersecurity are still essential in the loop of the phishing detection pipeline.

1 Introduction

Despite the availability of anti-phishing technologies, phishing attacks are still thriving and have caused data breaches of personal sensitive information and
private company data. Phishing attacks were reported to double in 2020 [3], and have caused significant financial losses of roughly between $60 million and $3 billion per year in the United States [28]. With the rapidly evolving nature of phishing, it would be ideal to have an automated detection system which could quickly adapt to phishing data changes and robustly detect these attacks.

Phishing is a cyber-attack that aims at stealing sensitive information by impersonating a legitimate person, company, or organization. By using social engineering and psychological tactics, phishing attackers exploit human vulnerabilities by sending messages that seem authentic and usually have a sense of urgency, persuading users to follow the attacker’s instruction in the message. While phishing attacks nowadays are also advertised through voice messages and SMS messages, reports show that phishing is still dominated by the traditional ones delivered via emails [3].

During the past decade, machine learning has developed rapidly and has successfully performed a broad range of tasks. Machine learning techniques have shown to be effective in detecting phishing attacks. However, building a machine learning model is not a trivial task, and requires data scientists with knowledge of the relevant domain. In the past couple of years, Automated Machine Learning (AutoML) has gained much attention, enabling non-ML experts in building a machine learning model.

The emergence of AutoML frameworks brings us to the question of whether AutoML-generated models could outperform manually trained machine learning models on phishing data, how AutoML could assist non-experts in building machine learning models, and to what extent we could automate the whole process of building an ML-based detection system. There are a number of past studies that have tried to investigate whether the existence of AutoML frameworks would affect the roles of human experts in the ML development pipeline [10]. However, to the best of our knowledge, we are the first to discuss this topic specifically in the case of phishing detection systems.

2 Automated Machine Learning

There has been significant research conducted in the areas of machine learning and deep learning since 1995, resulting in the development of various tools, such as Weka (1990s), scikit-learn (2007-2010), Tensorflow (2015), Keras (2015), and so on [55]. The emergence of these tools has enabled multidisciplinary research and the application of machine learning and deep learning techniques, which has shown promising results, demonstrating the potential of machine learning models to solve various problems. However, it has also become evident that developing a machine learning model is a complex task, requiring intuition, experience, and technical expertise to tune the model’s hyperparameters. The heavy reliance of machine learning development on human experts motivates researchers to explore the possibility of inventing a technique to automate the development of machine learning models. Such attempts focusing on AutoML projects have been initiated by researchers and machine learning practitioners,
and were followed by startups that sell AutoML frameworks as part of their business models. Some of the first AutoML tools are AutoWEKA [32] which is based on Weka [23], followed by auto-sklearn [18] and TPOT [48] which are both built on the scikit-learn library on Python [51]. Various AutoML frameworks have also emerged as the product of the ChaLearn AutoML challenge competitions [1] between 2015 and 2018, which is still conducted every year until now.

There are some existing studies which focus on performing a thorough literature review on the comparison and discussions on past works in AutoML approaches and tools [25, 59]. In general, AutoML tools aim to automate various aspects of the machine learning pipeline, including data preprocessing, feature engineering, model generation, and model evaluation. A standard full ML pipeline is shown in Figure 1. Based on previous literature reviews of AutoML tools [59, 25], we could divide AutoML frameworks based on which aspect of the machine learning pipeline it tries to automate; namely data preparation, feature engineering, model generation, and model evaluation.

2.1 Data Preparation

Data preparation includes data collection and preprocessing of the data, such as data cleaning and data augmentation. Automated data collection is necessary for tasks in which the data needs to be extended and continuously updated. Data cleaning aims to remove noises and handle missing values in the data. Meanwhile, data augmentation enhances the model’s performance by generating new data based on the existing data and by preventing the trained model from over-fitting.

2.1.1 Data Collection

Plenty of open datasets are now being shared among researchers, especially image datasets (e.g. MNIST handwritten digital dataset [35]). However, it remains a challenge to obtain a high quality dataset for some specific tasks, including phishing websites and email datasets which require anonymization. This issue can be solved with two different approaches: data searching and data synthesis. Some automated methods that help in the performance of data searching
include learning-based self-labeling methods for unlabeled data \[8\] and synthetic minority over-sampling technique (SMOTE) \[7\] to deal with dataset class imbalances. Meanwhile, various data generation techniques also exist to automate data synthesis process, for example using Generative Adversarial Networks (GAN) \[30, 47, 5\] and reinforcement learning-based methods in data simulators \[53\].

2.1.2 Data Cleaning

Data cleaning is an essential process to eliminate data noises, which could negatively affect machine learning model performances if not removed. However, data cleaning is generally a costly task, since it requires experts with specialist knowledge. Over time, there have been efforts to automate the process of data cleaning. Initially, data cleaning was performed through crowd-sourcing \[25\]. To enhance efficiency, a past work proposed a data cleaning technique in which data scientists define specific data cleaning operations based on a small subset of the data, and apply these cleaning operations to the full dataset. Past studies attempted to further improve this method using machine learning techniques, such as boosting and hyperparameter optimization, to find the best data cleaning operation pipeline or combination \[33, 34\]. To be applicable for real world data, the data cleaning methods should be able to clean data steadily. Several past works have proposed a technique to evaluate data cleaning algorithms that can perform continuously, and to orchestrate cleaning workflows that can learn from past cleaning tasks.

2.1.3 Data Augmentation

Data augmentation aims to enrich the dataset by generating new data by transforming the existing data. Past studies have proposed various methods to perform neural-based transformations on image data, such as adversarial noise \[44\], neural style transfers \[45\], and GAN-based techniques \[2\]. Meanwhile, there are two approaches to textual data augmentation: data warping and synthetic over-sampling \[60\]. Recently, various methods have been proposed to search for augmentation policies for different tasks using reinforcement learning \[11\], and various other improved algorithms \[37, 38\].

2.2 Feature Engineering

Feature engineering in supervised machine learning problems is defined as the process of finding explanatory variables that are predictive of the classification outcome \[59\]. This process is typically performed in a trial-and-error fashion and often requires extensive knowledge of the relevant domain. Feature engineering holds an important role, since its quality affects the machine learning model’s performance heavily \[13\]; however, it is also a time-consuming task. To help with this process, some AutoML tools provided automated feature engineering methods with the goal of constructing new feature sets that give the
best machine learning model performance.

The automated feature engineering task can be formally defined as follows \[59\]. Given a feature set with \( m \) number of features, \( F = [f_1, f_2, ..., f_m] \), a target vector, \( y \), and a machine learning algorithm, \( M \), let \( P_M(F, y) \) indicate the performance of the model with the corresponding algorithm, target vector, and feature set. Assume there are \( k \) transformation functions, \( t_1, t_2, ..., t_k \) which can be applied to the features, and a sequence of transformations of the features \( s = t_1(t_2(...(f_1))) \). The goal of the automated feature engineering process is to find a set of transformation sequences \( S = [s_1, ..., s_r] \) to produce a new set of features, \( F_{new} \), which satisfies \( \text{arg max}_S P_M(F_{new}, y) \). The goal of automated feature engineering is to obtain the best set of features and feature transformations which give the best classification performance from structured data.

Traditionally, new features are constructed manually by performing some standard transformations, such as standardization, normalization, or feature discretization. To improve the efficiency of such processes, automatic feature construction methods using decision trees \[19, 61\], genetic algorithms \[56\], and annotation-based approaches \[54\], have been proposed to search and evaluate the best combination of transformations.

Besides the construction of new features, feature engineering can also be performed by reducing the feature dimensionality to extract the most informative features and reduce redundancies in the feature set. This process is performed by applying mapping functions, such as principal component analysis (PCA) or linear discriminant analysis (LDA). In recent studies, there have been other improved methods proposed to perform feature extraction, e.g., using autoencoder-based algorithms \[45\] and unsupervised feature extraction methods \[20\].
2.3 Model Generation and Evaluation

There are two important elements in model generation: search space and the optimization method. The search space defines the structure and design principles of the machine learning models. Given a certain model, we could apply hyperparameter optimization (HPO), which aims to find the optimal training-related hyperparameters (e.g., learning rate), and architecture optimization (AO), to obtain the best hyperparameters associated with the model's structure or design (e.g., number of neighbors for $k$-NN, or number of layers or neural architectures for deep neural networks).

Traditional hyperparameter optimization strategies usually do not make any assumptions about the search space. One of the simplest hyperparameter optimization methods is grid search; a brute-force method to find the best set of hyperparameters, given a finite set of values for each hyperparameter specified by users. Another simple alternative to grid search is random search, which relies on sampling from a user-specified set of hyperparameter values under a certain budget constraint. There is also another kind of hyperparameter optimization method which performs "optimization from samples" [9], e.g., particle swarm optimization (PSO) [15] and evolutionary algorithms [4], which are both inspired by biological behaviours. Meanwhile, Bayesian optimization has emerged as the most advanced hyperparameter optimization method used in AutoML frameworks. Bayesian optimization builds a probabilistic model, which maps different hyperparameter configurations to their performance with some degree of uncertainty.

Besides hyperparameter optimization, finding the best model architecture is also an important and non-trivial task when building a machine learning classifier. With the emerging research in neural networks in the past decade, neural architecture search (NAS) has gained great interest in the AutoML community. There are three essential components of a neural architecture search (NAS): the search space of neural architectures, architecture optimization, and model evaluation methods. An intuitive method to perform model evaluation is to assess model performance after training and the neural network has converged. However, this takes an extensive time and resources due to the amount of computation needed. Past studies have focused on finding methods to accelerate the model training and model evaluation process. The first approach is by using low-fidelity model evaluation by reducing the model size. Another technique that has been proposed is weight sharing, which can make the model training time faster by utilizing knowledge regarding weights of prior network architectures. Past studies also proposed the use of surrogate-based methods which could estimate the black-box function of a neural network model, making it easier to obtain the best model configuration and performance. Another proposed approach is early stopping, which was initially used in classical ML to prevent over-fitting. More recent studies have improved early stopping to perform computation on smaller set of data, making it faster to compute. We will not cover this topic in detail in our paper. However, more thorough discussions on studies on hyperparameter optimization and architecture optimization,
especially neural architecture search (NAS), are covered by Waring et al. [59] and He et al. [25] in their literature reviews.

3 Phishing Detection Systems

Machine learning has shown to be effective in detecting phishing attacks based on past studies [12]. While attackers have been using various strategies to conduct phishing attacks, emails still remain a primary delivery method for attackers. In this section, we provide the general workflow of phishing detection systems and how it interacts with external parties, e.g. users and blacklist providers. The phishing detection systems workflow is shown in Figure 3.

Phishing attacks start with the attacker broadcasting emails that contain a message which tries to convince the receivers to proceed further by clicking on the provided link. Automated phishing detection systems inspect this email’s raw data and analyze the message, URL, content or visual appearance of the associated web page, and page hosting information. After automatically fetching this information, the feature extraction process is performed. Various past studies have focused on investigating the performance of features extracted from the phishing emails and websites. The output of the feature extraction process is a vector that contains values associated with each specific feature, each representing an email or website. Afterwards, a classifier is built, which predicts whether
Table 1: Phishing Dataset Details

| Task  | Num of Rows | Num of Features | Num of Classes | Details |
|-------|-------------|-----------------|----------------|---------|
| eml1a | 3668        | 3               | 2              | Raw phishing email dataset [58] with feature extraction proposed in [16] |
| eml1b | 3668        | 23              | 2              | Raw phishing email dataset [58] with feature extraction proposed in [6] |
| eml1c | 3668        | 791             | 2              | Raw phishing email dataset [58] with feature extraction proposed in [22] |
| eml1d | 3668        | 10              | 2              | Raw phishing email dataset [58] with feature extraction proposed in [52] |
| eml1e | 3668        | 579             | 2              | Raw phishing email dataset [58] with feature extraction proposed in [57] |
| url1  | 9205        | 3               | 2              | Phishing email dataset and feature extraction proposed in [10] |
| url2  | 96800       | 500             | 2              | Phishing website URL dataset and feature extraction proposed in [40] |
| url3  | 76728       | 12              | 2              | Phishing website URL dataset and feature extraction proposed in [42] |
| web1  | 15185       | 79              | 5              | Phishing website HTML dataset and feature extraction proposed in [41] |

can AutoML outperform humans?

The advancement of AutoML frameworks raises the question of whether AutoML can outperform humans in building machine learning models for detecting phishing. To answer this question, we performed an experiment to compare the performances of the models built using AutoML and the ones that are manually crafted.

4.1 Dataset and Performance Metrics

To perform a comparison between AutoML and manually built machine learning models, we tested the models on various phishing email, URL, and website datasets. Besides using publicly available datasets [16, 40, 42, 41, 46], we also performed feature extraction proposed in past studies [16, 6, 22, 52, 57, 58] on a raw phishing email dataset compiled by Verma et al. [58] to construct new sets of phishing email data. In each task of this experiment, we trained machine learning models using a specific dataset and compared the performance of models constructed with and without AutoML frameworks. Further details of each task are provided in Table 1.

While various metrics were measured during this experiment, we are particularly interested in the following performance metrics:

- **Accuracy**
  
  Accuracy measures the number of correct predictions divided by the total
size of the data, and can be expressed as:

\[
\text{accuracy}(y, \hat{y}) = \frac{1}{n_s} \sum_{i=0}^{n_s} 1(\hat{y}_i = y_i)
\]  

(1)

where \(\hat{y}_i\) is the predicted value of the \(i\)-th sample, \(y_i\) is the corresponding true value, and \(n_s\) is the total number of samples. In a multi-class classification setting, e.g. Task url3, this formula would calculate the subset accuracy, or the percentage of samples which are classified correctly.

- **AUC score**
  AUC, or Area Under the ROC Curve, is the total area under the ROC (receiver operating characteristic) curve, which would depict the model’s performance in identifying the positive and negative examples for each class and finding the best threshold to separate between both examples. In a binary classification task, positive and negative examples would refer to phishing and legitimate samples. In a multi-class classification, we chose the *one-vs-one* heuristic, where the dataset is split into one dataset for each class versus every other class [24], and the final AUC score is obtaining by computing the average AUC of all possible pairwise combinations of classes.

- **F1-score**
  Due to the precision-recall trade-off, it is challenging to have both precision and recall high, especially in imbalanced datasets. F1-score calculates the harmonic mean of recall (true positive rate) and precision, measuring the model’s performance correctly performing classification while heavily penalizing low recall or low precision scores.

- **Training duration**
  The training duration refers to the total amount of time each framework or algorithm takes to process the given dataset, until a classification model is produced. We do not include the time each model takes to perform prediction on the testing dataset, as it is generally negligible.

4.2 **Experiment Constraints and Setup**

In this study, we compared the performance of various mature open source AutoML frameworks, which are briefly described as follows:

- **AutoGluon** [14]
  AutoGluon uses a multi-layer stack ensemble, in which multiple models are ensembled and stacked in multiple layers. The main difference between AutoGluon and existing AutoML frameworks is that AutoGluon utilizes almost every trained model to produce the final prediction instead of only selecting the best model.
• **auto-sklearn** [17]
  Auto-sklearn was a winner of the ChaLearn AutoML Challenge 1 in 2015-2016 and ChaLearn AutoML Challenge 2 in 2017-2018. It uses Bayesian optimization to obtain the best machine learning pipeline. Auto-sklearn features automatic ensemble construction and uses meta-learning to increase the probability of finding a good pipeline by warm-starting the search procedure.

• **GAMA** [21]
  GAMA supports configurable AutoML pipelines, which allow the selection of optimization and post-processing algorithms. By default, GAMA searches over linear ML pipelines and creates a model ensemble in the post-processing step. These pipelines can be optimized with an asynchronous evolutionary algorithm or ASHA.

• **H2OAutoML** [36]
  H2OAutoML performs a random search, which is followed by a model stacking stage. This framework uses the H2O machine learning package by default, which supports distributed training.

• **hyperopt-sklearn** [31]
  Hyperopt-sklearn allows various search strategies, including random search, and various sequential model based optimization (SMBO) techniques. These techniques include Tree of Parzen Estimators (TPE), Annealing and Gaussian Process Trees.

• **TPOT** [49]
  TPOT constructs machine learning pipelines of arbitrary length using scikit-learn algorithms [51] and allows the use of XGBoost algorithm. During its search, pre-processing and stacking are both considered. While the model’s pipeline length is arbitrary, TPOT performs multi-objective optimization, in which it aims to keep the number of pipeline components minimal while optimizing the main selected metric. TPOT also provides support for sparse matrices, multiprocessing, and custom pipeline components.

To ensure that our evaluation was performed fairly, we trained and tested all the models using the OpenML AutoML Benchmarking Framework [20] to make sure that the models were developed under the same constraints and with the same setup. Evaluating and comparing AutoML systems is challenging due to the subtle differences in problem definition, e.g. the design of the hyperparameter search space or the way time budgets are defined. The OpenML AutoML Benchmarking toolkit aims to address this issue by providing a standardized environment to perform in-depth experiments comparing a wide range of AutoML frameworks.

For each task in Table 1 we performed random splits on the whole dataset, then allocated 75% for training and 25% for testing respectively. All tasks were run using the same computing resources, with 8-core Intel Xeon and 32 GB
RAM. The maximum runtime for each task was set to 3,600 seconds (1 hour). Each task could use a maximum of 8 cores when multiprocessing is available. To reduce biases resulting from outlier results, we repeated the experiment 10 times for each task and observed the consistency of the performance in the evaluation metrics. We selected the default metrics to optimize, which were the AUC score for binary classification tasks and log loss for multi-class classification tasks.

We also selected several traditional machine learning algorithms to compare with the models built using the aforementioned AutoML frameworks. The selected algorithms are Logistic Regression, SVM, KNN, Decision Tree, Random Forest, Multi-layer Perceptron, and Gaussian Naive Bayes, which are available in the scikit-learn Python package [51]. To train these models, we also used the same dataset split and computing resources as the one used in the AutoML setting, with a maximum of 8 core when multiprocessing was available. For each task in Table 1, we ran 10 experiments to obtain the best model for each algorithm and to observe any variance in the models’ performance. We manually defined a set of model hyperparameters and performed random searches to obtain the best model, optimizing the AUC score in binary classification tasks and log loss in multi-class classification tasks. Unlike the AutoML experiment setting, we did not set the maximum runtime for each task as this feature was not available. However, we set the number of iterations during the random search for each task. The iteration number was manually set to 20 for SVM and 100 for all other algorithms.

4.3 Results

We computed the average performance metrics of each model and task in all experiments. To perform comparisons between models built using AutoML frameworks and non-AutoML algorithms, we selected two models from each task which would represent the best model built using the AutoML framework and non-AutoML algorithms. The best performance in this case would be in terms of the average AUC score as this metric was optimized during the model search process.

Comparisons of the accuracy and AUC score of the best model built using AutoML and non-AutoML frameworks are provided in Figure 4 and Figure 5. As shown in these figures, the performances in terms of AUC score and accuracy are almost similar between models built using AutoML and manually developed ML models. However, there are some exceptions in Task em1_1a and Task em1_2 in which AutoML-based models significantly outperformed manually built models. The AUC score difference is between 13.5% to 23.3%, and the accuracy difference is between 11.6% to 22.9%.

In terms of training time, we found that it was much faster to train the models manually and achieve this level of performance without AutoML frameworks. Figure 6 provides a summary of the comparison of training duration between AutoML frameworks and traditional ML algorithms.
Figure 4: Accuracy

Figure 5: AUC Score

Figure 6: Duration
5 When Does AutoML Outperform Humans?

To gain a better understanding of the results in Section 4, we analyzed the complexity of performing classification on the datasets assigned to each task using the DCoL library [50]. The aim of this experiment is to understand in what kind of classification task AutoML frameworks provide better results.

The complexity measures that are computed can be grouped into several categories, which are briefly described as follows.

- **Measures of the overlaps from different classes based on the discriminative power of the features**, including the maximum Fisher’s discriminant ratio (F1), directional-vector maximum Fisher’s discriminant Ratio (F1v), volume of per-class bounding boxes overlap (F2), maximum individual feature efficiency (F3), and collective feature efficiency (F4).

- **Measures of linearity**, which includes the minimised sum of the error distance of a linear classifier (L1), training error of a linear classifier (L2), and non-linearity of a linear classifier (L3).

- **Neighborhood measures**, which includes the fraction of points on the class boundary (N1), ratio of intra/inter class nearest neighbor distance (N2), leave-one-out error rate of the one-nearest neighbor classifier (N3), non-linearity of the one-nearest neighbor classifier (N4), and maximum covering spheres fraction (T1).

- **Measures of dimensionality**, which includes the average number of points per dimension (T2).

- **Measures of class imbalance**, which includes the entropy of class proportions (C1) and imbalance ratio (C2).

A complete analysis of task complexity is provided in Figure 7. In this figure, darker cells are associated with higher complexity. In general, a higher parameter value usually indicates that the classification task is more complex, with an exception for F1, F1v, F3, F4, and T2 complexity measures where a higher measure value corresponds to simpler classification tasks. Higher complexity means that it is more difficult to achieve good classification performances. Further details on complexity measures are not discussed in our paper. However, we refer to [26, 27, 39] for those interested in more detailed explanations regarding measuring supervised classification complexity.

To observe the relationship between a task’s complexity and classification performance, we performed a correlation test between AutoML performance gain or improvement ($\text{performance}_{AutoML} - \text{performance}_{nonAutoML}$) and a complexity measure based on one of the parameters previously mentioned. Full results of this analysis is provided in Table 7 (Appendix B). A summary of the correlation test with statistically significant results ($p < 0.05$) is shown in Table 2.
Based on the correlation test, there are negative relationships between the F1 complexity measure (maximum Fisher’s discriminant ratio) and the performance gain in terms of AUC score. Note that a higher Fisher’s discriminant ratio would correspond to a simpler classification task. The negative relationship between F1 complexity measure and the AUC score gain would indicate that there would be a larger AUC score difference between AutoML-based models and manually built (non-AutoML) models in more complex tasks (with lower F1 complexity measure). The same goes with F3 and F4 complexity measures, in which both would have higher values when the classification task is simpler. Thus, a negative correlation between the F3 complexity measure and AUC score gain would indicate that AutoML-based models would outperform more significantly in more complex classification tasks (with lower F3 complexity measure). The correlation test also shows that AutoML-based models would outperform manually built models, in terms of accuracy, in more complex classification tasks with lower F4 complexity measure.

Furthermore, the correlation test also showed a positive correlation between some of the neighbor-based complexity measures (N1, N3, N4) and the AUC score performance gain. Unlike the previous complexity measures, a higher neighbor-based complexity measure indicates a more complex classification task. The results in Table 2 show that the AutoML-based models would be more likely to outperform manually developed classification models in a more complex classification task with higher N1, N3, and/or N4 complexity measures. These results show that AutoML-based models would be most beneficial when used to build classification models in complex settings where the features are not quite discriminative, and in datasets with overlapping classes or relatively high degrees of non-linearity.

Referring back to the results in Section 4, we found that this result is consistent with empirical findings. As shown in Figure 4 and Figure 5, AutoML
| Complexity Measure | Performance Gain | Correlation | p-value   |
|-------------------|------------------|-------------|-----------|
| F1                | AUC Score        | -0.70909    | 0.021666  |
| F3                | AUC Score        | -0.67273    | 0.033041  |
| F4                | Accuracy         | -0.70909    | 0.021666  |
| N1                | AUC Score        | 0.745455    | 0.01333   |
| N3                | AUC Score        | 0.721212    | 0.018573  |
| N4                | AUC Score        | 0.769697    | 0.009222  |

Table 2: Correlation between Complexity Measure and Performance Gain ($p < 0.05$)

Frameworks significantly outperformed manually crafted ML models in Task eml_1a and Task eml_2. As shown in Figure 7, Task eml_1a is deemed to be complex as indicated by its feature-based complexity measures (F1, F3, F4) and neighborhood complexity measures (N1, N3, N4). Meanwhile, the complexity of Task eml_2 is indicated by the feature-based measures (F1, F3, F4). This confirms that AutoML-based models outperform manually built ML models in these types of complex classification tasks.

6 Automating Phishing Detection with AutoML Frameworks

In this section, we discuss further on the AutoML-based models’ performances, followed by a discussion on the opportunity and challenges of the use of AutoML frameworks in automated phishing detection systems, and a highlight on the study limitations and potential future works.

6.1 AutoML-based Models’ Performances

In this section, we analyze further the AutoML-based models’ performances in each classification task, and observe the relationship between a model’s performance and complexity level of the classification task. We provide more details on the average accuracy, AUC score, and duration of each framework in Table 4 in Appendix A. The average AUC score, accuracy, and F1 score of each task is provided in Figure 8, Figure 9, and Figure 10. In these figures, we also computed the standard deviation and the confidence interval of each metric in every classification task.

We performed a correlation test to observe the relationship between the performance of an AutoML-based model on a specific classification task and a measure indicating the task’s complexity, as shown in Table 6 (Appendix B). We provide a summary of the correlation analysis with statistically significant results in Table 3. As shown in this table, there are five complexity measures that have a significant correlation with AutoML-based models’ performances, namely F4, L3, T1, C1, and C2 complexity measures. A higher F4 complexity measure would indicate a simpler classification task, whereas higher L3, T1, C1,
Table 3: Correlation between Complexity Measure and AutoML-based Models’ Performance ($p < 0.05$)

| Complexity Measure | Performance Metric | Correlation | p-value |
|--------------------|--------------------|-------------|---------|
| F4                 | Accuracy           | 0.648485    | 0.04254 |
| L3                 | Accuracy           | -0.69697    | 0.025097|
| L3                 | F1 Score           | -0.68485    | 0.028883|
| L3                 | AUC Score          | -0.7697     | 0.009222|
| T1                 | Accuracy           | 0.830303    | 0.00294 |
| T1                 | F1 Score           | 0.781818    | 0.007547|
| C1                 | F1 Score           | -0.69176    | 0.026678|
| C1                 | AUC Score          | -0.834      | 0.003705|
| C2                 | F1 Score           | -0.73055    | 0.016499|
| C2                 | AUC Score          | -0.88572    | 0.000649|

Figure 8: AutoML-based Models’ AUC Score per Task

or C2 complexity measure indicates that the classification task is more complex.

The results in Table 3 are quite intuitive in terms of the relationship between F4, L3, C1, C2 complexity measure and the performance metrics, where the performance would improve in simpler classification tasks. Interestingly, there is a positive correlation between AutoML-based model performance (accuracy and F1 score) and T1 complexity measure, indicating that the model seems to have better performance when classifying on datasets with higher fraction of maximum covering spheres.

6.2 Opportunity and Challenges

Machine learning models built using AutoML frameworks have shown to outperform manually built models (Figure 4 and Figure 5). It is also shown that performance gain correlates with several complexity measures, such as F1, F3, F4, N1, N3, and N4 complexity measures (Table 2), indicating that human experts would benefit AutoML frameworks greatly in automating machine learning development, when dealing with complex classification tasks particularly
Figure 9: AutoML-based Models’ Accuracy per Task

Figure 10: AutoML-based Models’ F1 Score per Task
with dataset features having low discriminative power, datasets with high class overlap, and dataset with high degree of non-linearity.

However, several challenges remain to be addressed in using AutoML frameworks for implementing a full machine learning pipeline of phishing detection systems. AutoML frameworks currently focus on supervised classification tasks with labeled datasets, which are not always available in the case of phishing classification tasks. Furthermore, most AutoML frameworks build stacked or ensembles of ML models, which are difficult to implement and update in an incremental learning setting. With these limitations, the role of data scientists and security experts is still crucial in observing whether the classification models need retraining due to the existence of concept drift, or if the features in the dataset are no longer representative of the attacks. It is also shown in Figure 4 and Figure 5 that while AutoML built models were able to outperform manually built ML models, the performance differences are not significant, except in several complex cases. Furthermore, the time needed to train manually built ML models is significantly lower when compared with training AutoML-based models (Figure 6).

7 Conclusion

Our study shows that AutoML frameworks are able to consistently produce models that perform similarly or better than manually built ML models. However, challenges remain in implementing these frameworks to fully automate phishing detection due to the support provided only on supervised classification problems, which consequently lead to the need for labeled data, and the inability to update the generated models incrementally. This indicates that experts with domain knowledge are still needed in-the-loop of the full phishing detection pipeline. Despite its limitations, our study also reveals that AutoML frameworks are able to outperform manually developed machine learning models in certain complex classification tasks, indicating that there are opportunities for utilizing AutoML to improve human-guided phishing detection systems. Future studies which further explore the collaboration of human experts and AutoML, as well as the design of the human-in-the-loop system architectures would be beneficial in improving phishing detection.

Acknowledgement

This work has been supported by the Cyber Security Cooperative Research Centre Limited, whose activities are partially funded by the Australian Government’s Cooperative Research Centres Programme.

Rizka Widyarini Purwanto was supported by a UNSW University International Postgraduate Award (UIPA) scholarship. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the scholarship provider.
A sincere thank you to Muhammad Johan Alibasa for his constructive feedback and discussions on the manuscript.

References

[1] AutoML@ChaLearn. https://automl.chalearn.org/. Accessed: 2021-03-29.

[2] Antreas Antoniou, Amos Storkey, and Harrison Edwards. Data augmentation generative adversarial networks. *arXiv preprint arXiv:1711.04340*, 2017.

[3] APWG. *APWG Phishing Trends Reports*. Anti Phishing Working Group, 2021.

[4] Thomas Back. *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*. Oxford university press, 1996.

[5] Christopher Bowles, Liang Chen, Ricardo Guerrero, Paul Bentley, Roger Gunn, Alexander Hammers, David Alexander Dickie, Maria Valdés Hernández, Joanna Wardlaw, and Daniel Rueckert. Gan augmentation: Augmenting training data using generative adversarial networks. *arXiv preprint arXiv:1810.10863*, 2018.

[6] Madhusudhanan Chandrasekaran, Krishnan Narayanan, and Shambhu Upadhyaya. Phishing Email Detection Based on Structural Properties. In *NYS Cyber Security Conference*, volume 3, 2006.

[7] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

[8] Brendan Collins, Jia Deng, Kai Li, and Li Fei-Fei. Towards scalable dataset construction: An active learning approach. In *European conference on computer vision*, pages 86–98. Springer, 2008.

[9] Andrew R Conn, Katya Scheinberg, and Luis N Vicente. *Introduction to derivative-free optimization*. SIAM, 2009.

[10] Anamaria Crisan and Brittany Fiore-Gartland. Fits and Starts: Enterprise Use of AutoML and the Role of Humans in the Loop. *arXiv:2101.04296 [cs]*, January 2021. arXiv: 2101.04296.

[11] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation strategies from data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 113–123, 2019.
[12] Avisha Das, Shahryar Baki, Ayman El Aassal, Rakesh Verma, and Arthur Dunbar. SoK: A Comprehensive Reexamination of Phishing Research From the Security Perspective. IEEE Communications Surveys & Tutorials, 22(1), 2019.

[13] Pedro Domingos. A Few Useful Things to Know About Machine Learning. Communications of the ACM, 55(10):78–87, 2012.

[14] Nick Erickson, Jonas Mueller, Alexander Shirkov, Hang Zhang, Pedro Larroy, Mu Li, and Alexander Smola. AutoGluon-Tabular: Robust and Accurate AutoML for Structured Data. arXiv preprint arXiv:2003.06505, 2020.

[15] Hugo Jair Escalante, Manuel Montes, and Luis Enrique Sucar. Particle swarm model selection. Journal of Machine Learning Research, 10(2), 2009.

[16] Ian Fette, Norman Sadeh, and Anthony Tomasic. Learning to Detect Phishing Emails. In Proceedings of the 16th International Conference on World Wide Web, 2007.

[17] Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Springenberg, Manuel Blum, and Frank Hutter. Efficient and Robust Automated Machine Learning. In Advances in Neural Information Processing Systems 28, pages 2962–2970. Curran Associates, Inc., 2015.

[18] Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Tobias Springenberg, Manuel Blum, and Frank Hutter. Efficient and robust automated machine learning. NIPS’15, page 2755–2763, Cambridge, MA, USA, 2015. MIT Press.

[19] João Gama. Functional trees. Machine Learning, 55(3):219–250, 2004.

[20] P. Gijsbers, E. LeDell, S. Poirier, J. Thomas, B. Bischl, and J. Vanschoren. An Open Source AutoML Benchmark. arXiv preprint arXiv:1907.00909 [cs.LG], 2019. Accepted at AutoML Workshop at ICML 2019.

[21] Pieter Gijsbers and Joaquin Vanschoren. GAMA: Genetic Automated Machine learning Assistant. Journal of Open Source Software, 4(33):1132, 2019.

[22] Christopher N Gutierrez, Taegyu Kim, Raffaele Della Corte, Jeffrey Avery, Dan Goldwasser, Marcello Cinque, and Saurabh Bagchi. Learning from The Ones That Got Away: Detecting New Forms of Phishing Attacks. IEEE Transactions on Dependable and Secure Computing, 15(6), 2018.

[23] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The weka data mining software: An update. SIGKDD Explor. Newsl., 11(1):10–18, November 2009.

[24] David J Hand and Robert J Till. A simple generalisation of the area under the roc curve for multiple class classification problems. Machine learning, 45(2):171–186, 2001.
[25] Xin He, Kaiyong Zhao, and Xiaowen Chu. AutoML: A Survey of The State-of-the-art. Knowledge-Based Systems, 212:106622, January 2021.

[26] Tin Kam Ho and Mitra Basu. Complexity Measures of Supervised Classification Problems. IEEE Transactions on Pattern Analysis and Machine Intelligence, 24(3):289–300, 2002.

[27] Tin Kam Ho, Mitra Basu, and Martin Hiu Chung Law. Measures of Geometrical Complexity in Classification Problems. In Data Complexity in Pattern Recognition, pages 1–23. Springer, 2006.

[28] Jason Hong. The state of phishing attacks. Commun. ACM, 55(1):74–81, January 2012.

[29] Ozan Irsoy and Ethem Alpaydın. Unsupervised feature extraction with autoencoder trees. Neurocomputing, 258:63–73, 2017.

[30] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 4401–4410, 2019.

[31] Brent Komer, James Bergstra, and Chris Eliasmith. Hyperopt-sklearn: Automatic Hyperparameter Configuration for Scikit-Learn. In Proc. SciPy, 2014.

[32] Lars Kotthoff, Chris Thornton, Holger H. Hoos, Frank Hutter, and Kevin Leyton-Brown. Auto-WEKA 2.0: Automatic model selection and hyperparameter optimization in WEKA. Journal of Machine Learning Research, 18(25):1–5, 2017.

[33] Sanjay Krishnan, Michael J Franklin, Ken Goldberg, and Eugene Wu. Boostclean: Automated error detection and repair for machine learning. arXiv preprint arXiv:1711.01299, 2017.

[34] Sanjay Krishnan and Eugene Wu. Alphaclean: Automatic generation of data cleaning pipelines. arXiv preprint arXiv:1904.11827, 2019.

[35] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11):2278–2324, 1998.

[36] Erin LeDell and Sebastien Poirier. H2O AutoML: Scalable Automatic Machine Learning. 7th ICML Workshop on Automated Machine Learning (AutoML), July 2020.

[37] Chen Lin, Minghao Guo, Chuming Li, Xin Yuan, Wei Wu, Junjie Yan, Dahua Lin, and Wanli Ouyang. Online hyper-parameter learning for auto-augmentation strategy. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 6579–6588, 2019.
[38] Tom Ching Ling Chen, Ava Khonsari, Amirreza Lashkari, Mina Rafi Nazari, Jaspreet Singh Sanabee, and Mario A Nascimento. Uniformaugment: A search-free probabilistic data augmentation approach. arXiv preprint arXiv:2003.14348, 2020.

[39] Ana C Lorena, Luís PF Garcia, Jens Lehmann, Marcilio CP Souto, and Tin Kam Ho. How Complex is Your Classification Problem? A Survey on Measuring Classification Complexity. ACM Computing Surveys (CSUR), 52(5):1–34, 2019.

[40] Justin Ma, Lawrence K Saul, Stefan Savage, and Geoffrey M Voelker. Identifying Suspicious URLs: An Application of Large-Scale Online Learning. In Proceedings of the 26th Annual International Conference on Machine Learning, 2009.

[41] Mohammad Saiful Islam Mamun, Mohammad Ahmad Rathore, Arash Habibi Lashkari, Natalia Stakhanova, and Ali A Ghorbani. Detecting Malicious URLs Using Lexical Analysis. In International Conference on Network and System Security. Springer, 2016.

[42] Samuel Marchal, Jérôme François, Radu State, and Thomas Engel. PhishStorm: Detecting Phishing with Streaming Analytics. IEEE Transactions on Network and Service Management, 11(4), 2014.

[43] Qinxue Meng, Daniel Catchpoole, David Skillicom, and Paul J Kennedy. Relational autoencoder for feature extraction. In 2017 International Joint Conference on Neural Networks (IJCNN), pages 364–371. IEEE, 2017.

[44] Agnieszka Mikołajczyk and Michał Grochowski. Data augmentation for improving deep learning in image classification problem. In 2018 international interdisciplinary PhD workshop (IIPhDW), pages 117–122. IEEE, 2018.

[45] Agnieszka Mikołajczyk and Michał Grochowski. Style transfer-based image synthesis as an efficient regularization technique in deep learning. In 2019 24th International Conference on Methods and Models in Automation and Robotics (MMAR), pages 42–47. IEEE, 2019.

[46] Rami M Mohammad, Fadi Thabitah, and Lee McCluskey. An Assessment of Features Related to Phishing Websites Using an Automated Technique. In 2012 International Conference for Internet Technology and Secured Transactions. IEEE, 2012.

[47] Tae-Hyun Oh, Romnachai Jaroenresri, Changil Kim, Mohamed Elgharib, Frédéric Durand, William T Freeman, and Wojciech Matusik. Learning-based video motion magnification. In Proceedings of the European Conference on Computer Vision (ECCV), pages 633–648, 2018.

[48] Randal S. Olson, Nathan Bartley, Ryan J. Urbanowicz, and Jason H. Moore. Evaluation of a tree-based pipeline optimization tool for automating
data science. In Proceedings of the Genetic and Evolutionary Computation Conference 2016, GECCO ’16, page 485–492, New York, NY, USA, 2016. Association for Computing Machinery.

[49] Randal S. Olson, Nathan Bartley, Ryan J. Urbanowicz, and Jason H. Moore. Evaluation of a Tree-based Pipeline Optimization Tool for Automating Data Science. In Proceedings of the Genetic and Evolutionary Computation Conference 2016, GECCO ’16. ACM, 2016.

[50] A Orriols-Puig, Núria Macià, and Tin Ho. DCoL: Data Complexity Library in C++ (Documentation), 2010.

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

[52] Venkatesh Ramanathan and Harry Wechsler. PhishGILLNET—Phishing Detection Methodology Using Probabilistic Latent Semantic Analysis, AdaBoost, and Co-training. EURASIP Journal on Information Security, 2012(1), 2012.

[53] Nataniel Ruiz, Samuel Schulter, and Manmohan Chandraker. Learning to simulate. arXiv preprint arXiv:1810.02513, 2018.

[54] Parikshit Sondhi. Feature construction methods: a survey. sifaka. cs. uiuc. edu, 69:70–71, 2009.

[55] Anh Truong, Austin Walters, Jeremy Goodsitt, Keegan Hines, C. Bayan Bruss, and Reza Farivar. Towards Automated Machine Learning: Evaluation and Comparison of AutoML Approaches and Tools. 2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI), pages 1471–1479, November 2019. arXiv: 1908.05557.

[56] Haleh Vafaie and Kenneth De Jong. Evolutionary feature space transformation. In Feature Extraction, Construction and Selection, pages 307–323. Springer, 1998.

[57] Rakesh Verma and Nabil Hossain. Semantic Feature Selection for Text with Application to Phishing Email Detection. In International Conference on Information Security and Cryptology. Springer, 2013.

[58] Rakesh M Verma, Victor Zeng, and Houtan Faridi. Data Quality for Security Challenges: Case Studies of Phishing, Malware and Intrusion Detection Datasets. In Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, 2019.
[59] J. Waring, C. Lindvall, and R. Umeton. Automated Machine Learning: Review of the State-of-the-Art and Opportunities for Healthcare. Artificial Intelligence in Medicine, 104, 2020.

[60] Sebastien C Wong, Adam Gatt, Victor Stamatescu, and Mark D McDonnell. Understanding data augmentation for classification: when to warp? In 2016 international conference on digital image computing: techniques and applications (DICTA), pages 1–6. IEEE, 2016.

[61] Zijian Zheng. A comparison of constructing different types of new feature for decision tree learning. In Feature Extraction, Construction and Selection, pages 239–255. Springer, 1998.
### A Model Performances

In this section, we provide more details regarding the AutoML and traditional ML model performances, as shown in Table 4 and Table 5.

| Task | Framework | Accuracy | AUC Score | Duration (sec) |
|------|-----------|----------|-----------|----------------|
| eml_a | AutoGluon | 0.866    | 0.865     | 3492.86        |
|       | auto-sklearn | 0.865  | 0.869     | 3504.94        |
|       | GAMA       | 0.749    | 0.865     | 3239.22        |
|       | H2OAutoML  | 0.865    | 0.871     | 2442.72        |
|       | hyperopt-sklearn | 0.865 | 0.73      | 3619.92        |
|       | TPOT       | 0.86     | 0.834     | 3315.22        |
|       |            |          |           |                |
| eml_b | AutoGluon | 0.908    | 0.913     | 3525.68        |
|       | auto-sklearn | 0.895  | 0.913     | 3601.26        |
|       | GAMA       | 0.908    | 0.908     | 3239.44        |
|       | H2OAutoML  | 0.899    | 0.906     | 2602.88        |
|       | hyperopt-sklearn | 0.887 | 0.821     | 429.83         |
|       | TPOT       | 0.94     | 0.908     | 3633.41        |
|       |            |          |           |                |
| eml_c | AutoGluon | 0.965    | 0.967     | 3241.95        |
|       | auto-sklearn | 0.946  | 0.967     | 3212.53        |
|       | GAMA       | 0.942    | 0.975     | 3289.61        |
|       | H2OAutoML  | 0.942    | 0.891     | 343.08         |
|       | hyperopt-sklearn | 0.938 | 0.862     | 3431.54        |
|       | TPOT       | 0.915    | 0.947     | 3619.78        |
|       |            |          |           |                |
| eml_d | AutoGluon | 0.923    | 0.952     | 3272.78        |
|       | auto-sklearn | 0.916  | 0.948     | 3398.93        |
|       | GAMA       | 0.92     | 0.947     | 3239.68        |
|       | H2OAutoML  | 0.93     | 0.954     | 2442.72        |
|       | hyperopt-sklearn | 0.897 | 0.871     | 31.57          |
|       | TPOT       | 0.915    | 0.947     | 3619.78        |
|       |            |          |           |                |
| eml_e | AutoGluon | 0.968    | 0.981     | 3491.24        |
|       | auto-sklearn | 0.97    | 0.985     | 3509.78        |
|       | GAMA       | 0.942    | 0.975     | 3289.61        |
|       | H2OAutoML  | 0.942    | 0.891     | 343.08         |
|       | hyperopt-sklearn | 0.938 | 0.862     | 3431.54        |
|       | TPOT       | 0.86     | 0.87      | 3418.89        |
|       |            |          |           |                |
| eml_f | AutoGluon | 0.971    | 0.984     | 3461.46        |
|       | auto-sklearn | 0.97    | 0.985     | 3600.33        |
|       | GAMA       | 0.942    | 0.984     | 3239.53        |
|       | H2OAutoML  | 0.972    | 0.984     | 2483.41        |
|       | hyperopt-sklearn | 0.972 | 0.967     | 879.84         |
|       | TPOT       | 0.924    | 0.984     | 3610.52        |
|       |            |          |           |                |
| url_a | AutoGluon | nan      | nan       | 12113.62       |
|       | auto-sklearn | 0.969  | 0.993     | 3596.35        |
|       | GAMA       | 0.97     | 0.993     | 3305.78        |
|       | H2OAutoML  | 0.964    | 0.993     | 3309.22        |
|       | hyperopt-sklearn | 0.931 | 0.916     | 3609.97        |
|       | TPOT       | 0.972    | 0.985     | 3724.93        |
|       |            |          |           |                |
| url_b | AutoGluon | 0.961    | 0.994     | 2912.35        |
|       | auto-sklearn | 0.952  | 0.991     | 3602.72        |
|       | GAMA       | 0.952    | 0.991     | 3257.65        |
|       | H2OAutoML  | 0.953    | 0.992     | 3281.59        |
|       | hyperopt-sklearn | 0.946 | 0.946     | 1812.67        |
|       | TPOT       | 0.952    | 0.991     | 3764.12        |
|       |            |          |           |                |
| url_c | AutoGluon | 0.961    | 0.994     | 3178.55        |
|       | auto-sklearn | 0.952  | 0.986     | 3600.84        |
|       | GAMA       | 0.952    | 0.986     | 3272.36        |
|       | H2OAutoML  | 0.953    | 0.988     | 3375.62        |
|       | hyperopt-sklearn | 0.949 | 0.974     | 1094.88        |
|       | TPOT       | 0.952    | 0.986     | 3767.93        |
|       |            |          |           |                |
| url_d | AutoGluon | 0.942    | 0.987     | 3438.36        |
|       | auto-sklearn | 0.941  | 0.988     | 3599.88        |
|       | GAMA       | 0.945    | 0.989     | 3240.61        |
|       | H2OAutoML  | 0.947    | 0.99      | 3199.18        |
|       | hyperopt-sklearn | 0.937 | 0.935     | 417.4          |
|       | TPOT       | 0.939    | 0.89      | 3766.47        |

Table 4: AutoML-based Model Performance
| Task   | Framework        | Accuracy | AUC Score | Duration (sec) |
|--------|-----------------|----------|-----------|----------------|
| eml_1a | Logistic Regression | 0.749    | 0.749     | 0.72           |
|        | SVM              | 0.749    | 0.749     | 2.179          |
|        | KNN              | 0.749    | 0.749     | 2.179          |
|        | Decision tree    | 0.749    | 0.749     | 4.59           |
|        | Random forest    | 0.749    | 0.749     | 5.64           |
|        | Multilayer perceptron | 0.749 | 0.749     | 18.032         |
|        | Gaussian NB      | 0.749    | 0.736     | 0.001          |
| eml_1b | Logistic Regression | 0.883    | 0.883     | 2.709          |
|        | SVM              | 0.834    | 0.828     | 3.608          |
|        | KNN              | 0.752    | 0.642     | 2.195          |
|        | Decision tree    | 0.896    | 0.86      | 1.777          |
|        | Random forest    | 0.906    | 0.908     | 13.047         |
|        | Multilayer perceptron | 0.749 | 0.304     | 23.342         |
|        | Gaussian NB      | 0.863    | 0.849     | 0.002          |
| eml_1c | Logistic Regression | 0.899    | 0.899     | 3.535          |
|        | SVM              | 0.917    | 0.937     | 1.877          |
|        | KNN              | 0.903    | 0.919     | 1.963          |
|        | Decision tree    | 0.897    | 0.894     | 1.801          |
|        | Random forest    | 0.916    | 0.95      | 19.309         |
|        | Multilayer perceptron | 0.749 | 0.934     | 19.411         |
|        | Gaussian NB      | 0.839    | 0.915     | 0.001          |
| eml_1d | Logistic Regression | 0.925    | 0.925     | 6.144          |
|        | SVM              | 0.917    | 0.937     | 1.877          |
|        | KNN              | 0.903    | 0.919     | 1.963          |
|        | Decision tree    | 0.897    | 0.894     | 1.801          |
|        | Random forest    | 0.916    | 0.95      | 19.309         |
|        | Multilayer perceptron | 0.749 | 0.934     | 19.411         |
|        | Gaussian NB      | 0.839    | 0.915     | 0.001          |
| eml_2  | Logistic Regression | 0.603    | 0.603     | 1.711          |
|        | SVM              | 0.397    | 0.402     | 14.276         |
|        | KNN              | 0.707    | 0.685     | 3.336          |
|        | Decision tree    | 0.741    | 0.752     | 1.647          |
|        | Random forest    | 0.741    | 0.752     | 11.034         |
|        | Multilayer perceptron | 0.603 | 0.845     | 36.178         |
|        | Gaussian NB      | 0.381    | 0.466     | 0.003          |
| url_1  | Logistic Regression | 0.935    | 0.935     | 4.384          |
|        | SVM              | 0.708    | 0.984     | 18731.06       |
|        | KNN              | 0.934    | 0.975     | 50131.41       |
|        | Decision tree    | 0.943    | 0.962     | 3847.875       |
|        | Random forest    | 0.971    | 0.99      | 126085.8       |
|        | Multilayer perceptron | 0.977 | 0.997     | 942.103        |
|        | Gaussian NB      | 0.731    | 0.769     | 0.733          |
| url_2  | Logistic Regression | 0.915    | 0.915     | 25.734         |
|        | SVM              | 0.733    | 0.721     | 298.13         |
|        | KNN              | 0.911    | 0.968     | 79.309         |
|        | Decision tree    | 0.94      | 0.95      | 11.277         |
|        | Random forest    | 0.955    | 0.992     | 279.35         |
|        | Multilayer perceptron | 0.94  | 0.945     | 282.544        |
|        | Gaussian NB      | 0.747    | 0.769     | 0.017          |
| url_3  | Logistic Regression | 0.94     | 0.94     | 154.087        |
|        | SVM              | 0.94     | 0.94     | 154.087        |
|        | KNN              | 0.94     | 0.94     | 154.087        |
|        | Decision tree    | 0.94     | 0.94     | 154.087        |
|        | Random forest    | 0.94     | 0.94     | 154.087        |
|        | Multilayer perceptron | 0.94 | 0.94     | 154.087        |
|        | Gaussian NB      | 0.94     | 0.94     | 154.087        |
| web_1  | Logistic Regression | 0.923    | 0.923    | 3.061          |
|        | SVM              | 0.659    | 0.756    | 15.578         |
|        | KNN              | 0.913    | 0.966    | 51.269         |
|        | Decision tree    | 0.999    | 0.947    | 2.183          |
|        | Random forest    | 0.934    | 0.958    | 20.192         |
|        | Multilayer perceptron | 0.919 | 0.973     | 47.325         |
|        | Gaussian NB      | 0.588    | 0.965    | 0.005          |

Table 5: Manually Developed (non-AutoML) Model Performance
## B Correlation Analysis

In this section, we provide results on the analysis of correlation between the complexity measures and the AutoML performances (Table 6), and correlation between the complexity measures and the classification performance gain when using AutoML frameworks (Table 7).

| Complexity Measure | Performance Metric | Correlation | p-value |
|--------------------|--------------------|-------------|---------|
| F1                 | Accuracy           | 0.151515    | 0.676065|
| F1                 | F1 Score           | 0.187679    | 0.603216|
| F1                 | AUC Score          | -0.21212    | 0.556306|
| Flv                | Accuracy           | -0.24848    | 0.488776|
| Flv                | F1 Score           | -0.32121    | 0.365468|
| Flv                | AUC Score          | -0.55152    | 0.098401|
| F2                 | Accuracy           | 0.066465    | 0.985858|
| F2                 | F1 Score           | -0.12284    | 0.735313|
| F2                 | AUC Score          | 0.071116    | 0.845218|
| F3                 | Accuracy           | 0.515152    | 0.127553|
| F3                 | F1 Score           | 0.56366     | 0.089724|
| F3                 | AUC Score          | 0.260006    | 0.467089|
| F4                 | Accuracy           | 0.648485    | 0.042544|
| F4                 | F1 Score           | 0.587879    | 0.078478|
| F4                 | AUC Score          | 0.284848    | 0.425038|
| L1                 | Accuracy           | 0.418182    | 0.229113|
| L1                 | F1 Score           | 0.393939    | 0.259998|
| L1                 | AUC Score          | 0.139094    | 0.709932|
| L2                 | Accuracy           | -0.32121    | 0.365468|
| L2                 | F1 Score           | -0.17576    | 0.627188|
| L2                 | AUC Score          | -0.12727    | 0.726057|
| L3                 | Accuracy           | -0.69697    | 0.025097|
| L3                 | F1 Score           | -0.68468    | 0.028883|
| L3                 | AUC Score          | -0.7697     | 0.009222|
| N1                 | Accuracy           | -0.11515    | 0.75142 |
| N1                 | F1 Score           | -0.04242    | 0.907364|
| N1                 | AUC Score          | 0.29697     | 0.404702|
| N2                 | Accuracy           | 0.090069    | 0.802772|
| N2                 | F1 Score           | -0.15152    | 0.670065|
| N2                 | AUC Score          | -0.29697    | 0.404702|
| N3                 | Accuracy           | -0.10303    | 0.776998|
| N3                 | F1 Score           | -0.06667    | 0.854813|
| N3                 | AUC Score          | 0.260006    | 0.467089|
| N4                 | Accuracy           | -0.28485    | 0.425038|
| N4                 | F1 Score           | -0.13039    | 0.709932|
| N4                 | AUC Score          | 0.10303     | 0.776998|
| T1                 | Accuracy           | 0.830303    | 0.002944|
| T1                 | F1 Score           | 0.721818    | 0.007547|
| T1                 | AUC Score          | 0.587879    | 0.078578|
| T2                 | Accuracy           | 0.127273    | 0.726057|
| T2                 | F1 Score           | 0.151515    | 0.670065|
| T2                 | AUC Score          | 0.2         | 0.579584|
| C1                 | Accuracy           | -0.5366     | 0.109784|
| C1                 | F1 Score           | -0.69176    | 0.026678|
| C1                 | AUC Score          | -0.834      | 0.002705|
| C2                 | Accuracy           | -0.57539    | 0.081792|
| C2                 | F1 Score           | -0.73555    | 0.016409|
| C2                 | AUC Score          | -0.88572    | 0.000649|

Table 6: Correlation between Complexity Measure and AutoML Performance
| Complexity Measure | Performance Gain | Correlation | p-value |
|--------------------|------------------|-------------|---------|
| F1                 | Accuracy         | -0.21212    | 0.556306|
| F1                 | F1 Score         | -0.38182    | 0.276255|
| F1                 | AUC Score        | -0.70909    | **0.021666** |
| F1v                | Accuracy         | 0.272727    | 0.445838|
| F1v                | F1 Score         | 0.353333    | 0.346594|
| F1v                | AUC Score        | 0.060661    | 0.986743|
| F2                 | Accuracy         | 0.536092    | 0.109784|
| F2                 | F1 Score         | 0.316789    | 0.37248 |
| F2                 | AUC Score        | 0.38144     | 0.276771|
| F3                 | Accuracy         | -0.62424    | 0.053718 |
| F3                 | F1 Score         | -0.41818    | 0.229113 |
| F3                 | AUC Score        | -0.67273    | **0.033041** |
| F4                 | Accuracy         | -0.70909    | **0.021666** |
| F4                 | F1 Score         | -0.44242    | 0.200423 |
| F4                 | AUC Score        | -0.53939    | 0.107593 |
| L1                 | Accuracy         | 0.357576    | 0.318376 |
| L1                 | F1 Score         | -0.45655    | 0.186905 |
| L1                 | AUC Score        | -0.49091    | 0.149656 |
| L2                 | Accuracy         | 0.575758    | 0.081553 |
| L2                 | F1 Score         | 0.115152    | 0.75142  |
| L2                 | AUC Score        | 0.115152    | 0.75142  |
| L3                 | Accuracy         | -0.16364    | 0.651477 |
| L3                 | F1 Score         | 0.078788    | 0.824717 |
| L3                 | AUC Score        | -0.06667    | 0.854813 |
| N1                 | Accuracy         | 0.563636    | 0.089724 |
| N1                 | F1 Score         | 0.539394    | 0.107593 |
| N1                 | AUC Score        | 0.745455    | **0.01333** |
| N2                 | Accuracy         | -0.15152    | 0.670665 |
| N2                 | F1 Score         | -0.47879    | 0.161524 |
| N2                 | AUC Score        | -0.33333    | 0.346594 |
| N3                 | Accuracy         | 0.624242    | 0.053718 |
| N3                 | F1 Score         | 0.490909    | 0.149656 |
| N3                 | AUC Score        | 0.721212    | **0.018573** |
| N4                 | Accuracy         | 0.127273    | 0.729057 |
| N4                 | F1 Score         | 0.612121    | 0.059872 |
| N4                 | AUC Score        | 0.769697    | **0.009222** |
| T1                 | Accuracy         | 0.10303     | 0.776998 |
| T1                 | F1 Score         | -0.57576    | 0.081553 |
| T1                 | AUC Score        | -0.38182    | 0.276255 |
| T2                 | Accuracy         | 0.03003     | 0.933773 |
| T2                 | F1 Score         | -0.4303     | 0.214492 |
| T2                 | AUC Score        | -0.26061    | 0.467089 |
| C1                 | Accuracy         | -0.47195    | 0.168458 |
| C1                 | F1 Score         | -0.0194     | 0.957589 |
| C1                 | AUC Score        | -0.16163    | 0.655533 |
| C2                 | Accuracy         | -0.38144    | 0.276771 |
| C2                 | F1 Score         | -0.05819    | 0.873149 |
| C2                 | AUC Score        | -0.21335    | 0.553971 |

Table 7: Correlation between Complexity Measure and Performance Gain with AutoML