AutoML Applications in Business:

A Case Study Using BrewAI

Authored By:

Fethi Rabhi[1]
School of Computer Science and Engineering,
University of New South Wales, Australia

Alan Ng[2]
Individual Researcher and Consultant, Hong Kong

Nikolay Mehandjiev[3]
Alliance Manchester Business School, The University of Manchester, UK

8 October 2021

Contents

School of Computer Science and Engineering

FACULTY OF ENGINEERING

Professor Fethi A. Rabhi
f.rabhi@unsw.edu.au

Alliance Manchester Business School

Professor Nikolay Mehandjiev
n.mehandjiev@manchester.ac.uk

Abstract

The demand for simple solutions that deliver meaningful results without the need to be operated by machine learning (ML)-experts has given rise to the field of automated machine learning (AutoML). It enables domain scientists to apply ML without the need to understand and learn the technologies that support these techniques in detail. This paper investigates the state-of-the-art in the area of AutoML particularly in the context of business applications. One of the problems identified is that most existing AutoML solutions are tied to a particular platform or need specialised staff to operate them. We then proceed to explore a new AutoML tool called BrewAI, which addresses these issues by offering platform-independent facilities which do not rely on specialised staff. BrewAI is designed to be a simple and cost-effective solution using service-oriented design principles and autonomous software services. We then explore the usability and model explainability offered by BrewAI by applying it to analyse a reference dataset. We conclude that BrewAI offers usable interface for the application of machine learning models which does not require any data pre-processing and modelling skills. BrewAI also offers model explainability facilities by showing necessary details about the data and the model which are understandable by business users. The paper concludes by generalising our findings to the class of AutoML tools, presenting the challenges facing them and outlining areas of future work.

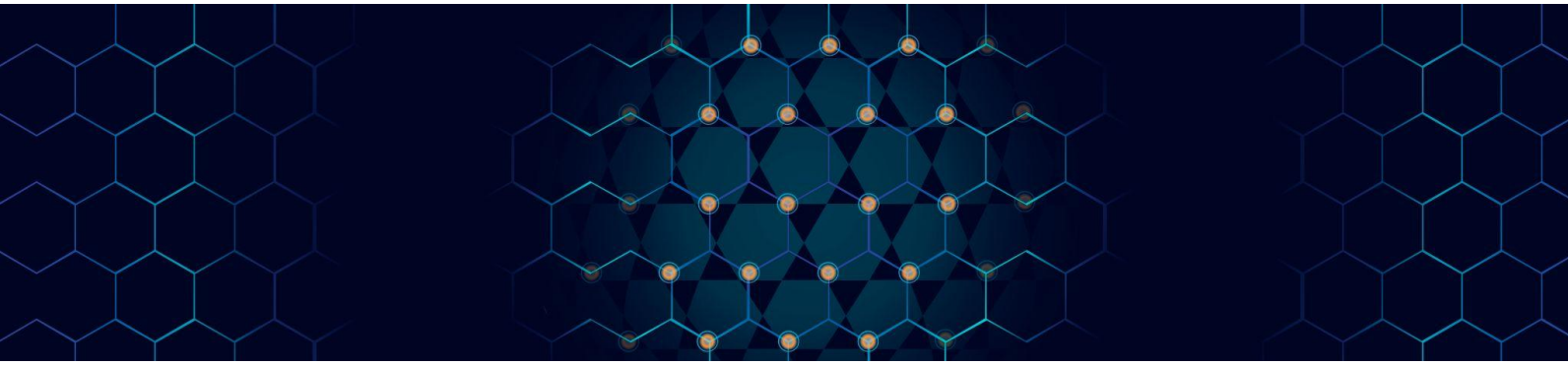
1. Introduction

For many years, the area of machine learning (ML) has been the preserve of ML scientists creating a wide variety of models and algorithms and applying them to new and emerging datasets. In particular, the popularity of deep learning methods has enabled key advances in many application domains, such as computer imaging, speech recognition, and search optimisation. As massive amounts of data are being made available by big technology companies as well as public agencies, many businesses are heavily investing in the development of practical applications of ML techniques for improving their business operations or creating new revenue streams, possibly leveraging their own private enterprise data.

However, the use of ML techniques is still fraught with several technical difficulties. The process of taking an ML model from conceptualisation to deployment to solve a business problem is a complex, time-consuming iterative process comprising a series of steps that involves data collection and integration, data pre-processing, feature selection and transformation, model training, model evaluation, tuning, and deployment. This pipeline process needs to be supported by appropriate computational resources during the model training and inference phases.

Despite the availability of a huge panoply of technologies, many challenges remain such as working out which ML techniques to apply to which problem, how to ensure data quality and how to fine-tune ML parameters properly. Further, ML is a continuous cycle that requires ongoing model monitoring and drifts detection even after the model has been deployed to production to ensure that the model's performance does not decline over time. Addressing these challenges is critical for the successful implementation of end-to-end ML pipelines at scale.

The demand for simple solutions that deliver meaningful results without the need to be operated by ML-experts has given rise to the field of automated machine learning (AutoML). The goal of AutoML is to make ML more systematic and efficient by automating the various phases of the ML pipeline thereby minimising human involvement in the orchestration of the ML pipeline. It enables domain scientists to apply ML without the need to understand and learn the technologies that support these techniques in detail. The purpose of this paper is to investigate the state-of-the-art in the area of AutoML particularly in the context of business applications. It presents our experiences in developing a practical application using the BrewAI AutoML system as a case study. It draws general conclusions about recent developments in this space and outlines areas of future work.



2. Literature Review

Basic concepts

According to [1], AutoML refers to the automation of several activities related to ML such as automating data collection and experiment design; automating data clean up and missing data imputation; automating feature selection and transformation; automating model discovery, criticism, and explanation; automating the allocation of computational resources; automating hyperparameter optimisation (HPO), automating inference and automating model monitoring and anomaly detection.

As described in [2] and illustrated in Fig. 1, this AutoML pipeline can be broadly broken into four phases: data preparation, feature engineering, model generation, and model evaluation.

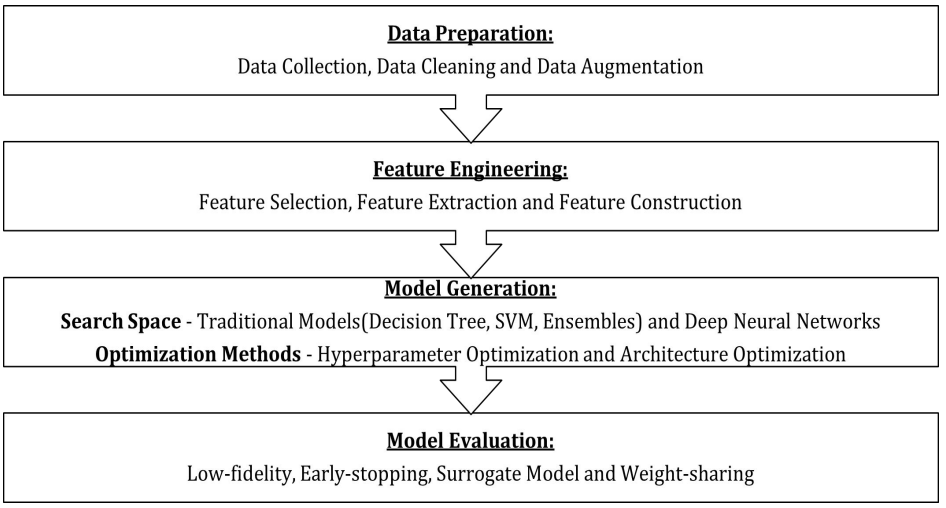


Fig.1. An overview of AutoML pipeline (adapted from He et al. [2])

Data preparation

The first phase of an ML pipeline is data preparation which typically comprises data collection, data augmentation and data cleaning. **Data collection** involves preparing the data set for building a model and here data can be ingested from multiple sources. A commonly occurring problem in data collection is handling imbalanced datasets, for example in detection of fraudulent credit-card transactions, a dataset of credit card transactions may only have a small percentage of fraudulent observations. Techniques such as under-sampling the majority class or over-sampling the minority class are often applied to address this problem. **Data augmentation** techniques such as SMOTE [3, 4] are often preferred over the former techniques to avoid over-fitting of a model which involves creation of synthetic data based on the original data. Ingested data can be in multiple languages, have special characters, represented in different scales, may contain outliers or there could be missing values in data. A good data set is critical to the accuracy of a model, so the process of data cleaning is used to cleanse data. Another important task performed in the data preparation phase help users better understand the data characteristics, identify hidden relationships and employs visual and programmatic methods to collect descriptive statistics, numerical summaries, create plots of distribution (e.g. histograms, boxplots) and conduct bivariate and multivariate analysis (through creating heatmaps, scatter plots etc.).

Feature Engineering

Feature engineering is the process of extracting useful and relevant features from raw data. One important use-case for feature engineering is to resolve the curse of dimensionality where too many features lead to a sparse dataset. Feature engineering can comprise three kinds of tasks – feature extraction, feature selection and feature construction. Feature extraction reduces the dimensionality of dataset by reducing redundant features using prominent algorithms such as principal component analysis (PCA), t-distributed stochastic embedding, feature selection uses a ranking score to rank and select the most important features while feature creation expands on original feature space to create more meaningful features. In automatic feature engineering, hierarchical feature extractors are learned in an end-to-end fashion from data rather than manually designed. Recent works on automatic feature generation such as the one reported in [5] focus on designing different search strategies that prune as many of the candidates to be evaluated as possible, while aiming to keep the most useful interactive features.

Model generation

Fig. 1 shows that model generation is divided into search space which defines the design principles of ML models and optimisation methods. Search space consists of traditional ML models (e.g., SVM and KNN), and neural architectures. In this paper, we will primarily focus on neural architectures. optimisation methods are classified into hyperparameter optimisation (HPO) and architecture optimisation (AO), where the former indicates the training-related parameters (e.g., the learning rate and batch size), and the latter indicates the model-related parameters (e.g., the number of layers for neural architectures). Some researchers refer to AutoML as neural architecture search (NAS) but it is more general so NAS can be considered to be a sub-field of AutoML.

At the heart of an AutoML system is the process of generating a model which consists of several steps, one of which is Hyperparameter optimisation (HPO). In statistics, a hyperparameter captures the prior belief before data is observed so these parameters need to be initialised before training an ML model. In particular, deep neural networks depend on a wide range of hyperparameter choices about the neural network's architecture, regularisation, and optimisation. HPO improves over the default settings provided by common ML libraries and allows general-purpose pipelines to be adapted to datasets from specific application domains. Common HPO techniques include Grid Search, Random Search [6], and Bayesian Optimisation [7] and many existing AutoML tools use variations of these techniques. For example, BOHB used in Auto-Pytorch [8] combines Bayesian optimisation (BO) with Hyperband (HB) [9] and has been shown to outperform BO and HB on many tasks. It also achieves speed ups of up to 55x over Random Search.

Feature Engineering and HPO led to the need for increasingly more complex neural architectures so the process of designing such architectures had to be automated as well leading to Architecture Optimisation (AO) methods. These methods may also be referred to as search strategy [10] or search policy [11]. The commonly used AO methods contain reinforcement learning (RL) [12–16], evolution-based algorithm (EA)[17–23] ,and gradient descent (GD) [24–26], surrogate model-based optimisation (SMBO) [6, 27–32], and hybrid AO methods [33–37].

AutoML tools

There is a huge diversity in the tools available to support AutoML, however, efforts in this field are somewhat fragmented. Current automation tools and techniques target individual phases of the ML pipeline.

For example, the data preparation phase is well supported by a multitude of R packages and Python libraries aimed at automating different tasks in this step. For example, automated EDA tools such as autoEDA, DataExplorer here aim to make data exploration phase fast and easy as possible [38]. A complete survey of the various R packages to support EDA can be found in [38]. Similarly, techniques such as BoostClean, AlphaClean [39, 40] have been applied to automate the process of data cleaning. Automated data augmentation techniques for textual, audio and image data have received much attention in recent years [38]. Automation efforts in the space of feature engineering target a type of feature engineering task e.g. decision-tree based automated feature construction methods

Auto-WEKA [41] is one of the first AutoML systems based on the well-known WEKA machine learning toolkit. TPOT [42] automatically constructs and optimises tree-based machine learning pipelines from a small set of fixed ML components that are connected in predefined ways. Auto-sklearn [43] is similar but adds several improvements such as meta-learning for warm starting the optimisation and automatic ensembling. Inspired by Auto-sklearn, Auto-PyTorch [8] uses an ensembling method to implement an automated post-hoc ensemble model selection [44] for efficient optimisation. Microsoft offers NNI [45] as an open-source package to be used within a Python environment. Besides Python, several AutoML tools based on R such as mlrMBO [46], parsnip [47] are surveyed in [38].

As mentioned earlier, several companies are now developing their own AutoML systems that aim to assist organisations to deploy ML pipelines with minimal effort and costs. Big tech companies are offering AutoML products such as Azure Machine Learning [48] and Amazon SageMaker Autopilot [49] and Google's AutoML [50]. Automated Artificial Intelligence (AutoAI) [51] is an IBM product (part of Watson) which extends the automation of model building towards automation of the full ML life cycle. It puts more focus on preparing data for training, choosing the features, and the best performing pipelines can be put into production to process new data, and deliver predictions based on the model training. AutoAI-TS [52] is a framework for time-series operating as a new service in AutoAI.

One of the problems with these solutions is that they are part of a much bigger system, often tied to a particular platform or cloud infrastructure or need to be installed on a user's desktop. Some solutions are more focused on AutoML and offered in a platform-independent manner. They include H2O Driverless AI [53, 54] which supports fully- or semi-automated feature engineering and selection, model tuning and training of predictive models and DataRobot (which has recently acquired Algorithmia). Still, these solutions may be tied to specific infrastructures that bring high costs to Small and Medium Enterprises (SMEs) and government organisations and may not be easy to deploy and operate by non-expert staff.

Performance of AutoML tools

There are many techniques that can be used to boost AutoML tools. In practical applications, it is often necessary to trade off two or more objectives, such as the performance of a model and IT resource management. This is discussed in detail in [55].

One of the key ideas used to improve the performance of AutoML tools is meta-learning. As different configurations are being explored (HPO, pipeline components and/or network architecture components), meta-learning is the process was based on model evaluation, better configurations are selected through a process of learning. In other words, meta-learning helps build AutoML systems that continuously improve over time [56]. Proposed meta-learning methods include MAML [57], Reptile [58], SNAIL [59], and Relational Meta-Learning[60] . Meta-learning is used in many AutoML tools such as Oracle AutoML [61].

There are many other areas that offer potential to improve performance in AutoML tools, some of them involve user input. For example, VolcanoML [62] introduces and implements basic building blocks that decompose a large search space into smaller ones and allows users to utilise these building blocks to compose an execution plan for the AutoML problem at hand.

Other performance-enhancing techniques work behind the scenes. A number of offline data pre-processing technologies exist such as Avro [63], Parquet [64], or TFRecord [65] which facilitate extracting features from raw data, validating data [66], and converting data to binary formats to enable higher throughput data ingestion. Some batch computing frameworks such as Apache Spark [67], Beam [68], and Flume [69] are also commonly used for offline pre-processing. There have been attempts to build simple data loading systems that could be shared between multiple machine learning jobs. For example, the tf.data API provides generic operators that can be parameterised by user-defined functions, composed, and reused across multiple ML domains [70].

The use of parallel computing techniques is also important in improving the performance of AutoML tools without user intervention. For example, the data pre-processing stage could leverage parallelism and pipelining to overlap pre-processing with model training computations. Determining the optimal degree of parallelism and amount of data to prefetch is often challenging as it depends on the nature of the workload and the hardware resources available.

There is also a growing interest in well-designed AutoML benchmarks to take reproducibility and comparability of AutoML approaches into account [71]. For example, HPOlib [72] provides benchmarks for hyperparameter optimisation, ASlib [73] for meta-learning of algorithm selection and NASBench-101 [74], NASBench-1 Shot 1 [75], and NASBench-201 [76] for neural architecture search. LCBench1 [8], a new benchmark for studying multi-fidelity optimisation w.r.t. learning curves on a joint optimisation space of architectural and training hyperparameters across 35 datasets.

Business applications of AutoML

There is no doubt that AutoML tools are becoming increasingly popular and arousing more interest in the business community[1]. There are many published examples of using AutoML in business applications e.g. forecasting bank failures by policymakers and central banks [77]. A more recent paper discusses other examples of applications of AutoML in industry and discusses future research trends [78].

Whilst the goal of scientific research is to create AutoML tools that aim for full automation, commercial interests in AutoML aim to offer some form of "semi-automation" in assisting organisations deploy ML pipelines at lower costs. For this reason, most business applications tend to deal with supervised learning problems (classification and regression), feature vector representations and homogeneous datasets (same distribution in the training, validation, and test set) [79].

Some of the other challenges reported when using AutoML in the business sector are:

- **How to relate ML to business objectives:** non-technical users requirements (e.g., business KPIs and policy compliance) are often not aligned with what technical users want (e.g., model accuracy and training time) [80]
- **Usability:** non-technical users need to be able to use the system without ML expertise
- **Need for transparency:** non-technical users do not necessarily understand the black-box nature of ML [81].
- **Incomplete pipelines:** many AutoML pipeline libraries have been proposed, but most of them only focus on some parts of the AutoML pipeline ([2], Fig. 1). e.g TPOT [42], Auto-WEAK [41], and Auto-Sklearn [43] are built on top of scikit-learn [82]
- **Data quality:** most progress has been done on model building but the bottleneck is now on the data side as data quality is key to producing good models for industry.
- **Testing of models** is also another problem, new techniques from software engineering are needed
- **Performance:** to achieve good performance, businesses need more sophisticated solutions which need to be weighed against cost considerations (hardware and resources available).

Most studies point out that the most difficult and hard to automate part is understanding the problem domain and exploration of existing data sets. Usually, much more time is spent on data preparation and exploration than on model tuning. In this paper, we investigate new opportunities for addressing these issues via a new AutoML tool namely BrewAI.

---

[1] https://www.arnnet.com.au/article/691087/how-low-code-platforms-enable-machine-learning/

3. BrewAI Case Study

The review of existing systems has demonstrated some limitations in terms of platform and expected skills. New generation of tools are now appearing which address these limitations. Here we focus on one of these tools called BrewAI [83] and explore its features and the ways in which it alleviates the issues with the existing tools. BrewAI is designed to be a simple and cost-effective solution that delivers ML functionalities for organisations that don't have a specialised staff or alternatively used by specialised staff with the intention of reducing time to market with AI models. Like other AutoML systems, BrewAI simplifies the creation and deployment of ML models. Starting from just a simple spreadsheet, a user can train, build and deploy a commercial-grade ML model within an IT infrastructure with minimal efforts. This section first presents a walkthrough of its user interface. It is followed by a technical overview of its architecture and its application in a dataset example.

BrewAI User Interface

An important component is the BrewAI User Interface which displays results at different stages of the ML pipeline in a way that is easily comprehended by the user. The user can also direct the different stages like training via simple button clicks. There are five stages to compute the prediction results from the AutoML model, illustrated in Fig. 2. There is no restriction in the order to follow when performing these five stages, users can jump into any stage to check the previous actions in that specific stage.

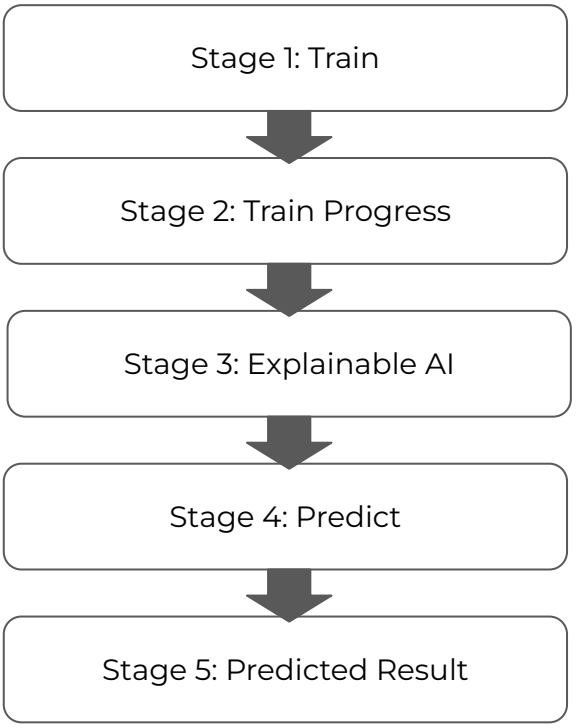


Fig. 2. BrewAI's five stages for AutoML

Stage 1 - Train. In this stage, users can upload the tabular data file to the BrewAI webpage through the interface. Fig. 3 and Fig. 4 show BrewAI's interfaces for training dataset upload and model training submission. After clicking the "Load Data" button (see Fig. 3), a dataset preview will be shown (see Fig. 4), users will then select the target column for prediction. There is a checkbox for users to enable the hyperparameter tuning feature in BrewAI if they want a more accurate AutoML model, otherwise, disabling hyperparameter tuning will get a less accurate but more time-efficient model. The model will be built after clicking the "Submit Model" button (see Fig. 4). BrewAI will automatically define the type of machine learning tasks (regression or classification), handle the data pre-processing, and build the AutoML model.

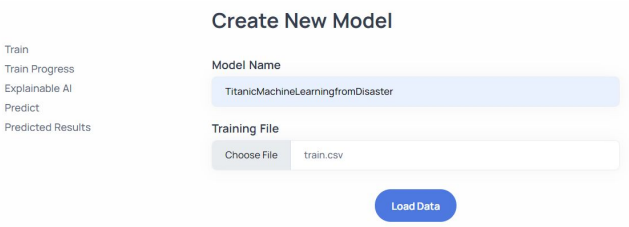


Fig. 3. BrewAI's Interface for training dataset upload.

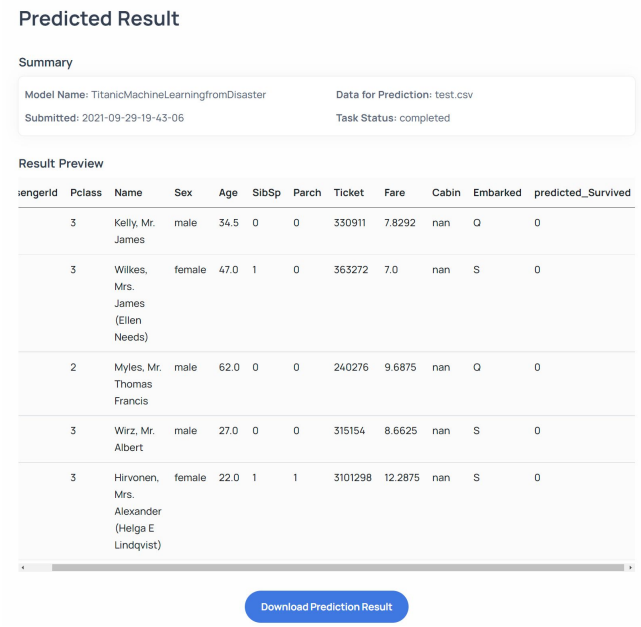


Fig. 4. BrewAI's Interface for loaded dataset review and model training submission

Stage 2 – Train Progress. This stage is to provide an interface for users to see the status of data processing and model building. Users could see a parallel model-building workflow If they submitted multiple models in stage 1(see Fig. 5). No action is required from the users in this stage.

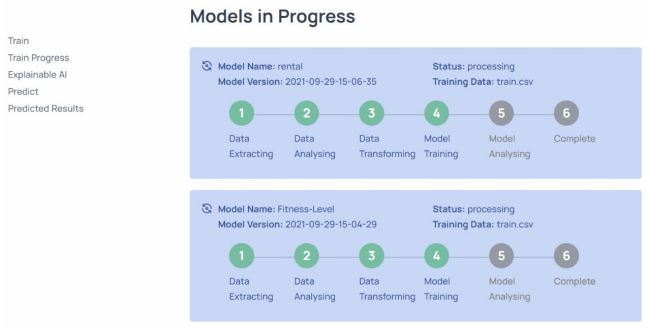


Fig. 5. BrewAI's Interface for the status of data processing and model building

Stage 3 – Explainable AI. This stage is to show explainable details of the data and model after model training is completed. The explainable data shows the details of data quality and data type for each input feature. The explainable model shows the details of the class distribution, model performance, confusion matrix, performance by class, feature importance, and hyperparameter optimisation. Fig. 6 and Fig. 7 show BrewAI's interfaces for data and model explanation.

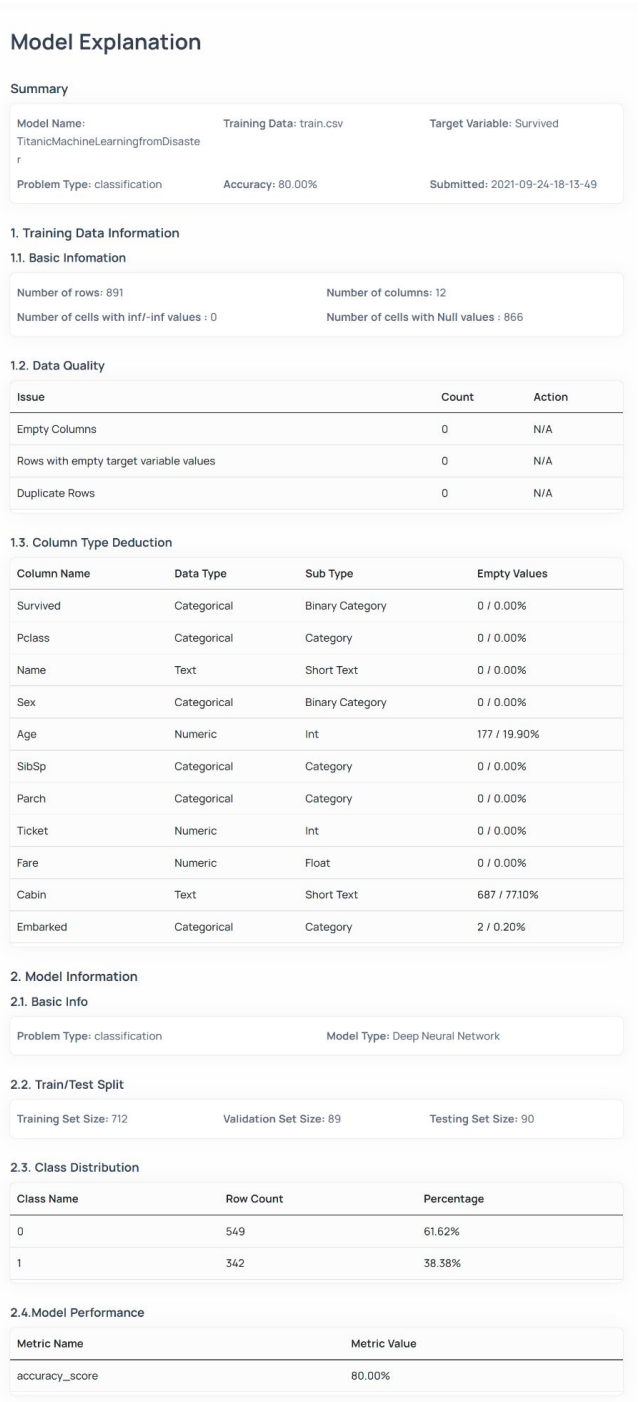
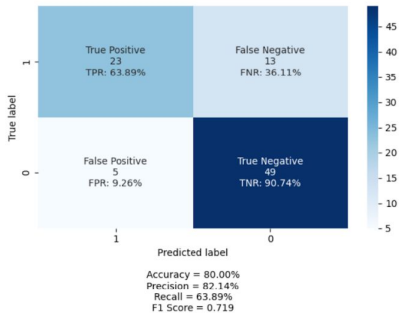


Fig. 6. BrewAI's Interface (part 1) for data, model, and hyperparameter explanation.

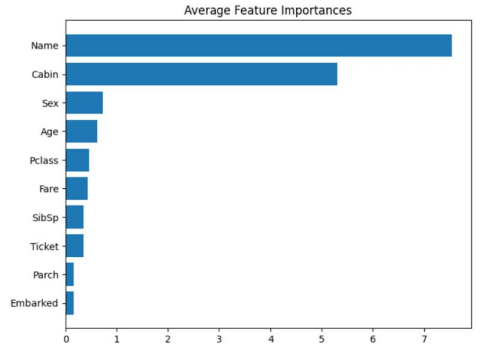
2.5. Confusion Matrix



2.6. Performance by Class

Class Name	Precision	Recall	F1 Score
1	82.14%	63.89%	71.87%
0	79.03%	90.74%	84.48%

2.7. Feature Importance



2.8. Hyperparameter Tuning

2.8.1. Hyperparameter Search Space

Learning Rate Range	[1e-05, 0.01]
Hidden Layer Number Range	[1, 3]
Hidden Layer Neuron Number Range	[3, 960]
Dropout Rate Range	[0.2, 0.5]

2.8.2. Hyperparameter Search Trials

Total Trials	Pruned Trials	Completed Trials
100	84	16

2.8.3. Best Hyperparameters Selected

Best Learning Rate	9.098213461305304e-05
Best Number of Hidden Layers	2
Best Neuron Numbers of Each Hidden Layer	[495, 360]
Best Dropout Rates of Each Hidden Layer	[0.452353266331664, 0.3991544514828167]

Fig. 7. BrewAI's Interface (part 2) for data, model, and hyperparameter explanation.

Stage 4 – Predict. In this stage, users can select a specific model trained in stage 2 to predict the test dataset. Users are allowed to select any previously trained model to do the prediction.

Stage 5 – Predicted Results. After finishing the prediction in stage 4, users can explore the prediction results in this stage (see Fig. 8). BrewAI also allows users to preview and download previously predicted results to csv files by clicking buttons (see Fig. 9).

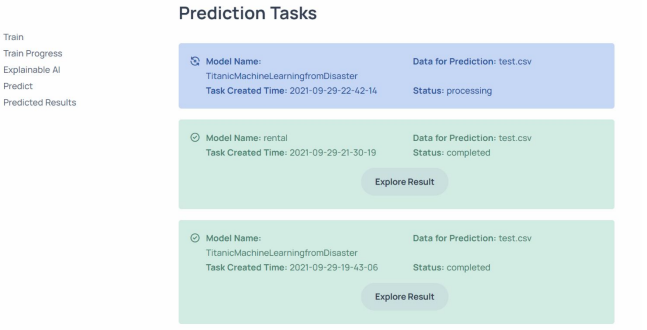


Fig. 8. Interface for result exploration.

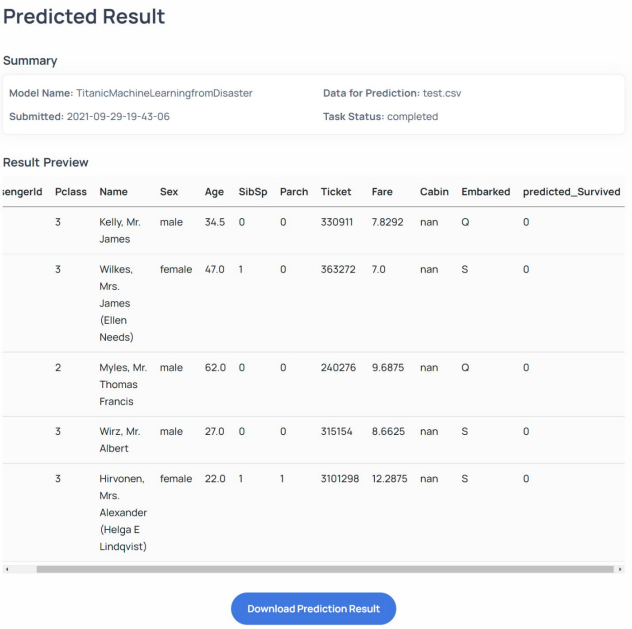
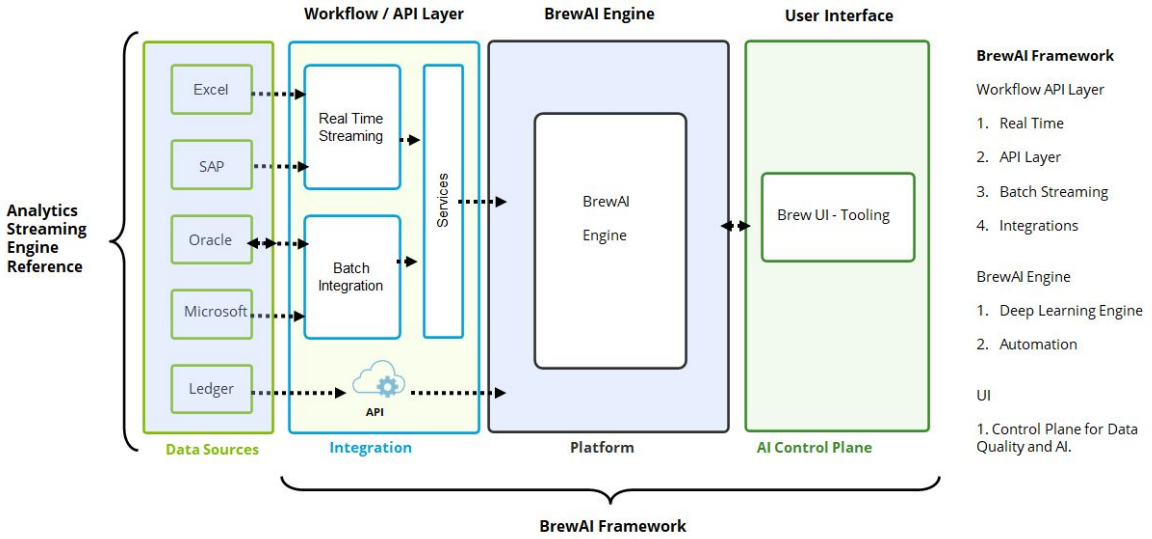


Fig. 9. Right: Interface for result preview and download.

Technical overview

BrewAI's software architecture is based on service-oriented design principles in which autonomous software services can operate and communicate independently from each other. This architecture is illustrated in Fig 10.



The BrewAI engine which is at the heart of the system is responsible for tackling supervised learning problems using deep learning methods. It can deal with different data types including numerical, text, categorical, and binary. New data types such as images are planned to be released in the current roadmap [84]. The engine is built over several other systems. Its code base relies on the PyTorch [85] library. Hyperparameter optimisation is automatically conducted using Optuna [86] and HyperOpt [87].

To determine which features are important [88], BrewAI uses different attribution techniques including Integrated Gradients for feature attribution and Conductance for layer and neuron attribution in order to better understand the neural network predicting survival. These basic building blocks for attribution can be utilised to improve model interpretability, breaking the traditional "black-box" characterisation of neural networks and delving deeper into understanding how and why they make their decisions.

As shown in the architecture diagram, BrewAI has the ability to aggregate data from different sources using a Workflow/API layer, each data source can be independently accessed to encode and feed data into the model. This is supported by an Apache Airflow Engine [89] which allows the definition, scheduling, and monitoring of a wide range of data processing pipelines. Airflow also provides many plug-and-play operators that are ready to execute tasks on Google Cloud Platform, Amazon Web Services, Microsoft Azure, and many other third-party services.

BrewAI gives DevOps engineers and data scientists the ability to observe and control multiple machine learning tasks at the same time for maximum efficiency. This is achieved via the editable data pipeline features provided by Apache Airflow. By accessing Airflow's WebUI or Python APIs, the DevOps and the performance management team can edit and review BrewAI's AutoML pipelines e.g. creating pipelines involving multiple heterogeneous data sources and combine them into one dataset, and then submit them for training and making predictions.

Data scientists can also have access to BrewAI's pipelines to further enhance the data processing, such as adding data validation and cleansing activities to pipelines. Fig. 11 shows the WebUI of Apache Airflow for editing BrewAI's AutoML pipeline.

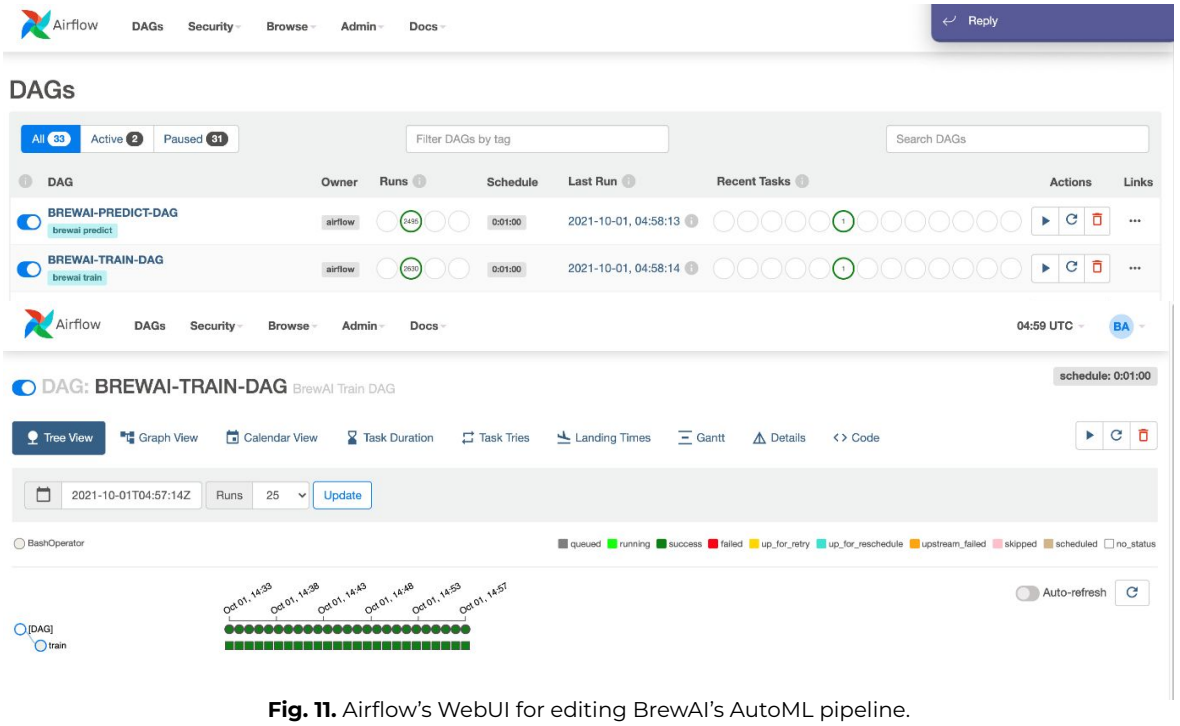


Fig. 11. Airflow's WebUI for editing BrewAI's AutoML pipeline.

All BrewAI software components are virtualised in containers using Kubernetes [90]. This allows them to be deployed on a scalable cloud platform (e.g. Amazon's EC2). The use of an elastic cloud means the system can adapt to different data sizes and loads.

Experiences Using BrewAI on Sample Datasets

Description of datasets used and experimental system

In this case study, we first perform an analysis task using a publicly available dataset to evaluate the following four aspects:

1. The model-building experience and the user interface's usability by non-technical experts
2. The model's explainability
3. The model's performance
4. The model's transparency and understandability

The analysis task is a binary classification with the Titanic dataset [91] which is a tabular dataset consisting of 11 columns of features and 981 samples in the training dataset, and 1309 samples in the testing dataset. The feature contains integers, string, float, and mixtures of string, symbols, and numbers. The machine learning task is to predict if a passenger survived based on the given information of the person.

Accessibility and Usability of BrewAI

Accessing BrewAI requires an internet connection and a browser to log in to their server. All the interfaces would render on the web page and the computation would run on the backend server, installation on the local computer is not required. We went through five stages in the BrewAI process mentioned in Fig. 2. Throughout the whole implementation process, we mainly use three controls from the interfaces to implement the AutoML task: 1. dropdown selection for selecting prediction target and models, 2. Confirm buttons for confirming actions and 3. Textbox for naming models. The model building and data pre-processing are fully automatic, except for the actions that were required for uploading dataset and selecting prediction target.

Model Explainability of BrewAI

After the data pre-processing and model building finish, a user can access an explainable AI page (see Fig. 6 & Fig. 7 – stage 3) to see the explainable features. Table 1 shows a summary of what explainable features are available in BrewAI divided according to data and model feature groups and types.

Table 1. BrewAI's explainable features for data and model

Explainable feature group	Explainable feature type	Explainable feature	How BrewAI explains it in the case study
Data Information	Basic Information	Number of Rows	Show the value of the count
	Basic Information	Number of cells with inf/-inf values	Show the value of the count
	Basic Information	Number of columns	Show the value of the count
	Basic Information	Number of cells with Null values	Show the value of the count
	Data Quality	Empty Columns	Show the value of count and what action was taken
	Data Quality	Rows with empty target variable values	Show the value of count and what action was taken
	Data Quality	Duplicate Rows	Show the value of count and what action was taken
	Feature(column)	Feature (column) Name	Show each feature name
	Feature(column)	Feature (column) Data Type	Show how BrewAI classifies the feature type: Categorical, Numeric, text, etc.
	Feature(column)	Feature (column) Data Sub Type	Show how BrewAI classifies the feature subtype: binary, short/long text, integer, float, etc.
	Feature(column)	Feature (column) Empty Values	Show count and percentage of empty value for each feature
Model Information	Basic Info	Problem Type	Show the type, e.g., classification or regression
	Basic Info	Model Type	Show the type, e.g., deep neural network
	Basic Info	Train/Test Split	Show sizes of training, validation, and testing data
	Class Distribution	Class Distribution	Show class name, sample count, and percentage
	Model Performance	Metric type	Show the type of performance, e.g., accuracy, R2
	Model Performance	Metric value	Show performance values, e.g., accuracy, f1, R2 scores
	Model Performance	Performance detail	Show confusion matrix in a chart
	Model Performance	Performance by Class	Show each target class and the relevant performance values
	Feature Importance	Feature Importance	Show all ranked features' importance in a bar chart
	Hyperparameter Tuning	Hyperparameter Search Space	Show hyperparameter items and the search range, e.g., learning rate, hidden layer number, dropout rate, etc.
	Hyperparameter Tuning	Hyperparameter Search Trials	Show how many trials (including pruned and completed trials) have been done for hyperparameter search
	Hyperparameter Tuning	Best Hyperparameters Selected	Show what hyperparameters have been selected

Model performance of BrewAI on Kaggle

BrewAI automatically splits the training, validation, and test datasets (derived from the uploaded training dataset) and provides a performance based on the test dataset. Once a model training had completed, we can see the accuracy and confusion matrix on BrewAI's webpage (see Fig. 6 & Fig. 7). In this case study, the accuracy score is 80% based on BrewAI's evaluation.

We uploaded the test dataset (the original test dataset, not the one derived from the training dataset) to BrewAI and downloaded the predicted value as a csv file (see Fig. 9 - stage 5). The predicted result was then submitted to Kaggle leaderboard of "Titanic - Machine Learning from Disaster" competition [91] for performance evaluation. The accuracy score on the Kaggle leaderboard for the test dataset was 0.76315 which is similar to the evaluation from BrewAI. Based on the Kaggle leaderboard data of this competition (extracted on 7th Oct 2021), the median value of the accuracy score among Kaggle competitors is 0.77511, around 88% of Kaggle competitor's accuracy scores fell between 0.75 to 0.8. Therefore, BrewAI's AutoML prediction ability is at the average level among 20779 Kaggle competitors in this case study. Participating in Kaggle competition usually requires extensive data science knowledge for data processing and model-building, the result of this case study shows that BrewAI is able to provide similar predictive power as an average Kaggle participant in an automated manner with less effort.

Model transparency and understandability of BrewAI

Drozdal et al.'s study [92] identify what information needs on the AutoML interfaces for data scientists to establish trust in AutoML systems. We evaluated the BrewAI's model transparency and understandability based on a table in Drozdal et al's study. The model transparency items of AutoML in the table were identified and ranked by 21 participants with prior experience with machine learning. We evaluate each item to understand the model transparency of BrewAI. Table 2 shows how many model-transparency items BrewAI can provide from Drozdal et al.'s study.

Table 2: BrewAI's model transparency item checklist (from Drozdal et al.'s study)

Importance Rank	Type	Aspect	Description	Available In BrewAI
4	Data	Raw data	View the meanings of each column in the raw data	Yes
5	Data	Raw data	Visualise each column's distribution in the raw data	Planned
6	Data	Raw data	Visualise the raw data - view overall distributions	Planned
8	Data	Raw data	View the raw data - statistics of individual distributions	Planned
9	Data	Raw data	Visualise outliers in the raw data	Planned
10	Data	Raw data	View statistics of missing values in the raw data	Yes
11	Data	Pre-processed data	View statistics of the pre-processed data	Planned
13	Data	Pre-processed data	Visualise data after pre-processing	Planned
15	Data	Raw data	View statistics of outliers in raw data	Planned
16	Data	Feature engineering	View how existing features were engineered into new features	Planned
19	Data	Raw data	View the raw data table	Yes

19	Data	Raw data	View the raw data table	Yes
21	Data	Pre-processed data	View the pre-processed data table	Planned
24	Data	Raw data	See how data was split (test vs. train/holdout)	Yes
1	Model	Model evaluation	View evaluation metrics	Yes
2	Model	Model evaluation	View visualisations of model performance	Yes
12	Model	Feature engineering	Effect of engineered features	Yes
14	Model	Pipeline	Show adopted models in output pipelines	Yes
17	Model	Pipeline	Ability to edit a pipeline	Yes
18	Model	Model evaluation	Compare differences between pipelines	Yes
22	Model	Model evaluation	Compare one model against other models	Yes
23	Model	Feature engineering	View new engineered features	Planned
26	Model	Hyperparameters	See model's hyperparameters	Yes
3	Process	Pre-processed data	Know how raw data was pre-processed	Planned
7	Process	Pipeline	View process of how a pipeline is created	Yes
20	Process	Pipeline	Show which types of models considered for model selection	Planned
25	Process	Feature engineering	Know how features were engineered	Planned
27	Process	Hyperparameters	Know how hyperparameter optimisation was performed	Yes

We found out that there are 14 out of 27 items of model transparency feature that BrewAI is available to show. The outstanding features are planned in BrewAI's roadmap [84]. In practice, most business users do not understand the technical aspects related to data handling and model-building in the ML pipeline. This is why BrewAI focuses on a simple and clear interface that provides cosine information about the input data and model performance, which business users concern are most interested in.

Overall Evaluation

For the usability aspect, BrewAI does not require any data pre-processing and modeling skills to apply machine learning models. The interface consists of only simple controls which are easy enough for business users to use. The model building processing is fully automatic without worrying about parameter and pipeline settings. The only requirement for using BrewAI is that users need to understand the target they want the AutoML model to learn. Although BrewAI only works with tabular/structural data, users can still transform any other type of data into a tabular form for classification and regression tasks.

For the model explainability and understandability aspect, BrewAI can show necessary details about the data and model in a way that business users can understand. Users can have a summary of their datasets without any programming skills or manual data analysis. There is limited explainability and control about the data pipeline and model generation process, but the assumption is that most business users only focus on the data and results such as data quality, performance, and feature importance that BrewAI can provide.

For the model performance aspect, the case study shows that non-expert users with the BrewAI model still achieve an average result in a Kaggle competition without data pre-processing and model building techniques.

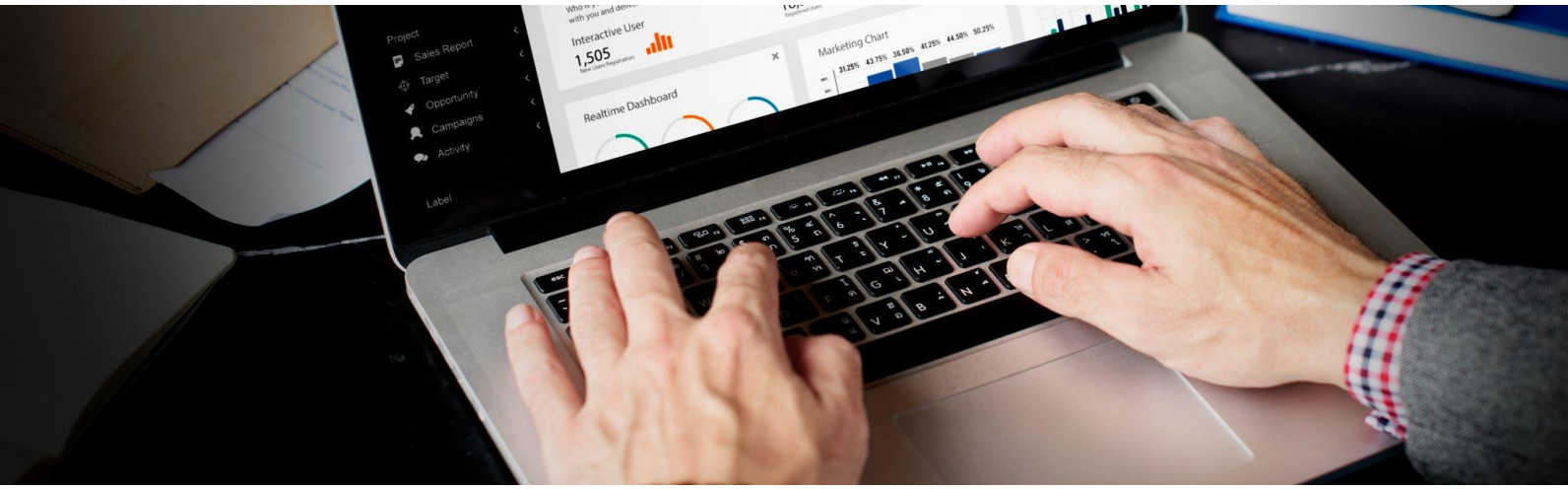
4. Conclusions and Future Work

Summary

This paper has reviewed the landscape of automating the application of machine learning methods and in particular existing work that is concerned with the development of a new type of tool called AutoML tools. As there is a huge variety in the number of proposed solutions, this paper has focused on those specifically targeted at business applications and which do not have a high entry barrier. A case study is performed using an existing solution (BrewAI) to determine its AutoML capabilities and positioning within the current offerings. Using some practical datasets, the evaluation shows that the tool has the ability to analyse data sets in an intuitive manner while it offers a flexible and scalable architecture without a loss in performance. Table 3 summarises the comparison with other tools.

Table 3: Comparing BrewAI with other AutoML tools

Tool Name	License/Dep Costs	Models Used	Expertise needed	Deployment	Completeness of pipeline	Integration
DataRobot	Annual License and Per Model Deployment	Wide range	ML Scientist	Premises/ Cloud	Complete	APIs
H2O	Annual License and Per Model Deployment	Wide range	ML Science	Premises /Cloud	Complete	APIs
Google AutoML	Per timed usage	Regression and Classification	ML Scientist and DevOps	Google Cloud	Partial	GCP services
AWS Sage	Per timed usage	Regression and Classification	ML Scientist and DevOps	AWS	Partial	AWS services
Azure	Per timed usage	Regression and Classification	ML Scientist and DevOps	Azure	Partial	Azure services
BrewAI	Annual License and Per Model Deployment	Deep Learning	Business Analyst, ML Scientist and DevOps	Premises/ Cloud	Complete	Workflow engine (Plugins for various systems: SQL, Apache Spark, cloud storage, etc.)



Future Research areas

There is no doubt that AutoML research work is likely to intensify especially when there are still many unresolved issues amongst them those listed in a recent survey [2]:

- Flexible search space. Although these search spaces have been proven effective for generating well-performing neural architectures, all of them are based on human knowledge and experience, which inevitably introduce human bias.
- Exploring more application areas: as AutoML techniques have had success in new areas such as network compression, federate learning, image caption, recommendation system, and searching for loss and activation functions, they have the potential to be applied in a wider range of areas.
- Interpretability: providing users with meaningful results is still and challenge and increasing the mathematical interpretability of AutoML is an important future research direction.
- Reproducibility: providing ML without incurring considerable resource consumption is also an important area of research.
- Robustness: most training datasets are well-labelled. However, in real-world situations, the data inevitably contain noise (e.g., mislabelling and inadequate information). Even worse, the data might be modified to be adversarial with carefully designed noises. Deep learning models can be easily fooled by adversarial data,
- Joint HPO and AO: there is a tremendous overlap between the methods used in HPO and AO. Future work can look at jointly optimising both hyperparameters and architectures.
- Complete AutoML pipeline: achieving a complete AutoML pipeline is still problematic.
- Lifelong learning: the system should be able to reuse prior knowledge to solve new tasks. We already mentioned meta-learning but unsupervised learning is still an active research area. Some work also looks at how to train a model using only new data while preserving its original capabilities

Regarding the last point, AutoML tools work on the assumption that we have labelled data, but in some cases, only a portion of the data may have labels or even none at all. Liu et al. [93] proposed a general problem setup, namely unsupervised neural-architecture search (UnNAS), to explore whether labels are necessary for architecture search. They experimentally demonstrated that the architectures searched without labels are competitive compared with those searched with labels.

On the business side of AutoML, the main issues are achieving the right balance between several often-conflicting forces [93]. One of them is how to express the problem not in terms of an ML task but as a set of business objectives with associated measures such as competitiveness, successfulness, and financial benefits [94]. Another is how to achieve transparency (explanations), usability (UI Design, UI aids) and performance (information quality) at the same time. Finally, establishing trust in AutoML is an important issue trust [92]. These issues are all interlinked e.g. adding business objectives may reduce the usability and decrease performance, adding more transparency may obscure and decrease trust, adding more usability may decrease performance etc. In some cases, compliance with regulations such as those associated with automated financial trading [95] is another important consideration.

In particular, [92] stresses the importance to provide the ability to "personalise" AutoML in different contexts. Differences in background knowledge, skills, work practices, and experience levels make it difficult to claim that AutoML tools ought to be designed as "one size fits all" [96] for every organisation. Some recent research by Arya et al. [97] allow for a degree of personalisation to accommodate individual preferences or different domains of use by defining explanation methods for different audiences and domains. We anticipate that, some form of AutoML with "human in the loop" is likely to be the prevalent approach when targeting business applications in the future.

5. Acknowledgements

We wish to thank BrewAI for sponsoring this research project and in particular Gavin Whyte, Andy Zeng and Mark Fordree for their help and advice. We also wish to acknowledge Aarthi Natarajan's contribution in writing this paper.

6. References

1. Automated Machine Learning : Methods, Systems, Challenges. Springer International Publishing : Imprint: Springer, Cham (2019).

2. He, X., Zhao, K., Chu, X.: AutoML: A survey of the state-of-the-art. Knowledge-Based Systems. 212, (2021). https://doi.org/10.1016/j.knosys.2020.106622.

3. Chawla, N. v, Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P.: SMOTE: Synthetic minority over-sampling technique. The Journal of artificial intelligence research. 16, 321–357 (2002).

4. Fernández, A., García, S., Herrera, F., Chawla, N. v: SMOTE for Learning from Imbalanced Data: Progress and Challenges, Marking the 15-year Anniversary. The Journal of artificial intelligence research. 61, 863–905 (2018).

5. Luo, Y., Wang, M., Zhou, H., Yao, Q., Tu, W.-W., Chen, Y., Dai, W., Yang, Q.: AutoCross: Automatic Feature Crossing for Tabular Data in Real-World Applications. In: Proceedings of the 25th ACM SIGKDD International Conference on knowledge discovery & data mining. pp. 1936–1945. ACM (2019).

6. Falkner, S., Klein, A., Hutter, F.: BOHB: Robust and Efficient Hyperparameter optimisation at Scale. (2018).

7. Mockus, J.: Application of Bayesian approach to numerical methods of global and stochastic optimisation. Journal of global optimisation. 4, 347–365 (1994).

8. Zimmer, L., Lindauer, M., Hutter, F.: Auto-Pytorch: Multi-Fidelity MetaLearning for Efficient and Robust AutoDL. IEEE Transactions on Pattern Analysis and Machine Intelligence. 43, 3079–3090 (2021). https://doi.org/10.1109/TPAMI.2021.3067763.

9. Li, L., Jamieson, K., DeSalvo, G., Rostamizadeh, A., Talwalkar, A.: Hyperband: A novel bandit-based approach to hyperparameter optimisation. Journal of machine learning research. 18, 1–52 (2018).

10. Elsken, T., Metzen, J.H., Hutter, F.: Neural Architecture Search: A Survey. (2018).

11. Yu, K., Sciuto, C., Jaggi, M., Musat, C., Salzmann, M.: Evaluating the Search Phase of Neural Architecture Search. (2019).

12. Zhong, Z., Yan, J., Wu, W., Shao, J., Liu, C.-L.: Practical Block-Wise Neural Network Architecture Generation. Presented at the September (2018). https://doi.org/10.1109/CVPR.2018.00257.

13. Baker, B., Gupta, O., Naik, N., Raskar, R.: Designing Neural Network Architectures using Reinforcement Learning. ArXiv. abs/1611.02167, (2017).

14. Zoph, B., Vasudevan, V., Shlens, J., Le, Q.: Learning Transferable Architectures for Scalable Image Recognition. Presented at the September (2018). https://doi.org/10.1109/CVPR.2018.00907.

15. Zoph, B., Le, Q. v: Neural Architecture Search with Reinforcement Learning. (2016).

16. Pham, H., Guan, M., Zoph, B., Le, Q., Dean, J.: Efficient Neural Architecture Search via Parameters Sharing. In: Dy, J. and Krause, A. (eds.) Proceedings of the 35th International Conference on Machine Learning. pp. 4095–4104. PMLR (2018).

17. Stanley, K.O., Miikkulainen, R.: Evolving Neural Networks through Augmenting Topologies. Evol. Comput. 10, 99–127 (2002). https://doi.org/10.1162/106365602320169811.

18. Miikkulainen, R., Liang, J., Meyerson, E., Rawal, A., Fink, D., Francon, O., Raju, B., Shahrzad, H., Navruzyan, A., Duffy, N., Hodjat, B.: Chapter 15 - Evolving Deep Neural Networks. In: Kozma, R., Alippi, C., Choe, Y., and Morabito, F.C. (eds.) Artificial Intelligence in the Age of Neural Networks and Brain Computing. pp. 293–312. Academic Press (2019). https://doi.org/https://doi.org/10.1016/B978-0-12-815480-9.00015-3.

19. Real, E., Moore, S., Selle, A., Saxena, S., Suematsu, Y.L., Tan, J., Le, Q., Kurakin, A.: Large-Scale Evolution of Image Classifiers. (2017).

20. Real, E., Aggarwal, A., Huang, Y., Le, Q. v: Regularized Evolution for Image Classifier Architecture Search. Proceedings of the AAAI Conference on Artificial Intelligence. 33, 4780–4789 (2019). https://doi.org/10.1609/aaai.v33i01.33014780.

21. Suganuma, M., Shirakawa, S., Nagao, T.: A Genetic Programming Approach to Designing Convolutional Neural Network Architectures. (2017).

22. Elsken, T., Metzen, J.H., Hutter, F.: Efficient Multi-objective Neural Architecture Search via Lamarckian Evolution. (2018).

23. Xie, L., Yuille, A.: Genetic CNN. (2017).

24. Liu, H., Simonyan, K., Yang, Y.: DARTS: Differentiable Architecture Search. (2018).

25. Ahmed, K., Torresani, L.: MaskConnect: Connectivity Learning by Gradient Descent. In: Computer Vision – ECCV 2018. pp. 362–378. Springer International Publishing, Cham (2018).

26. Shin, R., Packer, C., Song, D.: Workshop track-ICLR 2018 DIFFERENTIABLE NEURAL NETWORK ARCHITECTURE SEARCH.

27. Mendoza Hector and Klein, A. and F.M. and S.J.T. and U.M. and B.M. and D.M. and L.M. and H.F.: Towards Automatically-Tuned Deep Neural Networks. In: Hutter Frank and Kotthoff, L. and V.J. (ed.) Automated Machine Learning: Methods, Systems, Challenges. pp. 135–149. Springer International Publishing, Cham (2019). https://doi.org/10.1007/978-3-030-05318-5_7.

28. Zela, A., Klein, A., Falkner, S., Hutter, F.: Towards Automated Deep Learning: Efficient Joint Neural Architecture and Hyperparameter Search. (2018).

29. Hutter, F., Hoos, H.H., Leyton-Brown, K.: Sequential Model-Based optimisation for General Algorithm Configuration. In: Learning and Intelligent optimisation. pp. 507–523. Springer Berlin Heidelberg, Berlin, Heidelberg (2011).

30. Klein, A., Falkner, S., Bartels, S., Hennig, P., Hutter, F.: Fast Bayesian optimisation of Machine Learning Hyperparameters on Large Datasets. (2016).

31. Bergstra, J., Yamins, D., Cox, D.D.: Making a Science of Model Search: Hyperparameter optimisation in Hundreds of Dimensions for Vision Architectures. In: Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28. pp. I–115–I–123. JMLR.org (2013).

32. Falkner, S., Klein, A., Hutter, F.: Workshop track-ICLR 2018 PRACTICAL HYPERPARAMETER optimisation FOR DEEP LEARNING.

33. Yang, Z., Wang, Y., Chen, X., Shi, B., Xu, C., Xu, C., Tian, Q., Xu, C.: CARS: Continuous Evolution for Efficient Neural Architecture Search. (2019).

34. Maziarz, K., Tan, M., Khorlin, A., Georgiev, M., Gesmundo, A.: Evolutionary-Neural Hybrid Agents for Architecture Search. (2018).

35. Chen, Y., Meng, G., Zhang, Q., Xiang, S., Huang, C., Mu, L., Wang, X.: Reinforced Evolutionary Neural Architecture Search. (2018).

36. Sun, Y., Wang, H., Xue, B., Jin, Y., Yen, G.G., Zhang, M.: Surrogate-Assisted Evolutionary Deep Learning Using an End-to-End Random Forest-Based Performance Predictor. IEEE transactions on evolutionary computation. 24, 350–364 (2020).

37. Wang, B., Sun, Y., Xue, B., Zhang, M.: A Hybrid Differential Evolution Approach to Designing Deep Convolutional Neural Networks for Image Classification. (2018).

38. Staniak, M., Biecek, P.: The landscape of R packages for automated exploratory data analysis. The R journal. 11, 347–369 (2019).

39. Krishnan, S., Franklin, M.J., Goldberg, K., Wu, E.: BoostClean: Automated Error Detection and Repair for Machine Learning. ArXiv. abs/1711.01299, (2017).

40. Krishnan, S., Wu, E.: AlphaClean: Automatic Generation of Data Cleaning Pipelines. (2019).

41. Kotthoff Lars and Thornton, C. and H.H.H. and H.F. and L.-B.K.: Auto-WEKA: Automatic Model Selection and Hyperparameter optimisation in WEKA. In: Hutter Frank and Kotthoff, L. and V.J. (ed.) Automated Machine Learning: Methods, Systems, Challenges. pp. 81–95. Springer International Publishing, Cham (2019). https://doi.org/10.1007/978-3-030-05318-5_4.

42. Olson Randal S. and Moore, J.H.: TPOT: A Tree-Based Pipeline optimisation Tool for Automating Machine Learning. In: Hutter Frank and Kotthoff, L. and V.J. (ed.) Automated Machine Learning: Methods, Systems, Challenges. pp. 151–160. Springer International Publishing, Cham (2019). https://doi.org/10.1007/978-3-030-05318-5_8.

43. Feurer Matthias and Klein, A. and E.K. and S.J.T. and B.M. and H.F.: Auto-sklearn: Efficient and Robust Automated Machine Learning. In: Hutter Frank and Kotthoff, L. and V.J. (ed.) Automated Machine Learning: Methods, Systems, Challenges. pp. 113–134. Springer International Publishing, Cham (2019). https://doi.org/10.1007/978-3-030-05318-5_6.

44. Caruana, R., Niculescu-Mizil, A., Crew, G., Ksikes, A.: Ensemble Selection from Libraries of Models. (2004).

45. microsoft/nni: An open source AutoML toolkit for automate machine learning lifecycle, including feature engineering, neural architecture search, model compression and hyper-parameter tuning., https://github.com/Microsoft/nni, last accessed 2021/09/30.

46. Bischl, B., Richter, J., Bossek, J., Horn, D., Thomas, J., Lang, M.: mlrMBO: A Modular Framework for Model-Based optimisation of Expensive Black-Box Functions. (2017).

47. CRAN - Package parsnip, https://cran.r-project.org/web/packages/parsnip/index.html, last accessed 2021/09/30.

48. Barga, R.: Predictive Analytics with Microsoft Azure Machine Learning. Apress, Berkeley, CA (2015).

49. Das, P., Ivkin, N., Bansal, T., Rouesnel, L., Gautier, P., Karnin, Z., Dirac, L., Ramakrishnan, L., Perunicic, A., Shcherbatyi, I., Wu, W., Zolic, A., Shen, H., Ahmed, A., Winkelmolen, F., Miladinovic, M., Archembeau, C., Tang, A., Dutt, B., Grao, P., Venkateswar, K.: Amazon SageMaker Autopilot: A white box AutoML solution at scale. In: Proceedings of the 4th Workshop on Data Management for End-To-End Machine Learning, DEEM 2020 - In conjunction with the 2020 ACM SIGMOD/PODS Conference. Association for Computing Machinery, Inc (2020). https://doi.org/10.1145/3399579.3399870.

50. Wong, C., Houlsby, N., Lu, Y., Gesmundo, A.: Transfer Learning with Neural AutoML. (2018).

51. Wang, D., Ram, P., Weidele, D., Liu, S., Muller, M., Weisz, J., Valente, A., Chaudhary, A., Torres, D., Samulowitz, H., Amini, L.: AutoAI: Automating the End-to-End AI Lifecycle with Humans-in-the-Loop. In: Proceedings of the 25th International Conference on intelligent user interfaces companion. pp. 77–78. ACM (2020).

52. Shah, S.Y., Patel, D., Vu, L., Dang, X.-H., Chen, B., Kirchner, P., Samulowitz, H., Wood, D., Bramble, G., Gifford, W.M., Ganapavarapu, G., Vaculin, R., Zerfos, P.: AutoAI-TS: AutoAI for Time Series Forecasting. In: Proceedings of the 2021 International Conference on Management of Data. pp. 2584–2596. Association for Computing Machinery, New York, NY, USA (2021). https://doi.org/10.1145/3448016.3457557.

53. H2O.ai | AI Cloud Platform, https://www.h2o.ai/, last accessed 2021/09/22.

54. Darren Cook: Practical Machine Learning with H2O. O'Reilly Media (2016).

55. Feurer Matthias and Hutter, F.: Hyperparameter optimisation. In: Hutter Frank and Kotthoff, L. and V.J. (ed.) Automated Machine Learning: Methods, Systems, Challenges. pp. 3–33. Springer International Publishing, Cham (2019). https://doi.org/10.1007/978-3-030-05318-5_1.

56. Vanschoren, J.: Meta-Learning. In: Hutter Frank and Kotthoff, L. and V.J. (ed.) Automated Machine Learning: Methods, Systems, Challenges. pp. 35–61. Springer International Publishing, Cham (2019). https://doi.org/10.1007/978-3-030-05318-5_2.

57. Finn, C., Abbeel, P., Levine, S.: Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. (2017).

58. Nichol, A., Achiam, J., Schulman, J.: On First-Order Meta-Learning Algorithms. (2018).

59. Mishra, N., Rohaninejad, M., Chen, X., Abbeel, P.: A Simple Neural Attentive Meta-Learner. (2017).

60. Yao, H., Wu, X., Tao, Z., Li, Y., Ding, B., Li, R., Li, Z.: Automated Relational Meta-learning. (2020).

61. Yakovlev, A., Moghadam, H.F., Moharrer, A., Cai, J., Chavoshi, N., Varadarajan, V., Agrawal, S.R., Idicula, S., Karnagel, T., Jinturkar, S., Agarwal, N.: Oracle AutoML. Proceedings of the VLDB Endowment. 13, 3166–3180 (2020). https://doi.org/10.14778/3415478.3415542.

62. Li, Y., Shen, Y., Zhang, W., Jiang, J., Ding, B., Li, Y., Zhou, J., Yang, Z., Wu, W., Zhang, C., Cui, B.: VolcanoML: Speeding up End-to-End AutoML via Scalable Search Space Decomposition. (2021).

63. Welcome to Apache Avro!, https://avro.apache.org/, last accessed 2021/09/30.

64. Apache Parquet, https://parquet.apache.org/, last accessed 2021/09/30.

65. TFRecord and tf.train.Example | TensorFlow Core, https://www.tensorflow.org/tutorials/load_data/tfrecord, last accessed 2021/09/30.

66. Breck, E., Polyzotis, N., Roy, S., Whang, S., Zinkevich, M.: Data Validation for Machine Learning. In: Talwalkar, A., Smith, V., and Zaharia, M. (eds.) Proceedings of Machine Learning and Systems 2019, MLSys 2019, Stanford, CA, USA, March 31 - April 2, 2019. mlsys.org (2019).

67. Zaharia, M., Chowdhury, M., Franklin, M.J., Shenker, S., Stoica, I.: Spark: Cluster Computing with Working Sets. In: Proceedings of the 2nd USENIX Conference on Hot Topics in Cloud Computing. p. 10. USENIX Association, USA (2010).

68. Apache Beam, https://beam.apache.org/, last accessed 2021/09/30.

69. Welcome to Apache Flume — Apache Flume, https://flume.apache.org/, last accessed 2021/09/30.

70. Murray, D.G., Simsa, J., Klimovic, A., Indyk, I.: tf.data: A Machine Learning Data Processing Framework. Proc. VLDB Endow. 14, 2945–2958 (2021).

71. Lindauer, M., Hutter, F.: Best Practices for Scientific Research on Neural Architecture Search, (2019).

72. Eggensperger, K.: Towards an Empirical Foundation for Assessing Bayesian optimisation of Hyperparameters. Presented at the (2013).

73. Bischl, B., Kerschke, P., Kotthoff, L., Lindauer, M., Malitsky, Y., Fréchette, A., Hoos, H., Hutter, F., Leyton-Brown, K., Tierney, K., Vanschoren, J.: ASlib: A benchmark library for algorithm selection. Artificial Intelligence. 237, 41–58 (2016). https://doi.org/https://doi.org/10.1016/j.artint.2016.04.003.

74. Ying, C., Klein, A., Real, E., Christiansen, E., Murphy, K., Hutter, F.: NAS-Bench-101: Towards Reproducible Neural Architecture Search. (2019).

75. Zela, A., Siems, J., Hutter, F.: NAS-Bench-1Shot1: Benchmarking and Dissecting One-shot Neural Architecture Search. (2020).

76. Dong, X., Yang, Y.: NAS-Bench-201: Extending the Scope of Reproducible Neural Architecture Search. ArXiv. abs/2001.00326, (2020).

77. Agrapetidou, A., Charonyktakis, P., Gogas, P., Papadimitriou, T., Tsamardinos, I.: An AutoML application to forecasting bank failures. Applied economics letters. 28, 5–9 (2021).

78. Li, Y., Wang, Z., Ding, B., Zhang, C.: AutoML: A Perspective Where Industry Meets Academy. In: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining. pp. 4048–4049. Association for Computing Machinery, New York, NY, USA (2021). https://doi.org/10.1145/3447548.3470827.

79. Guyon Isabelle and Sun-Hosoya, L. and B.M. and E.H.J. and E.S. and L.Z. and J.D. and R.B. and S.M. and S.M. and S.A. and T.W.-W. and V.E.: Analysis of the AutoML Challenge Series 2015–2018. In: Hutter Frank and Kotthoff, L. and V.J. (ed.) Automated Machine Learning: Methods, Systems, Challenges. pp. 177–219. Springer International Publishing, Cham (2019). https://doi.org/10.1007/978-3-030-05318-5_10.

80. Mao, Y., Wang, D., Muller, M., Varshney, K., Baldini, I., Dugan, C., Mojsilović, A.: How Data ScientistsWork Together With Domain Experts in Scientific Collaborations: To Find The Right Answer Or To Ask The Right Question? Proceedings of the ACM on human-computer interaction. 3, 1–23 (2019).

81. Wang, Q., Ming, Y., Jin, Z., Shen, Q., Liu, D., Smith, M., Veeramachaneni, K., Qu, H.: ATMSeer: Increasing Transparency and Controllability in Automated Machine Learning. In: Proceedings of the 2019 CHI Conference on human factors in computing systems. pp. 1–12. ACM (2019).

82. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, É.: Scikit-Learn: Machine Learning in Python. J. Mach. Learn. Res. 12, 2825–2830 (2011).

83. Commercial grade deep learning tooling - brewai, https://www.brewai.com/, last accessed 2021/09/30.

84. Whyte, G., Zeng, A.: BrewAI Features Development Road Map, Internal Company Report. (2021).

85. PyTorch, https://pytorch.org/, last accessed 2021/10/04.

86. Akiba, T., Sano, S., Yanase, T., Ohta, T., Koyama, M.: Optuna: A Next-generation Hyperparameter optimisation Framework. In: Proceedings of the 25th ACM SIGKDD International Conference on knowledge discovery & data mining. pp. 2623–2631. ACM (2019).

87. Bergstra, J., Komer, B., Eliasmith, C., Yamins, D., Cox, D.D.: Hyperopt: a Python library for model selection and hyperparameter optimisation. Computational science & discovery. 8, 14008 (2015).

88. Dhamdhere, K., Sundararajan, M., Yan, Q.: How Important Is a Neuron? (2018).

89. Apache Airflow, https://airflow.apache.org/, last accessed 2021/09/30.

90. Kubernetes, https://kubernetes.io/, last accessed 2021/09/22.

91. Titanic - Machine Learning from Disaster | Kaggle, https://www.kaggle.com/c/titanic, last accessed 2021/09/30.

92. Drozdal, J., Weisz, J., Wang, D., Dass, G., Yao, B., Zhao, C., Muller, M., Ju, L., Su, H.: Trust in AutoML: exploring information needs for establishing trust in automated machine learning systems. In: Proceedings of the 25th International Conference on intelligent user interfaces. pp. 297–307. ACM (2020).

93. Liu, C., Dollár, P., He, K., Girshick, R., Yuille, A., Xie, S.: Are Labels Necessary for Neural Architecture Search?, (2020).

94. Cisneros Cabrera, S., Mehandjiev, N., Felfernig, A., Sampaio, P., Kununka, S.: A Laddering Approach to Explore the Motivations of Taking Computer Advice for Supply Networks Formation. In: PACIS 2020 Proceedings. 218, (2020).

95. Rabhi, F.A., Mehandjiev, N., Baghdadi, A.: State-of-the-Art in Applying Machine Learning to Electronic Trading. In: Proceedings of International Workshop on Enterprise Applications, Markets and Services in the Finance Industry (FinanceCom 2020). pp. 3–20 (2020).

96. Hou, Y., Wang, D.: Hacking with NPOs: Collaborative Analytics and Broker Roles in Civic Data Hackathons. Proceedings of the ACM on human-computer interaction. 1, 1–16 (2017).

97. Arya, V., Bellamy, R.K.E., Chen, P.-Y., Dhurandhar, A., Hind, M., Hoffman, S.C., Houde, S., Liao, Q.V., Luss, R., Mojsilović, A., Mourad, S., Pedemonte, P., Raghavendra, R., Richards, J., Sattigeri, P., Shanmugam, K., Singh, M., Varshney, K.R., Wei, D., Zhang, Y.: One Explanation Does Not Fit All: A Toolkit and Taxonomy of AI Explainability Techniques. (2019).